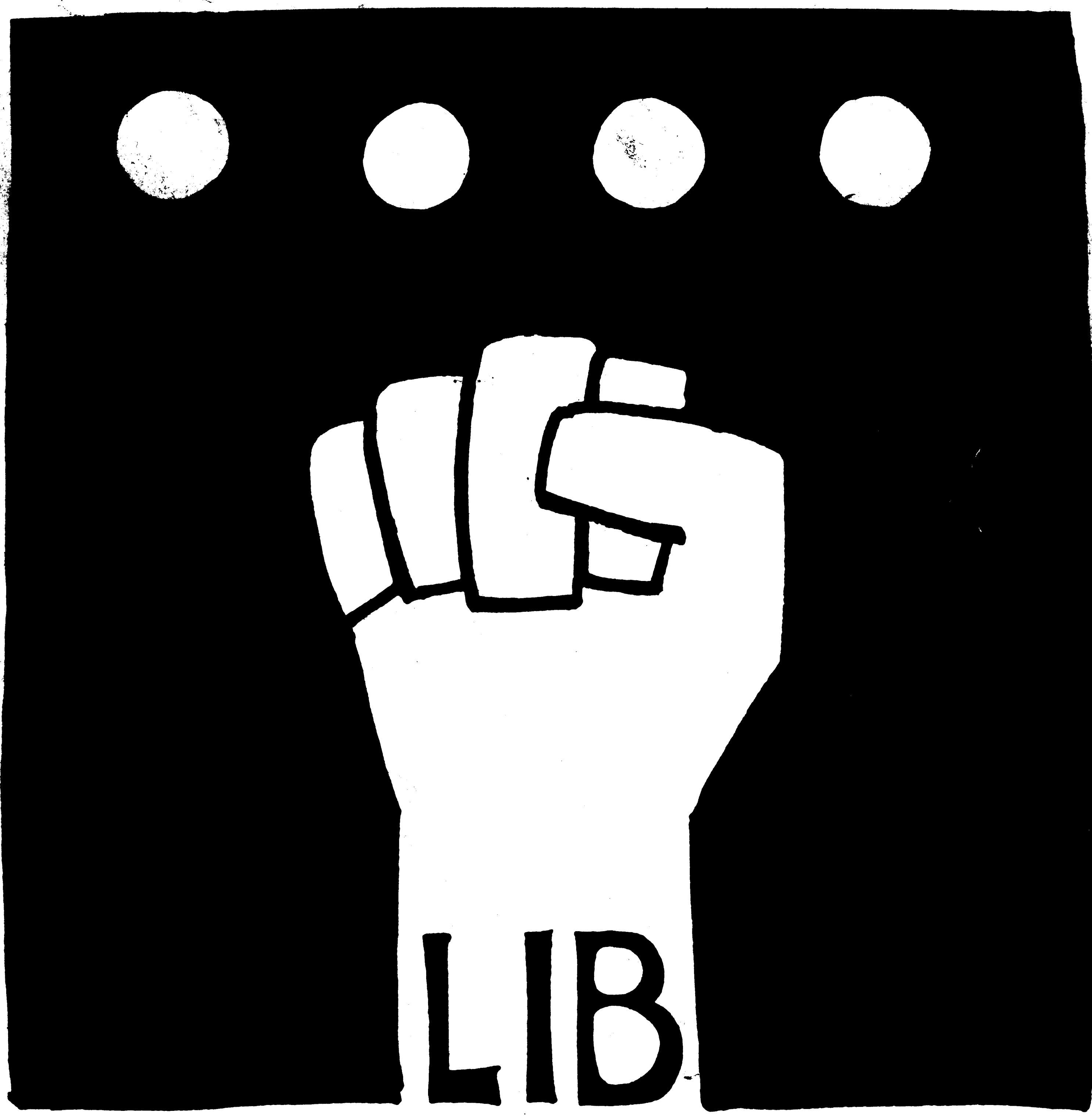
You can and must understand computers NOW

QA
76
N43++

COMPUTER



SEVEN DOLLARS.

First Edition.

BUSINESS
QA
76
N43++

COM

(c)1974 Theodor H. Nelson.
All rights reserved.

FIFTH PRINTING March '77
Copies available $7.00 for one, $50. for ten (with order) OR wholesale from: the distributors
702 S Michigan So Bend IN 46618

Any nitwit can understand computers, ...
Unfortunately, due to ridiculous historical circumstances, computers have been made a mystery to most of the world. And this situation does not seem to be improving. You hear more and more about computers, but to most people it's just one big blur. The people who know about computers often seem unwilling to explain things or answer your questions. Stereotyped notions develop about computers operating in fixed ways-- and so confusion increases. The chasm between laymen and computer people widens fast and dangerously.

This book is a measure of desperation, so serious and abysmal is the public sense of confusion and ignorance. Anything with buttons or lights can be palmed off on the layman as a computer. There are so many different things, and their differences are so important; yet to the lay public they are lumped together as "computer stuff," indi[...] and beyond understanding or criticism. It's as if [...] couldn't tell apart camera from exposure meter or [...] or car from truck or tollbooth. This book is theref[...] to the premise that

2

DATE DUE
LIBRARY ...

EVERYBODY SHOULD UNDERSTAND COMPUTERS.

It is intended to fill a crying need. Lots of everyday people have asked me where they can learn about computers, and I have had to say nowhere. Most of what is written about computers for the layman is either unreadable or silly. (Some exceptions are listed nearby; you can go to them instead of this if you want.) But virtually nowhere is the big picture simply enough explained. Nowhere can one get a simple, soup-to-nuts overview of what computers are really about, without technical or mathematical mumbo-jumbo, complicated examples, or talking down. This book is an attempt.

(And nowhere have I seen a simple book explaining to the layman the fabulous wonderland of computer graphics which awaits us all, a matter which means a great deal to me personally, as well as a lot to all of us in general. That's discussed on the flip side.)

Computers are simply a necessary and enjoyable part of life, like food and books. Computers are not everything, they are just an aspect of everything, and not to know this is computer illiteracy, a silly and dangerous ignorance.

Computers are as easy to understand as cameras. I have tried to make this book like a photography magazine-- breezy, forceful and as vivid as possible. This book will explain how to tell apples from oranges and which way is up. If you want to make cider, or help get things right side up, you will have to go on from here.

I am not a skillful programmer, hands-on person or eminent professional; I am just a computer fan, computer fanatic if you will. But if Dr. David Reuben can write about sex I can certainly write about computers. I have written this like a letter to a nephew, chatty and personal. This is perhaps less boring for the reader, and certainly less boring for the writer, who is doing this in a hurry. Like a photography magazine, it throws at you some rudiments in a merry setting. Other things are thrown in so you'll get the sound of them, even if the details are elusive. (We learn most everyday things by beginning with vague impressions, but somehow encouraging these is not usually felt to be respectable.) What I have chosen for inclusion here has been arbitrary, based on what might amuse and give quick insight. Any bright highschool kid, or anyone else who can stumble through the details of a photography magazine, should be able to understand this book, or get the main ideas. This will not make you a programmer or a computer person, though it may help you talk that talk, and perhaps make you feel more comfortable (or at least able to cope) when new machines encroach on your life. If you can get a chance to learn programming-- see the suggestions on p. -- it's an awfully good experience for anybody above fourth grade. But the main idea of this book is to help you tell apples from oranges, and which way is up. I hope you do go on from here, and have made a few suggestions.

I am "publishing" this book myself, in this first draft form, to test its viability, to see how mad the computer people get, and to see if there is as much hunger to understand computers, among all you Folks Out There, as I think. I will be interested to receive corrections and suggestions for subsequent editions, if any. (The computer field is its own exploding universe, so I'll worry about up-to-dateness at that time.)

192 58

SUMMARY OF THIS BOOK

Man has created the myth of "the computer" in his own image, or one of them: cold, immaculate, sterile, "scientific," oppressive.

Some people flee this image. Others, drawn toward it, have joined the cold-sterile-oppressive cult, and propagate it like a faith. Many are still about this mischief, making people do things rigidly and saying it is the computer's fault.

Still others see computers for what they really are: versatile gizmos which may be turned to any purpose, in any style. And so a wealth of new styles and human purposes are being proposed and tried, each proponent propounding his own dream in his own very personal way.

This book presents a panoply of things and dreams. Perhaps some will appeal to the reader...

THE COMPUTER PRIESTHOOD

Knowledge is power and so it tends to be hoarded. Experts in any field rarely want people to understand what they do, and generally enjoy putting people down.

Thus if we say that the use of computers is dominated by a priesthood, people who spatter you with unintelligable answers and seem unwilling to give you straight ones, it is not that they are different in this respect from any other profession. Doctors, lawyers and construction engineers are the same way.

But computers are very special, and we have to deal with them everywhere, and this effectively gives the computer priesthood a stranglehold on the operation of all large organizations, of government bureaux, and anything else that they run. Members of Congress are now complaining about control of information by the computer people, that they [...]t the information even though it's on computers. [...] is it seems a small matter that in ordinary companies [...]d" personnel can't get straight questions answered [...]er people; but it's the same phenomenon.

It is imperative for many reasons that the appalling gap between public and computer insider be closed. As the saying goes, war is too important to be left to the generals. Guardianship of the computer can no longer be left to a priesthood. I see this as just one example of the creeping evil of Professionalism,* the control of aspects of society by cliques of insiders. There may be some chance, though, that Professionalism can be turned around. Doctors, for example, are being told that they no longer own people's bodies.** And this book may suggest to some computer professionals that their position should not be as sacrosanct as they have thought, either.

This in not to say that computer people are trying to louse everybody up on purpose. Like anyone trying to do a complex job as he sees fit, they don't want to be bothered with idle questions and complaints. Indeed, probably any group of insiders would have hoarded computers just as much. If the computer had evolved from the telegraph (which it just might have), perhaps the librarians would have hoarded it conceptually as much as the math and engineering people have. But things have gone too far. People have legitimate complaints about the way computers are used, and legitimate ideas for ways they should be used, which should no longer be shunted aside.

In no way do I mean to condemn computer people in general. (Only the ones who don't want you to know what's going on.) The field is full of fine, imaginative people. Indeed, the number of creative and brilliant people known within the field for their clever and creative contributions is considerable. They deserve to be known as widely as, say, good photographers or writers.

> "Computers are catching hell from growing multitudes who see them uniformly as the tools of the regulation and suffocation of all things warm, moist, and human. The charges, of course, are not totally unfounded, but in their most sweeping form they are ineffective and therefore actually an acquiescence to the dehumanization which they decry. We clearly need a much more discerning evaluation in order to clarify the ethics of various roles of machines in human affairs."
>
> Ken Knowlton
> in "Collaborations with Artists-- a Programmer's Reflections"
> in Nake & Rosenfeld, eds., Graphic Languages (North-Holland Pub. Co.), p. 399.

* This is a side point. I see Professionalism as a spreading disease of the present-day world, a sort of poly-oligarchy by which various groups (subway conductors, social workers, bricklayers) can bring things to a halt if their particular new increased demands are not met. (Meanwhile, the irrelevance of each profession increases, in proportion to its increasing rigidity.) Such lucky groups demand more in each go-round-- but meantime, the number who are permanently unemployed grows and grows.

** Ellen Frankfort, Vaginal Politics. Quadrangle Books. Boston Women's Health Collective, Our Bodies, Ourselves. Simon & Schuster.

Cornell University Library
QA 76.N43
Computer lib :you can and must understan
3 1924 017 642 889

This side of the book, Computer Lib proper (whose title is nevertheless the simplest way to refer to both halves), is an attempt to explain simply and concisely why computers are marvelous and wonderful, and what some main things are in the field.

The second half of the book, Dream Machines, is specially about fantasy and imagination, and new techniques for it. That half is related to this half, but can be read first; I wanted to separate them as distinctly as possible.

The remarks below all refer to this first half, the Computer Lib half of the book.

FANDOM

With this book I am no longer calling myself a computer professional. I'm a computer fan, and I'm out to make you one. (All computer professionals were fans once, but people get crabbier as they get older, and more professional.) A generation of computer fans and hobbyists is well on its way, but for the most part these are people who have had some sort of an In. This is meant to be an In for those who didn't get one earlier.

The computer fan is someone who appreciates the options, fun, excitement, and fiendish fascination of computers. Not only is the computer fun in itself, like electric trains; but it also extends to you a wide variety of possible personal uses. (In case you don't know it, the price of computers and of using them is going down as fast as every other price is going up. So in the next few decades we may be reduced to eating soybeans and carrots, but we'll certainly have computers.)

Somehow the idea is abroad that computer activities are uncreative, as compared, say, with rotating clay against your fingers until it becomes a pot. This is categorically false. Computers involve imagination and creation at the highest level. Computers are an involvement you can really get into, regardless of your trip or your karma. They are toys, they are tools, they are glorious abstractions. So if you like mental creation, toy trains, or abstractions, computers are for you. If you are interested in democracy and its future, you'd better understand computers. And if you are concerned about power and the way it is being used, and aren't we all right now, the same thing goes.

THE SOCIETY

Which brings us to our next topic.

There is no question of whether the computer will remake society; it has. You deal with computers perhaps many times a day-- or worse, computers deal with you, though you may not know it. Computers are going into everything, are intertwined with everything, and it's going to get more and more so. The reader should have a sense of the dance of options, the remarkably different ways that computers may be used; by extension, he should come to see the extraordinary range of options which confront us as a society in our future use of them. Indeed, computers have with a swoop expanded the options of everything.

But a variety of inconvenient systems already touch on our lives, nuisances we must deal with all the time; and I fear that worse is to come. I would like to alert the reader, in no uncertain terms, that the time has come to be openly attentive and critical in observing and dealing with computer systems; and to transform criticism into action. If systems are bad, annoying and demeaning, these matters should be brought to the attention of the perpetrators. Politely at first. But just as the atmospheric pollution fostered by GM has become a matter for citizen concern and attack through legitimate channels of protest, so too should the procedural pollution of inconsiderate computer systems become a matter for the same kinds of concern. The reader should realize he can criticize and demand;

THE PUBLIC DOES NOT HAVE TO TAKE
WHAT'S BEING DISHED OUT.

There is already a backlash against computers, and the spirit of this anti-computer backlash is correct, but should be directed against very specific kinds of things. The public should stop being mad at "computers" in the abstract, and start being mad at the people who make inconvenient systems. It is not "the computer," which has no intrinsic style or character, which is at fault; it is people who use "the computer" as an excuse to inconvenience you, who are at fault. The mechanisms of legitimate public protest-- sit-ins and so on-- should perhaps soon be turned to complaint over bad and inhuman computer systems.

The question is, will the crummier trends continue? Or can the public learn, in time, what good and beautiful things are possible, and translate this realization into an effective demand? I do not believe this is an obscure or specialized issue. Its shadow falls across the future of mankind, if any, like a giant sequoia. Either computer systems are going to go on inconveniencing our lives, or they are going to be turned around to make life better. This is one of the directions that consumerism should turn.

I have an axe to grind: I want to see computers useful to individuals, and the sooner the better, without necessary complication or human servility being required. Anyone who agrees with these principles is on my side, and anyone who does not, is not.

THIS BOOK IS FOR PERSONAL FREEDOM,
AND AGAINST RESTRICTION AND COERCION.

That's really all it's about. Many people, for reasons of their own, enjoy and believe in restricting and coercing people; the reader may decide whether he is for or against this principle.

A chant you can take to the streets:

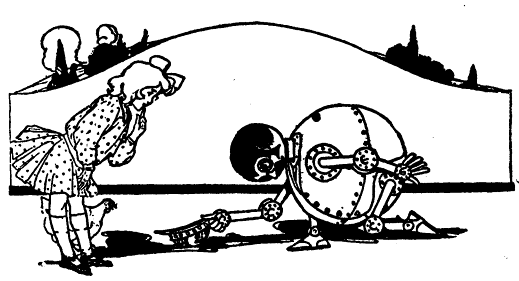
COMPUTER POWER TO THE PEOPLE!
DOWN WITH CYBERCRUD!

THE FUTURE, IF ANY

Simply as a matter of citizenship, it is essential to understand the impact and uses of computers in the world of the future, if any; and to have a sense of the issues about computers that confront us as a people-- especially privacy and data banks, but also strange new additions to our economic system ("the checkless society"), our political system (half-baked vote-at-home proposals), and so on. I regret that there is not room to cover these here.

Various companies are seeking wide public support for the sorts of things they are trying to bring about. Legislation will be proposed on which the views of the public should have a hearing. It is important that these be understood sensibly by some part of the electorate before they are made too permanent, rather than made matters of dumb assent.

Finally, and most solemnly, computers are helping us understand the unprecedented danger of our future (see "The Club of Rome," p.68). The human race may have only a short time left on earth, even if there is no war. These studies must be seen and understood by as many intelligent men of good will as possible.

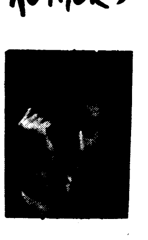


THEREFORE

Welcome to the computer world, the damndest and craziest thing that has ever happened. But we, the computer people, are not crazy. It is you others who are crazy to let us have all this fun and power to ourselves.

COMPUTERS BELONG TO ALL MANKIND.

AUTHOR'S CREDENTIALS



B.A., philosophy, Swarthmore; graduate study U. of Chicago; M.A., sociology, Harvard. Mostly self-taught in computers.
Member of editorial board, Computer Decisions magazine; listed in New York Times' Who's Who in Computers; member of Association for Computing Machinery since 1964.

Research assistant, Communication Research Institute, 1962-3. Instructor in sociology, Vassar College, 1964-6.
Senior staff researcher, Harcourt, Brace & World Publishers, 1966-7. Consultant to Bell Telephone Laboratories, Whippany, N.J., 1967-8.
Consultant to CBS Laboratories, Stamford, Ct., 1968-9. Proprietor of The Nelson Organization, Inc., New York City, 1969-72.
Lecturer in art, instructional resources and computer science, U. Illinois at Chicago Circle, 1973-6.
Co-founder of the Itty Bitty Machine Co. computer store, Evanston, Illinois, 1976.
Venture Fund lecturer, Swarthmore College, spring 1977.

PHOTO BY ROGER FIELD.

WHERE IT'S AT

Computers are where it's at.

Recently a bank employee was accused of embezzling a million and a half dollars by clever computer programming. His programs shifted funds from hundreds of people's accounts to his own, but apparently kept things looking innocent by clever programming tricks. According to the papers, the program kept up appearances by redepositing the stolen amount in each account just as interest payments were about to be calculated, then withdrawing it again just after. ("Chief Teller Is Accused of Theft of $1.5 Million at a Bank Here." New York Times, 23 March 73, p. 1.) The alleged embezzlement was discovered, not by bank audit, but by records found on the premises of a raided bookmaker.

In a recent scandal that has rocked the insurance world, an insurance company appears to have generated thousands of fictitious customers and accounts by computer, then bilked other insurance companies-- those who re-insured the original fictitious policies-- by fictitious claims on the fictitious misfortunes of the fictitious policy-holders.

In April of 1973, according to the Chicago radio, a burglary ring had a "computerized" list of a thousand prospective victims.

There have been instances where dishonest university students, nevertheless able programmers, were able to change their course grades, stored on a central university computer.

It is not unheard of for ace programmers to create grand incomprehensible systems that run whole companies, systems they can personally play like a piano, and then blackmail their firms.

A friend of a friend of the author is an ace programmer at the Pentagon, supposedly a private supervising colonels. On days he is mad at his boss, he says, the army cannot find out its strength within 300,000 men. Or three million if he so chooses.

This awkward state of affairs, obviously spanning both the American continent and most realms of endeavor, has come about for various reasons.

First, the climate of uncomprehension leads men in management to treat computer matters as "mere technicalities"-- a myth as sinister as the public notion that computers are "scientific"-- and abandon the kind of scrutiny they sensibly apply to any other company activities.

Second, most of today's computer systems are inherently leaky and insecure-- and likely to stay that way awhile. Getting things to work on them involves giving people extraordinary and invisible powers. (Eventually this will change, but watch out for the meantime.)

The obvious consequence is simply for the computer people to be allowed to take over altogether. It may indeed be that computer people -- the more well-informed and visionary ones, anyway-- can see the farthest, and appreciate most deeply the better ways things can go, and the steps that have to be taken to get there. (And Boards of Managers can at least be partially assured that hanky-panky at the lower levels will be prevented, if men in charge know where the bodies are buried.)

That seems to be how it's going. Examples:

The president of Dartmouth College, John Kemeny, is a respected computerman and a developer of one of the important computing languages, BASIC (see p. 16).

The new president of the Russell Sage Foundation, Hugh Cline, used to teach computing at Columbia.

It's probably the same in industry. In other words, more and more, for better and for worse, things are being run by people who know how to use computers, and this trend is probably irreversible.

In some ways, of course, this is a sinister portent. In private industry it's not so bad, since the danger is more of embezzlement and botch-up than of public menace. But then there's the problem of the government. The men who manage the information tools are more and more in charge of government, too. And if we can have a Watergate without computers, just wait. (See "Burning Issues," p. 58)

The way to defend ourselves against computer people is to become computer people ourselves. Which of course is the point. We must all become computer people, at least to the extent that we have already become Automobile People and Camera People-- that is, informed enough to tell when one goes by or when someone points one at you.

MANY MANSIONS

The future is going to be full of computers, for good or ill. Many computer systems are being prepared by a variety of lunatics, idealists and dreamers, as well as profit-hungry companies and unimaginative clods, all for the benefit of mankind. Which ones will work and which ones we will like is another matter. The grand and dreamy ones bid fair to reorganize drastically the lives of mankind.

For instance, Doug Engelbart at Stanford Research Institute has a beautiful system, called NLS, that will allow us to use computers as a generalized postoffice and publication system. From your computer terminal you just sign onto Engelbart's System, and you're at once in touch with lots of writings by other subscribers, which you may call to your screen and write replies to.

(These grander and dreamier applications are discussed on the other side of this book.)

But the plain computer visions are grand enough.

The great world of time-sharing, for instance. ("Time-sharing" means that the computer's time is shared by a variety of users simultaneously. See p. 45.) If you have an account on a time-sharing computer, you can sign on from your terminal (see p. 14) over any telephone, no matter where you are, and at once do anything that particular computer allows-- calling up programs in a variety of computer languages, dipping into data on a variety of subjects as easily as one now consults a chart.

For instance, at Dartmouth College-- where time-sharing is perhaps farthest advanced as a way of life-- the user (any Dartmouth student, for instance) can just sit down at a terminal and write a simple program (in Dartmouth's BASIC language, for instance) to analyze census data. Since Dartmouth has a complete file on its time-sharing system of the detailed sample from the 1970 census, the program can buzz through that and report almost immediately the numbers of divorced Aleuts or boy millionaires in the sample, or (more significantly) the relative incomes of different ethnic groups when categorized according to the questioner's interests.

But simple time-sharing is only the beginning. Networks of computers are now coming into being. Most significant of these is the ARPANET (financed by ARPA, the Defense Department's Advanced Research Projects Agency, it is nonetheless nonmilitary in character). Dozens of large time-sharing computers around the country are being tied into the Arpanet, and a user of any of these can reach directly into the other computers of the network-- using their programs, data or other facilities. Arpanet enthusiasts see this as the wave of the future.

MINI MANSIONS

But while computers and their combinations grow bigger and bigger, they also grow smaller and smaller. A complete computer the size of an Oreo cookie is now available, guaranteed for twentyfive years (and very expensive). But its actual heart, the Intel microprocessor, is only sixty bucks now, and just wait (see Microprocessors, p. 44). By 1980 there should be as many programmed and programmable objects in your house as you now have TVs, radios and typewriters; that's a conservative estimate. But just what these devices will all be doing-- ah, there's the question that has many people talking to themselves.

OTHER COMING THINGS?

There are a lot of tall stories about what computers will do for the world. Among the most threatening, I think, are glowing reports of "scientific" politics (don't you believe it). We hear how computers will bring "science" to government, helping, for example, to redraw the lines of election districts. (See Cybercrud, p. 8 .)

Then you may also have heard that computers are going to be our new mentors and companions, tutoring us, chatting with us and perhaps lulling us to sleep-- like Hal in 2001. Worried? Good. (See "The God-Builders," flip side.) (P. DM 12)

CHUTZPAH DEPARTMENT

A college student broke through the security of the Pacific Telephone computer system from a terminal and, according to Computerworld (6 June 73), stole over a million dollars worth of equipment by ordering it delivered to him! (Penthouse, December 73, claims he was in highschool and it was only nine hundred thousand, but you get the idea.)

After serving a few weeks in jail, he has formed his own computer-security consulting company.

More power to him.

The new breed has got to be watched.

This is the urgency of this book. Remember that the man who writes the payroll program can write himself some pretty amazing checks-- perhaps to be mailed out to Switzerland, next year.

From here on it's computer politics, computer dirty tricks, computer wonderlands, computer everything.

For anyone concerned to be where it's at, then, this book will provide a few suggestions. Now is the time you either know or you don't.

Enough power talk. Knowledge is power. Here you go. Dig in.

LESSON 1: GETTING THINGS STRAIGHT

The greatest hurdle for the beginner (or "layman") is making an effort to grasp particulars of that which he hears about.

A. WHAT IS ITS NAME? Every system or proposal or project has a name of some sort. Make an effort to learn it, or you're stuck trying to refer to "that computerish thing."

(And don't be a snob about acronyms, those all-cap names and terms sprung from the foreheads of other words, like ILLIAC and PLATO and CAI. There's a need for them. Short words are too general to use for names, and long phrases are too unwieldy.)

B. IN WHAT PARTICULAR WAY DOES IT EMPLOY THE COMPUTER? For record-keeping? For looking stuff up quickly or fancily? For searching out combinations? For making up combinations and testing their properties? For enacting complex phenomena? As automatic typewriters? To play music, or just to store the written notes?

It is hoped that you will become sensitive to these distinctions, and be able to understand and remember them after somebody explains them.

Otherwise you're stuck just referring to "that computer business," and you're in with the rest of the sheep.

CORNELL UNIVERSITY LIBRARIES ★ ITHACA N.Y. ★

(Incidentally —)

People ask me often where they can learn about "science." As in all fields, magazines are usually the best sources of general orientation.

Science Digest is kind of helpful for a start, although unfortunately they print summaries of every fool study that generalizes to the hearts of all humanity from two dozen Iowa State freshmen.

Scientific American is the favorite. Some stuff is hard to read but some ,isn't; the pictures and diagrams are terrific.

Science & Technology magazine seems to me one of the better ones-- breezy, informative, not trivial.

Science magazine is read by most actual scientists, and if you have a lively curiosity and can guess at the meanings of words, will tell you an incredible amount. (This is a main source for the science articles in the New York Times, which in turn...) Their articles on politics of science, and the future, are very interesting, important, and depressing. You have to join Am. Assn. for the Advancement of Science, Washington, D.C.

Daniel S. Greenberg's Science and Government Report (sorry-- $35 a year) is what really tells it. Greenberg is the man who knows, both what is shaping up in science and the insane governmental confusions and floundering responses and grandstanding and pork-barrel initiatives...

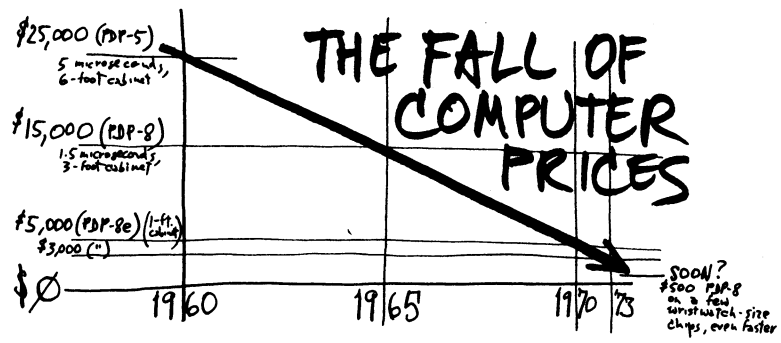
Greenberg is, incidentally, one of the finest writers of our time and a great humorist.

Science and Government Report, Kalorama Station (really?), Box 21123, Washington, D.C. 20009.

This is the wall that the handwriting is on.

ASPECTS OF THIS BOOK

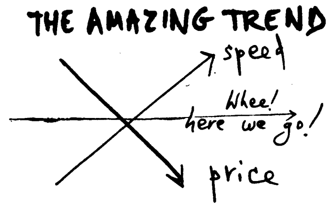
The explanations-- not yet fully debugged-- are intended for anybody. The listings of expensive products and services are intended not only as corroborative detail, for a general sense of what's available, but also for business people who might find them helpful, for affluent individuals and clubs who want to try their hand, and finally as a box score of how the prices are coming down. Because we are all going to be able to afford these things pretty soon.



This diagram shows the amazing and unique way prices drop in the computer field. The prices shown are for the first minicomputer, the PDP-5 (and its hugely popular offspring, the PDP-8); but the principle has held throughout the field, and the downward trend will probably accelerate due to the new big integrated circuits.

Another example: an IBM 7090, a very decent million-dollar computer in 1960, was put up for sale at a modish Parke-Bernet "used computer auction" in 1970. If I remember aright, they could not get a $1000 bid, because today's machines are so much smaller, faster and more dependable.

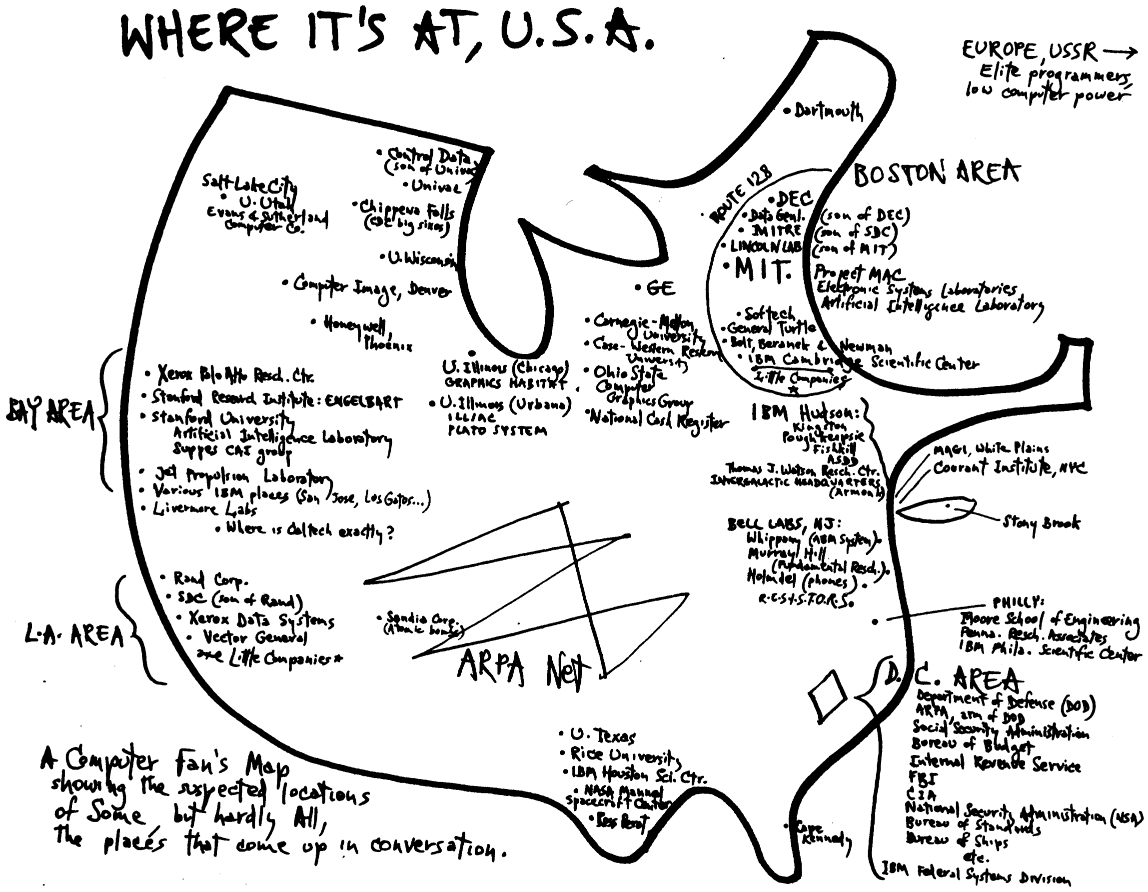
THE AMAZING TREND



WHERE IT'S AT IN THIS BOOK

2	INTRO
4	"Where It's At"
6	Sources of Information
8	CYBERCRUD
9	THE MYTH OF THE COMPUTER
10	The Power and the Glory
11	THE DEEP DARK SECRET (Computer Basics Reduced to One Easy Page)
12	THE NEW ERA
13	INTERACTIVE SYSTEMS
14	TERMINALS
15	COMPUTER LANGUAGES: Prelude
16	1. BASIC
18	2. TRAC® Language
22	3. APL
26	DATA STRUCTURES
27	Binary Patterns
30	COMPUTER LANGUAGES: Postscript
32	ROCK BOTTOM: Inner Languages of Computers; Computer Architecture
34	BUCKY'S WRISTWATCH, a sample machine-language program
35	The Assembler
36	Your Basic Computer Structure: THE MINICOMPUTER
38	BIG COMPUTERS
40	GREAT COMPUTERS: Sketches of Some Specific Machines
43	List of Mini Makers
44	MICROPROCESSORS (The New Third Kind of Computer)
45	ADVANCED PROGRAMS
45	OPERATING SYSTEMS
45	TIME-SHARING
46	COMPUTER PEOPLE
47	Program Negotiation
47	Suggestions for Writers
48	Fun and Games
50	How Computer Stuff is Bought and Sold
51	How Computer Companies are Financed, Sometimes
52	IBM
57	Digital Equipment Corporation
57	Peripherals for Your Mini
58	SIMULATION
58	OPERATIONS RESEARCH
58	GREAT ISSUES
58	MILITARY USES OF COMPUTERS
59	The ABM System
60	DNA
62	DAMN THAT COMPUTER!
64	STUFF YOU MAY RUN INTO
68	THE CLUB OF ROME

WHERE IT'S AT, U.S.A.



A Computer Fan's Map showing the suspected locations of Some but hardly All, The places that come up in conversation.

THE BUCK STOPS HERE

Everywhere in the world people can pretend that your ignorance, or position, or credentials, or poverty, or general unworthiness, are the reasons you are being pushed around or made to feel small. And because you can't tell, you have to take it.

And of course we can do the same thing with computers. Yes, we can do it in spades. (See "Cybercrud," p. 8.) But many of us do not want to. There has to be a better way. There has to be a better world.

BUSINESS
QA
76
N43++

YOUR INFORMATION SOURCES

There are several major places you get information in the computer field: friends, magazines, bingo cards, conferences and conference proceedings.

FRIENDS.

Friends we can't help with. But you might make some at conferences. Or join a computer club?

MAGAZINES.

The principal magazines are (first few listed roughly by degree of general interest):

Datamation. $15 a year or free. The main computer magazine, a breezy, clever monthly. Lots of ads, interesting articles the layman can read with not much effort. Twits IBM.
Subscriptions are $15 if you're not a computer person, free if you are. Datamation, 35 Mason St., Greenwich CT 06830.

Computer Decisions. Some $7 a year or free. Some nice light articles, as well as helpful review articles on different subjects. Avoids technicalities. Computer Decisions, 50 Essex St., Roselle Park NJ 07662.

Computers and Automation. Avoids technicalities but quite a bit of social-interest stuff. Nobody gets it free; something like $7.50 a year. Berkeley Enterprises, Inc., 815 Washington St., Newtonville, Mass. 02160.

Computerworld (actually a weekly tabloid paper). Not free: $9 a year. More up-to-the-minute than most people have time to be. Computerworld, Circ. Dept., 797 Washington St., Newton, Mass. 02160.

Computing Surveys. Excellent, clearly written introductory articles on a variety of subjects. Any serious beginner should definitely subscribe to Computing Surveys. (See ACM, below.)

Communications of the ACM. High-class journal about theoretical matters and events on the intellectual side of the field. (See ACM, below.) → "CACM."

Computer Design. $18/yr. or free. Concentrates on parts for computers, but also tells technical details of new computers and peripherals. Computer Design, Circulation Dept., P.O. Box A, Winchester, Mass. 01890.

Data Processing magazine. Oriented to conventional business applications of computers. $10. North American Publishing Co., 134 N. 13th St., Philadelphia, Pa. 19107.

Computer. (Formerly IEEE Computer Group News.) $12/yr. Thoughtful, clearly written articles on high-level topics. Quite a bit on Artificial Intelligence (see flip side). IEEE Computer Society, 16400 Ventura Blvd., Encino CA 91316.

Here are some other magazines that may interest you. No particular order.

PCC. Delightful educational/counterculture tabloid emphasizing computer games and fun. Oriented to BASIC language. $4/yr. from People's Computer Company, P.O. Box 310, Menlo Park, CA 94025.

Computing Reviews. Prints reviews, by individuals in the field, of most of the serious computer articles. Useful, but subject to individual biases and gaps. (See ACM, below.)

The New Educational Technology. $5/yr. Presumably concentrates on activities of its publisher: General Turtle, Inc., 545 Technology Square, Cambridge, MA 02139: wonderful computer toys for schools and the well-heeled.

The Honeywell Computer Journal. Something like $10 a year. Honeywell Information Systems, Inc., Phoenix, Arizona. Showcase magazine of miscellaneous content; readable, nicely edited. Has unusual practice of including microfiche (microfilm card) of entire issue in a pocket.

IBM Systems Journal. Showcase technical journal of miscellaneous content, especially arcana about IBM products. $5/yr. IBM, Armonk, NY 10504.

IBM Journal of Research and Development. Showcase technical journal of miscellaneous content. $7.50/year. IBM, Armonk, NY 10504.

Journal of the ACM. A highly technical, math-oriented journal. Heavy on graph theory and pattern recognition. (See ACM, below.) ("JACM")

Digital Design. $15 or free. About computer parts and designs. Digital Design, Circ. Dept., 167 Corey Road, Brookline, Mass. 02146.

Infosystems. Aspiring mag. $20 or free. Hitchcock Publicatons, P.O. Box 3007, Wheaton, Ill. 60187.

Think. This is the IBM house organ. Presumably free to IBM customers or prospects. IBM, Armonk, NY 10504.

There are also expensive (snob?) magazines, bought by executives.

Computer Age. $95/yr. EDP News Services Inc., 514 10th St. N.W., Washington DC 20004.

Computer Digest. $36/yr. Information Group, 1309 Cherry St., Philadelphia PA 19107.

Data Processing Digest. $51/yr. 6820 la Tijera Blvd., Los Angeles CA 90045.

Hey now, here's a magazine called Computopia. Only $15 a year. Unfortunately in Japanese. Computer Age Co. Ltd., Kasumigaseki Bldg., Box 122, Chiyoda-Ku, Tokyo, Japan.

"COMPUTER TOYS" — A WARNING

A number of inexpensive gadgets purport to teach you computer principles. Many people have been disappointed, or worse, made to feel stupid, when they learn nothing from these. Actually the best these things really can do is give you an idea of what can be done with combinations of switches. From that to learning what computer people really think about is a long, long way.

SOME GOOD BOOKS & ARTICLES FOR BEGINNERS

The best review of what's happening lately, by none other than Mr. Whole Earth Catalog himself: Stewart Brand, "Spacewar: Fanatic Life and Symbolic Death among the Computer Bums." Rolling Stone, 2 December 72, 50-56. He visited the most hotshot places and reports especially on the fun-and-games side of things.

Gilbert Burck and the Editors of Fortune, The Computer Age. Harper and Row. Ignore the ridiculous full title, The Computer Age and Its Potential for Management; this book has nothing to do with management, but is a nice general orientation to the field.

Thomas H. Crowley, Understanding Computers. McGraw-Hill. This is the most readable and straightforward introduction to the technicalities around.

Jeremy Bernstein, The Analytical Engine. Random House, 1964. History of computers, well told, and the way things looked in 1964, which wasn't really very different.

Donald E. Knuth, The Art of Programming. (7 vols.) A monumental series, excellently written and widely praised, for anyone who wants to dig in and be a serious programmer. Three of the seven volumes are out so far, at about twenty bucks apiece. Vol. 1: Fundamental Algorithms. Vol. 2: Seminumerical Algorithms. Vol. 3: Sorting and Searching. Addison-Wesley.

BUMMERS

This is perhaps a minority view, but I think any introduction to computers which makes them seem intrinsically mathematical is misleading. Historically they began as mathematical, but now this is simply the wrong way to think about them. Same goes for emphasizing business uses as if that were all.

We will not name here any of the various disagreeable pamphlets and books which stress these aspects and don't make things very clear.

ABOUT FREE SUBSCRIPTIONS. Many of the magazines are free to "qualified" readers, usually those willing to state on a signed form that they influence the purchase of computers, computer services, punch cards, or the like. (They ask other questions on the form, but whether you influence purchase is usually what decides whether they send you the magazine.) It is also helpful to have a good-sounding title or company affiliation.

BINGO CARDS.

These are little postcards you find in all the magazines except the ACM and company ones. Fill in your name and an attractive title ("Systems Consultant" or "consultant" is good-- after all, someday someone may ask your advice) and circle the numbers corresponding to the ads that entice you. You'll be flooded with interesting, expensively printed, colorful, educational material on different people's computers and accessories. And note that senders don't lose: any company wants its products known.

However, a postoffice box is good, as it helps to avoid calls at home from salesmen, wasting their time as much as yours. If you are in a rural-type area where you can assume a company name with no legal difficulties, so much the better.

POPULAR COMPUTERS

That the field has not been popularized by its better writers may simply come from an honest doubt that ordinary people can understand computers.

I dispute that. Through magazines, millions of Americans have learned about photography. Through the popular science-and-mechanics type magazines, and more recently the electronics magazines, various other technical subjects have become widely understood.

So far nobody has opened up computers. This is a first attempt. If this book won't do it another one will.

And you better believe that Popular Computers magazine is not very far away. Soon a fully-loaded minicomputer will cost less than the best hi-fi sets. In a couple of years, thousands of individuals will own computers, and millions more will want to. Look out, here we go.

Woops, here it is. Popular Computing, $15 a year ($12 if prepaid), Box 272, Calabasas, CA 91302.

THE MAIN COMPUTER ORGANIZATIONS

ACM, the Association for Computing Machinery. This is the main computer professional society; the title only has meaning historically, as many members are concerned not with machinery itself, but with software, languages, theories and so on.

If you have any plans to stick with the subject, membership in the Association for Computing Machinery is highly recommended. ACM calls itself "The Society of the Computing Community." Thus it properly embraces both professionals and fans.

Dues for official students are $8 a year, $35 for others, which includes a subscription to Communications of the ACM, the official mag. Their address for memberships and magazines is ACM, P.O. Box 12105, Church St. Station, New York, NY 10249. (The actual ACM HQ is at 1133 Ave. of the Americas, New York, N.Y. 10036.)

They have stacked the deck so that if you want to subscribe to any ACM magazines you'd better join anyway. Here are the year prices:

	Member	Non-Member
Communications of the ACM	free	$35
Computing Surveys	$7	$25
Computing Reviews	$12.50	$35
Journal of the ACM	$7	$30

The one drawback to joining the ACM is all the doggoned mailing lists it gets you on. It's unclear whether there's anything you can do to prevent this, but there oughta be.

SIGs and SICs. For ACM members with special interests (and we all have them), the ACM contains subdivisions-- clubs within the club, of people who keep in touch to share their interests. These are called SICs (Special Interest Committees) and SIGs (Special Interest Groups). There are such clubs-- SICs and SIGs-- in numerous areas, including Programming Languages, Computer Usage in Education, etc. Encouraging these subinterests to stay within ACM saves a lot of trouble for everybody and keeps ACM the central society.

AFIPS.

AFIPS is the UN of computing. They sponsored the Joints, and now sponsor the NCC. Just as individuals can't join the UN, they can't join AFIPS, which stands for American Federation of Information Processing Societies. Depending on your special interests, though, you can join a member society.

The constituent societies of AFIPS are, as of June 1973: (If any turn you on, write AFIPS for addresses: AFIPS, 210 Summit Ave., Montvale NJ 07645.)

☆ ACM: the Association for Computing Machinery.
IEEE, the Institute of Electrical and Electronics Engineers. This is the professional society of electronics guys.
Simulation Councils. This is the professional society for those interested in Simulation (see p.58).
Association for Computational Linguistics. (Where language and computer types gather.)
American Association of Aeronautics and Astronautics.
American Statistical Association.
Instrument Society of America.
Society for Information Display. (See flip side.)
American Institute of Certified Public Accountants.
American Society for Information Science. (This group is mainly for electronified librarians and information retrieval types-- see flip side.)
Society for Industrial and Applied Mathematics.
Special Libraries Association.
Association for Educational Data Systems.

IFIP. This is the international computer society. Like AFIPS, its members are societies, so joining ACM makes you an IFIP participant.

IFIP holds conferences around the world. Fun. Expense.

THE SPRING JOINT

IS NO MORE.

CONFERENCES.

Conferences in any field are exciting, at least till you reach a certain degree of boredom with the field. Computer conferences have their own heady atmosphere, compounded of a sense of elitism, of being in the witches' cauldron, and the sure sense of the impact everything you see will have as it all grows and grows. Plus you get to look at gadgets.

Usually to go for one day doesn't cost much, and at the bigger ones you get lots of free literature, have salesmen explain their things to you, see movies, hear fascinating (sometimes) speakers.

THE JOINTS! The principal computer conferences have always been the Spring Joint Computer Conference, held in an Eastern city in May, and the Fall Joint Computer Conference, held in a Western city in November (the infamous Spring Joint and Fall Joint, or SJCC and FJCC). In 1973, because of poor business the previous year, the two were collapsed into one National Computer Conference (NCC) in June (Universal Joint?) The Joints have always been sponsored by AFIPS (see below). The National Computer Conference will henceforth be annual, at least for a while.

The cost of attending is high-- while it's just a couple of dollars to look at the exhibits, this rises to perhaps fifteen dollars to go to the day's technical sessions or fifty for the week (not counting lodging and eats)-- but it's very much worth it. The lower age limit for attendees is something like twelve, unfortunately for those with interested children.

Other important conferences: the annual ACM conference in the summer; BEMA (Business Equipment Mfrs. Assn.) in the fall and spring (no theory, but lots of gadgets); and other conferens on special subjects, held all the time all over. Lists of conferences and their whereabouts are in most of the magazines; Communications of the ACM and Computer Design have the biggest lists.

CONFERENCE PROCEEDINGS.

(such as 'Proc ACM 65,' 'Proc. SJCC 68,' 'Proc. NCC 73.')

As you may know, conferences largely consist of separate "sessions" in which different people talk on specific topics, usually reading out loud from their notes and showing slides.

Conference proceedings are books which result from conferences. Supposedly they contain what each guy said; in practice people say one thing and publish another, more formal than the actual presentation.

This leads to a curious phenomenon at the main computer conferences (SJCC, FJCC, ACM and now NCC). When you register they give you a book (you're actually paying perhaps $15 for it), containing all the papers that are about to be given, nicely tricked out by their authors. If you rush to a corner and look at the book it may change your notion of which sessions to go to.

Anyway, the resulting volumes of conference proceedings are a treasure trove of interesting papers on an immense variety of computerish and not-so-computerish subjects. Great for browsing. Expensive but wonderful. (Horrible when you're moving, though, as they are big and heavy.)

JOINT PROCEEDINGS. Proceedings for the Spring Joint and Fall Joint, from the fifties to 1972, are available from AFIPS Press, as are proceedings of the 1973 NCC. (AFIPS Press, 210 Summit Avenue, Montvale NJ 07645.) They cost $20-26 each after the conference is over; less in microfilm. (At the Joint Conferences, AFIPS Press often gives discounts, at their booth, on back Joint proceedings.)
⇨ If you want to spend money to learn about the field, Proceedings of the Joint Conferences are a fine buy.

Back ACM Proceedings. From the ACM.

Other Proceedings. Often sold at counters at conferences. Or available from various publishers. Join the ACM and you'll find out soon enough.

TRY TO GET TO THE NATIONAL JOINT. Just as every Muslim should go to Mecca, every computer fan should go to a National Joint (National Computer Conference, or NCC). The next two are (check the magazines):

May 1974, Chicago
May 1975, ~~San Francisco~~ ANAHEIM.

NO QUALIFICATIONS ARE NEEDED. Think of it as a circus for smart alecks, or, if you prefer, a Deep Educational Experience.

WHAT HAPPENS IF YOU TAKE COMPUTER COURSES?

There is a lot of talk about "best" ways of teaching about computers, but in most places the actual alternatives open to those who want to learn are fairly dismal.

Universities. Universities and colleges tend to teach computing with a mathematical emphasis at the start. Indeed, most seem to require that to get into the introductory computer course, you must have had higher math (at least calculus, sometimes matrix algebra as well). This is preposterous, like requiring an engineering degree to drive a car. (Gradeschool kids can learn to program with no prerequisites.)

⇨ It seems to be to cut down enrollment, since they're not set up to deal with all those people who want to learn about computers. (And why not?) Also it's a status thing; as if this restriction somehow should keep enrollment to students with "logical minds," whatever those are, or "mathematical sophistication," as if that were relevant.

"Computer schools," community and commercial colleges, on the other hand, tend to prepare students only for the most humdrum business applications-- keypunching (which is rapidly becoming obsolete), and programming in the COBOL language on IBM business systems. This gets you no closer to the more exciting applications of computers than you were originally.

Some experimental trends are more encouraging. Some colleges, for instance, offer "computer appreciation courses," with a wider introduction to what's available and more varied programming intended to serve as an introduction to this wider horizon.

Highschool courses seem to be cutting through the junk and offering students access to minicomputers with quickie languages, usually BASIC. Both Digital Equipment Corp. and Hewlett-Packard seem to be making inroads here.

Kiddie setups, rumored to exist in Boston and San Francisco, are geared to letting grade-school children see and play with computers. Also one company (General Turtle, see p.57) is selling computer toys intended to encourage actual programming by children.

CYBERCRUD

A number of people have gotten mad at me for coining the term "cybercrud," which I define as "putting things over on people using computers." But as long as it goes on we'll need the word. At every corner of our society, people are issuing pronouncements and making other people do things and saying it's because of the computer. The function of cybercrud is thus to confuse, intimidate or pressure. We have all got to get wise to this if it is going to be curtailed.

Cybercrud takes numerous forms. All of them, however, share the patina of "science" that computers have for the layman.

1a) COMPUTER AS MAGIC WORD

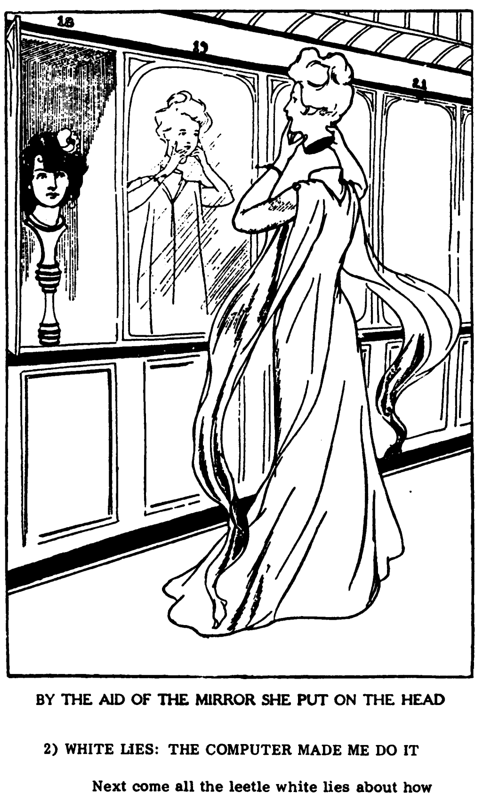
The most delicate, and seemingly innocent, technique is the practice of naming things so as spuriously to suggest that they involve computers. Thus there is a manufacturer of pot-pipes with "Data" in its name, and apparently a pornography house with a "Cyber-".

1b) COMPUTER AS MAGIC INGREDIENT

The above seems silly, but it is no less silly than talking about "computer predictions" and "computer studies" of things. The mere fact that a computer is involved in something has no bearing on its character or validity. The way things are done with computers affects their character and validity, just like the way things are done without computers. (Indeed, merely using a computer often has no bearing on the way things are done.)

This same technique is easily magnified to suggest, not merely that something involves computers, but is wholly done by computers. The word "computerize" performs this fatal function. When used specifically, as in computerize the billing operation, it can be fairly clear; but make it vague, as in computerize the office, and it can mean anything.

"Fully computerize" is worse. Thus we hear about a "fully computerized" print shop, which turns out to be one whose computers do the typesetting; but they could also run the presses, pay the bills and work the coffee machine. For practical purposes, there is no such thing as "fully" computerized. There is always one more thing computers could do.



BY THE AID OF THE MIRROR SHE PUT ON THE HEAD

2) WHITE LIES: THE COMPUTER MADE ME DO IT

Next come all the leetle white lies about how such-and-such is the computer's fault and not your decision. Thus the computer is made a General Scapegoat at the same time it's covering up for what somebody wants to do anyway.

"It has to be this way."

"There's nothing we can do; this is all handled by computer."

"The computer will not allow this."

"The computer won't let us."

The translation is, of course, THE STINKY LOUSY PROGRAM DOES NOT PERMIT IT. Which means in turn: WE DO NOT CHOOSE TO PROVIDE, IN OUR PROGRAMS AND EQUIPMENT, ANY ALTERNATIVES.

Now, it is often the case that good and sufficient reason exists for the way things are done. But it is also often the case that companies and the public are inconvenienced, or worse, by decisions the computer people make and then hide with their claim of technical necessity. (See p. 46: Dealing with computer people.)

3) YAGOTTAS: COMPUTER AS COERCER

More aggressively, cybercrud is a technique for making people do what you want. "The computer requires it," you say, and so people can be made to hand over personal information, secretaries can be intimidated into scouring the files, payment schedules can be artificially enforced.

THE GENERAL STATUS TRICK

Status tricks, combining the putdown and the self-boost, date back to times immemorial. But today they take new forms. The biggest trick is to elevate yourself and demean the listener at the same time, or, more generally, the technique is making people feel stupid while acting like a big cheese. Thus someoneone might say,

> "People must begin to get used to the objective scientific ways of doing things that computers now make necessary."

But the translation seems to be:

> "People must get used to the inflexible, badly thought out, inconvenient and unkind systems that I and other self-righteous individuals and companies are inflicting on the world."

YOU DON'T ALWAYS GOTTA

The uninformed are bulldozed, and even the informed are pressured, by the foolish myths of the clever, implacable and scientific computer to which they must adapt. People are told they have to "relate to the computer." But actually they are being made to relate to systems humans have designed around it, in much the same way a sword dance is designed around the sword.

When establishment computer people say that the computer requires you to be systematic, they generally mean you have to learn their system. But anyone who tells you a method "has to be changed for the computer" is usually fibbing. He prefers to change the method for the computer. The reasons may be bad or good. Often the computer salesman or indoctrinator will present as "scientific" techniques which were doped out or whomped up by a couple of guys in the back room.

Here is an example, as told to me. A friend of mine worked in a dress factory where they had a perfectly good system for billing and bookkeeping. Customers were listed by name and kept in alphabetical order. The fast pace of the garment industry meant that companies often changed names, and so various companies had a number of different names in the file. This bothered nobody because the people understood the system.

Then management bought a small computer, never mind what brand, and hired a couple of guys to come in and put the bookkeeping system on it.

Still okay. Indeed, small programming firms can sometimes do this sort of thing very well, because they can work flexibly with the people and don't necessarily feel committed to making it work a certain way.

Well, this was a nice instance where the existing system could have been exactly transferred to the computer. The fact that some customers had several names would certainly have been no problem; a program could have been written that allowed users to type any acceptable customer name, causing the computer to look up the correct account (and if desired, print its usual name and ask for verification).

But no. The guys did not answer employees' questions comprehensibly, nor did they want suggestions. They immediately decreed that since computers only worked with numbers (a fib, but a convenience to them), every customer would thenceforth have to be referred to by number.

After that the firm had nothing but trouble, through confusion over the multiple names, and my friend predicted that this would destroy the company. I haven't heard the outcome.

This story is not necessarily very interesting; it merely happened. It's not a made-up example.

Moral: until we overthrow the myth that people always have to adapt to computers, rather than the other way around, things will never go right. Adaptations should take place on both sides, darn it.

EVERYBODY DOES IT

Cybercrud is by no means the province of computer people alone. Business manipulators and bureaucrats have quickly learned the tricks. Companies do it to the public. The press, indeed, contributes (see Suggestions for Writers and Spokesmen, p. 47). But the computer people are best at it because they have more technicalities to shuffle around magically; they can put anybody down.

Now, computer people do deserve respect. So many things that people do with computers are hard. It can be understood that they want to be appreciated, and if not for the particulars, for the machismo (machinismo?) of coping with intricacy. But that is no excuse for keeping others in controlled ignorance. No man has a right to be proud that he is preserving and manipulating the ignorance of others.

"If it can't be done in COBOL,
I just tell people it can't be done by computer.
It saves a lot of trouble."

Attributed to somebody in Rochester.
(See COBOL, p. 31.)

BALDERDASH!

Dear Charter Subscriber,

Because [redacted]'s subscription roll is maintained by electronic computer, it is necessary for us to assign a common expiration date to all subscriptions. This enables us to distribute copies and mail renewal notices to all subscribers at the same time. Therefore, we are writing to inform you that your [redacted] Charter Subscription must be renewed now.

In the movie "Fail-Safe," they showed you lots of fake tape drives with the reels constantly turning in one direction. This they called a "computer." Calling any sinister box "a computer" is a widespread trick. Gives people the willies. Keeps 'em in line.

Dear Depositor:

PIFFLE. (the program requires it.)

Your bank is now utilizing a computer to provide you with better banking service. This new computer requires the use of a three part deposit slip. Enclosed you will find a supply of these new deposit slips. Please compare the account number on the deposit slips with the one imprinted on your checks to be sure the numbers are identical. If they agree, please start using them immediately. We recommend that you carry a few of these deposit slips in the cover of your checkbook.

If there are any questions about this new procedure, any one of our officers will be glad to help you.

You can buy little boxes with blinking lights that do nothing else but blink. They really put people uptight. "Are you recording what I say?" people ask. "Is it a computer?" They'll believe such a box is anything you tell them.

REASONS FOR CYBERCRUD (ALL BAD)

1) to manipulate situations.
2) to control others.
3) to fool.
4) to look like hot stuff.
5) to keep outsiders from seeing through something.
6) to sell something.
7) to put someone down.
8) to conceal.
9) general secretiveness.
10) low expectation of others' mentality.
11) seeking to be the broker and middleman for all relations with the computer.
12) vagueness sounds profound.
13) you don't have to show what you're not sure of.
14) your public image is monolithic.
15) you really don't know.

BEAUTIFUL BUNNY BOOTIES

Cybercrud is not aimed only at laymen. It can work even among insiders.

The operations manager of a national time-sharing service, for example, was fanatical about cleanliness. In order to assure a Clean Computer Room, he said, and hence no dangerous dust near the tapes or disks, he made a rule requiring that anyone entering the computer room had to wear cloth booties over his shoes.

Booties were hung outside for those who had to enter.

"And I had the greatest time making his," says his wife, laughing. "With the cutest little bunny faces on them. The buttons were the hardest part to get-- you know, the ones with eyes that roll!" She laughs very hard as she tells this.

"Of course there was no need for it," he now chortles, "but it sure kept people out of the computer room."

(That's applied logic for you.)

" COMPUTERS
AND THEIR PRIESTS

" First get it through your head that computers are big, expensive, fast, dumb adding-machine-typewriters. Then realize that most of the computer technicians that you're likely to meet or hire are complicators, not simplifiers. They're trying to make it look tough. Not easy. They're building a mystique, a priesthood, their own mumbo-jumbo ritual to keep you from knowing what they-- and you-- are doing."

-- Robert Townsend, Up The Organization (Knopf), p. 36.

THE CARGO-CULT ASPECT

Outsiders are often prey to cybercrud they dream up themselves. I once knew a college registrar's office where they had been getting along fine for years with paper forms. The year before the computer was slated to arrive, they started using file cards filled out by hand, instead. Why? "Well, we thought that would make it easier for the computer. Computers use cards, don't they?"

Note that referring to a computer as if it were a living creature is not cybercrud; to say that a program "looks at" a device, "tries to" effect a procedure, and "goes to sleep," are all colorful brief ways of describing what really happens. (See Guidelines for Writers and Spokesmen, p. 47)

WHAT SECRET POWERS DOES THIS MAN POSSESS?

Cybercrud is, of course, just one branch of THE GREAT GAME OF TECHNOLOGICAL PRETENSE that has the whole world in its grasp.

"Man, woman, child — all is up against the Wall of Science."
Firesign Theater

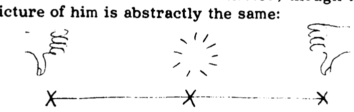
THE MYTH OF THE MACHINE: A DEEP CULTURAL ENGRAM

Public thinking about computers is heavily tinged by a peculiar image which we may call the Myth of the Machine. It goes as follows: there is something called the Machine, which is Taking Over The World. According to this point of view The Machine is a relentless, peremptory, repetitive, invariable, monotonous, inexorable, implacable, ruthless, inhuman, dehumanizing, impersonal Juggernaut, brainlessly carrying out repetitive (and often violent) actions. Symbolic of this is of course Charlie Chaplin, dodging the relentless, repetitive, monotonous, implacable, dehumanizing gears of a machine he must deal with in the film Modern Times.

Ordinarily this view of The Machine is contrasted with an idea of a Warm Human Being, usually an idealized version of the person thinking these thoughts.

The Machine ⟵⟶ Warm Human Being

But consider something. The model often goes further than this. The Machine is cold, the Human Being emotional and warm. Yet there is such a thing as being too emotional and warm. There is in fact a third type in the schema, the being who goes too far on the same scale. Strangely, he has at least three different names, though the picture of him is abstractly the same:



The Machine	Warm Human Being	"Bum" "Nigger" "Hippie"

Now, "bums," "niggers" and "hippies" are not real people. The words are derogatory slang for the destitute, for persons with any African ancestry, and for people dressing in certain styles. But the remarkable thing about the slang is that all three of these derogatory terms seem to have the same connotation in our culture: someone who is dirty, lazy and lascivious. In other words, whatever distinguishes The Machine from the Warm Human Being is carried too far by the bunch at the other end.

In other words, this conceptual continuum is a single, fundamental scale in our culture; why is unclear. Since most people consider themselves-- naturally!-- to be in the middle category, it acts as a sort of reference continuum of two bad things on either side.

It also has another effect: it supplies a derogatory way of seeing. On the right-hand side, it allows many Americans not to see, or to see only with disgust, the destitute and those with African ancestry and those dressing in hippie style. But this book isn't about that.

The left side of the continuum is our present concern. There, too, people refuse to see. What people mainly refuse to see is that machines in general aren't like that, relentless, repetitive, monotonous, implacable, dehumanizing. Oh, there are some machines like that, particularly the automobile assembly line. But the assembly line was designed the way it is because it gets the most work out of people. It gets the work it does out of people by the way it exerts pressure.

So here we see the same old trick: people building a system and saying it has to work that way because it's a machine, rather than because that's how I designed it.

To make the point clearer, let's consider some other machines.

The automobile is a machine, but it is hardly the repetitive, "dehumanized" thing we usually hear about. It goes uphill, downhill, left and right, fast and slow. It may be decorated. It is the scene of many warm human activities. And most importantly, automobiles are very much the extension of their owners, exemplifying life-style, personality, and ideology. Consider the Baja Buggy Volkswagen and the ostentatious cushy Cadillac. Consider the dashboard ornament and the bumper sticker. The Machine, indeed.

The camera is a machine, but one that allows its user to freeze and preserve the views and images of the world he wants.

The bicycle is a machine, but one that brings you into personal and non-polluting contact with nature, or at least that stylized kind of nature accessible to bicycle paths.

To sum up, then. The Machine is a myth. The bad things in our society are the products of bad systems, bad decisions and conceivably bad people, in various combinations. Machines per se are essentially neutral, though some machines can be built which are bad indeed, such as bombs, guns and death-camps.

The myth of The Machine is a curious aspect of our ideology. Is it especially American, or world-wide?

If we ignore this myth we can see each possible machine or system for what it is, and study how it ties in with human life for good or ill, fostering or lousing up such things as the good life, preservation of species, love and self-respect.

THE MYTH AND THE RORSCHACH

"The computer is the ultimate Rorschach test," Freed Bales said to me twelve years ago. Dr. Bales, a Harvard psychologist, was somewhat perturbed by the papers he was getting in his seminar on computer modelling in the social sciences. Somewhat nutty people in the seminar were writing somewhat nutty papers for him.

And truer words were never spoken. On this point I find Bales has been terribly, terribly right. The computer is an incredible projective test: what you see in the computer comes right off the back wall of your psyche. In over a decade in the field I have not ceased to marvel at the way people's personalities entwine with the computer, each making it his own-- or rejecting it-- in his own, often unique and peculiar way, deeply reflecting his concerns and what is in his heart. Yes, odd people are attracted to the computer, and the bonds that hold them are not those of casual interest.

In fact, people tend to identify with it.

In this light we may consider the often-heard remarks about computers being rigid, narrow, and inflexible. This is of course true in a sense, but the fact that some people stress it over and over is an important clue to something about them. My own impression is that the people who stress this aspect are the comparatively rigid, narrow and inflexible people.

Other computer experts, no less worthy, tell us the computer is a supertoy, the grandest play machine ever to be discovered. These people tend to be the more outgoing, generous and playful types.

In a classic study, psychiatrist Bruno Bettelheim examined a child who thought he was a machine, who talked in staccato monosyllables, walked jerkily and decorated the side of his bed with gears. We will not discuss here the probable origins and cure of this complex; but we must consider that identifying with machines is a crucial cultural theme in American society, an available theme for all of us. And it well may be that computer people are partaking of this same self-image: in a more benign form, perhaps, a shift of gears (as it were) from Bettelheim's mechanical child, but still on the same track.

Some of the computer high-school kids I've known, because of their youth, have been even more up-front about this than adults.

I know one boy, for instance, whose dream was to put a 33ASR Teletype on wheels under radio control, and alarm people at the computer conference by having it roll up to them and clatter out questions impersonally. (If you knew the kid -- aloof and haughty-seeming-- you might think that's how he approaches people in real life.)

I know a high-school boy (not a computer expert) who programmed a computer to type out a love story, using the BASIC "print" command, the only one he knew. He could not bring himself to write the love story on paper.

The best example I can think of, though, took place at the kids' booth (see p. 47) at a computer conference. One of the more withdrawn girls was sitting at an off-line video terminal, idly typing things onto the screen. When she had gone a sentence remained. It said:

I love you all, but at a distance.

(On the other side of this book, Dream Machines, we will carry this matter further. The most exciting things in the computer field are coming from people trying to realize their wildest dreams by computer: artificial intelligence, computer music, computer picture-making and so on.)

THE POWER AND THE GLORY

Forget what you've ever heard or imagined about computers. Just consider this:

The computer is the most general machine man has ever developed. Indeed, it should be called the All-Purpose Machine, but isn't, for reasons of historical accident (see nearby). Computers can control, and receive information from, virtually any other machine. The computer is not like a bomb or a gun, which can only destroy, but more like a typewriter, wholly noncommittal between good and bad in its nature. The scope of what computers can do is breathtaking. Illustrated are some examples (although having all this happen on one computer would be unusual). It can turn things on and off, ring bells, put out fires, type out on printing machines.

Computers are incredibly dogged. Computers can do things repeatedly forever, or an exact, immense number of times (like 4,901,223), doing something over and over, depending on whether it's finished or not. A computer's activities can be combined in remarkable ways. One activity, repeated over and over, can be part of another activity repeated over and over, which can be a part of still another activity, which can be repeated ad infinitum. THERE ARE DEFINITE LIMITATIONS on what computers can do, but they are not easy to describe briefly. Also, some of them are argued about among computer people.

THE ALL-PURPOSE MACHINE

Computers are COMPLETELY GENERAL, with no fixed purpose or style of operation. In spite of this, the strange myth has evolved that computers are somehow "mathematical."

Actually von Neumann, who got the general idea about as soon as anybody (1940s), called the computer

THE ALL-PURPOSE MACHINE.

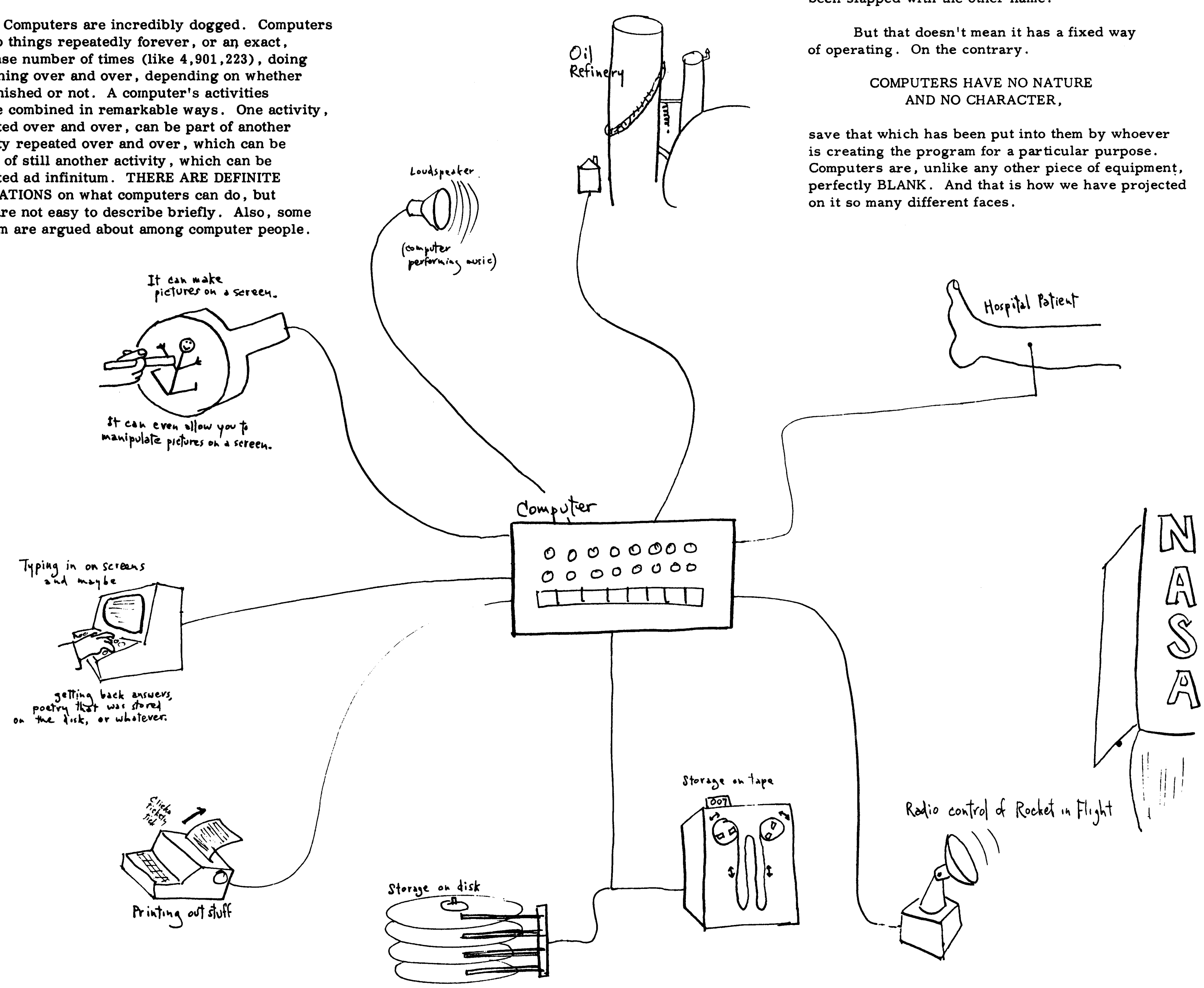
(Indeed, the first backer of computers after World War II was a maker of multi-lightbulb signs. It is an interesting possibility that if he had not been killed in an airplane crash, computers would have been seen first as text-handling and picture-making machines, and only later developed for mathematics and business.)

We would call it the All-Purpose Machine here, except that for historical reasons it has been slapped with the other name.

But that doesn't mean it has a fixed way of operating. On the contrary.

COMPUTERS HAVE NO NATURE
AND NO CHARACTER,

save that which has been put into them by whoever is creating the program for a particular purpose. Computers are, unlike any other piece of equipment, perfectly BLANK. And that is how we have projected on it so many different faces.



A HELPFUL COMPARISON

It helps sometimes to compare computers with typewriters. Both handle information according to somebody's own viewpoint.

Nervous Question	Helpful Parallel
"Can a Computer Write a Poem?"	"Can a Typewriter Write a Poem?" (Sure. Your poem.)
"Can't Computers Only Behave Mechanistically?"	"Can't Typewriters Only Behave Mechanistically?" (Yes, but carrying out your intent.)
"Aren't Computers Completely Impersonal?"	"Aren't Typewriters Completely Impersonal?" (Well, it's not like handwriting, but it's still what you say.)

Many ordinary people find computers intuitively obvious and understandable; only the complications elude them. Perhaps these intuitively helpful definitions may help your intuition as well.

1. Think of the computer as a WIND-UP CROSSWORD PUZZLE.

2. A COMPUTER IS A DEVICE FOR TWIDDLING INFORMATION. (So, what kinds of information are there? And what are the twiddling options? These matters are what the computer field consists of.)

3. A computer is <u>a completely general device, whose method of operation may be changed, for handling symbols in any specific way</u>.

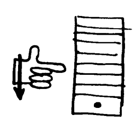
THE DEEP DARK SECRET

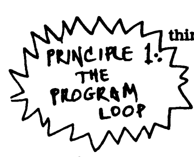
THE MAGIC OF THE COMPUTER PROGRAM

The basic, central magical interior device of the computer we shall call a program follower. A program follower is an electronic device (usually) which reads symbols specifying operations, carries out the step each specifies and goes on to the next.

The program follower reads down the list of instructions in the program, taking each instruction in turn and carrying it out before it goes on to the next.

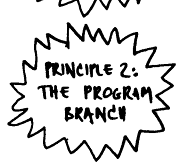
Now, there are program followers that just do that and nothing more; they have to stop when they get to the end of the list of instructions.





A true computer, however, can do several things more.

It can jump back to an earlier point in the program and go on from there. Repeating the program in this fashion is called a loop.



It can perform tests on symbols in the memory-- for instance, to see if a loop has been done enough times, or if some other part of the job has been finished-- and jump to some other program depending on these symbols. This is called a branch.

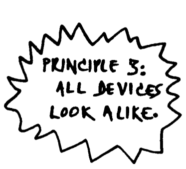
Finally, the computer can change the information stored in memory. For instance, it can place an answer in a specific part of memory.

WHAT, THEN, IS A (Digital) COMPUTER?

A device holding stored symbols
in a changeable memory,
performing operations on some of those symbols
in the memory,
in a sequence specified by other symbols
in the memory,
able to change the sequence
based on tests of symbols in the memory,
and able to change symbols in the memory.
(For example, do arithmetic and
store the result in the memory.)

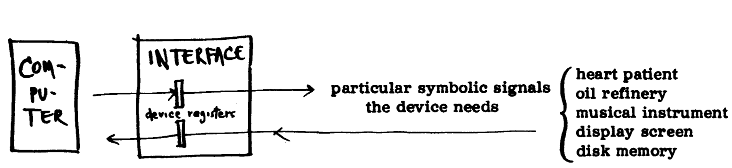
Rather than try to slip it to you or prove it in some fancy way, let's just state baldly: the power of such a machine to do almost anything surpasses all previous technical tricks in human history.

HOW CAN A COMPUTER CONTROL
SO MANY DIFFERENT THINGS?

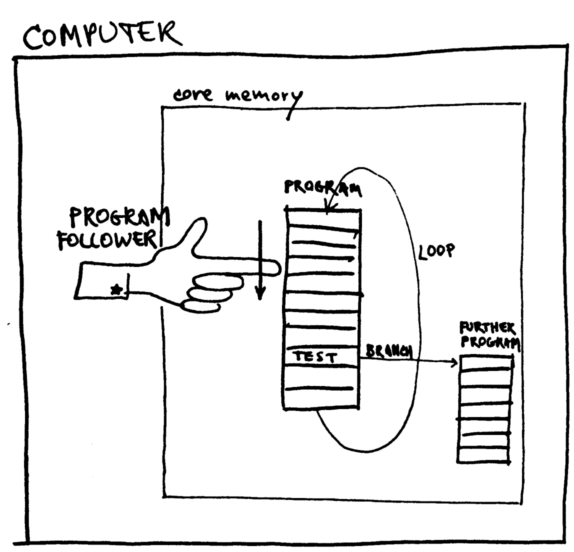


Answer. Different as they may seem, all devices are controlled in the same way. Every device has an interface, that is, its own special connection setup, and in this interface are the device registers.

These device registers look the same to the computer: the computer program simply moves information patterns into them or moves information patterns from them to see what they contain.



The computer, being a machine, doesn't know or care that device register 17 (say) controls a hog feeder, or device register 23 (say) receives information from smog detectors. But what you choose, in your program, to put into device register 17, controls what the hogs eat, and what comes into device register 23 will tell your program, you hope, about smog conditions. Choosing how to handle these things in your program is your business.

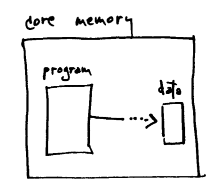


"What is an Interface?" asked the baby machine.
"Whatever Turns You On," said its dad.

HOW DOES THE LOOP WORK?

The computer does things over and over by changing a stored count, then testing the stored count against another number which is what the count should get to, and going to the beginning if the desired count has not been reached. This is called a loop. (If there's no way it can ever get out, that's an endless loop.) (Actually, the program loop is done the same way as a program branch: IF a certain count has not been reached, it branches BACK to the start of the loop.)

Other things besides programs may be stored in the memory. Anything besides programs are usually called data.



The instructions of programs use the data in different ways. Some programs use a lot of data, some use a little, some don't use any. It is one of the fascinating and powerful things about the computer that both the instructions of a program, and the data they work on, are stored as patterns of bits in the same memory, where they can be modified as needed. Indeed, the program can modify its own patterns of bits, a very important feature.

WHAT DO PROGRAMS LOOK LIKE?

In what forms are these programs stored, you ask? Well, they are written by people in computer languages, which are then stored in some form in the computer's fast core memory, where the program follower can act on them. But what does a computer language look like, you ask? Aha...

GO TO PAGE 16

(If you want to see what the bottom-most level looks like, with all the bits and things, skip ahead to p.35.)

WHATEVER IT MAY DO IN THE REAL WORLD,
to the computer program
it's just another device.

ANALOG COMPUTERS DISPOSED OF

There are two kinds of computers: analog and digital. (Also hybrid, meaning a combination.) Analog computers are so unimportant compared to digital computers that we will polish them off in a couple of paragraphs.

"Analog" is a shortened form of the word "analogy." Originally an "analog" computer was one that represented something in the real world by some other sort of physical enactment-- for instance, building a model of an economic system with tubes and liquids; this can demonstrate Keynesian economic principles remarkably well.

However, the term "analog" has come to mean almost exclusively pertaining to measurable electrical signals, and an "analog computer" is a device that creates or modifies measurable electric signals. Thus a hi-fi amplifier is an analog computer (it multiplies the signal), a music synthesizer is an analog computer (it generates and reshapes analog signals). Thus the term has deteriorated: almost anything with wires is an analog computer.

Analog computers cannot be truly programmed, only rewired.

Analog equipment is useful, important and indispensable. But it is simply not in the same class with digital computers, henceforth called "computers" in this book, which manipulate symbols on the basis of changeable symbolic programs.

"Analog computer" also means any way of calculating that involves measuring approximate readings, like a slide rule.

LET'S CALL A SPADE A SPADE

It's awfully easy to fool people with simple words, let alone buffalo them with weird technical-sounding gab. The thing about tech talk is that it can really be applied to any area. The trick lies in the arrangement of boxcar adjective nouns, and in the vague use of windy terms that have connotations in some particular technical area-- say, the space program.

Just consider. We might call a common or garden spade--

A PERSONALIZED EARTH-MOVING EQUIPMENT MODULE

A MINERALOGICAL MINI-TRANSPORT

A PERSONALIZED STRATEGIC TELLURIAN COMMAND AND CONTROL MODULE

AN AIR-TO-GROUND INTERFACE CONTOUR ADJUSTMENT PROBE

A LEVERAGED TACTILE-FEEDBACK GEOMASS DELIVERY SYSTEM

A MAN-MACHINE ENERGY-TO-STRUCTURE CONVERTER

A ONE-TO-ONE INDIVIDUALIZED GEOPHYSICAL RESTRUCTURIZER

A PORTABLE UNITIZED EARTHWORK SYNTHESIS SYSTEM

AN ENTRENCHING TOOL (Firesign Theater)

A ZERO-SUM DIRT LEVEL ADJUSTER

A FEEDBACK-ORIENTED CONTOUR MANAGEMENT PROBE AND DIGGING SYSTEM

A GRADIENT DISEQUILIBRATOR

A MASS DISTRIBUTION NEGENTROPRIZER

or Hey! A DIG-IT-ALL SYSTEM

AN EXTRA TERRESTRIAL TRANSPORT MECHANISM.

Spades, not words, should be used for shovelling. But words should help us unearth the truth.

In the computer field, the same things are often called by different names (for instance, the IBM 1800, a fairly ordinary minicomputer, is called by them the "IBM 1800 Data Acquisition and Control System"), different things are often called by the same names, and things can be inside-out and upside-down versions of each other in extraordinary variety. (Indeed, computer people may find this book inside-out, which is okay with me. Life is a Klein bottle.)

Sorting things out, then, means having a few basic concepts clear in your mind, and knowing when you see examples and variations of them.

Computer people often say that to understand computers you have to have a "logical mind."

There's no such thing. But saying such things intimidates many, especially those who have been told they do not have "logical minds."

What is meant, actually, is indeed important: in working with computers you must often work out the exact ramifications of specific combinations of things, without skipping steps.

But the other mode of thinking, the intuitive, has its place in the computer field too. Whichever your habitual style of mind, computers offer you food-- and utensils-- for thought.

HORRIBLE MISUNDERSTANDINGS

Some people think of computers as things that somehow mysteriously digest and assimilate all knowledge. "Just feed it to the computer," is the motto. But what you feed into the computer just sits there unless there's a program.

"How would you do that by computer?" is a question people often ask. The question should be, "how would you do that at all?" If there is a method for doing something which can be broken down into simple steps, and requires no human judgment, then maybe we can take those steps and program them on a computer. But maybe we can also think of a simpler way to get them done.

Then there is the idea that a computer is something you ask questions. This assumes, I guess, the earlier premise, that the computer has already digested and assimilated a lot of stuff and can sling it back at you in new arrangements.

Actually what must happen, to get "questions" answered, is this: there must be some program that puts input material into a data structure. (See "Data Structures.") Then you need programs that will count and trace, or whatever, through the data structure in ways you desire. Then you need a way to start these tracing-and-searching programs going through the data structure in ways you want. So you need a program accepting input from a keyboard, or whatever, and starting the other programs in operation...

COMPUTERS DEFINED JUST LIKE CAMERAS AND CARS

Just the way everyone can understand cameras, viz.:
"A camera is a device you point at something to willfully capture its appearance."

Just the way everyone can understand cars, viz.:
"A car is a device people get inside which then goes somewhere else, under the willful control of the driver."

Well, how about
"A computer is a device which manipulates information and external accessories, according to a plan willfully prepared by a planner."

INSPIRATIONAL MISCELLANY

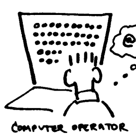
FICTIONS ABOUT WHAT COMPUTERS DO

Many people suppose there is nothing computers cannot do (see p. 45); some people, indeed, think there is nothing computers do not already do.

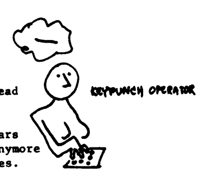
A couple of years ago, a leading picture magazine carried a piece about Stanford's Artificial Intelligence Laboratory, claiming that one "Shakey the Robot" had been developed to near-human intelligence and capabilities. This was pure bosh, since repudiated in the computer magazines, but a lot of people Out There in Readerland believed it. (See "The God-Builders," flip side.)

Once I had a long discussion with a somewhat wild-eyed young woman who believed that the government was monitoring her brain with computers. I think I persuaded her that even if this were feasible it would cost the government tens of thousands of dollars to do it, and that probably no existing government agency was that interested in her thoughts. I'm not sure she was persuaded.

SOME COMPUTER PEOPLE to distinguish among



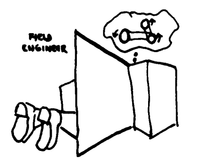
Computer operators turn 'em on and off, change programs, change disks and tapes, select modes of operations for programs that can do more than one thing. (See p. 38.)



Input typists (also called keypunch operators) are clerks who copy information into the computer (on terminals) or onto something the computer can read (punch cards, magnetic disk, etc.)

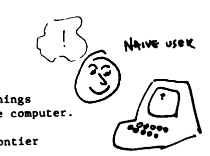
NOTE: these jobs may end in a few years when nothing else has to be copied anymore because users put things in themselves.

Computer programmers create exact plans for what the computer is to do, then change them till they work.



Computer repairmen, or "field engineers," fix computers and their accessories when something goes wrong electrically or in the gears.

They always wear tie clips, at least if they wear ties, so as not to get pulled into rotating machinery.



A NAIVE USER (no offense) is an ordinary person who doesn't need to know any of these things in order to do something useful with the computer.

Creating programs to help him is the frontier of computing.

WHAT YOU'VE SEEN PROBABLY WASN'T "A COMPUTER."

Get out of your head the notion that some one system you've seen showed you what Computers Are Really Like. Computer systems can be as different externally as bats and whales. (Yet it's the same kind of heartbeat, but that's no help in dealing with them.)

Then what is it computer people know, you may ask, that leads them to understand new systems quickly? Aha. Computer people simply adjust faster to whole new worlds.

THE AUTOMOBILE ANALOGY (more)

"The Interstate was bumper-to-bumper, but after we had lunch at the rest stop it cleared up till we got to the tollbooth. Then Harry got lost on the interchange, and we had to double back on the service road."

How incomprehensible to someone from 1905. Yet how simple-minded when you understand it. That's how it is with computers.

Computer talk sounds so strange and incomprehensible to you folks out there-- yet to us in here it's often as simple as the lines above-- if you know the fundamental concepts.

And nothing in the normal everyday world will have prepared you for them.

It's not jargon, but the simplest way to express thoughts in these areas.

USING A COMPUTER SHOULD ALWAYS BE EASIER THAN NOT USING A COMPUTER.

If it isn't, you
(or your company, or your state)
may have been sold a bill of goods.
OR they may have decided
your inconvenience is less important than something else.
In any case, you have a right to ask sharp questions.

WHAT IS THIS SYSTEM ABOUT?

Handy questions to size up what a computer is supposed to be doing.

What data does it contain?

Where is the data stored?

What other data will it link up to?

What information do you suppose can reasonably be derived from that?

What are the key input and output devices?

In what forms does information go in and out?

What do you suppose they might want to know?

THE DAMNED LIE

"Computers are rigid and inhuman."

A BETTER APPROXIMATION

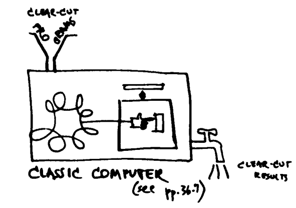
People are sometimes (all too often) rigid and inhuman. (Machines and animals are nonhuman-- the term "inhuman" applies only to people.)

"Rigid and inhuman" computer systems are the creation of rigid and inhuman people.

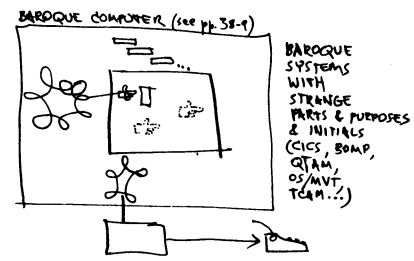
THE NEW ERA

A new era in computers is dawning.

The first, or Classic, computer era used straightforward equipment and worked on straightforward problems.



The second, or Baroque, computer era used intricate equipment for hard-to-understand purposes, tied together with the greatest difficulty by computer professionals who couldn't or wouldn't explain very well what they were doing.



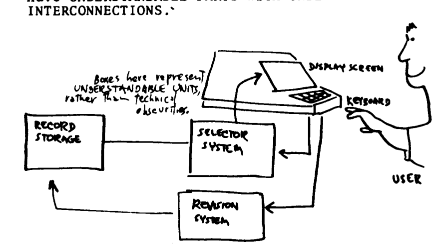
But a change is coming. No one company or faction is bringing it about, although some may feel it is not in their interest. I would like to call it here the DIAPHANOUS age of the computer.

By "diaphanous" I refer both to the transparent, understandable character of the systems to come, and to the likelihood that computers will be showing us everything (dia-, across everything, phainein, to show). (for later point see flip side.)

In the first place, COMPUTERS WILL DISAPPEAR CONCEPTUALLY, will become "transparent", in the sense of being parts of understandable wholes. Moreover, the "parts" of a computer system will have CLEAR CONCEPTUAL MEANING. In other words, COMPUTER SYSTEMS WILL BE UNDERSTANDABLE. Instead of things being complicated, they will become simple.

Now, many people think computers are by their nature incomprehensible and complicated-- unfortunately, that's because they have been MADE TO BE. Usually this is unintentional, but I fear not always. EXAMPLE. Instead of being told, "this is the mysterious XYZ computer, it has to have things just so, you have to fill out these RMQ forms to go into the V34...", you will hear such surprisingly simple things as "This system is set up for keeping track of who owes what to the company. On the screen you can get lists of accounts and outstanding bills and who owes them; if you point at one with the light pen, the printing machine over here will print a bill all set to go in the envelope."

In other words, systems will increasingly have UNDERSTANDABLE PARTS WITH UNDERSTANDABLE INTERCONNECTIONS.



What is responsible for this remarkable change?

For one thing, smaller and smaller companies are buying computer services, and they won't stand for ridiculous complications. For another thing, a number of people in the computer field have gotten sick of systems that make things hard for people. Finally, the price of computers, especially microprocessors (see p. 44) are coming down so fast that they can be tailored to fit people, rather than vice versa. But most of all, it's just high time, that's all.

BIBLIOGRAPHY

C.L. Freitas, "Making the Best Buy for the Small Business." Computer Decisions, March 73, 22-26.

Compares the relative costs of minicomputers and time-sharing; concludes that minis are the best buy.

Burton L. Katz, "Making Minicomputers Work in a Medium-sized Business." Data Processing, Winter 1971, 9-11.

Stresses the point that well-designed computer systems can be used by existing personnel of a firm, without excessive complication.

Frederic G. Withington, "Cosmetic Programming." Datamation, Mar 70, 91-95. How to make systems friendly on the outside.

INTERACTIVE SYSTEMS

Used to be that ordinary people had to deal with computers by filling out intricate forms, which were then translated into punch cards. The forms put things in weird categories (see "Coded-Down Data," p. 29.)

No longer.

Anyway, no longer necessary.

Computer systems can now give you action, excitement-- and explanations.

This is done through the magic of the TERMINAL. Terminals come in two conspicuous flavors (typewriter and screen or "boob tube") and also have two less-noticeable divisions ("Teletype" or "industry" versus "IBM type.")

Anyway, a terminal is something that allows a person and a computer to type at each other.

Now, computers are merely gadgets for twiddling information. They no more understand English, or human psychology, than puppies can read music. (See "Artificial Intelligence," p. 12-15) But the computer's program can, for instance, direct the computer to type out a simple question, and compare the user's answer with a simple set of alternatives. For example, suppose the user is visiting a hospital. A computer can sign him in without the abrasiveness of a receiving nurse, and with far more patience. The following might be a sample dialogue. (Here the computer types what's in caps, and the user's replies are in lower-case.)

```
DO YOU HAVE AN ACUTE PAIN?  (Y, N, DK)
     dk
YOUR ANSWER IS: DK FOR "DON'T KNOW."
     DOES THAT MEAN YOU'RE NOT SURE
     WHAT 'ACUTE' MEANS?  (ANSWER A)
     A PAIN COMES AND GOES? (ANSWER B)
     YOU HAVE A PAIN SORT OF ON THE
     BORDER?  (ANSWER C)
     c
IS THIS PAIN IN AN EXACT PLACE YOU
     CAN IDENTIFY? (Y,N,DK)
     y
```

An interactive system of this kind is called a conversational system, in that it "converses" with the user. The secret is that the alternatives in the computer program are few and carefully worked out beforehand: there are great pitfalls when there are too many alternatives, as when such conversational systems are used for teaching (see pp. DM 15-19).

Here is a straightforward example: a system I wish I had for balancing a checkbook. Note that the inner program for this conversational system could be written in any of the three languages presented later.

```
WHAT PROGRAM WOULD YOU LIKE TO RUN? ckbk
CHECKBOOK PROGRAM STARTS.
DO YOU WANT TO PROOF THE NEW STATEMENT FOR
     MARCH?  y
PLEASE LIST THE CHECKS THAT HAVE COME IN.
     231, 239, 240, 242, 244, 245.
SUM OF INCOMING CHECKS IS $345.72.  DO YOU
     WANT BREAKDOWN?  n
PRESUMABLY BANK IS CHARGING YOU .60 FOR
     SIX CHECKS.  ALSO MONTHLY CHARGE OF
     FIFTY CENTS (PLEASE CONFIRM).  y
ARE THERE ANY OTHER BANK CHARGES THIS
     MONTH?  n
ARE THE FOLLOWING DEPOSITS ON THIS STATE-
     MENT-- MARCH 1 SALARY, $854.00?  y
     GIFT FROM AUNT AGATHA, 14 MARCH,
     $25.00?  n
TOTAL ON STATEMENT SHOULD BE $1753.21.
     PLEASE CONFIRM.  y
YOUR CURRENT FLOAT IS $656.75.  DO YOU WANT
     BREAKDOWN?  y
CURRENT FLOAT AS FOLLOWS--
     NO. 241     IRVING'S RECORDS 7 MARCH $   6.75
     NO. 243     SINISTER & MALADROIT (LEGAL
                 FEES) 12 MARCH           $600.00
     NO. 246     DOGGIE HAIRDRESSERS
                 12 MARCH                 $  20.00
     NO. 247     SAM GRONK (REPAYMENT)
                 14 MARCH                 $  30.00
     TOTAL                                $656.75
ARE YOU DONE WITH CHECKBOOK PROGRAM?  y
```

(The part shown above is easy. Thinking out the ways for the user to correct his records, and/or the bank, is the tough part.)

COMPANIES THAT WILL SET UP WHOLE LITTLE BUSINESS SYSTEMS

A number of companies make minicomputers (partial list on p. 43); however, companies who want business systems built around minicomputers may want to investigate companies that will put together whole business systems for them around minis.

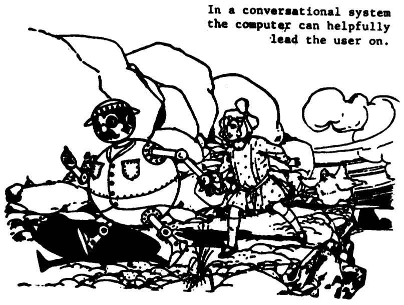
(It is hoped that one contribution of this book will be to give the reader a better idea of what to ask for.)

Two companies that seem to be in this business are:

Genesis One Computer Corporation, 99 Park Ave., NY 10016. Appears to use BASIC language (see pp. 16-17).

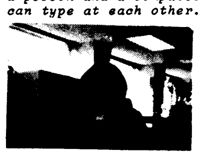
Qantel Corp. (offices in five major cities). Sells a minicomputer of their own manufacture, using a language called QIC (Qantel Interactive Code), which a salesman tells me is "just like BASIC" (see pp. 16-17). Minimum setup includes a display terminal, printer, computer and 6-million-character disk, at $31,000.

In a conversational system the computer can helpfully lead the user on.

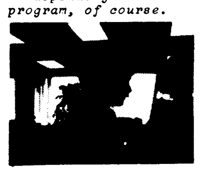


TERMINALS

A terminal is simply any device by which a person and a computer can type at each other.



Kids love terminals. This one is a video terminal or keyscope (see p. DM 108). It allows the computer to present textual or numeric information, play games with you, quiz you for information in a good-guy system, or whatever -- depending on the program, of course.



More expensive scopes (or computer displays) allow pictorial animation under the user's control (discussed throughout flip side). THE MAIN THING TO UNDERSTAND: what they do is decided by human beings, not "scientific principles." Human beings take note.

Types of available computer terminals are discussed in the next spread; more display terminals discussed p. DM 108

Below: a "bull pen" of terminals, all hooked up to the main computer at the Chicago Circle Campus, University of Illinois. What each person does at his terminal is normally independent of what any other person does, through time-sharing of the main computer. Installations more suited to time-sharing can have large numbers of terminals, all over a campus, a company or the world; see Time-Sharing, p. 45.

Motto 1 for the new era:

USING A COMPUTER SHOULD ALWAYS BE EASIER THAN NOT USING A COMPUTER.

Motto 2 for the new era:

THE NEW FRONTIER IN COMPUTERS IS CONCEPTUAL SIMPLICITY AND CLARITY.

People who delight in intricacy are going to have to learn some new tricks. Internal intricacy is fine, as long as the user doesn't have to deal with it.

Motto 3 for the new era (to computer people):

MAKING THINGS EASY IS HARD.

Motto 4 for the new era:

ANY SYSTEM FOR A SPECIFIC PURPOSE SHOULD BE TEACHABLE IN TEN MINUTES OR LESS.

Anyone who has been taught the use of some fixed-purpose computer system, such as an airline reservation system, may doubt this. But perhaps this book will clarify things somewhat.

A "GOOD-GUY SYSTEM" is a conversational computer system that is CLEAR, EASY TO USE, AND FRIENDLY.

ANY MAN OF COMMON SENSE CAN DESIGN A COMPUTER SYSTEM FOR A PURPOSE IMPORTANT TO HIM: the data structure, forms of information, general operations, record-keeping, and responses to on-line users.

But for some reason this is generally kept a secret.

"JOE TURKEY USER"

A good friend of mine, Jordan Young, is a former R.E.S.I.S.T.O.R. (see p. 47) and now a systems programmer (see p. 45) on the mighty Dartmouth time-sharing system, DTSS. (See p. 45.)

Jordan tells me that one of the more important people at Dartmouth is a mythical individual named Joe Turkey User. This estimable personage knows hardly anything about computers, makes a lot of mistakes, thinks he understands what you tell him when he doesn't, tends to hit the wrong keys on the terminal, and in general tends to screw up.

But the motto up there is: "If it's not simple enough for Joe Turkey User-- it's too complicated."

DTSS is a good-guy system.

YOUR FIRST COMPUTER CONTACT

When you first sit at a computer terminal, the feeling is one of sheer terror. Sweat and chills, jumpiness and sudden clumsy nervous motions, lunatic absentmindedness and stammering fear and awkwardness interfere with your ability to function or understand the person who is helping you.

It's perfectly normal.

A CONSIDERATE LAYOUT

CARSON PIRIE SCOTT & CO.

YOUR BILLING SUMMARY

NEW BALANCE

MINIMUM PAYMENT DUE

REGULAR-OPTION

Thank you, Carson's.

THE MOST IMPORTANT COMPUTER TERMS FOR THE '70s

Here are some phrases that will count in the new era of computing, when we will run into more and more computer systems set up for particular purposes.

on-line
: connected to a functioning computer. (Note that the computer may be in the typewriter or desk itself.) (As distinct from off-line, setting things up for processing later.)

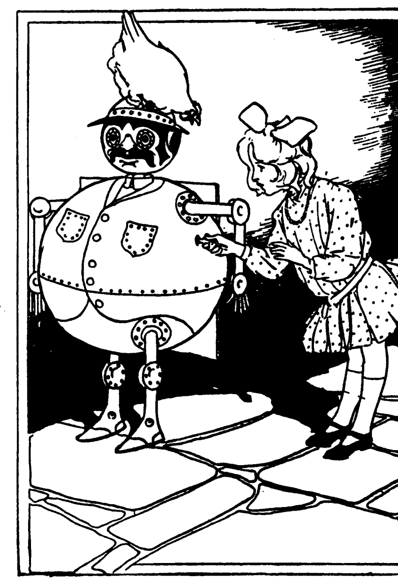
interactive
: not just connected, but responding to you. Interactive systems and programs can respond to your choices and requests, clarify what they want from you, etc.

remote
: referring to something far away, as distinct from local, right where you are. A computer can be either remote or local, e.g., on your desk.

front end (n.), front-end (adj.)
: whatever stands between you and a system. A front end can be the terminal in your office, for example. A front-end program is one which mediates between a user and some other system or program, perhaps collecting data for it by quizzing you.

dedicated
: set up for only one use. A big computer at a computing center has to have many uses; a little computer in your office can be dedicated. Dedicated computers are now hidden in all sorts of things: cash registers, for example (see "Microprocessors," p. 44).

turnkey (adj.)
: turned on with a key. Especially, turnkey systems, small computer systems that can just be turned on (key or not) and are fully set up, ready to run, programmed, etc.



real-time
: responding to events in the world as needed, without delays. Computer systems that control machinery, make airline reservations, predict the weather or respond to naive users are real-time. Systems that can catch up overnight are non-real-time.

"intelligent terminal"
: stupid term referring to any object that does more than act like a plain terminal. The term is stupid because it confuses distinctions. Some "intelligent terminals" have extra circuits for various purposes; others contain their own minicomputers; still others are ordinary terminals connected to front-end programs.

user-oriented
: set up for "users"-- people who are not programmers or input typists, but who actually need something done.

user level (n.), user-level (adj.)
: "where the user is" mentally; his level of involvement. User-level system, system set up for people who are not thinking about computers but about the subject or activity the computer is supposed to help with.

naive user (n.), naive-user (adj.)
: person who doesn't know about computers but is going to use the system. Naive-user systems are those set up to make things easy and clear for such people. (We are all naive users at some time or other; it's nothing to be ashamed of. Though some computer people seem to think it is.)

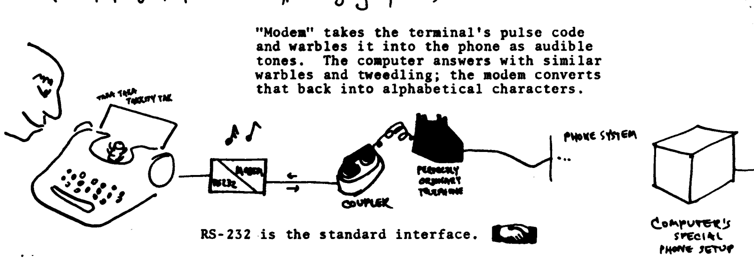
idiot-proof
: not susceptible to being loused up by a naive user. The hostility in this term may in some cases be real. Computer people sometimes forget, or do not wish to tolerate, the degree of confusion that naive users bring to the keyboard. This attitude is not just their problem but everybody's, since they lay it on us.

good-guy system
: term to be used here for naive-user systems that are friendly, helpful, simple and clear.

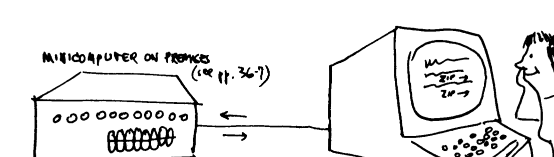
stand-alone system
: system (regardless of purpose) which doesn't have to be attached to anything else. (May contain its own computer.)

THE MIRACLE OF OVER-THE-PHONE TERMINALS
(Some people go ape just to see the typewriter going by itself)

"Modem" takes the terminal's pulse code and warbles it into the phone as audible tones. The computer answers with similar warbles and tweedling; the modem converts that back into alphabetical characters.



RS-232 is the standard interface.



YOU CAN HANG A TERMINAL EITHER ON A MINICOMPUTER (see p. 36) OR A BIG COMPUTER (see p. 38).

(What it does, of course, depends on the program, not the size or brand of computer.)

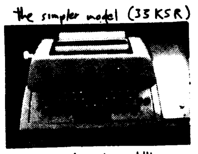
TWO KINDS OF TERMINALS

You would think the fundamental dichotomy among computer terminals was between those that print on paper and those that show you stuff on a screen. But it isn't. (That's like the difference between people and whales-- much greater outside than inside.)

Actually the fundamental distinction between terminals is between ASCII (pronounced "Askey") and IBM terminals. ASCII is a code and scheme of organization which was adopted by "the industry," under the blessing of the National Bureau of Standards. But IBM has pointedly ignored this standard.

The principal terminal of the ASCII type, in sheer numbers, is the model 33-ASR Teletype (trademark of Teletype Corp.), so this kind of terminal is called the "33 ASR type," or "Teletype-type," or we even say a given terminal "looks to the computer like a Teletype."

the simpler model (33 KSR)



Typing is done by a little spinning cylinder.

the fuller model (33ASR)

Paper Tape punch

Reader for punched Tape

IBM, however, seems to like changing its systems around a lot, for instance changing its codes when it brings out a new computer. (Fortunately, it just happens that they also sell adapters between them. Whew.) So IBM-type terminals are different by design.

There is one main type, however, exemplified by the IBM model 2741 terminal. Thus we say a terminal is an "IBM-type" or "2741-type" terminal.

Both Teletype- and IBM-type terminals come in either video-screen or printing models, from a variety of manufacturers.

Indeed, even the Selectric (IBM trademark), typing mechanism appears in some Teletype-type terminals.

There is a very important performance difference between ASCII and IBM terminals. The ASCII terminal can send each character typed by the user-- each "keystroke"-- to the computer immediately. This means that highly responsive programs can be written, which examine the user's input and can reply instantaneously, if need be, after anything the user types.

IBM-type terminals, however, require a "line feed" character or an "end of transmission" character to be typed by the user to make it the computer's turn. This locks the keyboard so the person can't use it. Then the computer must type something, ending with its own "unlock" signal that makes it the person's turn again.

Why this unwieldy design? Supposedly it results from the curious decision, in the design of IBM's 360 computer, to make all devices resemble the card reader as far as the computer is concerned. Just as the card reader reads punched cards till the last one is done, the IBM terminal is designed to send and receive characters until a "finished" condition is reached.

It makes sense to own your own:

SOME TERMINALS YOU MIGHT LIKE.

All are ASCII-type unless otherwise noted. Note: there are hundreds of types and brands of terminals available. These are just some thoughts.

PRINTING TERMINALS.

BEST BUY? The model 38 ASR Teletype gives you upper and lower case, and is otherwise similar to the standard model 33. $70 a month from RCA Service Company, Data Communications Div. (offices in major cities); $15/mo. for the coupler. 30-day cancellable but costs $50 to put in, $24 to take out.

There is a cute terminal that behaves just like the 33 ASR, but is faster and uses NCR pressure paper or a ribbon, interchangeably. The Extel Series A teleprinter from Extel Corp., 310 Anthony Trail, Northbrook, Ill. 60062.

If you like Selectrics, but want to go to ASCII, there is one weird possibility.

A firm called Tycom Systems Corporation (26 Just Road, Fairfield NY 07006) offers an interesting alternative. It happens that all Selectrics (anyway, Model I and Model II) have a seam around the midriff at which the typewriter can be unscrewed into two sections. Clever Tycom! They make a device which fits between, looks to the bottom like the top of the Selectric, and looks to the top like the bottom. Also, it turns the Selectric into a terminal, receiving ASCII codes from whatever computer you attach it to and causing the computer to type them, or sending out what you type to the computer in ASCII.

Curiously, IBM has given its blessing to this arrangement, meaning you can have this sandwich deal done to a Selectric you rent from IBM, and serviced under beefed-up IBM maintenance agreements ($72 per year, or $16.50 per hour, as of 1970).

DISPLAY TERMINALS (see pp. DM 20-1)
There are many brands. Some use video.

The earlier video terminals came with dreadful styling, like a 1940s science-fiction movie. But as an example of how the market is developing, one of the handsomest video terminals is the $1300 Mini-Tec from TEC Incorporated, 9800 North Oracle Road, Tucson, Ariz. 85704. It comes covered with wood-grain contact paper and looks very nice. (You should have seen their early models.)

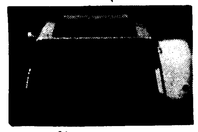
The Hazeltine 1000 video terminal rents for $49/mo. on a 1-year contract. LOWER-CASE OPTION; modem and coupler apparently not included. (Hazeltine, Greenlawn, NY 11740, with offices all over.)

Telephone Coupler with telephone handset in place

If you have no objection to ITT, they offer a portable video terminal with built-in modem and coupler, the Asciscope, for $65/month. Supposedly there's a long waiting list. (ITT Data Equipment and Systems Division, East Union Ave., East Rutherford, NJ 07073.)

For a display terminal in your car, see Kustom Electronics, Inc. (aren't they the rock-amp people?), Data Communications Division, 1010 West Chestnut, Chanute, Kansas 66720. They've already set up travelling terminals for the mobile constabulary of Kansas City (Mo.), Palm Beach and Nashville. (Communications, Jan. 73, ad p. 47.) Now, of course, you'll need a whole stationary radio setup to run that...

YOUR



Classic Teletype®

MISCELLANEOUS

Various firms rent terminals, some on a short-term basis. (Some terminal companies are bad news, keeping up their equipment badly and offering poor service, so watch it.)

(The day will come, let's hope it's soon, that you can rent a terminal overnight or for a weekend like a movie camera. But till people get a sense of how far and fast things are moving, we'll continue to schlock along haphazardly.)

Unfortunately rental people are hard to find, since they are usually local, and the Yellow Pages idiotically lump together every possible form of computer sales and service under "Data Processing Equipment and Supplies," and few firms further specify their business in the listing.

Here are some names (neither endorsed nor criticized):

Computer Planning & Supply, Chicago
TTS Systems, LA
Vardon & Associates, Dallas

A good outfit, that rents both ASCII and IBM-type terminals of their own manufacture, is Anderson Jacobson Co. (1065 Morse Ave., Sunnyvale, Calif. 94086, and major cities). They have a Selectric terminal, for instance, which rents for about $100 a month (about the same as the standard IBM 2741) but is portable.

To provide a memory with your ASCII or IBM-IBM-type terminal, an odd machine called the Techtran 4100 (about $1000 from Techtran Industries, 580 Jefferson Rd., Rochester, NY 14623) can be used for offline storage. It uses a magnetic cassette. Here are some things you can do with it:

type stuff into the Techtran,
  later squirt it to a computer at high speed
receive stuff from a computer at high speed,
  later type it back automatically on the terminal
type into the Techtran, correct it, and then have it typed back automatically-- no computer.

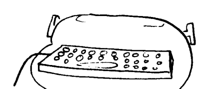
The question of whether the Techtran can be used with the Digi-Log has not been publicly resolved.

It happens that Anderson Jacobson (above) will rent you their 2741-type Selectric terminal, with a Techtran, for about $220 a month total. But they won't rent the Techtran separately.

A 2741-type Selectric terminal with memory, offering these same capabilities, is now available from IBM! It is the Communicating Mag Card Executive (CMC). Since the Mag Card Executive, to which they have added the communication feature, costs over $200 a month, figure the communication feature could cost another $100 or so monthly, or probably half again as much as the Anderson-Jacobson.

Honeywell (Honeywell Information Systems, Wellesley Hills, Mass.) has recently made available a Braille program to be used with "standard terminals" in their systems. (This may be the adaptation developed at MIT to do Braille on the 33 ASR.)

For those of us literary types who want upper and lower case but are stuck with 33ASRs, a LOWER-CASE CONVERSION KIT is available from Data Terminals and Communications, Campbell, California.



Something called the Compulaumate just sits on a Selectric and makes it a terminal. (Keys punched by solenoids.)

FURTHER POOP

If you're serious about keeping up with developments in the terminal area, you might want to subscribe to Terminals Review ($28/yr.), highly spoken of by Datamation. (GML Corp., 594 Marrott Rd., Lexington, MA 02173.)

A "CRT Survey" listing characteristics of 110 CRT displays (including both video terminals and fancier pictorial displays-- see flip side of this book) is available for ten bucks postpaid from Datapro Research Corp., One Corporate Center, Route 38, Moorestown, NJ 08057.

Standard display terminal offered with computers from DEC (see p. 57). It's the model VT05, $3000.

VIDEO TERMINALS WITHOUT THE VIDEO

A very hot item right now is a terminal called the "Digi-Log"-- actually several different models-- available from Digi-Log Systems, Inc., 666 Davisville Rd., Willow Grove, Pa. 19090.

This device fits in a briefcase. Basically it is a keyboard with a socket for the phone, and an antenna wire. You phone the computer, drop the phone handset in the slot, and clip the wire to the antenna of a TV set. Presto! On the TV set appears what you and the computer type at each other.

This is especially good for travelling salesmen (to communicate with their offices and ordering system via time-sharing computer) and executives who do computer work from the road. Also for people who want to show off remote computer systems.

Disadvantage: only 42 characters per line, which is awkward for some things, such as programming in Fortran.

Price: $1200 to $1400. They also lease, at rates as low as $40/month (3 years).

No lower-case as yet.

Also available on rental, supposedly, from Westwood Associates, Inc., 50 Washington Terrace, East Orange, NJ 07017.

Ann Arbor Terminals, Inc. (Ann Arbor, Mich.?) is said to offer a similar unit that is very nice.

The equivalent IBM-type terminal-- keyboard, coupler and clip to the TV-- is the IPSA-100, offered by I.P. Sharp Associates, Inc. (Bridge Administration Building, Bridge Plaza, Ogdensburg, NY 13669). Unfortunately it's much larger than the Digi-Log-- it comes in a medium-size suitcase -- and more expensive ($1700 up). However, they offer the APL character-set (see APL under "Magic Languages," p.22) as an option-- even a model with both normal and APL character-sets as a switch-selectable option (costs even more).

Recently, of all things, plans for a do-it-yourself unit of this type were announced in a popular electronics magazine (Don Lancaster, "TV Typewriter," Radio-Electronics, Sept. 1973, 43-52). This does not include the full plans, which are available for $2 from TV TYPEWRITER, Radio-Electronics, 45 E. 17th St., New York, NY 10003.

Supposedly this can be built for "around $120"-- probably a deal more-- if you are a skilled electronics builder or technician. But that looks to include a great deal of labor.

The finished unit holds up to 32 characters per line and up to 16 lines on the screen; a second memory can be added, to hold a second alternative screenful.

Upper case only.

TYPE RIGHTER:
The Magic Typewriters

A number of different systems are coming on the market to aid you in error-free typing.

IBM would have you call these "word processing systems," since that makes them sound of-a-piece with their dictation equipment. Actually they're text regurgitation systems, but let's just call them Magic Typewriters.

Prices of these things tend to run between $100 and $250 a month.

Generally these are being sold as secretarial aids, partly because they tend to be too ungainly for use by writers themselves. A principal use has been in large law offices, where contracts, wills and such are stored as "boilerplate" (standard sections of Document) and then modified slightly by the lawyer to justify the legal fees.

Such systems all basically consist of three things:

A typewriter, connected to some sort of magnetic memory, such as a tape, coated card or disk, and
editing circuitry, which responds to various acts by the user.

WHAT THEY DO: allow you to type stuff in, which is both typed on the paper and at the same time stored on the magnetic whatever. Small errors you correct as you type along, generally by backspacing.

When you want a clean copy-- Presto Wait-o! Put in clean paper, start the magnetic whatever at the beginning, and the typewriter retypes it without a mistake.

If you're lucky.

Unfortunately some of these systems are quite badly thought out. In one or two cases I am not sure whether they are designed as they are accidentally or on purpose. Neither interpretation is flattering to the manufacturer.

I have had extensive experience with two of these systems, the IBM Mag Tape Selectric and the IBM Mag Card Executive. Suffice it to say that if I believed that these systems were as cumbersome as they are by accident, then the sections in this book on IBM and its products might have a very different slant. As it is, these systems require a training period of (say) a week, and require such continuous attention to their curious mechanics that the user is given little opportunity to think of anything else. In both cases, in my opinion, the superficial plausibility of the initial design premises knots into tangled ramifications which verge on the preposterous. Much of this book was written on a Mag Card Executive-- and I'm damned sorry I bothered.

Some systems of this type are:

The IBM Mag Tape Selectric (MT/ST or MTST). Records on sprocketed 16mm mag film of the type used for movie sound recording, and you have two different tapes to get confused between.

The IBM Mag Card Executive. Records on a plastic Hollerith card (see p. 28) coated with magnetic oxide. Variable width of characters presents fascinating difficulties.

The IBM Mag Tape Selectric Composer (MT/SC, MTSC). Produces lovely results with the Selectric Composer, a very fancy Selectric. But has complications well beyond those of the Mag Tape Selectric. Even more variable widths than Mag Card Executive. Uses same mag-film cartridges as MTST.

(Note: for those who like the output from the above devices, but appreciate also the relative difficulty of their use, there is available a computer peripheral device which reads and writes these 16mm mag tape cartridges. I don't know who makes it, unfortunately.)

IBM's latest is called the Magnetic Memory Typewriter, and seems to store up to one page in a hidden memory. Apparently you can't set it aside, like the cards or tapes.

A firm called Redactron makes magic typewriters using either cassettes (audio-type) or mag cards (like the Mag Card Executive).

A firm called Savin does the same thing, using a Tycom Selectric Sandwich (see under "Printing Terminals," nearby).

Olivetti has one called the S-14 Word Processing System. Their cartridge (a disk?) stores, they say, 150 pages of typing.

Two other outfits in the field are Trendata and Quintype.

Woops! Here comes Sperry Remington! (Sperry Remington?) They have one too.

For those interested in this sort of thing, there is an International Word Processing Association (Maryland Road, AMS Building, Willow Grove, PA 19090.)

See also the Flip Side of the book for more high-performance text systems.

COMPUTER LANGUAGES

are what make computers go 'round.

If your computer only did one thing, then to start it you'd only need one button to press.

If your computer only did two dozen things, without variations, then you could let each operation be started by pressing one of the keys of the terminal, and that would be that.

But that's not what it's about.

We have lots of different things that we want computers to do, and we want one command to work on different varieties of data, or on the results of a previous command, or even to chew on another command itself; and so a computer language is a contrived method of giving commands to a computer that allows the commands to be entwined in a complex fashion.

This means having rules the computer can carry out and the person can remember.

This means having basic operations that can be built into bigger operations (routines, subroutines, subprograms, programs).

Thus a computer language is really a method by which a user can tie these programs together. Computer languages are built according to contrived sets of rules for tying programs together. Such rules are limited only by the imagination of their contrivers. Each computer language has its own contrived system of rules, and it may be completely different from the contrived rules tying together any other computer language. (That's one reason for here presenting three different computer languages, to show some of the mad variety that can exist.)

Computer languages tend to look like nothing else you've ever seen. Thus computer programs, which of course have to be written in these computer languages, look pretty weird. Some programs look like old train schedules (in multiple columns). Some look a little like printed poetry. In any case, A COMPUTER PROGRAM NO MORE LOOKS LIKE ITS RESULT THAN THAN THE WORD "COW" LOOKS LIKE A COW.

One of the central concepts of this book is that of a "program follower," a dynamic entity which somehow follows a program. Well, EVERY LANGUAGE HAS A PROGRAM FOLLOWER FOLLOWING ITS OWN PARTICULAR RULES. These rules are contrived for convenience, suitability to a purpose, and "aesthetics" of a sort-- often some form of stark compression. (The program followers wired into computers are some what more akin to one another; see "Rock Bottom," p. 32.) About all we can say languages have in common is: EVERY COMPUTER LANGUAGE ALLOWS LOOPS, TESTS AND BRANCHES, AND COMMUNICATION WITH EXTERNAL DEVICES, as mentioned on p. 11. Beyond that the differences are incredible.

So the basic secret of computer people is this: it's not that the necessarily know so much, but they can adapt to a whole new world of possibilities more quickly.

THREE [QUICKIE] COMPUTER LANGUAGES FOR YOU

Everyone should have some brush with computer programming, just to see what it is and isn't. What it is: casting mystical spells in arcane terminology, whose exact details have exact ramifications. What it isn't: talking or typing to the computer in some way that requires intelligence by the machine. What it is: an intricate technical art. What it isn't: science.

Why three languages? Because one would look too much alike. Only by perusing several do you get any sense of the variety they take.

These three languages make it possible in principle for you to learn computers with no coaching. All you need (in principle) is your own terminal, and time-sharing accounts with firms running BASIC (most of them do), TRAC Language (for availability see p. 21), and/or APL (for partial list of sources see p. 25).

Why these three? Several good reasons. One, they can be used from a terminal, which means that you could in principle get a terminal in your home and play with the computer from over the telephone. But this is expensive, and at worst fraught with accidental financial liabilities, so the possibility is minor right now. Nevertheless, it should be practical and inexpensive fairly soon.

These languages have been chosen because they are important, very different from each other, very powerful, influential and highly regarded in the field, interactive from time-sharing systems, and very suitable for making interactive programs and "good-guy systems."

Each may be used to create programs for science, business or recreation.

Because these languages can be used from a terminal, and thus learned quickly, we might call them Quickie languages.

Note: interactive languages mean you, the programmer, can change your program from the terminal; interactive programs are those which interact with users, which is different. However, these languages are quite suitable for both.

Another reason for these three: they represent, in a way, several major types.

BASIC is a widespread and fairly standard language-- that is, it is available on computers everywhere. Moreover, it looks rather like Fortran, which is the most important "scientific" computer language.

TRAC Language, though well-known among researchers, has mighty powers that are not so well known. Moreover, it achieves its powers through the simple and highly consistent following of a few simple principles, and is thus both very easy to learn and an elegant intellectual triumph for its inventor.

Moreover, it is a so-called "list language," meaning that it can handle information having extremely varied and changing form-- a very important feature to those of us interested in computer applications like picture-making and text handling, which use amorphous and busy types of data. (See "Data Structures," pp. 26+.)

APL is another elegant language, also worked out handsomely from certain basic ideas by a very thoughtful and inspired inventor.

In the contemplation of these three languages you may begin to see the influence of the individual human mind in the computer field, quite contrary to the stereotype. I would like to stress here that each of these three languages represents somebody's individual personal achievement, and is in turn a foundation upon which others, writing programs, can build their own.

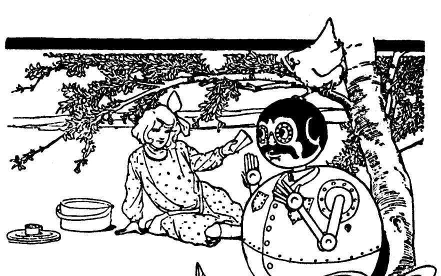
Two of these languages permit the creation of interactive programs that work on a line-by-line basis; in addition, TRAC Language (pp. 18-21) permits the creation of systems that react to any character the user types in, rather than waiting for the carriage return at the end of a line. This permits you to program user-level systems that are even more responsive.

IF YOU'RE SCARED. Don't worry, it's not a test. Flip the pages and look at the examples. (In particular, you might look for the same program which appears in each language: a program to cause the computer to print "HELP, I AM TRAPPED IN A LOOP" forever.)

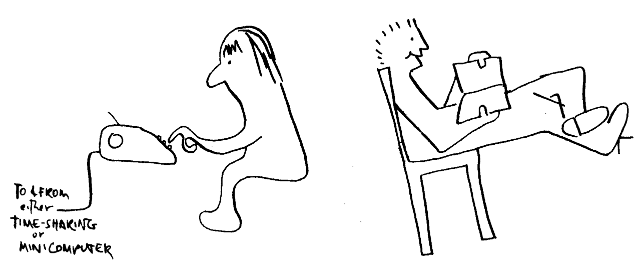
This book is organized so you can look at it or skip it in any order, so there is no particular reason you have to fight through the next three chapters if you want to press on. But if you want to study these languages, by all means do so.

Languages that can be used from a terminal are called on-line languages. There are a number of other popular on-line languages: JOSS (the original), FOCAL, LOGO, SPEAKEASY. I'm just sorry there's no room for them here.

Some popular non-interactive languages are briefly described on pp. 30-31.



Input to computers
is much easier
from interactive terminals.



The best way to start programming is to have a terminal running an interactive language, and a friend sitting nearby who already knows the language and has something else to do but can be interrupted with questions.

And you just try stuff.

Till more and more you get the feel of it.

And find yourself writing programs that work. **THE BEST WAY TO LEARN.**

PROGRAMS VS. SYSTEMS:
A Vague Guideline to a Vague Distinction

A "program" runs on an ordinary computer, without necessarily interacting with the outside world;

a "system" involves a whole setup, of which the computer and a program in it are just the central things.

A computer language is a system for tying together the fundamental operations of computers for larger tasks. Each computer language fits together according to its own principles, based in part on the personality and preoccupations of the person or people who designed it.

Modern computer languages generally can handle all the main kinds of programming: text handling, number crunching, storing files on disk memory and getting them back, and controlling whatever external devices you may have. Even making pictures in some way or other.

In this book we will try to give you a smattering of all these.

THREE Quickie Computer Languages:

BASIC (pp. 16-17)

TRAC® Language (pp. 18-21)

APL (pp. 22-5).

"COMPUTER TEXT EDITORS"

> The Moving Finger writes; and, having writ,
> Moves on: nor all your Piety nor Wit
> Shall lure it back to cancel half a Line,
> Nor all your Tears wash out a Word of it.
>
> *Khayyam/Fitzgerald*

Numerous interactive programs exist for editing text at computer terminals-- in other words, for doing what Magic Typewriters do, but using a computer instead of a small special-purpose machine.

Unfortunately most of these systems are dreadful. Dreadful, that is, for ordinary human beings. What computer people seem to think of as appropriate systems for handling text are totally unsuitable for people who care and think a lot about text, although they may be good for computer programmers.

Such systems allow you to insert text (with some difficulty), delete (with some difficulty), and rearrange (maybe).

Ordinarily the user must learn an explicit command language, some system of alphabetical commands that have to be typed in to effect any change in the material. Programmers think this is good for you and toughens the mind.

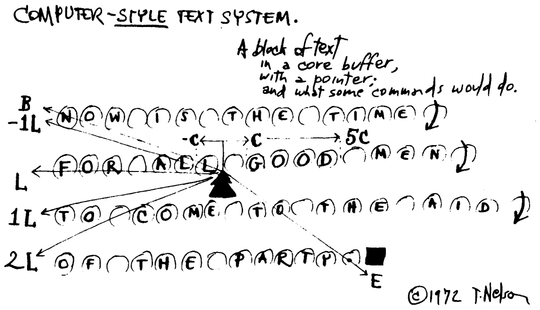
The text is usually stored as a series of alphabetical and punctuation codes in the computer's core memory. The area it occupies in the core memory is called a core buffer.

The program generally gives the user an imaginary "pointer," a marker specifying what point in the text the program is currently concerned with.

What is the pointer for? It specifies where the operations are to take place. "Insert," for example. If text is inserted, it will go into the place presently pointed at.

Many of the commands are concerned with controlling the current position of the pointer, moving it backward or forward by a specific number of characters (including punctuation marks and spaces) or lines (known to the program by the carriage-return codes interspersed in the text).

COMPUTER-STYLE TEXT SYSTEM.



In this simplified illustration, the pointer can be moved forward and backward in the text by various commands. Typing "B" moves the pointer to the beginning. "E" takes it to the end. "L" moves it to the beginning of the line it's presently on, and the commands "C" and "L," when given with numbers, tell the pointer to move forward or back the specified number of positions. For instance:

3C	Move forward 3 characters
-4C	Move backward 4 characters
2L	Move forward 2 lines
-2L	Move backward 2 lines

and so on. Note that these operations are not god-given, but that the particulars of how they behave and work together are determined by the personal quirks of who programmed them.

Another feature many of these programs have is called a "context editor" feature. So-called context editing moves the pointer from its present position to the next occurrence of a specific string of characters: for instance, the next occurrence of the word CHIAROSCURO. Often such commands permit you, by giving the command properly, to replace any given word or phrase with any other. It was drily remarked at a recent conference that this would allow a writer to change every occurrence of "or" in his writing to "and." Yet programmers seem to think this is a feature writers want.

(For programmers' purposes this is a very good facility; indeed, a whole computer language, SNOBOL, is built around it; -- see p. 31. But it has nothing to do with normal text.)

This type of thing is totally unsuited for the literary types of people who care most about text and its characteristics (connotations, twists) which can not be found by definable structured search. And who should not be forced to deal with explicit computer languages because it tends to interfere with the thought processes they are supposed to be pursuing, if not make them physically ill.

YOUR FIRST COMPUTER LANGUAGE: DARTMOUTH'S BASIC

The BASIC language, also called Dartmouth-Basic, was introduced in the sixties at Dartmouth College by John Kemeny and Thomas Kurtz. It was intended to be a simple and easy-to-learn introduction to computer programming, yet powerful enough to do useful things. It has grown in use, in recent years, both as the foremost beginner's language, and as a perfectly fine language for doing many simple kinds of work-- like custom business applications, statistics, and "good-guy" systems for naï. ɜ users as discussed elsewhere in this book.

Kemeny is now president of Dartmouth, and Kurtz runs their high-power time-sharing computer center, so BASIC has a permanent home base there.

Note that the name BASIC does not refer to the bottom-level or elemental languages of computers. BASIC has been contrived specifically to make programming quicker and easier. It is not "basic" to all computers; such bottom languages are called "machine language" or "assembler language" (see pp. 32-3).

The simplicity of the language begins at the program input, or editing, level. Each command of BASIC must be on a separate line, and each line must have a separate line number. Suppose you accidentally type in

```
50   IMPUGN Y
```

when you meant "INPUT" instead of "IMPUGN." You may replace that command at any time by typing the same line number and the new version of the line,

```
50   INPUT Y
```

which automatically replaces the previously line 50. If you want to get rid of the line entirely, you type

```
50
```

and an end-of-line code, and the whole line is gone.

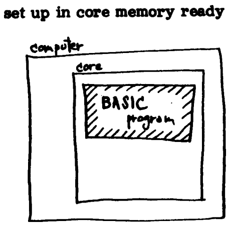
Example of a BASIC command:

```
153   LET X = Y
```

You can choose any line numbers you want, but the lines are automatically put in the order of their numbers. Since when you write a program you don't usually know at the outset what it will look like later, you try to leave enough gaps in the numbers at the start to fit in the instructions you might want to put between them later.

THE SETTING

To begin with, there must be a computer, and it must have a processor for the BASIC language, that is, a program for carrying out the operations of Dartmouth-BASIC. We will assume that this BASIC processor is all set up in core memory ready to go.



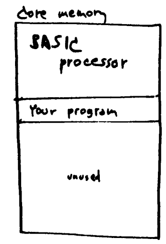
(Note: This is how it looks in a minicomputer. On a time-sharing system there's a lot of irrelevant other stuff going on, which we'll leave out.)

And we will assume, as previously mentioned, that you have some kind of a terminal-- that is, a device with a keyboard, some kind of place the computer can send messages to you and vice versa, and is more or less standard.

Now then: all that is needed is for you to understand the BASIC language, and you can program this computer within the confines of BASIC.

➡It is one of the strange aspects of this field that languages can be taught independently of discussions of the machine itself.

When you type in a program, the BASIC processor will do certain things to it (actually cook it down) and store it in core memory:



Every time you change one of the lines of the program the BASIC processor will insert, delete or replace lines as you have commanded, then rearrange whatever's left accordingly, in order of the line numbers.

Then when you tell the processor to start the program, by typing (with no line number)

```
RUN
```

the processor will start the program going at the command with the earliest line number, and your instructions will be executed according to the rules of BASIC.

Now we will consider some of the commands (or statements) of BASIC.

These two boys had never seen a computer before, but I loaded it up with the BASIC language processor, showed them a few basic commands and told them to turn it off when they were through.
I got back ten hours later and they were still at it. Too bad kids have such short attention spans.

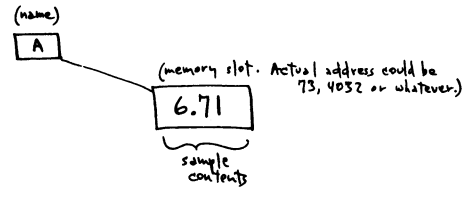
VARIABLES

The BASIC language, like a number of other languages, allows you to set aside places in core memory and give them names. These places may hold numbers. They can be used to count the number of times that things are done (or not done), to hold answers, numbers to test against, numbers to multiply by and so on.

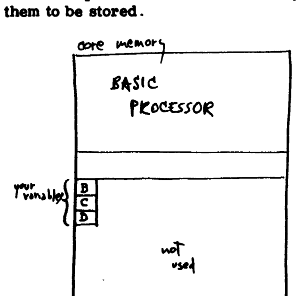
In BASIC, these places are given names of one alphabetical letter. That means you can have up to 26 of them. Examples:

A   E   I   O   U   sometimes Y   even X

Because these named spaces in memory may be used something like the way letters are used in algebra, we call them variables. In fact, each one is a place with a name.



If you use the names B,C and D for variables in your program, the BASIC processor will automatically set up places for them to be stored.



The END command

The END command in BASIC simply consists of the word END. It must come last in the program. Therefore it must have the highest line number. Example:

```
99   END
```

The PRINT command

Whenever the program follower gets to a PRINT command, it prints out on the terminal whatever is specified. Example:

```
97   PRINT "HAIL CAESAR.  BIRD THOU NEVER WERT"
```

When and if the program follower gets to this command, the terminal will print out

```
HAIL CAESAR.  BIRD THOU NEVER WERT
```

The GOTO command (pronounced "Go 2")

The GOTO command tells the program follower the number of the next command for it to do, from which it will go on. Example:

```
62   GOTO   99
```

which means that when a program follower gets to command #62, it must next jump to 99 and go on from there, unless that happens to be the END statement.

A SIMPLE SAMPLE PROGRAM

These are enough commands to write a sample program.

```
43   PRINT  "HELP, I AM CAUGHT IN A LOOP"
67   GOTO   43
68   END
```

The program will start at the first instruction, which happens in this case to be instruction number 43. That one prints a message. The next command, by line number, is 67. This tells the program follower to go back to 43, which it does.

```
43   PRINT  "HELP, I AM CAUGHT IN A LOOP"
67   GOTO   43
68   END
```

The result is that your terminal will print

```
HELP, I AM CAUGHT IN A LOOP
HELP, I AM CAUGHT IN A LOOP
HELP, I AM CAUGHT IN A LOOP
```

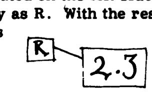
interminably, or until you do something drastic. It never gets to the END statement. (Two strategies for doing something drastic are usually to hold down the CONTROL button and type C, or hold down both CONTROL and SHIFT buttons, if you have them, and type P. One of these usually works.)

The LET command

The LET command puts something into a variable. Example:

```
43   LET R = 2.3
```

What is on the right side of the equals sign in the last statement, in this case 2.3, is stuffed into whatever location of core memory is designated on the left side, in this case a place known to you only as R. With the result that someplace in core memory is



The LET statement is an example of an assignment statement, which most computer languages have; an assignment statement assigns a specific piece of information (often a number, but often other things) to some name (often standing for a particular place in core memory).

The LET command in BASIC can also be used to do arithmetic. Example:

```
14   LET M = 2.3 + (12*7999.1)
```

(The asterisk has to be used for multiplication because traditionally terminals don't have a times-sign.) BASIC will work this out from right to left and store the result in M.

The INPUT command

The INPUT statement asks the person at the terminal for a number and then shoves it into a variable. Example:

```
41   INPUT Z
```

which causes the terminal to type a question mark, and wait. When the user has typed in a number followed by a carriage return, the BASIC processor stuffs the number into the variable and proceeds with the program. Here is a program using the INPUT statement.

Program waits here

```
10   PRINT "HOW OLD ARE YOU"
15   INPUT A
20   LET B = A/40.0
25   PRINT "YOUR AGE IS", B, "TIMES THE AGE
     OF THE EMPIRE STATE BUILDING."
30   END
```

This will cause the following to happen:

Program types:
    HOW OLD ARE YOU? 20
    your answer

Program types:
    YOUR AGE IS .5 TIMES THE AGE OF THE EMPIRE STATE BUILDING.

The IF command

The IF command is a way of testing what's stored in a variable. Example:

```
88   IF M = 40 then 63
```

This tests variable M to see if it contains the number 40. If M is indeed 40, the program follower jumps to line 63. If not, it goes right on and takes the next higher instruction after 88. The IF can test other relations than equality, including "less that," "greater than," "not equal," "less than or equal to," etc. For instance,

```
89   IF Q 7 then 102
```

will send the program follower to command 75 if variable Q contains a number less than 7. (Note that different BASICs for different computers may have slightly different rules here.)

The BASIC language, developed at Dartmouth, must not be confused with the underlying binary languages of individual computers (see "Rock Bottom," p.32). These underlying codes are called "machine languages" (or, in a dressed-up form, easier to use for programmers, "assembler language"). These are the basic languages, different for each machine. Dartmouth BASIC, or jut plain Basic, is a widely available, standardized, simple beginner's language.

ANOTHER PROFOUND EXEMPLARY PROGRAM

```
2    LET Z = 25
10   PRINT Z, " BOTTLES OF BEER IN THE WALL"
15   LET Z = Z - 1
62   IF Z = 0 GOTO 74
63   GOTO 10
74   PRINT "TIME TO GO HOME."
75   END
```

The program will start typing thusly:

    25 BOTTLES OF BEER IN THE WALL
    24 BOTTLES OF BEER IN THE WALL

and so on, until Z has reached 0; then it will type

    0 BOTTLES OF BEER IN THE WALL
    TIME TO GO HOME.

and then it will stop.

You will note that this program, like the one that printed "HELP, I AM CAUGHT IN A LOOP," has a loop, that is, a repeated sequence of operations. The first one was an endless loop, which repeated forever. This loop, however, is more well-behaved (by some people's standards), in that it allows an escape when a certain criterion has been reached-- in this case, printing a line of text 25 times with variants.

The reason we are able to escape from this loop is that we have a test instruction, IF statement number 62.

It is very important for the programmer to include tests which allow the program to get out of a loop. This may be couched as a motto, viz.:

LEAK BEFORE YOU LOOP.

AN AUTOMATIC LOOP

Indeed, for people who are big on program loops, BASIC provides a pair of instructions which handle the program loop completely. These are the FOR and NEXT instructions. We won't show them here, but they're not very hard. Using the FOR command, you can easily direct the computer to do something a million and one times, say. This can be exhilarating. You can even direct it to include that program in something to be done a billion times, resulting in a program loop that would be carried out over a trillion times. All in a short program! But of course this is just power on paper; we want our programs to be useful, and finish their jobs in the present century, and so such flights are just mental exercises.

FAST ANSWERBACK WITH BASIC (in some versions)

If you want a fast answer to a numerical question, you can do it without the line numbers. typing in

```
PRINT   3.1416 * 7124
```

will cause BASIC to print the answer right out and forget the whole thing.

TEXT STRINGS IN BASIC

The deluxe versions of the Dartmouth BASIC language have operations for handling text-- or what computerfolk call "strings," that is, strings of alphabetic characters and punctuation. These operations tend to begin with $ (standing for "$tring"?) and there's no room for them here.

But what they mean is that BASIC can type letters, count the nouns in Gone With The Wind, or print out the nine hundred million names of God.

If you write the program.

THIS IS A SERIOUS LANGUAGE,
AND CAN SAVE SOME COMPANIES A LOT OF MONEY

BASIC is a very serious language. Advanced versions of BASIC have instructions that allow users to put in alphabetical information, and store and retrieve all kinds of information from disks or tape. In other words, BASIC can be used for the fairly simple programming of a vast range of problems and "good-guy systems" mentioned elsewhere. Complete BASIC systems allowing complex calculations can be had for perhaps $3000; a general-purpose computer running BASIC with cassette or other mass storage, for business or other purposes, can now be had for some $6000. Allowing a few thousand dollars for programming specific applications in BASIC, simple systems can be created for a variety of purposes that some companies might say you needed a hundred-thousand-dollar system for.

This is serious business. Languages like BASIC must be considered by people who want simple systems to do understandable things in direct ways that are meaningful to them, and that don't disrupt their companies or their lives.

This has been a very hasty and brief presentation in which I have tried to convey the feeling of this important language. If you have the chance to learn it, by all means do.

SOME FUN THINGS TO TRY IN BASIC

Write a program that prints calendars.

Write a program that converts an input number to Roman Numerals.

Write a dialogue system that welcomes the user to the sanitarium, asks him questions, ignores the answers and insults him. (Use the INPUT statement for receiving numerical answers. Since the answers are ignored they can all be stored in one variable.)

WHERE TO GET IT

(Features of the BASIC language vary considerably from system to system. Which ones offer the highly desirable alphabetic commands and mass storage have to be checked out individually.)

BASIC is offered on many if not most time-sharing services, so you can use it from your home on a terminal. (But note that this can be expensive and even dangerous, if you're paying yourself; there are not presently adequate cost safeguards to prevent you from running up huge bills.)

BEST BUY? Rumors persist of a time-sharing service somewhere that offers BASIC for $5 an hour, total, with disk storage thrown in. I have not been able to verify this.

DEC offers minicomputer-based systems which time-share BASIC among several terminals simultaneously. (But you have to buy the whole big system.) The ones that run on the PDP-8 are marketed mainly to schools, and for this reason are called, somewhat peculiarly, EDUSYSTEMS Their multiterminal system for the PDP-11 is called RSTS (pronounced "Risstiss,") and is marketed mainly to businesses.

Hewlett-Packard offers BASIC, I believe, on all of its minicomputers. Of special interest is an odd computer called the Series 9800 Model 30. You're only allowed to program in BASIC. (It's actually a microprocessor; see p.44.)

Many other minicomputer manufacturers now offer BASIC. Data General's NOVA is one.

BIBLIOGRAPHY

Kemeny and Kurtz, BASIC Programming. Wiley, 1967.

DEC's Edusystem Handbook is a very nice introduction to BASIC, quite pleasant and whimsical; it may be a good introduction even if you're using other people's BASIC systems. It's $5 from DEC, Communications Services, Parker St., Maynard, Mass. 01754.

There is also a programmed text on BASIC by Albrecht (published by Wiley). For those of us who freeze at numerical-looking manuals, programmed texts can take away a lot of anxiety.

MY COMPUTER LIKES ME (when I speak in BASIC). This book has evidently been put together by the People's Computer Company, and has some idealistic fervor behind it. $1.19 from Dymax, Box 310, Menlo Park, Cal. 94025.

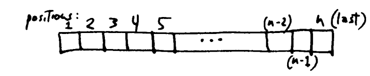
BASIC is a good example of an "algebraic" type of language, that is, one formulated more or less to look like high-school algebra and permit easy conversion of certain algebraic formulas into actual runnable programs.
The most widely-used language of this type is FORTRAN (see p.31). Thus BASIC is often referred to as a "Fortran-type language."
The kickeroo-- and if you understand this it's half the battle-- is that a line of BASIC or FORTRAN directs a certain event to take place, while a statement in algebra just describes relations.
The strange resemblance between the descriptive language (algebra) and the prescriptive language (Fortran or Basic) is that algebraic operations (which are just recombinations and restatements) can be mimicked by the computer language, and this early obsession of mathy computerfolk led to making the computer language look like a descriptive algebra. Especially with the weird use of the equals-sign to mean "is replaced now by." In hindsight this was a ridiculous idea; some of the more recent languages (such as APL) use a left-pointing arrow instead of an equals-sign, showing that an action is being called for, rather than a relationship being described.

ARRAYS,
an important data structure

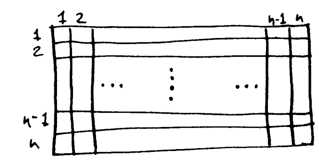
(available in BASIC, APL and many other languages)

Arrays are information setups with numbered positions. The positions can contain all sorts of different things, however: numbers, letters or other data, depending on the data structures allowed in the language.

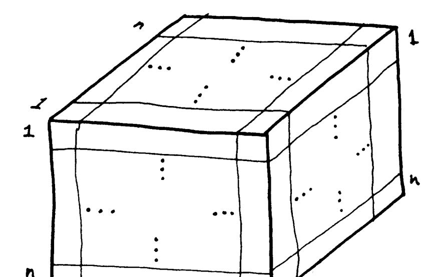
ONE-DIMENSIONAL ARRAY



TWO-DIMENSIONAL ARRAY



THREE-DIMENSIONAL ARRAY

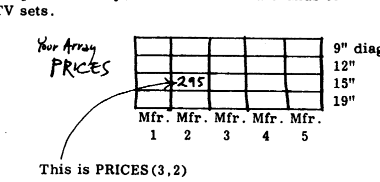


A one-dimensional array is like a row, a two-dimensional array is like a tabletop, a three-dimensional array is like a box, and for more dimensions you can't visualize.

Arrays are handy for working with a lot of different things one at a time. They can be given names just like variables.

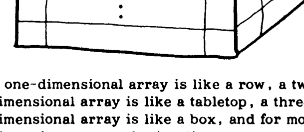
Suppose you have a one-dimensional array named SAM. Then in a program you can usually ask for the third element in SAM by referring to SAM(3). Better than that: you can refer by turns to every element of SAM by using a counting variable and changing its value. SAM(JOE) can be any one of the elements of the array, if we set the value of JOE, the counting variable, to the number of the position we want to point to.

For arrays having more than one dimension, the principle is the same. You may refer in a program to any space in the array by giving a number in parentheses, or subscript, specifying the space's position in each dimension. Suppose you have an array named PRICES, which gives the prices of, say, various sizes and brands of TV sets.



This is PRICES(3,2)
because it's the item in row 3, column 2.

Suppose you have a two-dimensional array giving the telephone numbers, salaries and ages of several different employees of a company. You have decided to call the array WHAM.



You can refer to any single entry in this array as WHAM(IRV,JOE), where IRV and JOE are two counting variables you've decided to set up.

If you set IRV and JOE both to 1, WHAM(IRV,JOE) is really WHAM(1,1), which refers you to the telephone number of employee A. If you change JOE to 2, that gives you WHAM(1,2), giving you B's phone; while WHAM(2,1) would be A's salary.

These are just the mechanics. What you choose to do with this sort of thing is your own affair. Counting around in arrays (and core memory, where they're stored) is called indexing.

THE SLEEPING GIANT
TRAC* Language

A mild-mannered man in Cambridge, Massachusetts, who owns his own very small business, is the creator of one of the most extraordinary and powerful computer languages there is, though lots of people in the field don't realize it. The language is fairly well-known among professionals, but its real power is hardly suspected.

If BASIC is a fairly conventional programming language, strongly resembling FORTRAN, TRAC (Text Reckoning and Compiling) Language is fairly unusual.

The name of it is "TRAC Language," not just TRAC — because it's a registered brand name (like Kleenex Tissues). Within the rules, the word "TRAC" is an adjective and not a noun. Thus TRAC is its first name, Language is its last; so we can refer to "TRAC Language" instead of having to precede it with the.

It is included here for several reasons.

1) It is extremely easy to learn, at least for beginners. Experienced programmers often have trouble with it.

2) It is extremely powerful for non-numeric tasks. In fact, it is ideal for building your own personal language.

3) It offers perhaps the best control of mass storage, and your own style of input-output, of any language.

4) It is superbly documented and explained with the new "The Beginner's Manual for TRAC Language," which is now available.

5) It is likely to catch on one of these days. (Some large corporations have been investigating it extensively.)

It is not so much the basic idea of TRAC Language, but the neatness with which the idea has been elaborated, that is so nice.
As a side point, here is an important motto for thinking in general about computers (and about other things in general):

MAKING THINGS FIT TOGETHER WELL
TAKES A LOT OF WORK AND THOUGHT.

Let Calvin Mooers' TRAC Language be a shining example.

TRAC Language is great for creating highly interactive systems for special purposes, including turnkey systems for inexperienced users and "good-guy" systems. It combines this with good facilities for handling text, and what is needed along with that, terrific control over mass storage. It is also excellent for simulating complex on-off systems; rumor has it that TRAC Language was used for simulating a major computer before it was built.

Against these advantages we must balance TRAC Language's less fortunate characteristics. For numerical operations it is extremely slow, if not terrible, compared to the most popular languages. The same applies to handling numerical arrays and controlling loops, which are comparatively awkward in TRAC Language.

Finally, many programmers are incensed by the number of parentheses that turn up in TRAC programs; in this it resembles the language LISP. But this is an aesthetic judgement.

The TRAC Language has been thought out in great detail for total compatibility of all parts. (Moreover, by standardizing the language exactly, Mooers heroically assures that programs can be moved from computer to computer without difficulty.)

In the well-thought-out ramifications of its basic concept, the TRAC Language is so elegant as to constitute a work of art. It beautifully fulfills this rule:

> "... the facilities provided by the language should be constructed from as few basic ideas as possible, and ... these should be general-purpose and interrelated in the language in a way which avoided special cases wherever possible." (Harrison, Data-Structures and Programming, pub. Scott, Foresman, p. 251.)

The fundamental idea of TRAC Language, which has been worked out in detail with the deepest care, thought and consistency, is this:

ALL IS TEXT.

That is, all programs and data are stored as strings of characters, in the same manner. They are labelled, stored, retrieved, and otherwise treated in the same way, as strings of text characters.

Data and programs are not kept in binary form, but remain stored in character form, much the way they were originally put in. The programs are examined for execution as text strings, and they call data in the form of text strings.

This gives rise to certain interesting kinds of compatibility.

a) Complete compatibility exists in the command structure: the results of one command can become another command or can become data for another command. ALMOST NOTHING CREATES AN ERROR CONDITION. If enough information is not supplied to execute a command, the command is ignored. If too much information is supplied, the extra is ignored.

b) Complete compatibility exists in the data: letters and numbers and spaces may be freely intermixed. Special terminal characters (like carriage returns and backspaces) are handled just like other characters, giving the programmer complete control of the arrangement of output on the page.

c) Complete compatibility also exists from one computer to another, so that work on one computer can be moved to another with ease. By the trademark TRAC, Mooers guarantees it — an innovation.

COMMAND FORMAT

A TRAC command has the following form. The cross-hatch or sharp-sign is the way this language identifies a command's beginning.

#(NM, arg2, arg3, arg4, ..)

NM is the name of any TRAC command. It counts as the first "argument," or piece of information supplied. Arg2, arg3, etc. are whatever else the command needs to know to be carried out.

We will look first at examples that use the arithmetic commands of TRAC Language, not because it is particularly good at arithmetic, which it isn't, but because they're the simplest commands. The arithmetic commands are AD (add), SU (subtract, ML (multiply), DV (divide). Each arithmetic command takes three arguments, the command name and two numbers. Examples:

#(AD, 1, 2)

is a command to add the numbers 1 and 2.

#(SU, 4, 3)

is a command to subtract the number 3 from the number 4.

#(ML, 632, 521)

is a command to multiply 632 by 521.

#(DV, 100, 10)

is a command to divide 100 by 10.

Now comes the interesting part.

The way TRAC commands may be combined provides the language's extraordinary power. This is based on the way that the TRAC processor examines the program, which is a string of character codes. Watch as we combine two AD instructions:

#(AD, 3, #(AD, 2, 5))

The answer is 10. Miraculous!

How can this be?

A comma ends an argument
in the TRAC language?
Ah, that all arguments
could be ended so easily.
--My grandfather.

* TRAC is a registered service mark of Rockford Research, Inc. Description of TRAC Language primitives adapted by permission from "TRAC, A Procedure-Describing Language for the Reactive Typewriter", copyright © 1966 by Rockford Research, Inc.

I am grateful to C.A.R. Kagan, of Western Electric Engineering Research Center, for his extensive (and finally successful) efforts to interest me in TRAC Language.

Mild-mannered Calvin Mooers steps into a phone booth, tears open his terminal, and

#(POW!)

IT'S SUPERLANGUAGE!

THE MAGIC SCAN

The secret of combining TRAC commands is that every command, when executed, is replaced by its answer; and whatever may result is in turn executed.

There is an exact procedure for this:

```
SCAN FROM LEFT TO RIGHT
     UNTIL A RIGHT PARENTHESIS;
→RESOLVE THE CONTENTS OF THE
     PAIRED COMMAND PARENTHESES
     (execute and replace by the command's result);
STARTING AT THE BEGINNING OF THE RESULT,
KEEP SCANNING LEFT-TO-RIGHT
     UNTIL A RIGHT PARENTHESIS.
WHEN YOU GET TO THE END, PRINT OUT
     WHAT'S LEFT.
```

The beauty part is how it all works so good.

An arithmetic example — so you get the procedure.

#(AD, 2, #(AD, 3, 4)) — first right parenthesis found.

execute what's in the command parentheses & replace with their answer, leaving:

7

#(AD, 2, 7) — scan to next right parenthesis

execute & replace

9

find no more parentheses print out what's left.

You might try this yourself on a longer example:

#(AD, #(SU, #(AD, 3, 4), #(SU, 7, 3)), 1)

Here is an interesting case:

#(AD, 1)

There's no third argument to add to the 1 — but that's okay in TRAC Language. 1 it remains.

PULLING IN OTHER STUFF

The core memory available to the use is divided into two areas, which we may call WORKSPACE and STANDBY.

#(ML, #(AD, 7, 3), #(SU, 16, 9))   WORKSPACE
STANDBY   Strings with Names

The Standby area contains strings of characters with names. Here could be some examples:

names	strings
HAROLD	54321
SUE	? ! *
PROGRAM	#(PS, HELP: I AM TRAPPED IN A LOOP)#(CL, PROGRAM)
GALOSHES	I MUSTN'T FORGET MY GALOSHES.

There is an instruction that moves things from the Standby area to the Workspace. This is the CALL instruction.

#(CL, whatever)

The CALL instruction pulls in a copy of the named string to replace it, the call instruction, in the work area. The string named in the call instruction also stays in the Standby area until you want to get rid of it. Example:

#(CL, HAROLD)

would be replaced by

54321

Suppose we say in a program

#(AD, 1, #(CL, HAROLD))

Then the result is:

54322

Now let's do a program loop using the CALL. If we type in to our TRAC processor

#(CL, PROGRAM)

it should type

HELP; I AM TRAPPED IN A PROGRAM LOOP
HELP; I AM TRAPPED IN A PROGRAM LOOP
HELP; I AM TRAPPED IN A PROGRAM LOOP

indefinitely.

Why is this? Let's go through the steps.

We noted that in our Standby area we had a string named PROGRAM which consisted of

#(PS, HELP; I AM TRAPPED IN A PROGRAM LOOP)#(CL, PROGRAM)

The TRAC processor scans across it to the first right parenthesis.

#(PS, HELP; I AM TRAPPED IN A PROGRAM LOOP)#(CL, PROGRAM)

and now executes this.

It happens that PS is the PRINT STRING instruction. PRINT STRING prints out its second argument, and forgets the rest. But the only argument after PS is

HELP; I AM TRAPPED IN A PROGRAM LOOP

so it prints that. If it had said

HELP, I AM TRAPPED IN A PROGRAM LOOP

the PRINT STRING command would only have printed

HELP

since a comma ends an argument in TRAC language.

Now, the PRINT STRING command leaves no result, so it is vaporized; all we have left in the work area is

#(CL, PROGRAM)

which is now scanned. But that's another CALL, and when it is executed by fetching the object called PROGRAM, its replacement in the work area is

#(PS, HELP; I AM TRAPPED IN A PROGRAM LOOP)#(CL, PROGRAM)

and guess what. We done it again.

(Another example of TRAC Language's consistency: suppose it executes the command

#(CL, EBENEZER)

when there is no string called EBENEZER. The result is nothing; so that command disappears, leaving no residue.)

THE FORM COMMANDS

Let us be a little more precise. The Standby area is really called by Mooers "forms storage," and a string-with-name that is kept there is called a form. One reason for this terminology is that these strings can consist of programs or arrangements that we may want to fit together and combine. Thus they are "forms".

1. CREATING A FORM

To create a form, you use the DEFINE STRING command:

#(DS, formname, contents)

The arguments used by DS give a name to the form and specify what you want to have stored in it. Example:

#(DS, ELVIS, 1234)

creates a form named ELVIS with contents 1234.

ELVIS — 1234

(Note that to get a program into a form without its being executed on the way requires some preparation. For this, "protection" is used; see end of article.)

It turns out that DEFINE STRING is the closest TRAC Language has to an assignment statement (as in BASIC, LET A = WHATEVER). If you want to use a variable A, say, to store the current result of something, in TRAC Language you create a form named A.

#(DS, A, WHATEVER)

Whenever the value of A is changed, you redefine form A.

2. CALLING A FORM.

As noted already,

#(CL, ELVIS)

will then be replaced by

1234

But a wonderful extension of this, that hasn't been mentioned yet, is

2A. THE IMPLICIT CALL.

You don't even have to say CL to call a form. If the first argument of a command — that is, the first string inside the command parentheses — is not a command known to TRAC Language, why, the TRAC processor concludes that the first argument may be the name of a form. So now if you type

#(AD, #(HAROLD), #(ELVIS))

it will first note, on reaching the right-paren of the HAROLD command, that since HAROLD is 54321, you evidently wanted this:

#(AD, 54321, #(ELVIS))

rescan of result

and then will do the same with ELVIS:

#(AD, 54321, 1234)

so that pretty soon it'll type for you

55555

This language is marvelously suited to data base management, management information systems, interactive query systems, and the broad spectrum of "business" programming.

For large-scale scientific number crunching, not so good.

With one exception: "infinite precision" arithmetic, when people want things to hundreds of decimal places.

Chugga chugga.

This implicit call is the trick that allows people to create their own languages very quickly. In not very long, you could create your own commands — say ZAPP, MELVIN and some more; and while at first it is more convenient to type in the TRAC format

#(ZAPP, #(MELVIN))

it is very little trouble in TRAC Language to create new syntaxes of your own like

ZAPP ! MELVIN

that are interpreted by the TRAC processor as meaning the same thing.

2B. FILLING IN HOLES.

Another thing the CALL command in TRAC Language does is fill in holes that exist in forms. Let us represent a hole as follows:

[ ]

Now suppose there is a TRAC form with a hole in it, like this.

WORD — H[ ]T

Additional arguments in the call get plugged into holes in the form. Examples:

call	result
#(CL, WORD)	HT
#(CL, WORD, O)	HOT
#(WORD, A)	HAT
#(WORD, OO)	HOOT

Now, a form can have a number of different holes. Let us denote these by

[1] [2] [3] [4] . . .

Now suppose we have a form

WORD — [1]H[2]T[3]

which we might call numerous ways:

call	result
#(WORD, W, I, E)	WHITE
#(WORD, , OO, OWL) (Note that putting nothing between two commas made nothing the argument.)	HOOTOWL
#(WORD, #(WORD, , O)S, O)	HOTSHOT

Perhaps you can think of other examples.

This fill-in technique is obviously useful for programming. If a form contains a program, its holes can be made to accept varying numbers, form names, text strings, other programs. Example: Suppose we want to create a new TRAC command, ADD, that adds three numbers instead of just two. Fair enough:

ADD — #(AD, [1], #(AD, [2], [3])) and there you are.

This brings up another example of how nicely TRAC Language works out. Suppose you have the following in forms storage:

ZOWIE — #(ZIP, [1], [2])
ZIP — #(ZAP, [1], [2])
ZAP — #(AD, [1], [2])

Try acting this one out with pencil and paper. Suppose you type in

#(ZOWIE, 5, 7)

It happens that the arguments 5 and 7 will be passed neatly from ZOWIE to ZIP to ZAP to the final execution of the AD; all through the smooth plugging of holes by the implicit call and the Magic Scan procedure of the TRAC processor.

TRAC Language is a so-called "list processing language" or "List Language." This term has come to mean any language for twiddling data having arbitrary and changing form. Two other prominent languages of this type are SNOBOL and LISP (see p. 31).

List languages are traditionally freaky.

TRAC Language is:

an interpretive language
(each step carried out directly by the processor without conversion to another form first);
an extensible language
(you can add your own commands for your own purposes);
a list-processing language
(for handling complex and amorphous forms of data that don't fit in boxes and arrays).
It is one of the few such languages that fits in little computers.

3. DRILLING THE HOLES

The holes (called by Mooers segment gaps) are created by the SEGMENT STRING instruction.

#(SS, formname, whatever1, whatever2 . . .)

where "formname" is the form you want to put holes in and the whatevers are things you want to replace by holes. Example: Suppose you have a form

INSULT → YOU ARE A CREEP

You make this more general by means of the SEGMENT STRING instruction:

#(SS, INSULT, CREEP)

resulting in

INSULT → YOU ARE A [ ]

which can be filled in at a more appropriate time.

Fuller example. Suppose we type into the TRAC processor the following:

```
#(DS, THINGY, ONE FOR THE MONEY AND TWO FOR THE SHOW)
#(SS, THINGY, ONE, TWO,  )
                       ↑
                       └── note space
```

We have now created a form THINGY and replaced parts of it with segment gaps. Since each of the later arguments of SEGMENT STRING specifies a differently numbered gap, we will have gaps numbered [1], [2], and [3]. The gap [1] will have replaced the word ONE, the gap [2] will have replaced the word TWO, and a lot of gaps numbered [3] will have replaced all the spaces in the form (since the fifth argument of SS was a space). The resulting form is:

THINGY
[1][3]FOR[3]THE[3]MONEY[3]AND[3][2][3]FOR[3]THE[3]SHOW

We can get it to print out interestingly by typing #(CL, THINGY, RUN, HIDE) (since after the call, the plugged-in form will still be in the forms storage.) This is printed:

RUNFORTHEMONEYANDHIDEFORTHESHOW

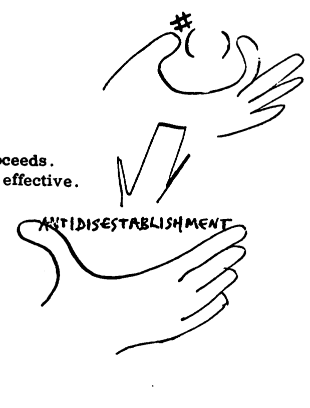
or perhaps, if we use a carriage return for the last argument, we can get funny results. The call

#(THINGY, NOT A FIG, THAT, [carriage return]
)

should result in

NOT A FIG
FOR
THE
MONEY
AND
THAT
FOR
THE
SHOW

In TRAC Language, every command
is replaced by its result
as the program's execution proceeds.
This is ingenious, weird and highly effective.



TEST COMMANDS IN TRAC LANGUAGE

There are test commands in TRAC Language, but like everything else they work on strings of characters. Thus they may work on numbers or text. Consider the EQ command (test if equal):

#(EQ, firstthing, secondthing, ifso, ifnot)

where "firstthing" and "secondthing" are the strings being compared, and ifso and ifnot are the alternatives. If firstthing is the same as secondthing, then ifso is what the TRAC processor does, and ifnot is forgotten. Example:

#(EQ, 3, #(SU, 5, 2), HOORAY, NUTS)

If it turns out that 3 is equal to #(SU, 5, 2), which it is, then all that would be left of the whole string would be

HOORAY

while otherwise the TRAC processor would produce NUTS.

To most computer people this looks completely inside-out, with the thing to do next appearing at the center of the test instruction. Others find this feature at-trac-tive.

DISK OPERATIONS

Now for the juicy disk operations. Storing things on disk can occur as an ordinary TRAC command.

#(SB, name, form1, form2, form3 . . .)

creates a place out somewhere on disk with the name you give it, and puts in it the forms you've specified. Example:

#(SB, JUNK, TOM, DICK, HARRY)

and they're stored. If you want them later you say

#(FB, JUNK)

and they're back.

Because you can mix the disk operations in with everything else so nicely, you can chain programs and changing environments with great ease to travel smoothly among different systems, circumstances, setups.

Here is a stupid program that scans all incoming text for the word SHAZAM. If the word SHAZAM appears, it clears out everything, calls a whole nother disk block, and welcomes its new master. Otherwise nothing happens. If you have access to a TRAC system (or really want to work on it), you may be able to figure it out. (RESTART must be in the workspace to begin.)

RESTART → #(DS, TEMP, #(RS))#(SS, TEMP,)#(RPT)

RPT → #(EQ, SHAZAM, #(TEST), (#(EVENT)))#(RPT)

TEST → #(CS, TEMP, (#(RESTART)))

EVENT → #(DA)#(FB, MARVEL)#(PS, WELCOME O MASTER)

In this example, however, you may have noticed more parentheses than you expected. Now for why.

PROTECTION AND ONE-SHOT

The last thing we'll talk about is the other two syntactic layouts.

We've already told you about the main syntactic layout of TRAC Language, which is

#(   )

It turns out that two more layouts are needed, which we may call PROTECTION and ONE-SHOT. Protection is simply

(   )

which prevents the execution of anything between the parentheses. The TRAC processor strips off these plain parentheses and moves on, leaving behind what was in them but not having executed it. (But it may come back.) An obvious use is to put around a program you're designing:

#(DS, PROG, (#(AD, A, B)))

safe — stripped, stripped

but other uses turn up after you've experimented a little. The last TRAC command arrangement looks like this

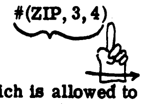
##(   )

and you can put any command in it, except that its result will only be carried one level

##(CL, ZOWIE, 3, 4)

results in (using the forms we defined earlier),

#(ZIP, 3, 4)



which is allowed to survive as is, because the moving finger of the TRAC scanner does not re-scan the result.

It is left to the very curious to try to figure out why this is needed.

> Whatever can be executed
> is replaced by
> its result.
> This may or may not
> yield something
> which is in turn
> executable.
> When nothing left is executable,
> what's left
> is printed out.
>
> That's the TRAC language.

FAST ANSWERBACK IN TRAC LANGUAGE

TRAC Language can be used for fast answerback to simple problems. Typing in long executable TRAC expressions causes the result, if any, to be printed back out immediately.

For naive users, however, the special advantage is in how easily TRAC Language may be used to program fast answerback environments of any kind.

A SERIOUS LANGUAGE; BUT BE WILLING TO BELIEVE WHAT YOU SEE

TRAC Language is, besides being an easy language to learn, very powerful for text and storage applications.

Conventional computer people don't necessarily believe or like it.

For instance, as a consultant I once had programmed, in TRAC Language, a system for a certain intricate form of business application. It worked. It ran. Anybody could be taught to use it in five minutes. The client was considering expanding it and installing a complete system. They asked another consultant.

It couldn't be done in TRAC Language, said the other consultant; that's some kind of a "university" language. End of project.

HOW TO GET IT

There have been, until recently, certain difficulties about getting access to a TRAC processor. Over the years, Mooers has worked with his own processors in Cambridge. Experimenters here and there have tried their hands at programming it, with little compatibility in their results. Mooers has worked with several large corporations, who said said they wanted to try processors to assess the value of the the language, but those endeavors brought nothing out to the public.

FINALLY, however, TRAC Language service is publically available, in a fastidiously accurate processor and with Mooers' blessing, on Computility™timesharing service. They run PDP-10 service in the Boston-to-Washington area. (From elsewhere you have to pay long distance.) The charge should run $12 to $15 per hour in business hours, less elsewhen. But this depends to some extent on what your program does, and is hence unpredictable. A licensed TRAC Language processor may be obtained from Mooers for your own favorite PDP-10. Processors for other computers, including minis, are in the planning stage.

TRAC Language is now nicely documented in two new books by Mooers, a beginner's manual and a standardization book (see Bibliography).

Since Mooers operates a small business, and must make a livelihood from it, he has adopted the standard business techniques of service mark and copyright to protect his interests. The service mark "TRAC" serves to identify his product in the marketplace, and is an assurance to the public that the product exactly meets the published standards By law, the "TRAC" mark may not be used on programs or products which do not come from Rockford Research, Inc.

Following IBM, he is using copyright to protect his documentation and programs from copying and adaptation without authority.

Mooers also stands ready to accommodate academic students and experimenters who wish to try their hands at programming a TRAC processor. An experimenter's license for use of the copyright material may be obtained for a few dollars, provided you do not intend to use the resulting programs commercially.

For information of all kinds, including lists of latest literature and application notes, contact:

Calvin N. Mooers
Rockford Research, Inc.
140-1/2 Mount Auburn Street
Cambridge, Mass. 02138 Tel. (617)876-6776

TRAC® PRIMITIVES*

OUTPUT.
- PS, string
  - PRINT STRING: prints out the second argument.

INPUT.
- RS
  - READ STRING: this command is replaced by a string of characters typed in by the user, whose end is signalled by a changeable "meta" character.
- CM, arg2
  - CHANGE META: first character of second argument becomes new meta character. May be carriage-return code.
- RC
  - READ CHARACTER: this command is replaced by the next character the user types in. Permits highly responsive interactive systems.

DISK COMMANDS.
- SB, blockname, form1, form2 ...
  - STORE BLOCK: under block name supplied, stores forms listed.
- FB, blockname
  - FETCH BLOCK: contents of named block are quietly brought in to forms storage from disk.

MAIN FORM COMMANDS.
- DS, formname, contents
  - DEFINE STRING. Discussed in text.
- CL, formname, plug1, plug2, plug3 ...
  - CALL: brings form from forms storage to working program. Plug1 is fitted into every hole (segment gap) numbered 1, plug2 into every hole numbered 2, and so on.
- SS, formname, punchout1, punchout2 ...
  - SEGMENT STRING: this command replaces every occurrence of punchout1 with a hole (segment gap) numbered 1, and so on.

INTERNAL FORM COMMANDS.

(All of these use a little pointer, or form pointer, that marks a place in the form. If there is no form remaining after the pointer, these instructions act on their last argument, which is otherwise ignored.)
- IN, formname, string, default
  - Looks for specified string IN the form, starting at pointer. If not found, pointer unmoved. (NOTE: string search can also be done nicely with the SS command.)
- CC, formname, default
  - CALL CHARACTER: brings up next character in form, moves pointer to after it.
- CN, formname, no. of characters, default
  - CALL N: brings up next N characters, moves pointer to after them.
- CS, formname, default
  - CALL SEGMENT: brings up everything to next segment gap, moves pointer to it.
- CR, formname
  - CALL RESTORE: moves pointer back to beginning of form.

MANAGING FORMS STORAGE
- LN, divider
  - LIST NAMES: replaced by all form names in forms storage, with any divider between them. Divider is optional.
- DD, name1, name2 ...
  - DELETE DEFINITION: destroys named forms in forms storage.
- DA
  - DELETE ALL: gets rid of all forms in forms storage.

TEST COMMANDS.
- EQ, firstthing, secondthing, ifso, ifnot
  - Tests if EQual: if firstthing is same as secondthing, what's left is ifso; if not equal, what's left is ifnot.
- GR, firstthing, secondthing, ifso, ifnot
  - Tests whether firstthing is numerically GReater than secondthing. If so, what's left is ifso; if not, what's left is ifnot.

OH YEAH, ARITHMETIC.

(All these are handled in decimal arithmetic, a character at a time, and defined only for two integers. Everything else you write yourself as a shorty program.)

$\left.\begin{array}{l}\text{AD}\\\text{SU}\\\text{ML}\\\text{DI}\end{array}\right\}$ mentioned in text.

BOOLEAN COMMANDS.

(Several exist in the language, but could not possibly be understood from this writeup.)

---

* Description of TRAC language primitives adapted by permission from "TRAC, A Procedure-Describing Language for the Reactive Typewriter," copyright © 1966 by Rockford Research, Inc.

BIBLIOGRAPHY

Calvin N. Mooers, The Beginner's Manual for TRAC® Language, 300 pages, $10.00, from Rockford Research, Inc. (See "Where to Get It.")

Calvin N. Mooers, Definition and Standard for TRAC® T-64 Language, 86 pages, $5.00, from Rockford Research, Inc.

Calvin N. Mooers, "TRAC, A Procedure-Describing Language for the Reactive Typewriter," Communications of the ACM, v. 9, n. 3, pp. 215-219 (March 1966). Historic paper, out of print. This paper is copyrighted, and the copyright is owned by Rockford Research, Inc., through legal assignment from the Association for Computing Machinery, Inc.

And for those who want to understand the depth of the standardization problem, Mooers offers freebie reprints of:

Calvin N. Mooers, "Accommodating Standards and Identification of Programming Languages," Communications of the ACM, v. 11, n. 8, pp. 574-576 (August 1968).

STARK & CLEVER APL

Some people call it a "scientific" language. Some people call it a "mathematical" language. Some people are most struck by its use for interactive systems, so to them it's an interactive language. But most of us just think of it as THE LANGUAGE WITH ALL THE FUNNY SYMBOLS, and here they are:

```
*ρ∪∩α⊂÷ω↑∊|∧\⊃⎕;⌈∘∇'Δ
¨¯<≠≤≥=>)∨:⊥_(↓⊤→⍳~○?⌊-
12384657]9.BF[UN←ITOQD+
PRVCAZ×WYEMO/XL,SJGKH
```

Enthusiasts see it as a language of inconceivable power with extraordinary uses. Cynics remark that it has all kinds of extraordinary powers for inconceivable uses-- that is, a weird elegance, much of which has no use at all, and some of which gets in the way.

This is probably wrong. APL is a terrific and beautiful triumph of the mind, and a very useful programming language. It is not for everybody, but neither is chess. It is for bright children, mathematicians, and companies who want to build interactive systems but feel they should stick with IBM.

APL is one of IBM's better products, probably because it is principally the creation of one man, Kenneth Iverson. It is mainly run on 360 and 370 computers, though implementations exist for the DEC PDP-10 and perhaps other popular machines. (Actually Iverson designed the language at Harvard and programmed it on his own initiative after moving to IBM; added to the product line by popular demand, it was not a planned product and might in fact be a hazard to the firm, should it catch on big.)

APL is a language of arrays, with a fascinating notation. The array system and the notation can be explained separately, and so they will.

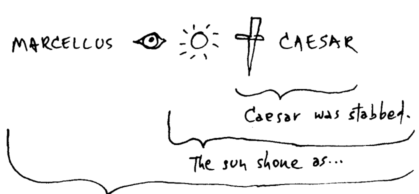
Let's just say the language works on things modified successively by operators. Their order and result is based upon those fiendish chicken scratches, Iverson notation.

THAT NIFTY NOTATION

The first thing to understand about APL is the fiendishly clever system of notation that Iverson has worked out. This system (sometimes called Iverson notation) allows extremely complex relations and computer-type events to be expressed simply, densely and consistently.

(Of course, you can't even type it without an IBM Selectric typewriter and an APL ball. Note the product-line tie-in.)

The notation is based on operators modifying things. Let's use alphabetic symbols for things and play with pictures for a minute.



In considering the successive meanings of this rebus we are proceeding from right to left, as you note, and each new symbol adds meaning. This is the general idea.

You will note, in this example, the curious arrangement whereby you can have several pictures, or operators, in a row. This is one of the fun features of the language.

TWO-SIDED OPERATORS

In old-fashioned notations, such as ordinary arithmetic, we are used to the idea of an operator between two things. Like

2 + 2

or in algebra,

x X y

These, too, occur in APL; indeed, APL can also nest two-sided operators-- that is, put them one inside the other, like the leaves of a cabbage. Old-fashioned notations nest with parentheses. But APL nests leftward. It works according to a very simple right-to-left rule.

x X y X 2 + 2

the result of this

is operated on by
the next thing and operator,
yielding another result

which is in turn operated on by
the next thing and operator,
yielding final result.

ONE-SIDED OPERATORS

We are also used to some one-sided operators in our previous life. For instance:

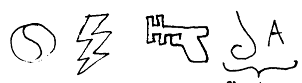
- 1

means the negation of 1;

- (- 1)

means negating that.

APL can also nest one-sided operators.



first operator is
applied to ~~A~~;

result is worked on
by second operator;

result is worked on
by third operator;

result is worked on by
fourth operator,
yielding final result.

SAME SYMBOLS WORK BOTH WAYS

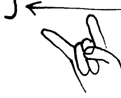
Now, one of the fascinating kickers of APL is the fact that most of the symbols have both a one-sided meaning and a two-sided meaning; but, thank goodness, they can be easily kept straight.

Here is a concrete example: the symbol ⌈ or "ceiling." Used one-sided, the result of operator ⌈ applied to something numerical is the integer just above the number it is applied to: ⌈7.2 is 8. Used two-sided, the result is whichever of the numbers it's between is larger: 10⌈6 is 10. (There is also ⌊ , floor, which you can surely figure out.)

Now, when you string things out into a long APL expression, Iverson's notation determines exactly when an operator is one-sided and when it is two-sided:

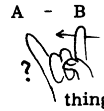
As you go from right to left,

another thing?
another op?
OP THING



you generally start with a thing on the right. Then comes an operator. If the next symbol is another thing, then the operator is to be treated as a two-sided operator (because it's between two things). If the object beyond the first operator is another operator, however, that means APL is supposed to stop and carry out the first operator on a one-sided basis. Example:

A - B



thing,
op,
thing.

Conclusion:
It's two-sided.
Interpretation:
"subtract B from A."

A + - B

thing,
op,
op--
stop.

Conclusion:
The first operator
is one-sided.
Interpretation:
"negate B."
Then take next symbol.

A WEIRD EXAMPLE, TO HELP WITH THE NOTATION.

Just for kicks, let us make up a notation having nothing to do with computers, using these Iverson principles:

1) If an operator or symbol is between two names of things, carry it out two-sidedly. If not, carry it out one-sidedly.

2) Go from right to left.

The best simple example I can think of involves file cards on the table (named A, B, C...) and operators looking like this:

0⤸ 45⤸ 90⤸ 180⤸ 45⤹ 90⤹ 180⤹

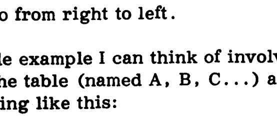
to which we may assign the following meanings:

ONE-SIDED: ROTATION OPERATORS

0⤸ A	do nothing to A
45⤸ A	rotate A clockwise 45°
90⤸ A	rotate A clockwise 90°
	etc.

TWO-SIDED: STAPLING OPERATORS

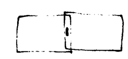
B 45⤸ A — staple A (thing named on the right) to B (thing named on the left) at a position 45° clockwise from middle of B's centerline.



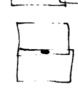
And equivalently for other angles.

Now, using these rules, and letting our things be any file cards that are handy, here are some results:

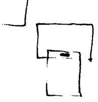
A 0⤸ B



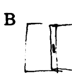
A 90⤸ B



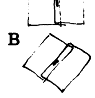
A 90⤸ 90⤸ B



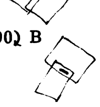
90⤸ A 90⤸ B



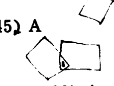
45⤹ A 90⤸ B



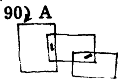
45⤹ A 90⤸ 90⤸ B



B 0⤸ 45⤸ A



C 45⤸ B 0⤸ 90⤸ A



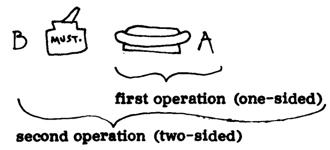
WOOPS!
One of these examples is wrong!
Can you find it?

It's hard to believe, but there you are. This notation seems adequate to make a whole lot of different stapled patterns.

Exercise! Use this nutty file card notation to program the making of funny patterns. Practice with a friend and see if you can communicate patterns through these programs, one person uncomprehendingly carrying out the other's program and being surprised.

The point of all this has been to show the powerful but somewhat startling way that brief scribbles in notations of this type can have all sorts of results.

Here is another example showing how we chug along the row of symbols and take it apart. Again, the alphabetical entities represent things.

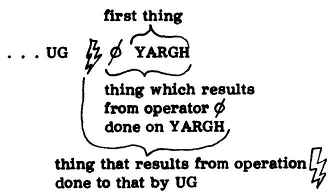


Try dividing up these examples:

One more thing needs to be noted. Not only can we work out the sequences of operations, from right to left, between the symbols; the computer can carry them out in a stable fashion. Which is of course essential.

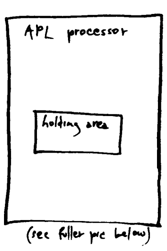
INSIDE

The truth of the matter is that APL in the computer is a continuing succession of things being operated on and replaced in the work area.



and so on.

What is effectively happening is that the APL processor is holding what it's working on in a holding area. The way it carries out the scan of the APL language, there only has to be one thing in there at a time.



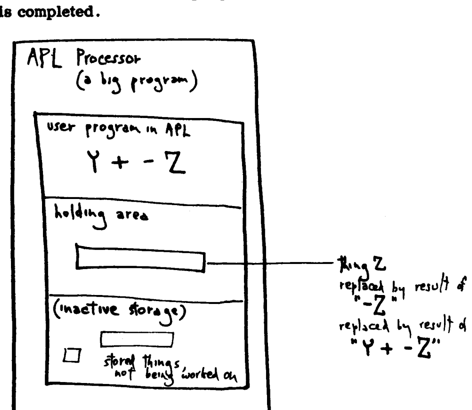
Suppose we have a simple user program,

Y + - Z

Starting at the right of this user program, the main APL program puts Z into the work area. That's the first thing. Then, stepping left in the user program, the APL processor follows the rules and discovers that the next operation makes it

- Z

which happens to mean, "the negation of Z." So it carries this out on Z and replaces Z with the result, -Z. Then, continuing to scan leftward, the APL processor continues to replace what was in the work area with the result of each operation in the successive lines of the user program, till the program is completed.



SOME APL OPERATORS

It would be insane to enumerate them all, but here is a sampling of APL's operators. They're all on the pocket cards (see Bibliography).

For old times' sake, here are our friends: (And a cousin thrown in for symmetry.)

+A	plain A (whatever A should happen to be)
A+B	A plus B (whatever A should happen to B, heh heh)
-B	negation of B
A-B	A minus B
×B	the sign of B (expressed as -1,0 or 1)
A×B	A times B

And here are some groovies:

!A	factorial A (1×2×3 ... up to A)
A!B	the number of possible combinations you can get from B, taken A at a time
?A	a random integer taken from array A
A?B	take some integers at random from B. How many? A.

But, of course, APL goes on and on. There are dozens more (including symbols made of more than one weird APL symbol, printed on top of each other to make a new symbol).

Consider the incredible power. Single APL symbols give you logarithms, trigonometric functions, matrix functions, number system conversions, logs to any arbitrary base, and powers of e (a mysterious number of which engineers are fond).

Other weird things. You can apply an operation to all the elements of an array using the / operator: +/A is the sum of everything in A, ×/A is the combined product of everything in A. And so on. Whew.

As you may suspect, APL programs can be incredibly concise. (This is a frequently-heard criticism: that the conciseness makes them hard to understand and hard to change.)

MAKE YOUR OWN

Finally and gloriously, the user may define his own functions, either one-sided or two-sided, with alphabetical names. For instance, you can create your own one-sided operator ZONK, as in

ZONK B

and even a two-sided ZONK,

A ZONK B

which can then go right in there with the big boys:

A ⌽ ZONK ⍳↓ B

Don't ask what it means, but it's allowed.

APL THINGS, TO GO WITH YOUR OPERATORS

As we said, APL has operators (already explained) and things. The things can be plain numbers, or Arrays (already mentioned under BASIC). Think of them as rows, boxes and superboxes of numbers:

2 4 6 8 10	a one-dimensional thing
2 4 3 5	a two-dimensional thing
2 4 6 8	a three-dimensional thing, seen from the front. Maybe we better look at the levels side by side: 1 3 2 4 5 7 6 8

APL can have Things with four dimensions, five and so on, but we won't trouble you here with pictures.

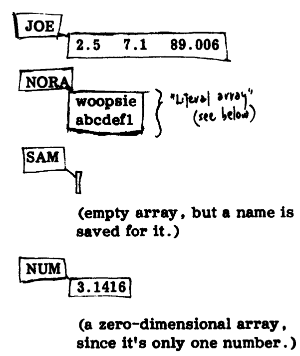
Oh yes, and finally a no-dimensional thing. Example:

75.2

It is called no-dimensional because there is only one of it, so it is not a row or a box.

Seriously, these are arrays, and Iverson's APL works them over, turns them inside out, twists and zaps through to whatever the answers are.

As in BASIC and TRAC, the arrays of APL are really stored in the computer's core memory, associated with the name you give them. The arrays may be of all different sizes and dimensionality:



Each array is really a series of memory locations with its label and boxing information-- dimensions and lengths-- stored separately. One very nice thing about APL is that arrays can keep changing their sizes freely, and this need be of no concern to the APL programmer. (The arrays can also be boxed and reboxed in different dimensions just by changing the boxing information-- with an operator called "ravel.")

STOP THE PRESSES!

An APL machine, a mini that does nothing but APL, is now available from a Canadian firm for the mere pittance of

THREE THOUSAND FIVE HUNDRED DOLLARS,

the price of many a mere terminal. This according to Computerworld, 10 Oct 73.

Run, don't walk, to Micro Computer Machines, Inc., 4 Lansing Sq., Willowdale, M2J 1T1, Ontario, Canada. That \$3500 gets you a 16K memory, the APL program, keyboard and numerical keyboard, and plasma display. Cassette (which apparently stores and retrieves arrays by name when called by the program) is \$1500 extra. RUNS ON BATTERIES. Sorry, no green stamps. (Note that the APL processor takes up most of the 16K, but you can get more.)

* * * * * * * * * * * * *

The rumor that IBM has APL on a chip, inside a Selectric -- which therefore does all these things with no external connection to any (external) computer-- remains unsubstantiated. The rumor has been around for some time.

But it's quite possible.

The thing is, it would probably destroy IBM's entire product line-- and pricing edifice.

Few people know all of APL, or would want to. The operations are diverse and obscure, and many of them are comprehensible only to people in mathematical fields. However, if you know a dozen or so you can really get off the ground.

As in BASIC, you can use subscripts to get at specific elements in arrays. Referring to the examples above, if you type

JOE[2]

you get back on your typewriter its value

7.1

and if you type

NORA[2,4]

you get back

d

There are basically four kinds of information used by APL, and all of them can be put in arrays. Three of these types are numerical, and arrays of them look like this on paper:

Integer arrays: 2 4 -6 8 10 2048

Scalar arrays: 2.5 -3.1416 0.001 2795333.1
(a scalar is something that can be measured on a ruler-like scale, where there are always points in betweeen.)

Logical arrays: 1 0 0 0 1 0 1
(these arrays of ones and zeroes are called "logical" for a variety of reasons; in this case we could call them "logical" simply because they are used for picking and choosing and deciding.)

These three numerical types of information may be freely intermixed in your arrays. One more type, however, is allowed. It's hard to figure out from the manuals, but evidently this type can't be mixed in with the others too freely. We refer to the alphabetical or "literal" array, as in

The quick brown fox jumped over the lazy dog.

Now, pre-written APL programs can print out literal information, and accept it from a user at a terminal, which is why APL is good for the creation of systems for naive users (see "Good-Guy Systems," p. 13).

Literal vectors may be picked apart, rearranged and assembled by all the regular APL operators. That's how we twiddle our text.

CRASHING THE SYMBOLS TOGETHER

Now that we know about the operators and the arrays, what does APL do?

It works on arrays, singly and in pairs, according to those funny-looking symbols, as the APL processor scans right-to-left.

IVERSON'S TAFFY-PULL

A number of basic APL operators help you stretch, squish and pull apart your arrays. Consider the lowly comma (called "ravel," which means the same as "unravel").

,A	forget A's old dimensions, make it one-dimensional.
A,B	make A and B one long one-dimensional array.

Here is how we make things appear and disappear. ("Compression.")

A/B — A must be a one-dimensional array of ones and zeroes. The result is those elements of B selected by the ones.
Example:
1 0 1/c a t
results in
c t

The opposite slash has the opposite effect, inserting extra null elements where there are zeroes:

1 1 0 1\3 5 9
results in
3 5 0 9

Here's another selector. This operator takes the first or last few of A, depending on size and sign of B:

B↑A

and B↓A is the opposite.

If you want to know the relative positions of numbers of different sizes in a one-dimensional array,

⍋ (name of array)

will tell you. It gives you the positions, in order of size, of the numbers. And ⍒ does it for descending order.

These are just samples. The list goes on and on.

SAMPLE PROGRAMS

Here is an APL program that types out backwards what you type in. First look at the program, then the explanation below.

```
    ∇ REV
[1]  I ← ⍞
[2]  ⍞ ← ⌽ I
    ∇
```

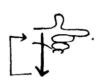
Explanation. The down-pointing triangles ("dels") symbolize the beginning and end of a program, which in this case we have called REV. On Line 1, the "Quote-Quad" symbol (on the right) causes the APL processor to wait for alphabetical input. Presumably the user will type something. The user's line of input is stuffed into thing or array I. The user's carriage return tells the APL processor he has finished, so it continues in the program. On the second line, APL takes array I and does a one-sided ⌽ to it, which happens to mean turning it around. Left-arrow into the quote-quad symbol means print it out.

Because of APL's compactness, indeed, this magnificent program can all go on one line:

```
    ∇ REV
[1]  ⍞ ← ⌽ I ← ⍞
    ∇
```

First the input goes into I, then the processor does a ⌽ I (reversal) and puts it out.

And here is our old friend, the fortune-cookie prisoner.



```
    ∇ INF
[1]  ⍞ ← 'HELP, I AM CAUGHT IN A LOOP'
[2]  → 1
    ∇
```

On line 1 the program prints out whatever's in quotes. And line 2 causes it to go back and do line 1 again. Forever.

THE TEST-AND-BRANCH IN APL

It should be mentioned at this point that branching tests are conducted in APL programs by specifying conditions which are either true or false, and APL's answer is 1 if true, 0 if false. (This is another thing these logical arrays are for.)

Example:

3>2

This operation leaves the number 1, because 3 is greater than 2. So you could branch on a test with something like

→7 × A > B

which branches to line 7 in the program if A is greater than B, and is ignored (as an unexecutable branch to line zero) if B is greater than A.

Some love it, some hate it.

THE APL ENVIRONMENT

Aside from the APL language itself, to program in APL you must learn a lot of "system" commands, alphabetical commands by which to tell the APL processor what you want to do in general -- what to store, what to bring forth from storage, and so on.

Ordinarily you have a workspace, a collection of programs and data which you may summon by name. When it comes-- that is, when the computer has fetched this material and announced on your terminal that it is ready-- you can run the programs and use the data in your workspace. You can also have passwords for your different workspaces, so others at other terminals cannot tamper with your stuff.

This is not the place to go into the system commands. If you're serious, you can learn them from the book or the APL salesman.

There are many, many different error messages that the APL processor can send you, depending on the circumstances. It is possible to make many, many mistakes in APL, and there are error messages for all of them. All of them, that is, that look to the computer like errors; if you do something permissible that's not what you intended, the computer will not tell you.

But it is a terminal language, designed to help people muddle through.

Good luck!

IVERSON'S STRANGE AND WONDERFUL CHOICES OF SYMBOLS

Iverson's notation is built around the curious principle of having the same symbols mean two things depending on context. (Goodness knows he uses enough different symbols; doubling up at least means he doesn't need any more.) It turns out that this notation represents a consistent series of operations in astounding combinations.

The overall APL language, really, is the carrying through of this notation to create an immensely powerful programming language. The impetus obviously came from the desire to make various intricate mathematical operations easy to command. The result, however, is a programming language with great power for simpler tasks as well.

Now, the consequences of this overall idea were not determined by God. They were worked out by Iverson, very thoughtfully, so as to come out symmetrical-looking and easy to remember. What we see is the clever exploitation of apparent but inexact symmetries in the ideas. Often APL's one-sided and two-sided pairs of operators are more suggestively similar than really the same thing.

When Iverson assigns one-sided and two-sided meanings to a symbol, often the two meanings may look natural only because Iverson is such an artist. Example:

two-sided	one-sided
A×B	×B
A times B	the sign of B

This makes sense. To argue that it is inherent in "taking away half the idea of multiplication," however, is dubious.

Some symmetries Iverson has managed to come up with are truly remarkable. The arrow, for instance. The left arrow:

A←B
Assignment statement: B (which may have been computed during the leftward scan) is assigned the name of A;

and the right arrow:

→B
The jump statement, where B (which may have been computed during the leftward scan) is a statement number; the program now goes and executes that line.

This symmetry is mystically interesting because the assignment and jump statements are so basic to programming.

Or consider this:

⎕←X
print X.

X←⎕
take input from the user and stuff it into X.

Another weird example: supposedly the conditional branch

→B/A

(one way of writing, "jump to A if B is true") is a special case of the "compression" operator. (Berry 360 primer, 72 and 165.) This is very hard to understand, although it seems clear while you're reading it.

On the other hand, there is every indication that APL is so deep you keep finding new truths in it. (Like the above paragraph.) The whole thing is just unbelievable. Hooray for all that.

APL FOR USER-LEVEL SYSTEMS
(See "Good-Guy Systems," p. 13)

Because APL can solicit text input from a user and analyze it, the language is powerful for the creation of user-level environments and systems-- with the drawback, universal to all IBM terminals, that input lines must end with specific characters. In other words, it can't be as fully interactive as computer languages that use ASCII terminals.

Needless to say, the mathematical elegance and power of the system is completely unnecessary for most user-level systems. But it's nice to know it's there.

APL is probably best for systems with well-defined and segregated files-- "array-type problems," like payroll, accounts and so on. It is not suited for much larger amorphous and evolutionary stuff, the way list languages like TRAC are. Don't use APL if you're going to store large evolving texts or huge brokerage data bases, like what tankers are free in the Mediterranean.

The quickest payoff may lie in using APL to replace business forms and hasten the flow of information through a company. A salesman on the road with an APL terminal, for instance, can at once enter his orders in the computer from the customer's office, checking inventory directly. If the program is up.

ROUND (an obscure and donnish joke)

ρ, the Greek letter "rho," is an APL operator for testing the size of arrays. When used in the one-sided format, it gives the sizes of each dimension of an array.

Thus

ρA, when A is $\begin{bmatrix}1 & 2\\ 3 & 4\end{bmatrix}$ is 2 2.

And now

ρ'YOUR BOAT'
equals 9, since there are 9 letters in the array 'YOUR BOAT';

ρρ'YOUR BOAT'
is 1,
since ρ 9 is 1, and

ρρρ 'YOUR BOAT'
is likewise 1.

This language is superb for "scientific" programming, including heavy number crunching and experimentation with different formulas on small data bases. (Big data bases are a problem.)

It is also not bad for a variety of simple business applications, such as payroll, accounting, billing and inventory.

FAST ANSWERBACK IN APL

If you want quick answers, the APL terminal just gives you the result of whatever you type in. For instance,

3 x 4

will cause it to print out

12

and the same goes for far less comprehensible stuff like

7 ≥ ⍋ ⌽ ? $\underbrace{1\ \ 2\ \ 3\ \ 4}_{\text{typed-in array}}$ (carriage return)

PROGRAMS IN APL

But the larger function of APL is to create programs that can be stored, named and carried out at a later time.

For this, APL allows you to define programs, a line at a time. The programs remain stored in the system as long as you want. Using the "Del" operator (∇), you tell the system that you want to put in a program. Del causes the terminal to help you along in various ways.

A nice feature is that you can <u>lock</u> your APL programs, that is, make them inaccessible and unreadable by others, whether they are programmers or not. In this case you define a program starting with the mystical sign <u>del-tilde</u> (⍫) instead of del (∇), and invoke the names of dark spirits.

APL, like BASIC, can be classed as an "algebraic" language-- but this one is built to please real mathematicians, with high-level stuff only they know about, like Inner and Outer Products.

Paradoxically, this makes APL terrific for teaching these deeper mathematical concepts, helping you see the consequences of operations and the underlying structure of mathematical things. Matrix algebra, for instance, can be visualized a lot better by working up to it with lesser concepts (like vectors and inner products) enacted on an APL terminal.

It would be really swell if someone would put together a tour-guide book of higher mathematics at the grade/highschool level for people with access to APL.

Interestingly, Alfred Bork (U. of Cal. at Irvine) is taking a similar approach to teaching physics, using APL as a fundamental language in his physics courses.

SNEAKY REPEATER STATEMENT IN APL?

One of the APL operators, "iota" (ι), seems to make its own program loop within a line. When used one-sided, it furnishes a series of ascending numbers up to the number it's operating on. This until the last one is reached.

<u>You type</u>: 3 x ι 7
<u>APL replies</u>: 3 6 9 12 15 18 21

In other words, one-sided iota looks to be doing its own little loop, increasing its starting number by 1, until it gets to the value on its right, and chugs on down the line with each.

Very sneaky way of doing a loop.

However! It isn't really looping, exactly. What the iota does is create a one-dimensional array, a row of integers from 1 up to the number on its right. This result is what then moves on leftward.

WHERE TO GET IT

IBM doesn't sell APL services. Their time-sharing APL is available, however, from various suppliers. Of course, that means you probably have to have an IBM-type terminal, unless you find a service that offers APL to the other kind-- an addition which seems to be becoming fashionable.

Usual charge is about ten bucks an hour connect charge, <u>plus</u> processing, which depends on what you're doing. It can easily run over $15 an hour, though, and more for heavy crunching or printout, so watch it.

The salesman will come to your house or office, verify that your terminal will work (or tell you where you can rent one), patiently show you how to sign on, <u>teach</u> you the language for maybe an hour if he's a nice guy, and proffer the contract.

→APL services are probably safer to sign onto, in terms of risked expenses, than most other time-sharing systems. (Though of course all time-sharing involves financial risk.) Because the system is restricted only and exactly to APL, you're not paying for capabilities you won't be using, or for massive disk storage (which you're not allowed in most APL services anyway), or for acres of core memory you might be tempted to fill.

→ In other words, APL is a comparatively straight proposition, and highly recommended if you have a lot of math or statistics you'd like to do on a fairly small number of cases. Also good for a variety of other things, though, including fun.

Different vendors offer interesting variations on IBM's basic APL\360 package, as noted below. In other words, they compete with each other in part by adding features to the basic APL\360 program, vying for your business. Each of the vendors listed also offers various programs <u>in</u> APL you can use interactively at an IBM-type terminal, in many cases using an ordinary typeball and not seeing the funny characters; though how clear and easy these programs are will vary.

And remember, of course, that you can do your own thing, or have others do it for you, using APL.

APL is also available on the PDP-10, and presumably other non-IBM big machines.

THE VENDORS

Scientific Time-Sharing Corporation (7316 Wisconsin Ave., Bethesda MD 20014) calls its version APL*PLUS. They'll send you a nice pocket card summarizing the commands.

APL*PLUS offers over twentyfive concentrators around the country, permitting local-call services in such metropolitan centers as Kalamazoo and Rochester. (Firms with offices in both cities, please note.)

They also have an "AUTOSTART" feature which permits the chaining of programs into grand complexes, so you don't have to call them all individually.

APL*PLUS charges the following for storage, if you can dig it: $10 PER MILLION BYTE-DAYS. (A byte is usually one character.) The census is probably taken once a day.

This firm also services ASCII terminals, which some people will consider to be a big help. That means you can have interactive users of APL programs at ASCII terminals, and that you can also program from the few APL terminals that aren't of the IBM type.

Time Sharing Resources, Inc. (777 Northern Blvd., Great Neck, N.Y. 11022) offers a lot of APL service, including text systems and various kinds of file handling, under the name TOTAL/APL.

Among the interesting features Time Sharing Resources, Inc. have added is an EXECUTE command, which allows an APL string entered at the keyboard in user on-line mode to be executed as straight APL. This is heavy.

Perhaps the most versatile-sounding APL service right now is offered by, of all people, a subsidiary of the American Can Company. American Information Services (American Lane, Greenwich CT 06830) calls their version VIRTUAL APL, meaning that it can run in "virtual memory"-- a popular misnomer for virtually unlimited memory-- and consequently the programmer is hardly subject to space limitations at all. Moreover, files on the AIS system are <u>compatible with other IBM languages</u>, so you can use APL to try things out quickly and then convert to Fortran, Cobol or whatever. (Or, conversely, a company may go from those other languages to APL without changing the way their files are stored on this service.) APL may indeed <u>intermix</u> with these other languages, how is unclear.

And the prices look especially good: $8.75 an hour connect, $15 a month minimum (actually their minimum disk space rental -- 1 IBM cylinder-- so for that amount you get a lot of storage). But remember there are still core charges, and $1 per thousand characters printed or transferred to storage.

In the West, a big vendor is Proprietary Computer Systems, Inc., Van Nuys, California.

TERMINALS

For an APL terminal, you might just want a 2741 from IBM (about a hundred a month, but on a year contract).

Or see the list under "Terminals" (p. 14), or ask your friendly APL company when you sign up.

Two more APL terminals, mentioned here instead of under "Terminals" for no special reason:

Tektronix offers one of its greenie graphics terminals (see flip side) for APL (the model 4013). This permits APL to draw pictures for you. It seems to be an ASCII-type unit.

Computer Devices, Inc. supposedly makes an an APL terminal using the nice NCR thermal printer, which is much faster and quieter than a mechanical typewriter. Spookier, though. And the special paper costs a lot of money.

BIBLIOGRAPHY

Iverson has a formal book. Ignore it unless you're a mathematician: Kenneth E. Iverson, <u>A Programming Language</u>. Wiley, 1962.

Paul Berry, <u>APL\360 Primer</u>, <u>Student Text</u>. Available "through IBM branch offices," or IBM Technical Publications Department, 112 East Post Road, White Plains, NY 10601. No IBM publication number on it, which is sort of odd. 1969.
→This is one of the most beautifully written, simple, clear computer manuals that is to be found. Such a statement may astound readers who have seen other IBM manuals, but it's true.

A.D. Falkoff and K.E. Iverson, <u>APL\360 Users' Manual</u>. Also available from IBM, no publication number.

POCKET CARDS (giving very compressed summaries) are available from both:
Scientific Time Sharing Corp. (see WHERE TO GET IT)
Technical Publications Dept., IBM, 112 East Post Road, White Plains, N.Y. 10601.
Ask for APL Reference Data card S210-0007-0. May cost a quarter or something.

Paul Berry, <u>APL\1130 Primer</u>. Adapted from 360 manual. Same pub. But for version of APL that runs on the IBM 1130 minicomputer.

Roy A. Sykes, "The Use and Misuse of APL." $2 from Scientific Time-Sharing Corp., 7316 Wisconsin Ave., Bethesda MD 20014.

A joker for you math freaks. Trenchard More, Jr., "Axioms and Theorems for a Theory of Arrays." <u>IBM Journal of Resch. & Devt.</u>, March 73, <u>135-157</u>. This is a high-level thing, a sort of massive set theory of APL, intended to make APL operators apply to arrays of arrays, and lead ultimately to the provability of programs.

"Get on Target with APL." A suggestive circular sales thingy. IBM G520-2439-0.

IBM has a videotaped course in APL by A.J. Rose. (Done 1968.)

⇨ What you really need to get started is Berry's Primer, Falkoff and Iverson's manual, and a pocket card. Plus of course the system and the friend to tutor you.

Power and simplicity do not often go together.

APL is an extremely powerful language for mathematics, physics, statistics, simulation and so on.

However, it is not exactly simple. It's not easy to debug. Indeed, APL programs are hard to understand because of their density.

And the APL language does not fit very well on minis.

APL is not just a programming language. It is also used by some people as a definition or description language, that is, a form of notation for stating how things work (laws of nature, algebraic systems, computers or whatever).

For instance, when IBM's 360 computer came out, Iverson and his friends did a very high-class article describing formally in APL just what 360s do (the machine's architecture). But of course this was even less comprehensible than the 360 programming manual.

Falkoff, A.D., K.E. Iverson and E.H. Sussenguth, "A Formal Description of System/360," <u>IBM Systems Journal</u>, v.3 no. 3, 1964.
The formal description in APL.

<u>IBM System/360 Operating System</u>: <u>Assembler Language</u>. Document Number C28-6514-X (where X is a number signifying the latest edition). IBM Technical Publications, White Plains New York.
The Manual.

DATA STRUCTURE: INFORMATION SETUPS

One of the commonest and most destructive myths about computers is the idea that they "only deal with numbers." This is TOTALLY FALSE. Not only is it a ghastly misunderstanding, but it is often an intentional misrepresentation, and as such, not only is it a misrepresentation but it is a damned lie, and anyone who tells it is using "mathematics" as a wet noodle to beat the reader with.

Computers deal with symbols and patterns.

Computers deal with symbols of any kind-- letters, musical notes, Chinese ideograms, arrows, ice cream flavors, and of course numbers. (Numbers come also in various flavors, simple and baroque. See chocolate box, p. 29.

Data structure means any symbols and patterns set up for use in a computer. It means what things are being taken into account by a computer program, and how these things are set up-- what symbols and arrangements are used to represent them.

The problem, obviously, is Representing The Information You Want Just The Way You Want It, in all its true complexities.

(This is often forbiddingly stated as "making a mathematical model"-- but that's usually in the rhetorical, far-fetched and astral sense in which all relations are "mathematical" and letters of the alphabet are considered to be a special distorted kind of number.)

Now it happens that there are many kinds of data structure, and they are interchangeable in intricate ways.

The same data, with all its relationships and intricacies, can be set up in a vast variety of arrangements and styles which are inside-out and upside-down versions of each other. The same thing (say, the serial number, 24965, of an automobile) may be represented in one data structure by a set of symbols (such as the decimal digits 2, 4, 9, 6, 5 in that order), and in another data structure by the position of something else (such as the 24965th name in a list of automobile owners registered with the manufacturer).

Furthermore, many different forms of data may be combined or twisted together in the same overall setup.

The data structure chosen goes a long way in imposing techniques and styles of operation on the program.

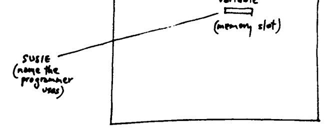
On the other hand, the computer language you use has a considerable effect upon the data structures you may choose. Languages tend to impose styles of handling information. The decision to program a given problem in a specific language, such as BASIC or COBOL or APL or TRAC Language, either locks you into specific types of data structure, or exerts considerable pressure to do it a certain way. In most cases you can't set it up just any way you want, but have to adjust to the language you are using-- although today's languages tend to allow more and more types of data.

Plainly, then, it is these overall structures that we really care about; but to understand overall structures, we need an idea of all the different forms of data that may be put in them.

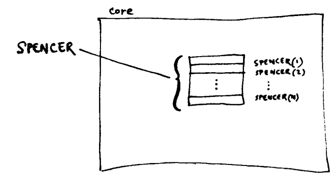
VARIABLES AND ARRAYS

The earliest data structures in computers, and still the predominating ones, are variables and arrays. (We met them earlier under BASIC, see pp. 16-17, and APL, see p. 22-5.)

A variable is a space or location in core memory. (For convenience, most programming languages allow the programmer to call a variable by a name, so that he doesn't have to keep track of its numerical address.)



An array (also called a table) is a section of core memory which the programmer cordons off for the program to put and manipulate data in. If SPENCER is the name of the array, then SPENCER(1) is the first memory slot in it, SPENCER(2) is the second, and so on up to however big it is.



(You can get a feel for how this ordinarily relates to input from outside-- see "How Data Comes, Goes, and Sits," nearby.)

The contents of a numerical field, or piece of data coming in, can simply be stuffed by the programmer into a variable.

The contents of a record, or unified set of fields, can get put into an array. The program can then pick into it for separate variables, if desired, or just leave them there to be worked on.

Then you twiddle your variables with your program as desired.

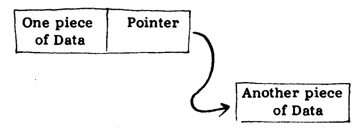
When you've done one record, you repeat. That's how lots of business programs go. Some other routine kinds, too.

FANCY STRUCTURES

Many forms of advanced programming are based on the idea that things don't have to be stored next to each other, or in any particular order.

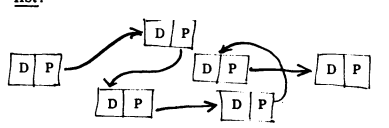
If things aren't next to each other, we need another way the program can tell how they belong together.

A pointer, then-- sometimes called a link-- is a piece of data that tells where another piece of data is, in some form of memory. Pointers often connect pieces of data.



A pointer can be an address in core memory; it can be an address on disk (diskpointer); it can point to a whole string of data, such as a name, when there is no way of knowing in advance how long the string may be (stringpointer).

A series of pieces of data which point to each other in a continuing sequence is called a threaded list.

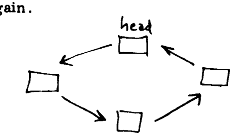


For this reason the handling of data held together by pointers-- even though it may make all sorts of different patterns-- is called list processing. (The (The term "list processing" might seem to go against common sense, as it might suggest something like, say, a laundry list, which is structured in a very simple blocklike form. But that's what we call it.)

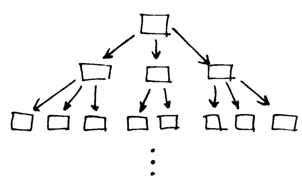
Prominent list-processing languages include SNOBOL, L^6 and LISP (see p. 31). There is argument as to whether TRAC Language is a list-processing language.

Here are some interesting structures that programmers create by list processing:

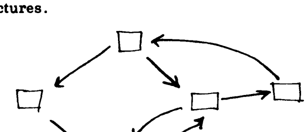
RINGS (or cycles). These are arrangements of pointers that go around in a circle to their first item again.



TREES. These are structures that fan out. (There are no rings in a tree structure, technically speaking.)



GRAPH STRUCTURES (sometimes called plexes). Here the word "graph" is not used in the ordinary way, to mean a diagrammatic sort of picture, but to mean any structure of connected points. Rings and trees are special cases of graph structures.



Graph structures can go any which way.

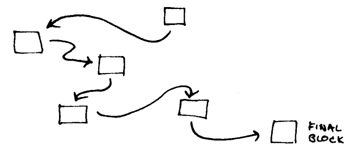
FAST-CHANGING DATA

One of the uses of such structures is in strange types of programs where the interconnections of information are changing quickly and unpredictably. Such operations happen fast in core memory. In this kind of programming (for which languages like LISP, SNOBOL and TRAC Language are especially convenient), the pointers are changed back and forth in core memory, every which way, all the time. Presumably according to the programmer's fiendish master plan-- if he's gotten the bugs out. (See Debugging, p. 30.)

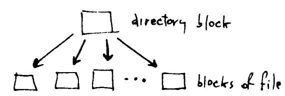
FANCY FILES

But these structures are not restricted to data in core memory. Complex and changeable files can be kept on disk in various ways by the same kind of threading (called "chaining" on mass storage).

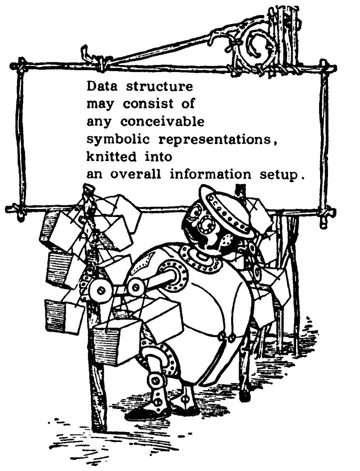
CHAINED FILE ON DISK



Another way of handling changeable files is through a so-called directory block, which keeps track of where all the other blocks are stored.



But these techniques, you see, may be used in both fast and slow operations, and for any purpose, so trying to categorize them tends not to be helpful. (Note also that these techniques work whether you're dealing with bits, or characters, or any other form of data.)



Note: By decent standards of English, the word data should be plural, datum singular. But the matter is too far gone: data is now utterly singular, like "corn" and "information," a granular collective which may be scooped, poured or counted.

But I draw the line at media. Media are many, "media" is plural!

A CLASSIC MISUNDERSTANDING

"Computers put everything into pigeonholes."

Wrong. People put things into pigeonholes. And designers of computer programs can set up lousy pigeonholes. If you let 'em. More sophisticated programming can often avoid pigeonholes entirely.

A Bit Is Not A Piece

People who want to feel With It occasionally use the term "bit" for any old chunk of information, like a name or address. This is Wrong. A Bit is the smallest piece of binary information, an item that can be one of two things, like heads or tails, X or O, one or zero; and all other information can be packed into a countable number of bits. (How many may depend on the data structure chosen.)

As a handy rule of thumb: every letter of the alphabet or punctuation mark is eight bits (see ASCII box); for heavy storage of everyday decimal numbers, every numerical digit can be further packed down (to four bits in BCD code).

A CONCRETE EXAMPLE. Suppose we want to represent the genealogy of the monarchs of England, so far as is known, in a computer data structure. NOTE THAT A DATA STRUCTURE IS DIFFERENT FROM A PROGRAM: if several programmers agree beforehand on a data structure, then they can go separate ways and each can write a program to do something different with it-- if they have really agreed on a complete and exact layout, which they may only think they've done.

First we consider the subject matter. Genealogy is conceptually simple to us, but as data is not as trivial as it might seem at first. Every person has two parents and a specific date of birth. Each pair of parents can have more than one child, and individual parents can at different times share parenthood with different other individuals.

Presumably we would like a data structure that allows a program to find out who was a given person's parent, who were a given person's children, what brothers and sisters each person had, and similar matters (so far as is known by historians-- another difficulty).

Note that just because it is simple to put this information in a wall chart, that does not mean it is simple to figure out an adequate data structure.

Note too, that any aspect of the data which is left out cannot then be handled by the program. What's not there is not there.

The easy way out is to use a language like, say, TRAC Language, and use its basic units (in this case, "forms") to make up a data structure whose individual sections would show parentage, dates, brothers and sisters and so on.

The braver approach is to try to set it up for something like FORTRAN or BASIC, languages which treat core memory more like a numerically-addressed array or block, as does rock-bottom machine language.

Let us assume that we have decided to use an array-type data structure, for instance to go with a program in the BASIC language on a 16-bit minicomputer. We do not have much room in core memory, so for each person in our data structure we are going to have to store a separate record on a disk memory, and call it into core memory as required.

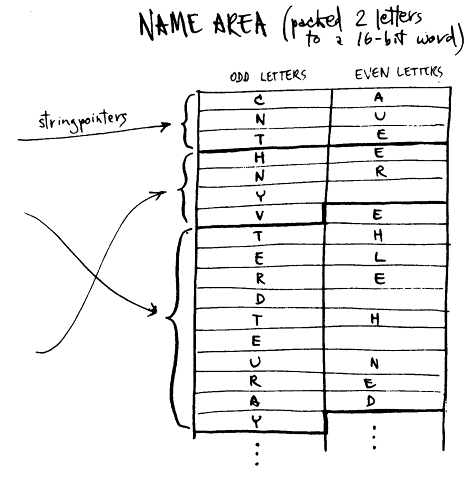
After much head-scratching, we might come up with something like the following. It is not a very good data structure. It is not a very good data structure on purpose.

It uses a block of 28 words, or 448 bits, per individual, not counting the length of his name, which is an additional 8 bits per character or space. However, this in itself is neither good nor bad. It's more than you might expect, but less than you might need.

(Incidentally, out of concern for storage space, some data fields are packed more than one to a 16-bit computer word. This is scornfully called bit-fiddling by computerfolk who work on big machines and don't have to worry about such matters.)

	1	monarch no. (if any)	sex — (1 bit)
individual's own	2	serial no.	
(name)	3,	stringpointer	→ (to name area)
	4	(two 16-bit words long)	
mother	5	serial no.	
father	6	serial no.	
brothers	7	serial no.	
(up to five)	8	⋮	
	9		
	10		
	11		
sisters	12	serial no.	
(up to five)	13	⋮	
	14		
	15		
	16		
date of 1st reign, if any	17	start (11 bits)	no. months
date of 2d reign, if any	18	start (11 bits)	no. months
female children,	19	serial no.	
up to five	20	⋮	
	21		
	22		
	23		
male children,	24	serial no.	
up to five	25	⋮	
	26		
	27		
	28		

As explained already, that was the basic block. We still have to keep the names somewhere, in a string area. Whether to keep this in core all the time, or on disk, is a decision we needn't go into here.



Here are some assumptions I have embodied in this data structure. That is, I had them in mind. (The parts you didn't have in mind are what get you later.)

- Parents and children of monarchs are included, as well as monarchs.
- All monarchs have a separate monarch number.
- No monarch reigned more than twice. (?)
- No monarch or parent of a monarch had more than five children of one sex. (Note the danger of these assumptions.)
- We are not interested in grandchildren of monarchs unless they are also monarchs, or siblings, or parents of monarchs.
- The information about the different people can be input in any order, as the years of reign can be stepped through by a program to find the order of reign.

If this seems like too much bother, that is in a way the point. Data structures must be thought out. Since computers have no intrinsic way of operating or of handling data (though particular languages will restrict you in particular ways), you will have to work all this out, and a carelessly chosen data structure will leave something out, or fail to distinguish among important differences, or otherwise have its revenge.

(For instance, if you haven't noticed yet: we left out legitimacy. For many purposes we want to know which kings were bastards.)

(Self-test: is five bits long enough to express the greatest number of months any English monarch reigned? -- see "Binary Patterns." Or do we have to fix this data structure on that score also?)

To give you a sense of the sort of program this data structure allows:

A program to ascertain how many kings were the sons of kings would look at each entry that had a monarch number, test whether the monarch was male, and if male, would look at the male parent's serial number. Then it would look up that parent's entry, and see whether it in turn had a monarch number, and if so, add one to the count it was making. Then it would go back to the entry it had been looking at, and step on to the one after that.

This is actually a pretty lousy data structure. The clumsiness of this approach to such data-- and you are welcome to think of a better one-- shows some of the difficulties of handling complex data about the real world. Things like lengths of names and numbers of relatives produce great irregularities, but make these kinds of data no less worth of our attention.

We could add lots of things to our data structure (and so make it more unwieldy). For instance, we might want to mark each serial number specially if it referred to someone who was the offspring of a monarch. We could simply set a particular bit to 1 in the serial number for them (called a flag or tag). We could also flag dates and genealogies that are regarded as uncertain. There is no limit to the exactness and complexity with which information may be represented. But doing it right can, as always, be troublesome.

A lot of computer people want to avoid dealing with complex data; perhaps you can begin to see why. But we must deal with the true complexities of information; therefore languages and systems that allow complex information structures must become better-known and easier to use.

THE FRONTIER: COMPLEX FILE STRUCTURE

The arrangements of whole files-- groups of records or other info chunks-- are up to the programmer. The structure of files is called, not surprisingly, file structure, and it is up to the programmer to decide how his files should be arranged.

Habits die hard. The notion of sequence-- even false, imposed sequence-- is deep in the racial unconscious of computer people. An interesting concrete term shows this nicely. Because computer people often think any file should have a basic sequence, they use the term inverted file for a file that has been changed from its basic sequence to another sequence. But increasingly, all the sequences are false and artificial. Where now are inverted files? All files are inverted if they're anything.

Fortunately, the final frontier of data structure is now increasingly recognized as the control of complex storage of files on disk memory. The latest fancy term for this is data base system, meaning planned-out overall storage that you can send your programs to like messengers.

The fact that IBM now has moved into this area (with its intricate "access methods" and all their initials) means complex storage control has finally arrived, although the pioneering work was done by Bachman at GE some years ago (see bibliography). Till the last few years, external storage, with pointers and everything, has not been conveniently under the programmer's control except in crude ways. Finally we are seeing systems beginning to get around that automatically handle complex file structures in versatile ways that programmers can use more easily.

data
damyata
dhayadvam
— T.S. Eliot, The Waste Land

" There is a growing feeling that data processing people would benefit if they were to accept a radically new point of view, one that would liberate the application programmer's thinking from the centralism of core storage and allow him the freedom to act as a navigator within a database. ... This reorientation will cause as much anguish among programmers as the heliocentric theory did among ancient astronomers and theologians."

Charles W. Bachman
(piece cited in Bibliography)

Remember the song that had a pointer data structure?

(in alphabetical order)

ANKLE BONE ←
BACK BONE ←
FOOT BONE ■
→ HEAD BONE
HIP BONE ←
KNEE BONE ←
NECK BONE ←
SHIN BONE ←
SHOULDER BONE ←
THIGH BONE ←

BIBLIOGRAPHY

Malcolm C. Harrison, Data-Structures and Programming. Scott, Foresman, 1973.

→This book can be recommended to ambitious beginners. It has useful summaries of different languages, as well as fundamental treatment of data structures as they intertwine with specific languages.

An obscure and intricate study of the interchangeability of data structures-- how they fundamentally interconvert-- has been the longtime research of one Anatol Holt, who calls his work Mem-Theory. Mem is from memory, and also, conveniently, a Hebrew letter.

This is an extremely ambitious study, as it in principle embraces not just much or all of computer science, but perhaps mathematics itself. Math freaks attention: Holt has said he intended to derive all of symbolic logic and mathematics from relations and pointer structures. Let's hear it for turning Russell on his head.

I don't know if Holt has published anything on it in the open literature or not.

However, he does have a game available which seems weirdly to embody these principles. The game of Mem is available for $6.50 postpaid ($6.86 to Pennsylvanians) from Stelledar, Inc., 1700 Walnut St., Phila. PA 19103. It has beautifully colored pieces, looks deceptively simple, and is unlike anything, except discrete abstractive thinking itself. Recommended.

Charles W. Bachman, "The Programmer as Navigator." CACM Nov 1973.

Bachman was the prime mover in the development of large linked disk data systems at General Electric; he is the Pioneer. This is about big n-dimensional stuff.

David Lefkovitz, File Structures for On-Line Systems. Spartan-Hayden Books, $12.

Alfonso F. Cardenas, "Evaluation of File Organization-- a Model and System." CACM Sep 73, 540-548. Not surprisingly, it turns out that different file organizations have different advantages.

Edgar H. Sibley and Robert W. Taylor, "A Data Definition and Mapping Language." CACM Dec 73, 750-759.

Example of current sophisticated approaches: a whole language for nailing the data just the way it should be. Has helpful further citations.

"ASCII and ye shall receive."
— the Industry

SOMETIMES IT JUST SITS THERE SOMETIMES IT COMES AND GOES.

Data usually has to be marshalled into rows, or even regiments and battalions, before it can go into a computer.

(Some people just get their data into a computer by sitting at a terminal and typing it in, perhaps answering questions typed to them by a front-end program. But they're the lucky ones. Most of us have to get the data set up on some kind of holding surface before it gets fed in. That's an input medium.)

DATA MEDIA

A data medium ("medium" is the singular of "media") is anything that holds the marks of data outside the core memory of a computer. Thus punched cards and punched paper tape may be used as input media, used for putting information into a computer. (Each medium needs a corresponding input or output device, to whisk across the surface and translate its marks or holes into the corresponding electronic pulses.)

There are three types of data media: input, output and storage media. An input medium carries the data in. An output medium receives the results of a program; for instance, a sheet of paper coming out of a printing device is an output medium, as is a punched card or punched paper tape.

Storage media are output media that may be used as input media later on. Thus punched cards and punched paper tape can be storage media. But the better storage media use magnetic recording (which is faster and less bulky), like magnetic tape and disks, or just plain "disks" as we generally call them. (See fuller list of mag media under "Peripherals," p. 57.)

The units and arrangements of data used for input, output and storage are in principle not necessarily the true ones of the data structure used by the program. The blocks and records of storage, for instance, may have irregular data with pointers sitting in them. (Unfortunately there is some carryover, in that programmers are tempted to use data structures which are easy to store and run in and out, rather than handling the true complexities of the subject. This is always a temptation.)

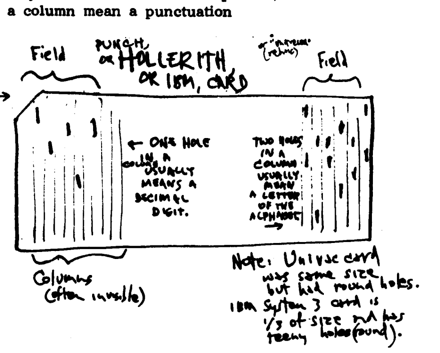
Let us consider the units and arrangements of data used for input and output and storage. These are, respectively, fields, records, files and blocks.

THE PUNCH CARD

Let's begin with a fun example: that hoary old medium for input and output, the punched (or "punch") card. The punch card will show us what a field is.

The punch card is generally believed to have been invented by Herman Hollerith (although the author's in-laws had bitter recollections to the contrary). It was first used on a broad scale to count up the census of 1890, and later became an early cornerstone of IBM, but that's another story.

The punches on a card represent a row of information (such as a row of typed letters). this is not obvious because the card is a rectangle rather than a line. However, the length of the card is actually divided into eighty positions, each of which may hold one number, alphabetic character or punctuation mark. These positions are actually narrow columns, eighty of them, with different positions in which holes may be punched. One hole in a column represents a numeral; which position in the column specifies what number. Two holes in a column generally mean a letter of the alphabet, three holes in a column mean a punctuation mark.



Data is punched into cards according to some plan associated with the program.

Beyond those simple matters there is no preordained arrangement for information on a punch card; it all depends on what the program calls for. But each separate piece or section of information-- each bunch of consecutive characters that together have a specific meaning -- are called a field.

A field can be a name, a number, an amount of money, an alphabetical code representing something, a numerical code representing something, or other stuff. When the cards go into the program, the program can pick off the information it needs one field at a time-- putting the field in columns 1 to 17 into one program variable, the field from columns nine to ten into another program variable, and so on.

The punch card is an important example of an input unit influencing the structure of computer programs. It is convenient to use fields on a punch card as the basic data structure of a program and say, "That's the way it has to be for the computer. In the worst cases we see the workings of the "punch card mentality" or "80-column mind" (see box).

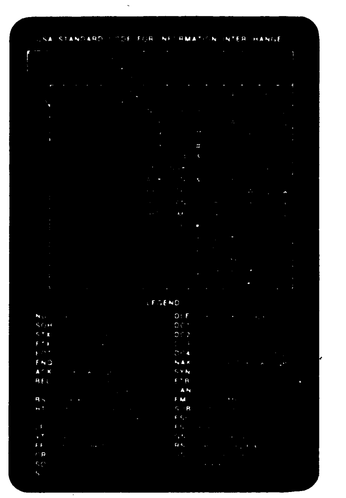
→People will often thrust a punched data card at you and ask, "What does this mean?" Who knows? It may have lettering banged along the top, showing what characters the holes represent, but if these characters don't show anything understandable, such as the person's name, you're in the dark. The card may have preprinted section lines dividing it up, but these are rarely self-explanatory. It's often impossible just to look at a punched card and tell by eye what the individual fields are for, or even where they begin and end; all that depends on the program. Only someone who understands the program, or at least knows what fields the card is divided into and what the characters represent there, can help.

Sometimes, in dismal systems we encounter day-to-day-- like for university registration -- a punch card will have a person's name in the first few columns, or worse, a personal serial number. Other information continues from there. These may or may not be recognizable, either from reading the holes by eye, or from designations pre-printed on the card.

"ASCII not, what your computer will do for you."
— IBM

ASCII code. You can figure out from the table the bit pattern for any letter, or what any given combination of seven bits means.

Example. Find the capital letter G in the table. For the first three bits of the code, look at the top of the column: 100. For the next four, look sideways to the left: 0111. So G is: 1000111.



(An eighth bit is used as a check on the number of ones in the code; this is called the parity bit, and either rounds to an even number of bits (even parity) or an odd number of bits (odd parity). Thus if a code comes through to the computer with a wrong number of ones, the computer can take remedial action.)

Those funny multiletter codes are for controlling terminals and like that.

Pocket card courtesy of Computer Transceiver Systems, Inc.

MAGNETIC STORAGE

The same principle of fields applies in other data media, especially magnetic tape and disk. We may extend the notion of a field to explain records and files.

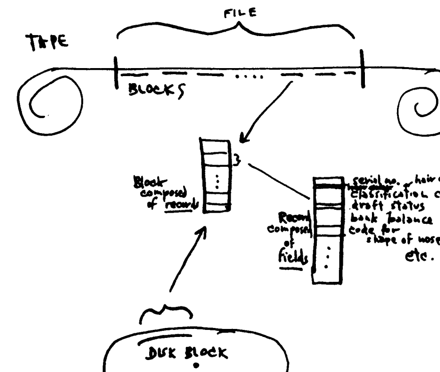
A field, generally speaking, is a section of positions on some medium reserved for one particular piece of information, or the data in it.

A record is a bunch of fields stored on some medium which have some organized use. (For instance, the accounting information held by an electric utility company about a particular customer is likely to be stored as a record with at least these fields: account number; last name; initials; address; amount currently owed.)

A file is a whole big complete bunch of information that is stored someplace. In many applications a file is composed of numerous similar, consecutive records. For instance, an electric company may well store the records for all of its customers on a magnetic tape, ordered by account number (account 000001 first).

Storing sequences of similar records in long files is typical of business programs, though perhaps this should begin to change. It's especially suited to batch processing, that is, handling many records in the same way at the same time. (See "System Programs.")

Now, the divisions of field, record and file are conceptual: they are what the programmer thinks about, based on the information needs of a specific computer program.



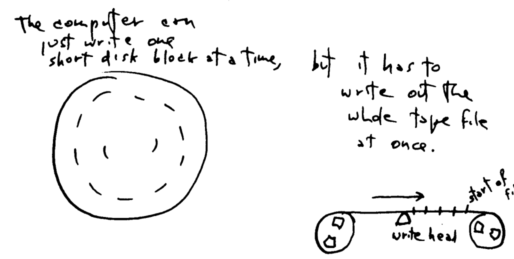
BLOCKS

A block is something else, which may be related only to quirks of the situation.

A block is a section of stored material, divided either according to the divisions of the data or peculiarities of the device holding it, such as a disk drive. Short records may be stored many to a block. If records are long they may be made up of many blocks.

→In particular, tape blocks can be almost any size, while disk blocks often have a certain fixed size (number of characters or bits) based on the peculiarities of the individual device. (This can be a pain in the neck.)

On the other hand, due to the quirks of magnetic recording, your program usually can't just change something in the middle of a block; the whole disk block or tape file has to be replaced. This is less trouble with a short disk block than a long tape file.



TRADITIONAL CONVEYER-BELT PROGRAMS

Many traditional business programs are of this type, reading in one data record at a time, doing something to it (such as noting that an individual has paid the exact amount of his gas) and writing out a new record for that customer on the current month's tape.

THE PROBLEM

Standardized fields, blocks and records are often necessary or convenient. But, on the other hand, the kinds of computer programs people find oppressive often have their roots in this kind of data storage and its associated styles of programming, especially the use of fixed-field records as the be-all and end-all. The more interesting uses of the computer (interactive, obliging, artistic, etc.) use a greater variety of data structures.

People's naive idea of "programming" is often a reasonable approximation to the notion of "data structure." Data structure is how information is set up. After it's set up, programs can twiddle it; but the twiddling options are based on how the information is set up to begin with.

THE MAGIC OF DATA

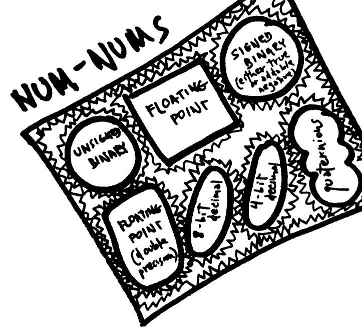
How does a computer program print something out on a printing machine? It sends the code for each letter out to the printing machine.

How does a computer program respond to something a user types in? It compares the codes that come in from the letters he types with a series of codes in memory, and when it finds a match between letters, numbers, words or phrases, branches to the corresponding action.

How does a computer program measure something? It takes in numerical codes from a device which has already made the measurements and converted them to codes.

DOES NOT COMPUTE!

Some TV writer's idea of a computer announces this when data are insufficient or contradictory. Ho hum.



CODED-DOWN DATA: AN IDEA WHOSE TIME HAS PASSED

Codes are patterns or symbols which are assigned meanings. Sometimes we make up special codes to cut down the amount of information that has to be stored. On your driver's license, for instance, they may reduce your hair color to one decimal digit (four bits of information), since there are less than nine possibilities for quick identification of hair-color anyway.

Obviously, codes can be any darn thing: any set of symbols that is less than what you started with. But by compressing information they lose information, so that subtleties disappear (consider the use of letters A to F to grade students). When you divide a continuum into categories, not just the fewness of the categories, but the places you draw the line-- called "breaks" or "cutting-points"-- present problems. Such chopping frequently blurs out important distinctions. Coding is always arbitrary, frequently destructive and stupid.

Lots of ways now exist to handle written information by computer. These often present better ways to operate than by using codes of this type. But many computer programmers prefer to make you use codes.

(NOTE: there are two other senses of "code" used hereabouts: 1) the binary patterns made to stand for any information, especially on input and output; 2) what computer programs consist of, that is, lines of commands.)

SOME POINTS

"Logical deduction" really consists of techniques for finding out what's already in a data structure.

"Logical inconsistency" means a data structure contradicts itself. Rarely does it happen that a computer helps you discover something new about a subject that you didn't suspect or see coming without the computer; after all, you have to set up a study in such a way as to make room to find things out, and you can only make room to find some things out.

THE PUNCH CARD MENTALITY

Punch cards are not intrinsically evil. They have served many useful purposes. But the punch-card mentality is still around. This will be seen in the programmer who habitually sets things up so we have to use punch cards (when other media, or interactive terminals, would be better); who insists on the user or victim putting down numbers (when with a little more effort the program could handle text, which is easier for the human, or even look up the information in data it has already); who insists that people's last names be cut down to eleven letters because he doesn't feel like leaving a longer field or handling exceptions in his program; who insists on the outsider cutting his information into snarfy little codes, when such digestion, if needed at all, could be better done by the program; and so on.

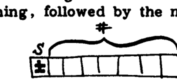
The punch card mentality is responsible for many of the woes that have been blamed on "computers."

IF YOU WANT NUMBERS, WE GOT 'EM

The basic kinds of number operations wired into all computers are few: just add (and sometimes subtract) binary numbers. However, up above the minicomputer range, a computer may have multiply, divide, and more. Fancier computers offer more types and operations on them.

PLAIN BINARY-- Very important for counting. Represents numbers as patterns of 1's and 0's (or X's and Ohs, if you prefer). How to handle negative numbers? Two ways:

TRUE NEGATIVE-- binary number with a sign bit at the beginning, followed by the number.

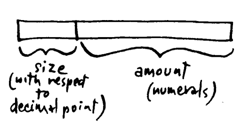


Trouble is, the arithmetic is harder to wire for this kind, because there are two zeroes (plus and minus) between 1 and -1.

ADDABLE NEGATIVE-- this system does a sort of flip and begins a negative number with all ones. It means that the machine doesn't have to have subtraction circuitry: you just add the flipped negative version of a number, and that actually subtracts it. This has now caught on generally. (It's usually called "twos complement negative," which has some obscure mathematical meaning.)

BCD (Binary-Coded Decimal)-- the accountant's numbering system. Used by COBOL (see p. 31). It's plain old decimal, with every numeral stored in four bits; the machine or language has to add them one numeral at a time, instead of crunching together full binary words.

FLOATING POINT-- the scientist's number technique for anything that may not come out even. Expresses any quantity as an amount and a size.



The "amount" part contains the actual binary numerals, the "size" is the number of places in front of or after the decimal point that the number starts. Very important for astronomical and infinitesimal matters, since a floating-point number can be bigger, say, than

9,876,543,210,000

or smaller than

.00000001234567

For some people even this isn't precise enough, so they program up "infinite precision arithmetic," which carries out arithmetic to as many places as they want. It takes much longer, though.

WHAT'S AVAILABLE IN MACHINES AND LANGUAGES

Some machines, like the 360, are more-or-less wired up to handle several number types: binary, floating point, BCD. Little machines usually only have plain binary, so other types have to be handled by programs built up from that fundamental binary.

Languages make up for this by providing programs to handle numbers in some or all of these formats. There are languages that offer even more kinds of numbers--

IMAGINARY numbers
(two-part numbers following certain rules)
QUATERNIONS
(like Imaginary numbers but worse)
and goodness knows what else.

On the other hand, some languages restrict what number facilities are available for simplicity's sake. BASIC, for instance, doesn't distinguish between integers (counting numbers) and those with decimal points; all numbers may have decimal points. TRAC Language only gives you integers to start, since it's easy enough to program other kinds of number behavior in (like infinite precision).

For historical reasons computers have been used mostly with numbers up to now; but that is going to be thoroughly turned around. Within a few years there may be more text-- written prose and poetry-- stored on computers than numbers.

During the recent massive lawsuit by Control Data against IBM, it was revealed that IBM had an awesome number of letters and communications stored on magnetic memory.

When I lived in New York, I had a driver's license with the staggering serial number

NO 5443 12903 3-4121-37

Now it may very well be, as in some serial numbers, that information is hidden in the number that Insiders can dope out, like my criminal record or automobile accidents, if any. (N is my initial, and two of the digits show my date of birth, a handy check against alteration by thirsty minors. But the rest of it is ridiculous.) The fact that that leaves 15 more decimal digits means (if no other codes are hidden) that New York State has provision in their license numbering for up to 999,999,999,999,999 inhabitants. It is doubtful that there will ever be that many New Yorkers, or indeed that many human beings while the species endures.

In other words, either New York State is planning on having many, many more occupants, or an awfully inefficient code has been adopted, meaning a lot of memory space is wasted holding those silly big numbers for millions of drivers. However, that doesn't represent a lot of money. 10 million decimal spaces these days fits on a couple of disk drives. But it's an awful pain in the neck when you want to cash a check.

INPUT AND OUTPUT CODES

Data has to get inside the machine somehow, and results have to get back out. Two main types of codes-- that is, standardized patterns-- exist, although what forms of data programs work on inside varies considerably. (The input data can be completely transformed before internal work starts.)

1. ASCII (pronounced "Askey," American Standard Code for Information Exchange. This allows all the kinds of numbers and alphabets you could possibly want (for instance, Swahili) for getting information in and out of computers.

ASCII is used to and from most Teletype terminals and keyscopes.

However, ASCII is also used for internal storage of alphabetical data in many non-IBM systems, andit is also the running form of a number of programming languages, such as TRAC language (see p. 18), TECO ~~(see p.)~~, and GRASS (see p. 31).

IBM's deliberate undermining of the ASCII code is a source of widespread anger. (See IBM, p. 52.)

2. EBCDIC (pronounced "Ebsadick,") Extended Binary Coded Decimal. This was the code IBM brought out with the 360, passing ASCII by. (IBM seems to think of compatibility as a privilege that must be earned, i.e., paid for.) EBCDIC also allows numbers, the English alphabet, and various punctuation marks. This is used to and from most IBM terminals ("2741 type").

And Also:

HOLLERITH, meaning the column patterns that go in on punched cards. (They can also come out that way, if you want them to.)

CARD-IMAGE BINARY. If for some reason you want exact binary patterns from your program, they can be punched out as rows or columns on punch cards.

STERLING. Just to show you how comical things can get, the original PL/I specifications (see p. 31) allowed numbers to be input and output in terms of Pounds, Shillings and Pence (12 pence to the shilling, 20 shillings to the pound). No provision was made for Guineas (the 21-shilling unit), or farthings, unfortunately.

MAGIC LANGUAGES

A computer language is a system for casting spells. This is not a metaphor but an exactly true statement. Each language has a vocabulary of commands, that is, different orders you can give that are fundamental to the language, and a syntax, that is, rules about how to give the commands right, and how you may fit them together and entwine them.

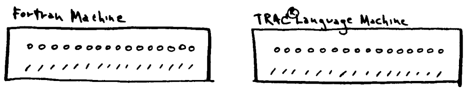
Learning to work with one language doesn't mean you've learned another. You learn them one at a time, but after some experience it gets easier.

There are computer languages for testing rocketships and controlling oil refineries and making pictures. There are computer languages for sociological statistics and designing automobiles. And there are computer languages which will do any of these things, and more, but with more difficulty because they have no purpose built in. (But each of these general-purpose languages tends to have its own outlook.)

Most programmers have a favorite language or two, and this is not a rational matter. There are many different computer languages-- in fact thousands-- but what they all have in common is acting on series of instructions. Beyond that, every language is different. So for each language, the questions are

WHAT ARE THE INSTRUCTIONS?
and
HOW DO THEY FIT TOGETHER?

Most computer languages involve somehow typing in the commands of your spell to a computer set up for that language. (The computer is set up by putting in a bigger program, called the processor for that language.)



(Same computer with different language processors loaded in its core memory)

Then, after various steps, you get to try your program.

Once you know a language you can cast spells in it; but that doesn't mean it's easy. A spell cast in a computer language will make the computer do what you want--

IF it's possible to do it
with that computer;
IF it's possible to do it
in that language;
IF you used the vocabulary
and rules of the language
correctly;
and IF you laid out in the spell
a plan that would effectively
do what you had in mind.

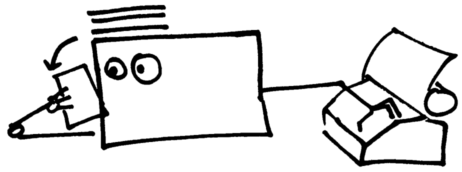
BUT if you make a mistake in casting your spell, that is a BUG. (As you see from the IFs above, many types of bug are possible.) Program bugs can cause unfortunate results. (Supposedly a big NASA rocket failed in takeoff once because of a misplaced dollar sign in a program.) Getting the bugs out of a program is called debugging. It's very hard.

DESIGNING COMPUTER LANGUAGES

Every programmer who's designed a language, and created a processor for it, had certain typical uses in mind. If you want to create your own language, you figure out what sorts of operations you would like to have be basic in it, and how you would like it all to fit together so as to allow the variations you have in mind. Then you program your processor (which is usually very hard).

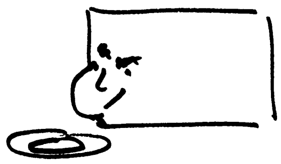
★AN INTERPRETER★

carries out each instruction
as it's encountered.



★ A COMPILER★

chews the instructions
of the language
into another form
to be processed later.



An Interpreter carries out,

A Compiler sets up.

HOW DO COMPUTER LANGUAGES WORK?

Basically there are two different methods.

A compiling language, such as FORTRAN or COBOL, has a compiler program, which sits in the computer, and receives the input program, or "source program," the way the assembler does. It analyzes the source program and substitutes for it an object program, in machine language, which is a translation of the source program, and can actually be run on the computer. The relation of the higher language is not one-to-one to machine language: many instructions in machine language are often needed to compile a single instruction of the source program. (A source program of 100 lines can easily come out a thousand lines long in its output version.) Moreover, because of the interdependency of the instructions in the source program, the compiler usually has to check various arrangements all over the program before it can generate the final code.

Most compilers come in several stages. You have to put the first stage of the compiler into the computer, then run in the source program, and the first stage puts out a first intermediate version of the program. Then you put this version into a second stage, which puts out a second intermediate version; and so on through various stages. This is done fairly automatically on big computers, but on little machines it's a pain.

(In fact, compilers tend to be very slow programs; but that depends on the amount of "optimizing" they do, that is, how efficient they try to make the object program.)

An interpretive language works differently. There sits in core a processor for the language called an interpreter; this goes through the program one step at a time, actually carrying out each operation in the list and going on to the next. TRAC and APL are interpretive; it's a good way to do quickie languages.

Interpreters are perhaps the easier method of the two to grasp, since they seem to correspond a little better to the way many people think of computers. That doesn't mean they're better. For programs that have to be run over and over, compiling is usually more economical in the long run; but for programs that have to be repeatedly changed, interpreters are often simpler to work with.

A BLACK ART

Making language processors, especially compilers, is widely regarded as a black art. Some people have tricks that are virtual trademarks (see below).

Actually, the design of a language-- especially the syntax, how its commands fit together-- strongly influences the design of its processor. BASIC and APL, for instance, work left-to-right on each line, and top-to-bottom on a program. Both act on something stored in a work area. TRAC, on the other hand, works left-to-right on a text string that changes size like a rubber band. Other languages exhibit comparable differences.

MIXED CASES AND VARIATIONS (for the whimsical)

There are a lot of mixed cases. A load-and-go compiler (such as WATFOR) is put into the computer with the program, compiles it, and then starts it going immediately. An interpretive compiler looks up what to do with a given instruction by interpreting it into a series of steps, but compiling them instead of carrying them out. (A firm called Digitek is well known for making very good compilers of this type.) An incremental compiler just runs along compiling a command at a time; this can be a lot faster but has drawbacks.

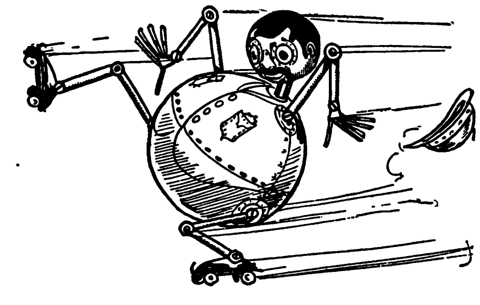
BIBLIOGRAPHY.

David Gries, Compiler Construction for Digital Computers.
Not for beginners, but a beautiful book. Good on abstract theory of languages, too.

DEBUGGING

A program is like a nose;
Sometimes it runs, sometimes it blows.

Attributed to Howard Rose.
(Datamation, 1 Sep 71, 33.)



According to the grapevine...

a prestigious Southern university
had a program
where the number of months
was carelessly set to 10
(as a dimension in an array).
In November,
nobody got their checks
till this error was found.

candid photos

Debugging means changing and fixing your program till it works the way you want it to.

This is the part of programming people like the least.

You run your program and then try to find out what went wrong. It could be a mistake in the basic thinking ("logic error"), or a clerical error in the particular choice of commands to carry out a well-thought-out process ("coding error").

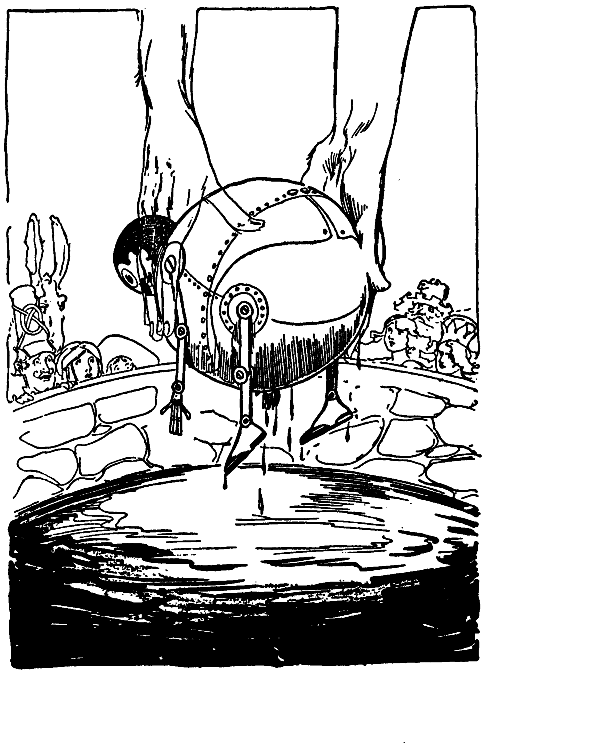
Some systems allow you to debug interactively, from a terminal. This helps a lot. You can run parts of your program, get it to stop at certain points to let you look around, and so on.

No program is ever fully debugged.

-- folk saying

For every bug that goes out,
two more bugs go in.

-- folk saying



THE GREAT COMPUTER LANGUAGES

A certain number of computer languages are very widely accepted and used; I list them here. If you want to learn any of them, I believe that Daniel McCracken has written a manual on every one of them. (Not the variants listed, though.)

Why their names are always spelled with capital letters I don't know. (Generally they get let down in longer articles, though.)

Good Old FORTRAN

FORTRAN was created in the late fifties, largely by John Backus, as an algebraic programming system for the old IBM 704. (However, the usual story is that it stands for FORmula TRANslator.)

Fortran is "algebraic," that is, it uses an algebraic sort of notation and was mostly suited, in the beginning, to writing programs that carried out the sorts of formulas that you use in highschool algebra. It's strong on numbers carried to a lot of decimal places ("scientific" numbers) and the handling of arrays, which is something else mathematicians and engineers do a lot (see Arrays under BASIC).

Fortran has grown and grown, however; after Fortran I came Fortran II, Fortran III and Fortran IV; as well as a lot of variants like Fortran Pi ("irrational, and somewhere between III and IV"), WATFOR and WATFIV.

The larger Fortrans-- that is, language processors that run on the bigger computers-- now have many operations not contemplated in the original Fortran, including operations for handling text and so on.

BASIC, presented earlier, is in some respects a simplified version of Fortran.

ALGOL LOST, AND PL/I

ALGOL is considered by many to be one of the best "scientific" languages; it has been widely accepted in Europe, and is the standard "publication language" in which procedures for doing things are published in this country. It is different from FORTRAN in many ways, but a key respect is this: while in FORTRAN the programmer must lay out at the beginning of his program exactly what spaces of core memory are to have what names, in ALGOL the spaces in core memory are not given names except within subsections of the program, or "procedures." When the program follower gets to a specific procedure, then the language processor names the spaces in core memory.

This has several advantages. One is that it can be used for so-called "recursive" programs, or programs that call new versions of themselves into operation. I guess we better not get into that. But mathematicians like it.

Originally this language was called IAL, for International Algebraic Language, but then as it grew and got polished by various international committees it was given its new name. (I don't know if anyone consciously named it after Algol, the star.)

It has gone through several versions. Algol 62, the publication language, is one thing; Algol 70, the 1970 version, is much more complicated and strange.

Several versions of ALGOL have gotten popular in this country. One, developed at the University of Michigan, is called MAD (Michigan Algorithm Decoder); its symbol is of course Alfred E. Newman. Another favorite (for its name, anyway) is JOVIAL (Jules' Own Version of the International Algebraic Language), developed under Jules Schwartz (and supposedly named without his consultation) at System Development Corporation.

When IBM announced its System 360 back in 1964, there had been hope that they would support the international language committees and make Algol the basic language of their new computer line. No such luck. Instead they announced PL/I (Programming Language I), a computer language that was going to be all things to all men.

In programming style it resembled COBOL, but had facilities for varieties of "scientific" numbers and some good data structure systems. It is available for the 360 and for certain big Honeywell computers; indeed, the operating system for MULTICS (see p. 45) was written in PL/I. Whether there are people who love the language I don't know; there are certainly people who hate it.

```
// JOB 1 INUSEINUSE   01 MAY 73 02.360 HRS
// ACT *****    01 MAY 73 02.360 HRS
// FOR PHONE   01 MAY 73 02.360 HRS
*IOCS(1443 PRINTER,KEYBOARD)
*LIST ALL
*ONE WORD INTEGERS
C     ALPHABETIC PHONE NUMBER PROGRAM
      DIMENSION J(7),NUMB(7),LETT(3,10),LINE(132),IN(80),NUM(10)
      DATA NW/3/,NR/10/,INL/80/,LINL/120/
      DATA IBL/' '/
      DATA LETT/'0','0','0','1','1','1','A','B','C','D','E','F','G','H',
     1 'I','J','K','L','M','N','O','P','R','S','T','U','V','W','X','Y'/
      DATA NUM/'0','1','2','3','4','5','6','7','8','9'/
      INL2=INL+1
C     READ PHONE NUMBER AND TITLE LINE
   10 READ(NR,11)(IN(I),I=1,INL)
   11 FORMAT(80A1)
C     FIND LAST NON-BLANK IN TITLE LINE
      DO 12 I=1,INL
      K=INL2-I
      IF(IN(K)-IBL)13,12,13
   12 CONTINUE
C     WRITE TITLE LINE
C     COMPUTE STARTING POINT ON PRINT LINE
   13 LINX=(K/8+1)*8
      WRITE(NW,14)(IN(I),I=1,K)
   14 FORMAT('1',80A1)
C     CONVERT ALPHA PHONE NUMBER TO NUMERIC
      DO 16 I=1,7
      IN1=IN(I)
      DO 15 K=1,10
      IF(IN1-NUM(K))15,16,15
   15 CONTINUE
   16 NUMB(I)=K-1
C     BLANK PRINT LINE
      DO 1 I1=1,LINL
    1 LINE(I1)=IBL
      DO 6 I1=1,3
      K=NUMB(1)+1
      J(1)=LETT(I1,K)
      DO 6 I2=1,3
      K=NUMB(2)+1
      J(2)=LETT(I2,K)
      DO 6 I3=1,3
      K=NUMB(3)+1
      J(3)=LETT(I3,K)
      DO 6 I4=1,3
      K=NUMB(4)+1
      J(4)=LETT(I4,K)
      DO 6 I5=1,3
      K=NUMB(5)+1
      J(5)=LETT(I5,K)
      DO 6 I6=1,3
      K=NUMB(6)+1
      J(6)=LETT(I6,K)
      DO 6 I7=1,3
      K=NUMB(7)+1
      J(7)=LETT(I7,K)
      IF(LINX+7-LINL)4,4,2

                                                  01 MAY 73
                                                   PAGE 002

    2 WRITE(NW,3)(LINE(L),L=1,LINX)
    3 FORMAT(1X,132A1)
      LINX=0
    4 DO 5 K=1,7
      LINX=LINX+1
    5 LINE(LINX)=J(K)
    6 LINX=LINX+1
      IF(LINX)8,8,7
    7 WRITE(NW,3)(LINE(L),L=1,LINX)
      GO TO 10
    8 CALL EXIT
      END

VARIABLE ALLOCATIONS
   J(I )=0006-0000  NUMB(I )=000D-0007  LETT(I )=002B-000E  LINE(I )=00AF-002C   IN(I )=00FF-00B0   NUM(I )=0109-0100
 INL2(I )=010A        INL(I )=010B        NR(I )=010C          I(I )=010D          K(I )=010E       IBL(I )=010F
 LINX(I )=0110         NW(I )=0111       IN1(I )=0112         I1(I )=0113       LINL(I )=0114        I2(I )=0115
   I3(I )=0116         I4(I )=0117        I5(I )=0118         I6(I )=0119         I7(I )=011A         L(I )=011B

STATEMENT ALLOCATIONS
 11    =0124  14    =0127  3     =012C  10    =0136  12    =0162  13    =016B  15    =01AA  16    =01B3  1     =01CB  2     =0294
 4     =02AF  5     =02B9  6     =02D2  7     =031B  8     =0334

FEATURES SUPPORTED
 ONE WORD INTEGERS
 IOCS

CALLED SUBPROGRAMS
 ISTOX   MRED   MWRT    MCOMP   MIOIX   SUBSC   TYPEN   HOLEB   PRNTN   EBPRT

INTEGER CONSTANTS
    1=011E      8=011F      7=0120     10=0121      3=0122      0=0123

CORE REQUIREMENTS FOR PHONE
 COMMON      0  INSKEL COMMON       0
  VARIABLES    286  PROGRAM     536


END OF COMPILATION
```

This program was a surprise from Alan Nelles, a student at Chicago Circle. He was amused by my practice of alphabetizing phone numbers, and wrote a program to do it automatically.

Premises of the program: you supply it with your phone number, and it prints out all the alphabetical combinations that could also be dialled to reach your telephone.

Language: Fortran.

Behold some of the combinations. The recipient picks out the one he likes from 2½ pages of them.

```
EGUIMAK EGUIMAL EGUIMBJ EGUIMBK EGUIMBL EGUIMCJ EGUIMCK EGUIMCL EGUINAJ EGUINAK EGUINAL [illegible] [illegible] [illegible] [illegible]
EGUINCK EGUINCL EGUIOAJ EGUIOAK EGUIOAL EGUIOBJ EGUIOBK EGUIOBL EGUIOCJ EGUIOCK EGUIOCL EGVGMAJ EGVGMAK EGVGMAL EGVGMBJ
EGVGMBK EGVGMBL EGVGMCJ EGVGMCK EGVGMCL EGVGNAJ EGVGNAK EGVGNAL EGVGNBJ EGVGNBK EGVGNBL EGVGNCJ EGVGNCK EGVGNCL EGVGOAJ
EGVGOAK EGVGOAL EGVGOBJ EGVGOBK EGVGOBL EGVGOCJ EGVGOCK EGVGOCL EGVHMAJ EGVHMAK EGVHMAL EGVHMBJ EGVHMBK EGVHMBL EGVHMCJ
EGVHMCK EGVHMCL EGVHNAJ EGVHNAK EGVHNAL EGVHNBJ EGVHNBK EGVHNBL EGVHNCJ EGVHNCK EGVHNCL EGVHOAJ EGVHOAK EGVHOAL EGVHOBJ
EGVHOBK EGVHOBL EGVHOCJ EGVHOCK EGVHOCL EGVIMAJ EGVIMAK EGVIMAL EGVIMBJ EGVIMBK EGVIMBL EGVIMCJ EGVIMCK EGVIMCL EGVINAJ
EGVINAK EGVINAL EGVINBJ EGVINBK EGVINBL EGVINCJ EGVINCK EGVINCL EGVIOAJ EGVIOAK EGVIOAL EGVIOBJ EGVIOBK EGVIOBL EGVIOCJ
EGVIOCK EGVIOCL EHTGMAJ EHTGMAK EHTGMAL EHTGMBJ EHTGMBK EHTGMBL EHTGMCJ EHTGMCK EHTGMCL EHTGNAJ EHTGNAK EHTGNAL EHTGNBJ
EHTGNBK EHTGNBL EHTGNCJ EHTGNCK EHTGNCL EHTGOAJ EHTGOAK EHTGOAL EHTGOBJ EHTGOBK EHTGOBL EHTGOCJ EHTGOCK EHTGOCL EHTHMAJ
EHTHMAK EHTHMAL EHTHMBJ EHTHMBK EHTHMBL EHTHMCJ EHTHMCK EHTHMCL EHTHNAJ EHTHNAK EHTHNAL EHTHNBJ EHTHNBK EHTHNBL EHTHNCJ
EHTHNCK EHTHNCL EHTHOAJ EHTHOAK EHTHOAL EHTHOBJ EHTHOBK EHTHOBL EHTHOCJ EHTHOCK EHTHOCL EHTIMAJ EHTIMAK EHTIMAL EHTIMBJ
EHTIMBK EHTIMBL EHTIMCJ EHTIMCK EHTIMCL EHTINAJ EHTINAK EHTINAL EHTINBJ EHTINBK EHTINBL EHTINCJ EHTINCK EHTINCL EHTIOAJ
EHTIOAK EHTIOAL EHTIOBJ EHTIOBK EHTIOBL EHTIOCJ EHTIOCK EHTIOCL EHUGMAJ EHUGMAK EHUGMAL EHUGMBJ EHUGMBK EHUGMBL EHUGMCJ
EHUGMCK EHUGMCL EHUGNAJ EHUGNAK EHUGNAL EHUGNBJ EHUGNBK EHUGNBL EHUGNCJ EHUGNCK EHUGNCL EHUGOAJ EHUGOAK EHUGOAL EHUGOBJ
EHUGOBK EHUGOBL EHUGOCJ EHUGOCK EHUGOCL EHUHMAJ EHUHMAK EHUHMAL EHUHMBJ EHUHMBK EHUHMBL EHUHMCJ EHUHMCK EHUHMCL EHUHNAJ
EHUHNAK EHUHNAL EHUHNBJ EHUHNBK EHUHNBL EHUHNCJ EHUHNCK EHUHNCL EHUHOAJ EHUHOAK EHUHOAL EHUHOBJ EHUHOBK EHUHOBL EHUHOCJ
EHUHOCK EHUHOCL EHUIMAJ EHUIMAK EHUIMAL EHUIMBJ EHUIMBK EHUIMBL EHUIMCJ EHUIMCK EHUIMCL EHUINAJ EHUINAK EHUINAL EHUINBJ
EHUINBK EHUINBL EHUINCJ EHUINCK EHUINCL EHUIOAJ EHUIOAK EHUIOAL EHUIOBJ EHUIOBK EHUIOBL EHUIOCJ EHUIOCK EHUIOCL EHVGMAJ
EHVGMAK EHVGMAL EHVGMBJ EHVGMBK EHVGMBL EHVGMCJ EHVGMCK EHVGMCL EHVGNAJ EHVGNAK EHVGNAL EHVGNBJ EHVGNBK EHVGNBL EHVGNCJ
EHVGNCK EHVGNCL EHVGOAJ EHVGOAK EHVGOAL EHVGOBJ EHVGOBK EHVGOBL EHVGOCJ EHVGOCK EHVGOCL EHVHMAJ EHVHMAK EHVHMAL EHVHMBJ
EHVHMBK EHVHMBL EHVHMCJ EHVHMCK EHVHMCL EHVHNAJ EHVHNAK EHVHNAL EHVHNBJ EHVHNBK EHVHNBL EHVHNCJ EHVHNCK EHVHNCL EHVHOAJ
EHVHOAK EHVHOAL EHVHOBJ EHVHOBK EHVHOBL EHVHOCJ EHVHOCK EHVHOCL EHVIMAJ EHVIMAK EHVIMAL EHVIMBJ EHVIMBK EHVIMBL EHVIMCJ
EHVIMCK EHVIMCL EHVINAJ EHVINAK EHVINAL EHVINBJ EHVINBK EHVINBL EHVINCJ EHVINCK EHVINCL EHVIOAJ EHVIOAK EHVIOAL EHVIOBJ
EHVIOBK EHVIOBL EHVIOCJ EHVIOCK EHVIOCL EITGMAJ EITGMAK EITGMAL EITGMBJ EITGMBK EITGMBL EITGMCJ EITGMCK EITGMCL EITGNAJ
EITGNAK EITGNAL EITGNBJ EITGNBK EITGNBL EITGNCJ EITGNCK EITGNCL EITGOAJ EITGOAK EITGOAL EITGOBJ EITGOBK EITGOBL EITGOCJ
EITGOCK EITGOCL EITHMAJ EITHMAK EITHMAL EITHMBJ EITHMBK EITHMBL EITHMCJ EITHMCK EITHMCL EITHNAJ EITHNAK EITHNAL EITHNBJ
EITHNBK EITHNBL EITHNCJ EITHNCK EITHNCL EITHOAJ EITHOAK EITHOAL EITHOBJ EITHOBK EITHOBL EITHOCJ EITHOCK EITHOCL EITIMAJ
EITIMAK EITIMAL EITIMBJ EITIMBK EITIMBL EITIMCJ EITIMCK EITIMCL EITINAJ EITINAK EITINAL EITINBJ EITINBK EITINBL EITINCJ
EITINCK EITINCL EITIOAJ EITIOAK EITIOAL EITIOBJ EITIOBK EITIOBL EITIOCJ EITIOCK EITIOCL EIUGMAJ EIUGMAK EIUGMAL EIUGMBJ
EIUGMBK EIUGMBL EIUGMCJ EIUGMCK EIUGMCL EIUGNAJ EIUGNAK EIUGNAL EIUGNBJ EIUGNBK EIUGNBL EIUGNCJ EIUGNCK EIUGNCL EIUGOAJ
EIUGOAK EIUGOAL EIUGOBJ EIUGOBK EIUGOBL EIUGOCJ EIUGOCK EIUGOCL EIUHMAJ EIUHMAK EIUHMAL EIUHMBJ EIUHMBK EIUHMBL EIUHMCJ
EIUHMCK EIUHMCL EIUHNAJ EIUHNAK EIUHNAL EIUHNBJ EIUHNBK EIUHNBL EIUHNCJ EIUHNCK EIUHNCL EIUHOAJ EIUHOAK EIUHOAL EIUHOBJ
EIUHOBK EIUHOBL EIUHOCJ EIUHOCK EIUHOCL EIUIMAJ EIUIMAK EIUIMAL EIUIMBJ EIUIMBK EIUIMBL EIUIMCJ EIUIMCK EIUIMCL EIUINAJ
EIUINAK EIUINAL EIUINBJ EIUINBK EIUINBL EIUINCJ EIUINCK EIUINCL EIUIOAJ EIUIOAK EIUIOAL EIUIOBJ EIUIOBK EIUIOBL EIUIOCJ
EIUIOCK EIUIOCL EIVGMAJ EIVGMAK EIVGMAL EIVGMBJ EIVGMBK EIVGMBL EIVGMCJ EIVGMCK EIVGMCL EIVGNAJ EIVGNAK EIVGNAL EIVGNBJ
EIVGNBK EIVGNBL EIVGNCJ EIVGNCK EIVGNCL EIVGOAJ EIVGOAK EIVGOAL EIVGOBJ EIVGOBK EIVGOBL EIVGOCJ EIVGOCK EIVGOCL EIVHMAJ
EIVHMAK EIVHMAL EIVHMBJ EIVHMBK EIVHMBL EIVHMCJ EIVHMCK EIVHMCL EIVHNAJ EIVHNAK EIVHNAL EIVHNBJ EIVHNBK EIVHNBL EIVHNCJ
EIVHNCK EIVHNCL EIVHOAJ EIVHOAK EIVHOAL EIVHOBJ EIVHOBK EIVHOBL EIVHOCJ EIVHOCK EIVHOCL EIVIMAJ EIVIMAK EIVIMAL EIVIMBJ
EIVIMBK EIVIMBL EIVIMCJ EIVIMCK EIVIMCL EIVINAJ EIVINAK EIVINAL EIVINBJ EIVINBK EIVINBL EIVINCJ EIVINCK EIVINCL EIVIOAJ
EIVIOAK EIVIOAL EIVIOBJ EIVIOBK EIVIOBL EIVIOCJ EIVIOCK EIVIOCL FGTGMAJ FGTGMAK FGTGMAL FGTGMBJ FGTGMBK FGTGMBL FGTGMCJ
FGTGMCK FGTGMCL FGTGNAJ FGTGNAK FGTGNAL FGTGNBJ FGTGNBK FGTGNBL FGTGNCJ FGTGNCK FGTGNCL FGTGOAJ FGTGOAK FGTGOAL FGTGOBJ
FGTGOBK FGTGOBL FGTGOCJ FGTGOCK FGTGOCL FGTHMAJ FGTHMAK FGTHMAL FGTHMBJ FGTHMBK FGTHMBL FGTHMCJ FGTHMCK FGTHMCL FGTHNAJ
FGTHNAK FGTHNAL FGTHNBJ FGTHNBK FGTHNBL FGTHNCJ FGTHNCK FGTHNCL FGTHOAJ FGTHOAK FGTHOAL FGTHOBJ FGTHOBK FGTHOBL FGTHOCJ
FGTHOCK FGTHOCL FGTIMAJ FGTIMAK FGTIMAL FGTIMBJ FGTIMBK FGTIMBL FGTIMCJ FGTIMCK FGTIMCL FGTINAJ FGTINAK FGTINAL FGTINBJ
FGTINBK FGTINBL FGTINCJ FGTINCK FGTINCL FGTIOAJ FGTIOAK FGTIOAL FGTIOBJ FGTIOBK FGTIOBL FGTIOCJ FGTIOCK FGTIOCL FGUGMAJ
FGUGMAK FGUGMAL FGUGMBJ FGUGMBK FGUGMBL FGUGMCJ FGUGMCK FGUGMCL FGUGNAJ FGUGNAK FGUGNAL FGUGNBJ FGUGNBK FGUGNBL FGUGNCJ
FGUGNCK FGUGNCL FGUGOAJ FGUGOAK FGUGOAL FGUGOBJ FGUGOBK FGUGOBL FGUGOCJ FGUGOCK FGUGOCL FGUHMAJ FGUHMAK FGUHMAL FGUHMBJ
FGUHMBK FGUHMBL FGUHMCJ FGUHMCK FGUHMCL FGUHNAJ FGUHNAK FGUHNAL FGUHNBJ FGUHNBK FGUHNBL FGUHNCJ FGUHNCK FGUHNCL FGUHOAJ
FGUHOAK FGUHOAL FGUHOBJ FGUHOBK FGUHOBL FGUHOCJ FGUHOCK FGUHOCL FGUIMAJ FGUIMAK FGUIMAL FGUIMBJ FGUIMBK FGUIMBL FGUIMCJ
FGUIMCK FGUIMCL FGUINAJ FGUINAK FGUINAL FGUINBJ FGUINBK FGUINBL FGUINCJ FGUINCK FGUINCL FGUIOAJ FGUIOAK FGUIOAL FGUIOBJ
FGUIOBK FGUIOBL FGUIOCJ FGUIOCK FGUIOCL FGVGMAJ FGVGMAK FGVGMAL FGVGMBJ FGVGMBK FGVGMBL FGVGMCJ FGVGMCK FGVGMCL FGVGNAJ
FGVGNAK FGVGNAL FGVGNBJ FGVGNBK FGVGNBL FGVGNCJ FGVGNCK FGVGNCL FGVGOAJ FGVGOAK FGVGOAL FGVGOBJ FGVGOBK FGVGOBL FGVGOCJ
FGVGOCK FGVGOCL FGVHMAJ FGVHMAK FGVHMAL FGVHMBJ FGVHMBK FGVHMBL FGVHMCJ FGVHMCK FGVHMCL FGVHNAJ FGVHNAK FGVHNAL FGVHNBJ
FGVHNBK FGVHNBL FGVHNCJ FGVHNCK FGVHNCL FGVHOAJ FGVHOAK FGVHOAL FGVHOBJ FGVHOBK FGVHOBL FGVHOCJ FGVHOCK FGVHOCL FGVIMAJ
FGVIMAK FGVIMAL FGVIMBJ FGVIMBK FGVIMBL FGVIMCJ FGVIMCK FGVIMCL FGVINAJ FGVINAK FGVINAL FGVINBJ FGVINBK FGVINBL FGVINCJ
FGVINCK FGVINCL FGVIOAJ FGVIOAK FGVIOAL FGVIOBJ FGVIOBK FGVIOBL FGVIOCJ FGVIOCK FGVIOCL FHTGMAJ FHTGMAK FHTGMAL FHTGMBJ
FHTGMBK FHTGMBL FHTGMCJ FHTGMCK FHTGMCL FHTGNAJ FHTGNAK FHTGNAL FHTGNBJ FHTGNBK FHTGNBL FHTGNCJ FHTGNCK FHTGNCL FHTGOAJ
FHTGOAK FHTGOAL FHTGOBJ FHTGOBK FHTGOBL FHTGOCJ FHTGOCK FHTGOCL FHTHMAJ FHTHMAK FHTHMAL FHTHMBJ FHTHMBK FHTHMBL FHTHMCJ
```

Below: Nelles' program to calculate the date of Easter. The language is Algol.

```
LEVEL 1JUL67                      OS ALGOL F                    DATE  FEB 07 1970
                                SOURCE PROGRAM                        PAGE 001
 SC      SOURCE STATEMENT

00000   'BEGIN' 'INTEGER' YEAR1,YEAR2,SUBSCR: START: ININTEGER(0,YEAR1);
00002       ININTEGER(0,YEAR2); 'IF' YEAR1>YEAR2 'THEN' 'GOTO' EXIT;
00004       SUBSCR:=YEAR2-YEAR1+1;
00005       'BEGIN' 'INTEGER' CURYR,CENT,M,L,A,B,C,D,E,KAY,K;
00006           'INTEGER' 'ARRAY' EASTER(/1:SUBSCR,1:2/);
00007           'INTEGER' 'PROCEDURE' MOD(X,Y); 'INTEGER' X,Y;
00009           MOD:=X-Y*(X'/'Y); CURYR:=YEAR1; KAY:=1;
00012           GO:CENT:=CURYR'/'100; M:=15+CENT-CENT'/'4-(8*CENT+13)'/'25;
00014               L:=4+CENT-CENT'/'4; A:=MOD(CURYR,4); B:=MOD(CURYR,7);
00017               C:=MOD(CURYR,19); D:=19*C+M; D:=MOD(D,30);
00020               E:=2*A+4*B+6*D+L; E:=MOD(E,7);
00022               'IF' E=6 'AND' D=29 'THEN' 'GOTO' APRL19;
00023               'IF' E=6 'AND' D=28 'AND' (M=2 'OR' M=5 'OR' M=10 'OR'
00023               M=13 'OR' M=16 'OR' M=21 'OR' M=24 'OR' M=29)'THEN'
00023             'GOTO' APRL18; EASTER(/KAY,1/):=D+E+22;
00025               EASTYR:EASTER(/KAY,2/):=CURYR; 'IF' CURYR>=YEAR2 'THEN'
00026                   'GOTO' SORT; CURYR:=CURYR+1; KAY:=KAY+1; 'GOTO' GO;
00030               APRL18:EASTER(/KAY,1/):=49; 'GOTO' EASTYR;
00032               APRL19:EASTER(/KAY,1/):=50; 'GOTO' EASTYR;
00034       SORT: 'FOR' KAY:=1 'STEP' 1 'UNTIL'SUBSCR-1 'DO'
00034           'BEGIN' 'FOR' K:=KAY+1 'STEP' 1 'UNTIL' SUBSCR 'DO'
00034               'IF' EASTER(/KAY,1/)>EASTER(/K,1/) 'THEN'
00034               'BEGIN' A:=EASTER(/KAY,1/);
00035                   EASTER(/KAY,1/):=EASTER(/K,1/);
00036                   EASTER(/K,1/):=A; A:=EASTER(/KAY,2/);
00038                   EASTER(/KAY,2/):=EASTER(/K,2/);
00039                   EASTER(/K,2/):=A 'END';
00040               'END';
00041           OUTSTRING(1,'('A. NELLES, MATH 280, EASTER TABLES')');
00042           SYSACT(1,15,1); 'FOR' KAY:=1 'STEP' 1 'UNTIL' SUBSCR 'DO'
00043               'BEGIN' 'IF' EASTER(/KAY,1/)<32 'THEN'
00043                   OUTSTRING(1,'('MARCH ')') 'ELSE'
00043                       'BEGIN' OUTSTRING(1,'('APRIL ')');
00044                           EASTER(/KAY,1/):=EASTER(/KAY,1/)-31
00044                       'END'; OUTINTEGER(1,EASTER(/KAY,1/));
00046                       OUTINTEGER(1,EASTER(/KAY,2/));SYSACT(1,15,1)
00047                   'END'; SYSACT(1,15,1) 'END'; 'GOTO' START;
00050    EXIT:OUTSTRING(1,'('ORIGINALITY')'); SYSACT(1,15,1) 'END'
```

YECCCH, IT'S COBOL

Research and hobby types hate COBOL or ignore it, but it's the main business programming language. Your income tax, your checking account, your automobile license-- all are presumably handled by programs in the COBOL language.

COBOL, or COmmon Business Oriented Language, was more or less demanded by the Department of Defense, and brought into being by a committee called CODASYL, which is apparently still going. COBOL uses mostly decimal numbers, is designed basically for batch processing (described elsewhere), and uses verbose and plonking command formats.

Just because it's standard for business programming doesn't mean it's the best or most efficient language for business programming; I've talked to people who advocate business programming in FORTRAN, BASIC, TRAC and even APL. But then you get into those endless arguments... and it turns out that a large proportion of business programmers only know Cobol, which pragmatically settles the argument.

There are people who say they've discovered hidden beauties in COBOL; for instance, that it's a splendid language for complex pointer manipulation (see Data Structures, p. 26). That's what makes horse racing.

JCL

Some call it Despicable, Some call it Home

"After you study it for six months, it makes perfect sense." --An IBM enthusiast.

JCL is a language with which you submit programs to an IBM 360 or 370 computer. "Submit" is right. Its complications, which many call unnecessary, symbolize the career of submission to IBM upon which the 360 programmer embarks. (See IBM, pp. 52-3, and 360, p. 41.)

SNOBOL

SNOBOL is the favorite computing language of a lot of my friends. It is a list-processing language, meaning it's good for amorphous data. (It derives from several previous list-processing languages, especially IPL-V and COMIT.)

SNOBOL is a big language, and only runs on big computers. The main concept of it is the "pattern match," whereby a string of symbols is examined to see if it has certain characteristics, including any particular contents, relations between contents, or other variations the programmer can specify; and the string substitution, where some specified string of symbols is replaced by another that the programmer contrives.

LISP

is probably the favorite language of the artificial-intelligence freaks (see p. 12). A fondness for LISP, incidentally, is not considered to reflect on your masculinity.

LISP is a "cult" language, and its adherents are sometimes called Lispians. They see computer activities in a somewhat different light, as composed of ever-changing chains of things called "cars" and "cudders," which will not be explained here.

LISP was developed by John McCarthy at MIT, based largely on the Lambda-notation of Alonzo Church. It allows the chaining of operations and data in deeply intermingled forms. While it runs on elegant principles, most people object to its innumerable parentheses (a feature shared to some extent by TRAC Language).

Joseph Weizenbaum, also of MIT, has created a language called SLIP, somewhat resembling LISP, which runs in FORTRAN. That means you can run LISP-like programs without having access to a LISP processor, which is helpful.

THEN, THERE'S ALWAYS MACHINE LANGUAGE

If you feel like making programs run fast, and not take up very much core memory, you go to machine language, the computer's very own wired-up deep-down system of commands (see p. 32). It takes longer, usually, but many people consider it very satisfying.

Then, of course, if you have a particular style and approach and set of interests, you will probably start building up a collection of individual programs for your own purposes.

Then you'll work out simplified ways of calling these into operation and tying their results and data together.

Which means you'll have a language of your own.

ROCK BOTTOM
THE WORLD BENEATH THE HIGHER LANGUAGES

Every computer is wired to accept a specific system of commands. When these commands are stored in the computer's memory, and the computer's program follower gets to them, they cause it to respond directly by electronic reflex. This is called machine language, the very language of the machine itself.

In most available computers the machine languages are binary, meaning composed of only two alternative symbols. Binary because it's a sensible way of organizing the machine's structure; it permits programs to be reduced to a single common form of information, and permits programs to be stored in binary memory. Each individual instruction or command ordinarily occupies one memory slot, though some computers have commands of varying length.

Different computers have different machine languages, but the instructions of all computers are basically similar. Big computers have more commands, with more variations, and carry them out faster; but those variations are just extra ways of saving steps, not qualitatively different features.

These deep-down operations ARE ALL THE THINGS THE COMPUTER EVER DOES. However, in their combinations these instructions can be woven into chains and diadems of complex actions.

ALL COMPUTER PROGRAMS ARE EVENTUALLY WRITTEN OR ENACTED IN THE MACHINE'S PARTICULAR BINARY LANGUAGE.

Now, it is entirely possible to write your programs at this level, considering and arranging rock-bottom commands. This is called machine-language programming (and assembly programming; see examples a little later on). Indeed, working at this level is very highly respected in some quarters. Others avoid it. This is a very serious matter of taste and what you're working on.

Higher-level languages, seen on earlier pages, have more convenient forms for people, but must be translated, either ahead of time or on a running basis, to the bottom-most codes that make things happen in the machine. All of them are built out of machine language. Writing the language processors, programs that enact or translate these higher-level languages, is considered a black art. (See p. 30.)

Every programmable device has a "machine language," or rock bottom code system that activates the thing directly; its program follower responds electrically to these codes, and enacts them one instruction at a time.

True computers are programmable devices that can modify their own instructions, change their sequence of operations and do other versatile stuff.

What the Computer Really Is
COMPUTER ARCHITECTURE
the Nuts and Bolts

Computers are basically alike. Ignore their appearances: a roomful of roaring cabinets may have a great deal in common with a small blinking box; indeed, they may have the same architecture, or structure, and therefore be the same computer.

The structure of computers, in their glorious similarities and fascinating differences, is called computer architecture.

(For the architecture of a beginner's computer, see p. 33; for the architecture of some famous computers, see pp. 40-3.)

Computer architecture covers three main things: registers (places where something happens to information); memories (places where nothing happens to information); their interconnections; and machine language, all the bottom-level instructions (for this last see "Rock Bottom," p. 32).

REGISTERS AND MEMORIES

Computers are made, basically, of two things: registers and memories. A register is where something happens to information; a memory is where nothing happens to information. Let's go over that slowly.

A register is a place where something happens to information: the information can be flipped around, tested, changed by arithmetic, or whatever. (We noted earlier that registers are what connect a computer to its accessories. They are also principal parts of the computer itself.)

A memory is a place where nothing happens to information. A program puts the information there, and there it stays till some program pulls it out again or replaces it.

A main or general register (often called the accumulator, for no good reason) is where the program brings things to be worked on, tested, compared, added to and so on. There can be several of them in a computer.

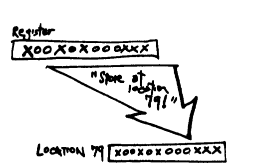
Other registers perform other functions in the computer; a given computer's design, or architechture, is largely the arrangement of registers and the operations that take place between them.

The reason we don't just have all registers-- and no memories at all-- is that registers traditionally cost more than memories. (However, some machines are being tried that have all working registers instead of memory. See STARAN, p. 43.)

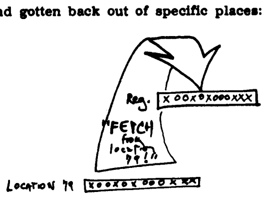
Memories come in all sizes and speeds. So lots of computers have big slow memories, such as disk memories, along with their small fast memories.

A memory consists of numerous holding places or storage locations, each holding one standard piece of information for the computer, a word having a specific number of bits (see p. .) We must stress: a "COMPUTER WORD" HAS NOTHING TO DO WITH ENGLISH WORDS OR ALPHABETICAL CHARACTERS. The term refers to a specific machine's standard memory slot, having a fixed number of bit positions.

One important reason for this standardization is that each holding place, or memory location, can be given a number or address. If every slot in the memory has an address, information can be stored in specific places:



and gotten back out of specific places:



A core memory has a definite rhythm or cycle, into which it divides the passing time. The memory cycle of a core memory is so important that its duration is often called the cycle time of the computer. A request to the core memory made at the beginning of the cycle is honored at the end of the cycle. Core cycles are very fast, being these days about one microsecond, or millionth of a second.

A core memory can only perform one act (store or fetch) during one memory cycle.

Core cycles during which nothing is requested of the memory simply go by.

One last point about core memories. The number which specifies an address to the memory is a binary pattern-- just like all the other information (see "Binary Patterns," p. 33). (Or more exactly, whatever binary pattern is supplied to the memory as the address to store or from which to fetch, that pattern will be treated as the address to store or from which to fetch, that pattern will be treated as a binary number whether it was supposed to be or not. It could be the alphabetic word GRINCH which got there by mistake (see "Debugging," p. 30), but the memory will treat it as an address number and go to the address specified by that pattern.

THEN WHAT ARE THE DIFFERENCES BETWEEN COMPUTERS?

The word length
  (number of bit-spaces in a main register and memory slot)
The number of main registers
  and what they can do; i.e., how they are set up and what operations can take place in and among them; i.e.,
  the Instruction Set (see nearby);
The amount of memory;
The accessories or peripherals;
The cycle time.

Here's the computer, then, in all its glory: a device with a symbolic program, stored in a memory, being stepped through by a program follower.

The commands of the program cause the program follower to carry out the individual steps requested by each command of the program.

THE ROCK BOTTOM PROGRAM FOLLOWER

How, you ask desperately, does this innermost program follower work? The one that is built into the computer?

Aha.

Basically it consists of two specific registers, the Program Counter (usually abbreviated PC) and the Instruction Register (usually abbreviated IR), and other electronic stuff, loosely termed "decoding logic."

(Since we are already visualizing the program follower as a little hand, let's think of the index finger as the program counter and imagine that the thumb can flip an instruction into a little cup, the Instruction Register or IR. What the heck.)

WHEN a program is set into operation, the binary pattern specifying its first address in memory is put into the program counter.

Then the instruction at that address is fetched to the program follower (that is, put in the instruction register), decoded and carried out.

THEN THE PROGRAM COUNTER AUTOMATICALLY HAS ONE ADDED TO IT, SO IT POINTS TO THE NEXT INSTRUCTION.

The instruction pulled from memory is held in the command or instruction register and there decoded by the system's electronics.

It is of no concern to the programmer how this is done electronically. (And indeed electronics is generally of little concern to computer people, unless they are trying to design or optimize computers or other devices themselves. Indeed, the electronic techniques are constantly changing.)

All we need to know is that an electrical decoding system (called the logic circuits) carries out the specific instruction-- for instance, by shutting off the path to the memory, turning on the adding circuit, and opening paths through the adding circuit and back to the main register.

Now that the program counter holds the number of the next instruction it in turn is accordingly fetched and executed.

And so it continues.

When an instruction calls for a jump or branch in the program, what happens?

The jump command causes a new number to be stuffed into the program counter, that's what, and so that's where the program goes next.

ALTERNATING CYCLES

Many instructions tell the program follower to take a data word (also a binary pattern) from memory and put it in a main register or vice versa.

Such an instruction is translated by the decoding logic into a request to the memory.

Since a core memory can only do one thing during one of its cycles, the next instruction in the program cannot be fetched until the data has moved to or from the memory.

Thus in many types of program the cycles alternate:

Instruction cycle (fetch the next)
Data cycle
  (data goes to or from memory),
Instruction cycle,
Data cycle,
  and so on.

FUNDAMENTAL OPERATIONS OF COMPUTERS
A GREAT MYSTERY IS ABOUT TO UNFOLD.

YOUR BASIC COMMANDS, NOW

(Computers exist which do little more than these, and yet they can in principle do anything fancier computers can do.)

TO BE SHOWN: The following are the rock-bottom basic operations of computers, available as specific instructions in all computers (with some variation).

The first seven listed below will be used in the extended example in the next spread.

LOAD a binary pattern from core memory to a main register.

STORE a binary pattern in core memory from a main register.

SEND OUT ("OUTPUT") a binary pattern to an external device.

BRING IN ("INPUT") a binary pattern from an external device.

ADD TWO binary patterns together. (This causes them to be treated as numbers, whether they were to begin with or not.)

JUMP--
  Go to another part of the program and forget you were here.

TEST TWO binary patterns against each other, and branch or not in the program depending on the result.

NOT TO BE SHOWN: Here are the rest of the utterly fundamental commands of computers. (These are not used in the forthcoming example.)

TEST ONE SPECIFIC binary pattern, and branch in the program depending on the result.

SET AN ACCESSORY IN OPERATION/TURN IT OFF.

REVERSE (or "COMPLEMENT") a binary pattern-- changing all the X's to O's and vice versa.

SLIDE (or "SHIFT") a binary pattern sidelong through a register.

FLIPPER (or "LOGICAL") operations between two binary patterns, especially--

OR (or "INCLUSIVE OR" or "IOR")-- result is an X where either original pattern was an X.
AND (or "MASK")-- result is an X only where both original patterns had an X.

FANCY OPERATIONS

The following operations are desirable but not strictly necessary, and many computers, especially minicomputers, don't have them all.

SUBTRACT. (Can also be done if necessary with combination of adds and flips.)

MULTIPLY. (Can also be done if necessary with combination of adds, shifts and tests.)

DIVIDE. (Can also be done if necessary with combination of subtracts, shifts and tests.)

MORE FLIPPER ("LOGICAL") operations:

XOR (or "EXCLUSIVE OR")-- result is an X only where one pattern had an X, but not both.
NAND-- reversed AND.
NOR-- reversed OR.

SUBROUTINE JUMP--
  "Go to another part of the program but rememember this place because you'll be coming back on your own."

RETURN FROM SUBROUTINE--
  "Go back to wherever it was in the program that you last came from."

PUSH (on Stack machines only, see p.)-- take a binary pattern and put it on top of the Stack.

POP (on Stack machines only, see p.)--- take whatever binary pattern is now on the top of the Stack.

ADD ONE (or "INCREMENT")-- (Useful when you're counting the number of times something has been done.)

SUBTRACT ONE (or "DECREMENT," not "excrement")-- (Also useful when you're counting the number of times something has beer done.)

ASTRONOMICAL/INFINITESIMAL ARITHMETIC (or "FLOATING POINT" arithmetic)-- operates on a certain number of Significant Digits and keeps separate track of the decimal point-- actually a Binary Point, since it's rarely if ever done decimally.

→Very important in the physical sciences.

Almost any operations can be "built in." The sky is of course the limit, since any electronic operation can be added to a computer's instruction-set if desired-- say, "turn on the electric blender" or "multiply quaternions"-- but the former is more easily done as an output instruction, and the latter as part of a program.

Somehow
LOADING, STORING,
MODIFYING
AND TESTING
BINARY PATTERNS
DOESN'T SEEM
TERRIBLY FRAUGHT
WITH POSSIBILITIES;
but the endless variations and ramifications make chess look like tic-tac-toe.

And part of the power, of course, is in the great speed, the teeny fraction of a second each step takes; five hundred operations yet take only about a thousandth of a second. So no matter how intricate the enactment to which these tiny steps are built, it still happens awfully fast.

A computer, then, internally just consists of certain places to work on information (main registers), certain places to keep it the rest of the time (memories), certain pathways and interconnections between them, an instruction-set having certain powers whose instructions can be operated on out of memory, and a program follower that carries out the instructions of that instruction-set.

INSTRUCTION-SET.

The system of command patterns designed and wired into a particular computer, each with its exact results.

(The instructions in the set are the vocabulary of a machine language.)

A WIND-UP CROSSWORD PUZZLE

We look at last at what really happens inside a given computer. It must be a specific computer because there is no single inner language for all computers. For simplicity's sake (like most introductory texts) we hereby present a fictitious machine,

THE ★ FIDO★

(Faithful Instrument, Domesticated and Obliging).

The FIDO is a twelve-bit machine. The main register (it has only one) is twelve bits long, and every memory slot is twelve bits long.

Every instruction is twelve bits long; every data word is twelve bits long, though of course much longer pieces of data can be put together by taking more than one twelve-bit word.

Some rudimentary instructions of the FIDO are listed in a nearby box. The instructions of the FIDO are of two types: plain ones that just use the main register (like CLEAR), and the divided ones, which select a memory slot or output device. On the FIDO these are divided into an operation code (opcode) of five bits--the bits that tell the program follower what the operation is to be; and an address of seven bits, specifying which memory slot (or external device) is to be operated on.

These seven bits allow exactly 128 different patterns, (from OOOOOOO to XXXXXXX), which means we can select among exactly 128 different memory slots. (See Binary Patterns, p. 33.) (H. H.)

The Fido comes with one row of lights and switches; the row of lights can show the contents of any specific working register or memory slot. When the computer is stopped, this is helpful for debugging programs (see p. 30.)

Ah, if only we could tell you all about the FIDO here! Its many more instructions. The option bits in the commands that allow fancy variations, or the option bits in the interfaces, spoken of earlier, which allow the program to give different commands to external devices.

But let's get on with a program for the FIDO. Thrill to the pulsating rhythms of...

BUCKY'S WRISTWATCH!

(see next page.)

BASIC INSTRUCTIONS OF THE FIDO COMPUTER.

For a revelation of its Secret Identity, See Below.

5 bits → OPCODE (Binary pattern selecting operation)
leaves 7 bits for → ADDRESS (Binary pattern selecting where to perform operation)

OPERATION CALLED FOR

X X X X X $\underbrace{\text{o o o o o o o}}_{\text{don't matter}}$

CLEAR AC
This instruction causes the AC to be filled with zeroes.

O O X O O $\underbrace{\text{o o o o o o o}}_{\text{address goes here}}$

ADD (from memory to AC)
This adds the contents of the specified memory location to the contents of the AC. Result remains in the AC. Whatever was in the memory before is still there. This instruction is also used to bring a new pattern to the AC, copying it from the specified memory location; but you have to CLEAR the AC first, so you're adding it to zero.

O X X O O $\underbrace{\text{o o o o o o o}}_{\text{address goes here}}$

STORE
This instruction copies the contents of the AC to the specified memory location. Whatever was in the memory location is destroyed.

Whatever was in the AC is still there too.

X X O O O $\underbrace{\text{o o o o o o o}}_{\text{address goes here}}$

INPUT*
This instruction copies the contents of a specified device register to the AC.

X X O O X $\underbrace{\text{o o o o o o o}}_{\text{address goes here}}$

OUTPUT*
This instruction copies the contents of the AC to a specified device register.

X O X O O $\underbrace{\text{o o o o o o o}}_{\text{address goes here}}$

JUMP
This instruction makes the program follower take its next instruction at the specified address and go on from there.

O O O O X $\underbrace{\text{o o o o o o o}}_{\text{address goes here}}$

TEST, SKIP IF EQUAL**
This is a common test instruction, permitting the program to branch depending on various conditions. The contents of the AC are compared with the specified core memory location. If they are not the same, the program continues and takes the next instruction in the normal fashion. IF the two patterns are the same, the program follower SKIPS the next instruction and goes on to the one after.

Whatever the next instruction is, then, determines the course of events if the two patterns turn out to be the same.

For instance, that middle instruction can be a JUMP instruction, taking the program to a whole nother part of core memory and a new series of events.

FIDO POCKET CARD

XXXXXooooooo	CLEAR
OOXOOooooooo	ADD from mem.
OXXOOooooooo	STORE in mem.
XXOOOooooooo	INPUT
XXOOXooooooo	OUTPUT
XOXOOooooooo	JUMP
OOOOXooooooo	TEST, SKIP IF EQUAL.

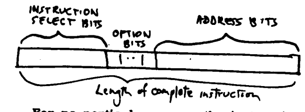
* Note: these instructions have been changed slightly to protect the innocent (you).

** This instruction does not exist on the PDP-8. Actually it offers a wider choice, which we can't go into here. Sophisticated instruction-packing makes the PDP-8 remarkably efficient considering its small 12-bit word length.

If you want information on the machine language and assembly language of any given machine, write the manufacturer for the programming manual. There may also be a pocket card.

INSTRUCTION LAYOUT

An occult aspect of computer design is the matter of how to pack into the so-many bits of an instruction word all the options the programmer should have.



For no particular reason the instruction select bits are usually on the left, the address bits on the right, and option bits (no room for them in this book, unfortunately) in the middle.

The number of bits in the address determines the number of places in the memory that the programmer can choose among. 15 bits in the address means a choice of 32,768 memory locations. 7 bits means a choice of only 128. (See "Binary Patterns," p. 33 .)

Generally a specific computer has more than one instruction layout.

Deciding what the instruction layouts are to be hinges on the architectural design of the computer (see p.32) and the instruction-set. It all gets worked out together.

It's ultimately a matter of design elegance, but the consequences are very concrete. An elegant instruction-set is easy to use and therefore saves a lot of time and money. (Anyone interested in studying the matter might want to compare the PDP-11, a 16-bit computer with a brilliantly designed instruction-set, with some other 16-bit computer.)

GUESS WHAT!

The FIDO is nothing but a stripped-down version of that beloved family pooch of computerdom,

The PDP-8.

(Described p.40 .)

If you buy a PDP-8 from Digital Equipment Corporation, you get all this and more. (Except for the external devices.) And the PDP-8, of course, allows much bigger memories than 128 slots, but that's too complicated for here.) Arf.

BINARY PATTERNS

are what the computer operates on deep down. "Binary" just means that only two symbols are used (just as "decimal" means that ten symbols are used). Patterns of binary symbols happen to be electrically convenient, so that's how computers are built, but that would change if some more convenient set of symbols came along.

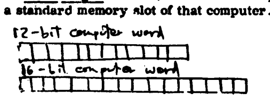
Binary patterns are very systematic and easy to deal with. Consider the number of binary symbols you can have in just four spaces. →LET'S USE THE LETTERS X AND O, AND PUT THEM IN ALPHABETICAL ORDER, SO YOU'LL SEE THAT WE'RE TALKING ABOUT PATTERNS, RATHER THAN NUMBERS.

```
O O O O
O O O X
O O X O
O O X X
O X O O
O X O X
O X X O
O X X X
X O O O
X O O X
X O X O
X O X X
X X O O
X X O X
X X X O
X X X X
```

You can see that the pattern repeats in certain interesting ways. Each column repeats itself as you read down; adding a new position to the left doubles the number of possible patterns you can have in the row.

These are the infamous "bits" you have heard of. As you can see, there is nothing hard or complicated about them. The number of bits in a thing are the number of spaces which can be either X or O.

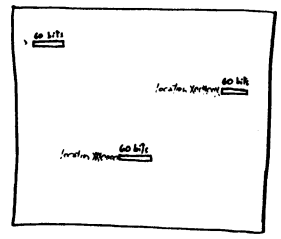
Now, the most basic fact about any computer is its word length: that is, the number of spaces in a standard memory slot of that computer.



A "12-bit computer" (like the PDP-8) has memory words that are all twelve bits long. A "16-bit computer" (like the PDP-11) has memory words that are all 16 bits long.

Actually computers with small word lengths like these are called minicomputers. Big computers have much bigger word lengths. The IBM 360 has a 32-bit word length. The Control Data 6600 has a 60-bit word.

Now, it is an interesting fact that not only are computer memories divided up into slots, or locations, of equal length,



but each of these locations has an address, that is, a number by which the contents of the location can be found. And these numbers are binary.

Many forms of information are kept in binary patterns which are not numbers. For instance, letters of the alphabet are usually stored as 8-bit patterns.

|X|X|O|X|O|O|O|X|

THE LETTER "Q"
(IN ASCII CODE)

All computers can
in principle do
the same things,
some faster.
However, some are
too slow or too small
ever to do what others can,
though the types of their
operations are similar.

Some computers (and their
languages and facilities)
are much more convenient
for programmers than others,
because their instruction-sets
are better.

This is no small matter.

(But it's a big matter of
taste and argument
among computer people.)

The big-point is,

AT THE BOTTOM PROGRAMS ARE BINARY

AND DATA IS BINARY,

since it's all stored in binary memory.

But since that suits few people's individual purposes, we build up HIGHER LANGUAGES AND DATA STRUCTURES. So that different users deal with different mechanics corresponding better and more conveniently to the structures that interest them.

However, we will have to stop using these X's and O's. It's not really done, so we will switch to the more usual way of writing binary patterns with 1's and zeroes. (Apologies to readers who hate numbers; but remember that these patterns, while we may write them out as 1's and zeroes, may represent wholly non-numerical kinds of information.) That means the letter Q is

|1|1|0|1|0|0|0|1|

but it's still the letter Q.

Of course, bits may also represent numerical information. And so we pass on to

BINARY NUMBERS.

These are the same old binary patterns, but when we decide to treat them as numbers, they are binary numbers.

Let's count. Note that these are the same combinations of bits as before, merely put in the more usual notation.

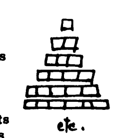
decimal number	binary number
0	000
1	001
2	010
3	011
4	100
5	101
6	110
7	111
08	1000
09	1001
10	1010
11	1011
12	1100
13	1101
14	1110
15	1111

As you observe, the higher numbers need more and more bits to hold them.

This brings up some interesting facts.

CERTAIN NUMBERS ARE SPECIAL because they are the number of things that can be specified by a certain number of bits.

Special number	
2	one bit
4	two bits
8	three bits
16	four bits
32	five bits
64	six bits
128	seven bits
256	eight bits
512	nine bits
1024	ten bits



("ONE K" is 1024; memories and everything else come in K's, or multiples of 1024.)

Actually the term "k," standing for "kilo-," should mean one thousand, and the term BK, or Binary K, is used by fussy people to stand for the very important nearby number 1024. But computer people generally use expressions ending in K for the following special numbers:

NUMBER	THAT'S HOW MANY COMBINATIONS FIT IN
2048 ("2K")	eleven bits
4096 ("4K")	twelve bits
8192 ("8K")	thirteen bits
16,384 ("16K")	fourteen bits
32,768 ("32K")	fifteen bits.

Above this number they increase very fast, and we generally have to look them up, but the idea is this: the number of bits used to select something limits the number of things you can select among. For instance, if you have a computer memory with 32K different locations, you need fifteen bits exactly to specify a location in memory.

Here are some ramifications:

• The word length of a computer determines how large a number it can hold. A computer with a twelve-bit word can only hold a number up to 4095 in one memory location (since we use 000 000 000 000, the first combination, to stand for zero); if we want to use longer numbers we have to set aside two or more word locations per number. (A 16-bit computer can hold a number up to 65,535 in one memory location.)

• In designing data structures, if you use binary codes (rather than, say, alphabetical characters), you have to allow enough bits for all the alternatives that might turn up.

• In the design of the wired-in instructions for a computer, therefore, the number of bits set aside to specify an address in core determines whether that instruction can select from the whole memory, or just a part of it.

★BUCKY'S WRISTWATCH★

There is a certain folk hero whom the people all call Bucky. It is said that he wears three wristwatches: one for where he is now, one for where he will be next, and one that tells what time it is at his home.

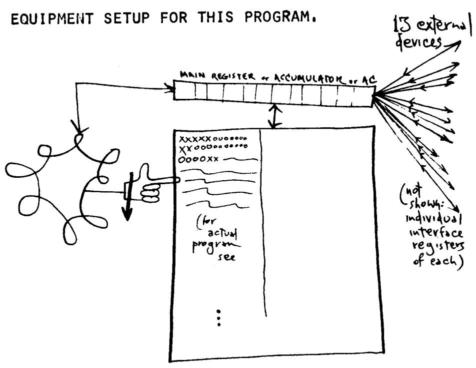
Well now. Here's an example of a little problem on which to try our FIDO computer.

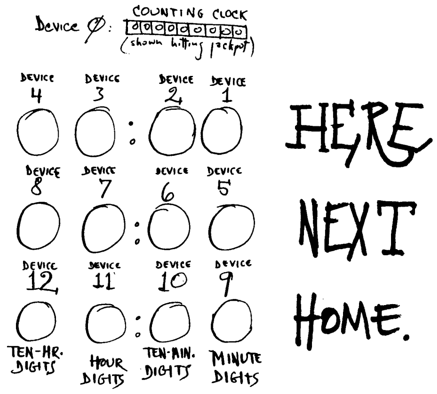
Let's wire up a magic wristwatch for Bucky the Folk Hero, one that will use a teeny FIDO on a chip (the coming thing), attached to three rows of numerical readouts (like those on pocket calculators).

This application is not so absurd as you might think.

It is obviously quite simple in principle.

It will let us see some of the ways that the rock-bottom machine languages of computers are used.





ABOUT THIS WONDERFUL PROGRAM.

Naturally this got saved for last, and what is presented here shows it.

The example was meant to be a case of not-very-numerical programming that would show the abstractness of it all. The program itself has no intrinsic quality related to the problem; that much should be visible.

Anyhow, I programmed this myself a few weeks ago in the FIDO language, and was very pleased with it, but then discovered a couple of appalling bugs. As time closed in on this project I asked my friend Mike O'Brien to code the program, and he kindly consented, taking time out of his previous weekend plans. Here is Mike's program, for which I am grateful.

HowEVer, after it was set in type, Mike realized that it too has some gross flaws and would not work as here presented. We thought of having a chocolate chip cookie contest for corrections, sending out chocolate chip cookies to entrants fixing it up, but we don't have such a computer and we wouldn't run the program if we had one anyway, so see if you can get the basic idea of it, and if you are a real wise guy fix the program for your own satisfaction, and that will be that.

The basic idea is that we have a FIDO, presumably on a single integrated circuit chip, attached to thirteen external devices (or peripherals, or input-output devices, or I/O devices or whatever). These devices are a timer or clock, which reaches zero once per minute-- this is a computer clock, meaning a timer, not something that people can read-- and the three rows of numerical readouts that are the desired Superwatch.

For simplicity's sake we assume here that each numeral is interfaced to do either input or output; thus the FIDO computer can ask any given numeral what it says, and change its contents.

The finished Wristwatch is going to give time on a twentyfour-hour basis, not twelve, like at NASA and suchlike places. After 12:59 comes 13:00. After 23:59 comes 01:00.

WOOPS!

01:00 should come on the clock after 24:59. The days will be a little short for Bucky, since we test the second digit for a 3 rather than a 4. This is called a logic error, meaning it was thought out incorrectly, rather than a "coding error" (meaning the program fails to carry out the steps that have been thought out).

So you begin to see, uh, heh heh, some of the, uh...

The bulk of the program is occupied with testing the numerals and changing them. However, in proportions of activity, the poor thing is going to spend most of its time saying, "Is it time yet? Is it time yet? Is it time yet?" (That's the second, third and fourth instruction.)

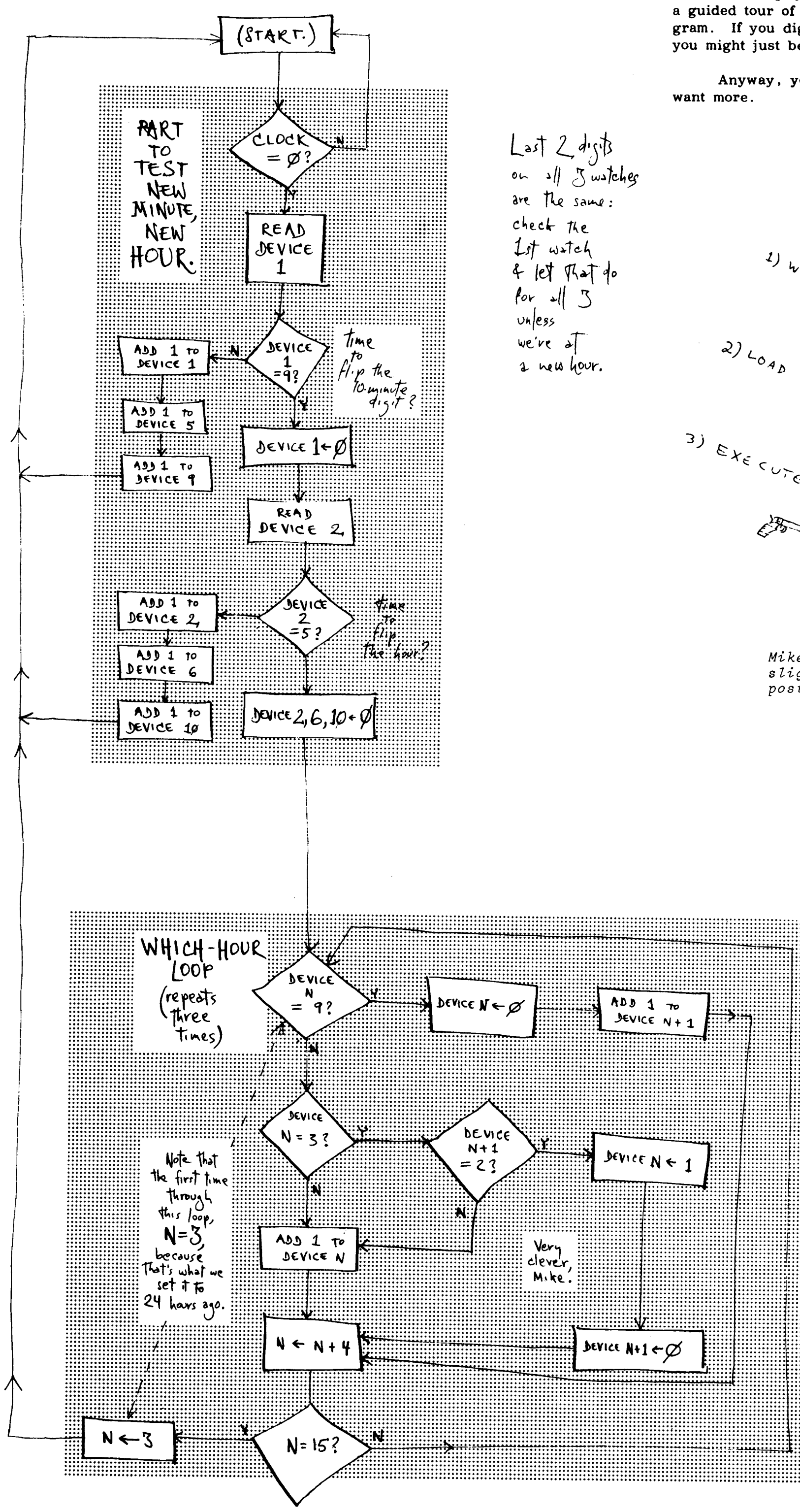
Because the FIDO selects the particular input-output device with the last seven bits of an input or output instruction, this has been done with "address modification" arithmetic: creating an output instruction to address a particular device by adding the instruction to the name of the device. This is an ancient and honorable programming trick.

In several cases, the program chooses a device to examine, or fill, by taking a blank input or output instruction (kept at locations X OXO XOX and X OXO XXO, respectively) and adds it, in the AC, to a counting number that is being used to step around in the array of numerals. (This counting number is "N," stored in location X OXO XXX.) (These instructions were put into the slots in octal form, as "6ØØØB" and "62ØØB" respectively. The slashes are meant to distinguish zeroes from Ohs. The "B" at the end (in the assembly listing) means that the assembler is supposed to translate these numbers to Binary, taking them three bits at a time: 6 Ø Ø Ø comes out to XXO 000 000 000.)

Note that in this flowchart

$A \leftarrow 3$

means, "stuff the number 3 into the variable A." A variable is a named location in core memory.

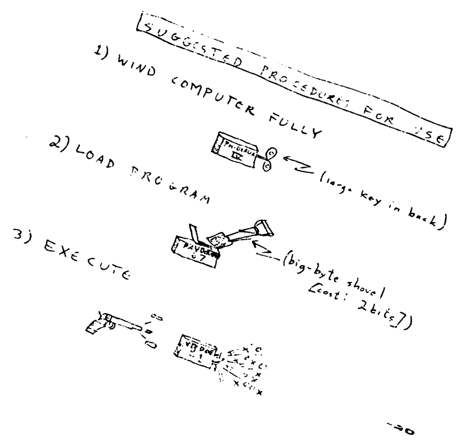


Last 2 digits on all 3 watches are the same: check the 1st watch & let that do for all 3 unless we're at a new hour.

Anyhow, what the program is really doing, when it finds the timer has reached zero, is, testing whether the rightmost digit is a nine. (It only has to test one, since minutes are the same round the world.) If it's not nine, it just adds one to each-- a part of the program called ADMIN, starting at XXO OXO. If it's nine, however, it sets the final digits all to zero, and then tests the tens digit to see if it's a five, meaning the end of an hour. (The number five has been ingenuously stored in a location which Mike has called FIVE, which assembled to slot number X OXO OXO. If you look there, you will see that the slot does, indeed, contain the binary pattern for the number 5.)

What a pity there is no time to take you on a guided tour of this profound, magnificent program. If you dig this sort of thing, however, you might just be able to dope it out.

Anyway, you've had your taste. Hope you want more.



Mike O'Brien's slightly disgruntled postscript to the program.

New-hour checks are performed on all 3 watches by the same loop.

Note that N is the number saying which device we're looking at, "DEVICE N" is its actual contents.

(Note that the variable called N lives in core location X O X O X X X — see next page.)

This is what the program looks like in the computer's core memory. (A printout like the following is called a machine-language listing.)

Since all the addresses are filled in, this program is said to be in absolute binary. If they weren't filled in, it would be called relocatable binary.

Machine-language listings come in different flavors. A binary listing (or dump) is generally in ones and zeroes. An octal listing groups the bits by threes and substitutes the numbers zero through seven for the different combinations of three bits. The other main kind, the hexadecimal listing or dump (an IBM thing), groups the bits by fours and substitutes the numbers 0-9 and the letters A to F, for the sixteen different combinations of four bits.

This is what the program looks like when you set it up for the Assembler, which is the easier way.

A program laid out like this is called an Assembly Listing. Studying it may help you debug (see p. 30).

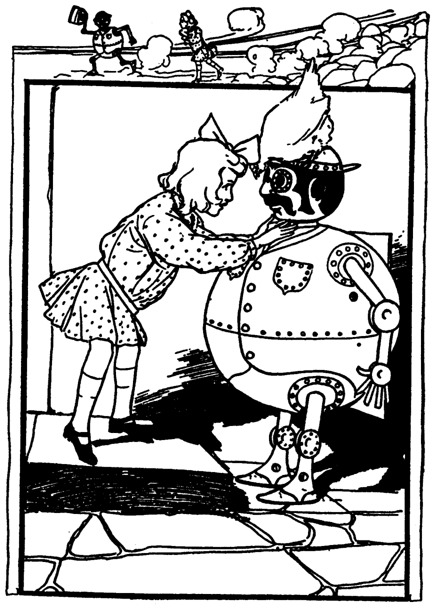
An easy-to-remember alphabetical code is used to represent each final instruction desired. Such an abbreviation is called a mnemonic; usually they're more cryptic. The mnemonics are turned by the assembler into the binary opcode.

You don't have to know the actual addresses in core memory, you just use alphabetical names or labels, and the Assembler figures out where they really go and puts in the binary addresses.

Desired numbers, such as 9, are plugged into the address parts of instructions.

YOUR OWN COMMENTS (here set off with slashes) can stay here too.

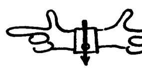
In this FIDO example, the Assembler follows two common practices: it recognizes a label because it ends in a comma, and recognizes a comment because it begins with a slash.



"THIS COPPER MAN IS NOT ALIVE AT ALL"

Bucky's Wristwatch in BINARY

Bucky's Wristwatch in ASSEMBLY LANGUAGE



ADDRESS (slot no. in core memory)	CONTENTS (actual veritable authentic BITS — here shown non-numerically) CORE MEMORY	LABELS (names programmer gave slots)	OP NAMES ('Mnemonics')	PROGRAMMER'S COMMENTS so he doesn't forget, or the next guy can tell.
OOO	XXXXXOOOOOOO	START,	CLEAR	
OOX	XXOOOOOOOOOO	CHKCL,	INPUT Ø	/CLOCK IS I/O SLOT #OOOOOOO.
OXO	OOOOXXOOXXOX		TEST ZERO	/A NEW MINUTE?
OXX	XOXOOOOOOOOX		JUMP CHKCL	/NO, CHECK CLOCK AGAIN.
XOO	XXOOOOOOOOOX		INPUT 1	/YES, READ MINUTE SLOT OF 1ST WATCH.
XOX	OOOOXXOXOOXX		TEST NINE	/IS IT A 9?
XXO	XOXOOOXXOOXO		JUMP ADMIN	/NO, GO TO MINUTE INCREMENTER
XXX	XXXXXOOOOOOO		CLEAR	/YES, SET EACH
OOX OOO	XXOOXOOOOOOX		OUTPUT 1	/TEN-MINUTE DIGIT
OOX OOX	XXOOXOOOOXOO		OUTPUT 4	/TO ZERO.
OOX OXO	XXOOXOOOXOOX		OUTPUT 9	
OOX OXX	XXOOOOOOOOXO		INPUT 2	/CHECK TEN-MINUTE DIGIT.
OOX XOO	OOOOXXOXOOXO		TEST FIVE	/NEW HOUR?
OOX XOX	XOXOOOXXOXXX		JUMP AD2TEN	/NO, GO TO TEN-MINUTE INCREMENTER.
OOX XXO	XXXXXOOOOOOO		CLEAR	/YES, SET EACH
OOX XXX	XXOOXOOOOOXO		OUTPUT 2	/TEN-MINUTE DIGIT
OXO OOO	XXOOXOOOOXXO		OUTPUT 6	/TO ZERO.
OXO OOX	XXOOXOOOXOXO		OUTPUT 10	
OXO OXO	OOXOOXOXOXXX	ROUND,	ADD N	/GET CLOCK-NUMBER COUNTER
OXO OXX	OOXOOXOXOXOX		ADD INPUT	/AND FORM INPUT INSTRUCTION
OXO XOO	OXXOOOXOOOOX		STORE IN1	/PUT IT WHERE IT BELONGS.
OXO XOX	OOXOOXOOXXXO		ADD ONE	/FORM OTHER INPUT INSTRUCTION.
OXO XXO	OXXOOOXOOXXX		STORE IN2	/PUT IT WHERE IT BELONGS.
OXO XXX	OXXOOOXXXXXO		STORE IN2P1	/HERE TOO.
OXX OOO	XXXXXOOOOOOO		CLEAR	
OXX OOX	OOXOOXOXOXXX		ADD N	/GET COUNTER AGAIN.
OXX OXO	OOXOOXOXOXXO		ADD OUTPUT	/AND FORM OUTPUT INSTRUCTION.
OXX OXX	OXXOOOXOXXOX		STORE OUT1	/PUT IT HERE WHERE IT BELONGS.
OXX XOO	OXXOOOXXOOOO		STORE OUT1P1	/AND HERE.
OXX XOX	OXXOOOXXXXOX		STORE OUT1P2	/HERE TOO.
OXX XXO	OOXOOXOOXXXO		ADD ONE	/FORM OTHER OUTPUT INSTRUCTION.
OXX XXX	OXXOOOXOXOXX		STORE OUT2	/PUT IT WHERE IT BELONGS.
XOO OOO	OXXOOXOOOOOO		STORE OUT2P1	/HERE TOO.
XOO OOX	OOOOOOOOOOOO	IN1,Ø		/BECOMES "INPUT N"
XOO OXO	OOOOXXOXOOXX		TEST NINE	/IS HOUR DIGIT A 9?
XOO OXX	XOXOOOXOOXOX		JUMP PAST	/NO, TEST AGAIN
XOO XOO	XOXOOOXXXXOO		JUMP AD10HR	/YES, GO FLIP 10-HOUR DIGIT
XOO XOX	OOOOXXOXOOOO	PAST,	TEST THREE	/IS HOUR DIGIT A 3?
XOO XXO	XOXOOOXOXXXX		JUMP INCHR	/NO, GO INCREMENT HOUR.
XOO XXX	OOOOOOOOOOOO	IN2,Ø		/BECOMES "INPUT N+1."
XOX OOO	OOOOXXOOXXXX		TEST TWO	/IS TEN-HOUR COUNTER A TWO?
XOX OOX	XOXOOOXOXXXX		JUMP INCHR	/NO, INCREMENT HOUR NORMALLY
XOX QXO	XXXXXOOOOOOO		CLEAR	/YES, IT WAS 23:59, SO SET
XOX OXX	OOOOOOOOOOOO	OUT2,Ø		/TIME TO Ø1:ØØ. "OUTPUT N+1" IS HERE.
XOX XOO	OOXOOXOOXXXO		ADD ONE	/SET AC TO 1.
XOX XOX	OOOOOOOOOOOO	OUT1,Ø		/AND "OUTPUT N" HERE.
XOX XXO	XOXOOXOOOOOX		JUMP INCN	/GO INCREMENT CLOCK-NUMBER COUNTER
XOX XXX	OOXOOXOOXXXO	INCHR,	ADD ONE	/ADD 1 TO HOUR
XXO OOO	OOOOOOOOOOOO	OUT1P1,Ø		/BECOMES "OUTPUT N".
XXO OOX	XOXOOXOOOOOX		JUMP INCN	/GO INCREMENT CLOCK-NUMBER COUNTER
XXO OXO	OOXOOXOOXXXO	ADMIN,	ADD ONE	/ADD 1 TO MINUTE DIGIT.
XXO OXX	XXOOXOOOOOOX		OUTPUT 1	/AND PUT IT
XXO XOO	XXOOXOOOOXOX		OUTPUT 5	/IN ALL
XXO XOX	XXOOXOOOXOOX		OUTPUT 9	/THE MINUTE DIGITS.
XXO XXO	XOXOOOOOOOOX		JUMP CHKCL	/THEN GO BACK TO CLOCK-WATCHING.
XXO XXX	OOXOOOOOXXXO	AD2TEN,	ADD ONE	/ADD 1 TO TEN-MINUTE DIGIT
XXX OOO	XXOOXOOOOOXO		OUTPUT 2	/AND PUT IT
XXX OOX	XXOOXOOOOXXO		OUTPUT 6	/IN ALL
XXX OXO	XXOOXOOOXOXO		OUTPUT 1Ø	/THE TEN-MINUTE DIGITS.
XXX OXX	XOXOOOOOOOOX		JUMP CHKCL	/THEN GO BACK TO CLOCK-WATCHING.
XXX XOO	XXXXXOOOOOOO	AD1ØHR,	CLEAR	/FIRST CLEAR
XXX XOX	OOOOOOOOOOOO	OUT1P2, Ø		/HOUR DIGIT (BECOMES "OUTPUT N")
XXX XXO	OOOOOOOOOOOO	IN2P1, Ø		/THEN GET TEN-HOUR DIGIT
XXX XXX	OOXOOXOOXXXO		ADD ONE	/AND ADD 1 TO IT.
X OOO OOO	OOOOOOOOOOOO	OUT2P1,Ø		/BECOMES "OUTPUT N+1".
X OOO OOX	XXXXXOOOOOOO	INCN,	CLEAR	/ROUTINE TO GET NEXT CLOCK NUMBER.
X OOO OXO	OOXOOXOXOXXX		ADD N	/ADDING FOUR TO CLOCK NUMBER
X OOO OXX	OOXOOXOXOOOX		ADD FOUR	/TAKES US TO NEXT CLOCK.
X OOO XOO	OOOOXXOXOXOO		TEST FTEEN	/HAVE WE RUN OUT OF CLOCKS (N=15)?
X OOO XOX	XOXOOXOOXOXX		JUMP STORN	/NO, GO STORE N AND RETURN
X OOO XXO	XXXXXOOOOOOO		CLEAR	/YES, SET
X OOO XXX	OOXOOXOXOXXX		ADD N	/N=3
X OOX OOO	OOXOOXOXOOOO		ADD THREE	/AND RETURN
X OOX OOX	OXXOOXOXOXXX		STORE N	/TO START OF PROGRAM
X OOX OXO	XOXOOOOOOOOX		JUMP CHKCL	/(WE'VE DONE CHECKING CLOCKS).
X OOX OXX	OXXOOXOXOXXX	STORN,	STORE N	/STORE NEW CLOCK-NUMBER COUNTER
X OOX XOO	XOXOOOOXOOXO		JUMP ROUND	/AND SERVICE NEXT CLOCK. END OF MAIN PROGRAM.
X OOX XOX	OOOOOOOOOOOO	ZERO, Ø		/ THESE ARE CONSTANTS.
X OOX XXO	OOOOOOOOOOOX	ONE, 1		
X OOX XXX	OOOOOOOOOOXO	TWO, 2		
X OXO OOO	OOOOOOOOOOXX	THREE, 3		
X OXO OOX	OOOOOOOOOXOO	FOUR, 4		
X OXO OXO	OOOOOOOOOXOX	FIVE, 5		
X OXO OXX	OOOOOOOOXOXO	NINE, 9		
X OXO XOO	OOOOOOOOXXXX	FTEEN, 15		
X OXO XOX	XXOOOOOOOOOO	INPUT, 6ØØØB		/RAW INPUT INSTRUCTION. (OCTAL)
X OXO XXO	XXOOXOOOOOOO	OUTPUT, 62ØØB		/RAW OUTPUT INSTRUCTION. (OCTAL)
X OXO XXX	OOOOOOOOOOOO	N, Ø		/COUNTER FOR WHICH CLOCK WE'RE ON.
	⋮			

IF THIS LOOKS FORMIDABLE, TRY OVER HERE.

Thank God for THE ASSEMBLER

Ten minutes after starting to program in Machine Language you will probably want Assembly Language.

It's a pain trying to get all the ones and zeroes right. (Exes and Ohs in the example. Same thing.)

It's a pain trying to keep track of binary numbers for where things are stored.

SO: let's give them alphabetical names. That's assembly language. (And the conversion program we put our alphabeticals into, to turn them back into the binary patterns that really run the machine-- that conversion program is called the Assembler.)

An assembler is a direct and non-tricky translator, intended mainly to handle the details of exact transposition between instruction code-words and the exactly corresponding machine-language program that you intend.

IT WORKS LIKE THIS: The assembler scans through the assembly-language program, testing the successive alphabetical characters. After finding the key punctuation marks or delimiters (shown as comma and slash for the FIDO assembler), it scans for the alphabetical instruction mnemonics, and translates them by a table in core memory into the corresponding binary codes. (It ignores everything on a line after a slash, which is lucky, since in the comments you may use words which are the same as instruction mnemonics.)

The assembler also counts the instructions, and (starting wherever you say) figures where in core memory the instructions (and any data or spaces you put in) go. Then it makes a list of these addresses, called a symbol table (also called a name list at less elegant places).

An assembler is the simplest form of compiler (see p. 30). Basically it translates an assembly-language program, which cannot be run directly, into a binary program which can.

Then from this symbol table it fills the resulting binary addresses into the binary commands of the program.

Aren't you glad you don't have to?

Generally the assembler then sends out the binary program to some external device, such as a disk memory or paper tape punch. Then it can be put into core memory when you want to run it.

(You can put a program into core memory one bit at a time through the front-panel switches; but nobody likes doing this except for teeny programs.)

(Note: an assembler for one computer (say the PDP-8) that runs on a different computer (say, the 360) is called a cross assembler.)

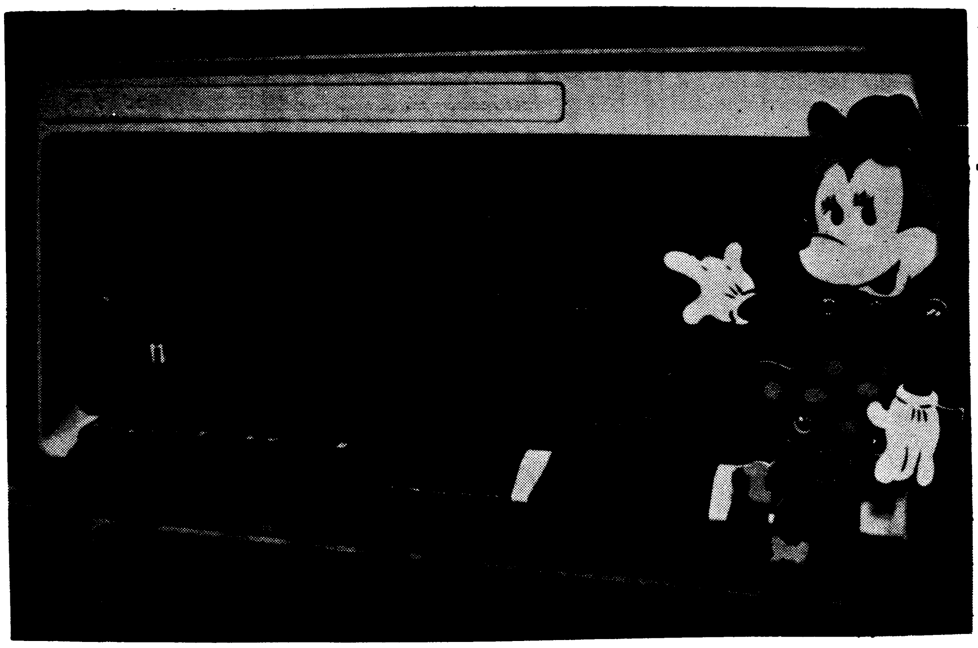
NOW YOU SEE WHY WE USE HIGHER COMPUTER LANGUAGES.

Most people don't like this stuff.

"Assembly language programming is good for the soul."

Folk saying

the MINI



© Walt Disney Productions

This is a PDP-11, one of the world's best-designed minicomputers (see p. 42). The PDP-11 is a 16-bit machine. Shown is Model 45, the fastest PDP-11, which has various special features. Stripped, with 4K of core memory (that's 4096 locations), it costs about $13 grand. A smaller PDP-11 goes for some $5000.

A minicomputer simply means a small computer, no different in principle from the big ones (see next spread), and it can do all the same things except as limited by speed and memory capacity.

(Mind, we are talking about real computers, not the little calculators you hold in your hand that just do arithmetic. A real computer is one which works on stored programs and all kinds of data, working not merely on numbers but on such other things as text, music and pictures if supplied with appropriate programs; see flip side.)

There is some argument over what constitutes a minicomputer; basically we will say it's any computer with a word length of 18 bits or less (see "Binary Patterns," p. 27). (Some companies, like Datacraft and Interdata, are trying to peddle their worthy computers as "minicomputers" even though they're 24 and 32 bits, respectively, but that's very odd. Interdata says any computer under ten thousand is a mini-- which means all computers will be minis by and by; a vexing thing to do to the term.)

Traditionally minicomputers come with much less. In the old days pretty much all the programs you got with it were an assembler (see p. 35) and a debugger (see p. 30) and a Fortran compiler (see p. 31) if you were lucky. Today, though, with minis having highly built-up software like (see pp. 40-42 for descriptions) the PDP-8, the PDP-11 and the Nova, you can get a lot of different assemblers, together with Fortran, BASIC, and a little disk or cassette operating system (see p. 45) to make your life a little easier.

The idea of owning a computer may seem strange to some people, but with prices falling as they are it makes perfect sense. Numerous individuals own minis, and as the price continues to drop the number will shoot up. For several families with children to pool together and buy one for the kids makes a lot of sense. One friend of mine has an 8, another is contemplating an 11. (I've been trying to get my own for years; perhaps this book...) Anyhow, the general price range is now $3000 to $6000 plus accessories, and that's dropping fast. Rental is usually a great mistake: prices are very high and after six months or so you'll have paid for it without owning it. (But names of rental places will be found in this book, and some of them may offer good arrangements.) Minis may now be had in quantity for $1000 each-- price of the PDP-8A in May 1974-- and soon that will be the consumer price.

Unfortunately, the price of the computer itself is dropping faster than that of the accessories, such as the basic terminal you'll need, which still weighs in at $1000-5000. Moreover, as soon as you want to do anything serious you'll need a disk (starting around $4500) or at least a cassette memory (starting around $1500). But these prices too will come way down as the consumer market opens.

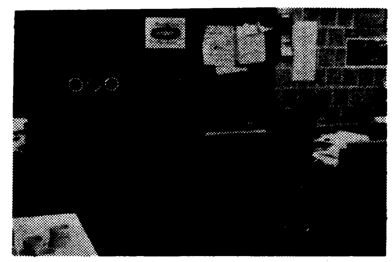
Some of us minicomputer freaks see little real need for big computers. Minicomputers are splendid for interactive and "good-guy" systems (see p. 13); as personal machines, to handle typing and bookkeeping; even for business systems, if you recognize the value of working out your own in BASIC or, say, TRAC Language.

Minicomputers are being put inside all manner of other equipment to handle complex control. (However, for repetitive simple tasks, the latest thing is microprocessors (see p. 44), which cost less but are harder to program.)

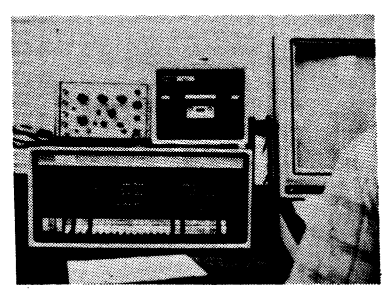
Minicomputers are now being found in highschools; active marketing to highschools is now being done by both DEC and Hewlett-Packard.

Children's museums in Brooklyn and Boston have recently obtained PDP-11s for the kids to interact with. In the Brooklyn case, the computer will even demonstrate the exhibit and help the child discover things about it, in ways worked out by Gordon Pask (see p. DM13).

In the future, networks of minis may be the systems to offer low-cost information services to the home (for speculations, see p. DM57). But minis will also start to make bigger and bigger incursions on the territory of the big machines. For instance, one group proposes a time-sharing system which will simply consist of Novas interconnected in a ring, the so-called STAR-RING, which will supposedly compete with big time-sharing.

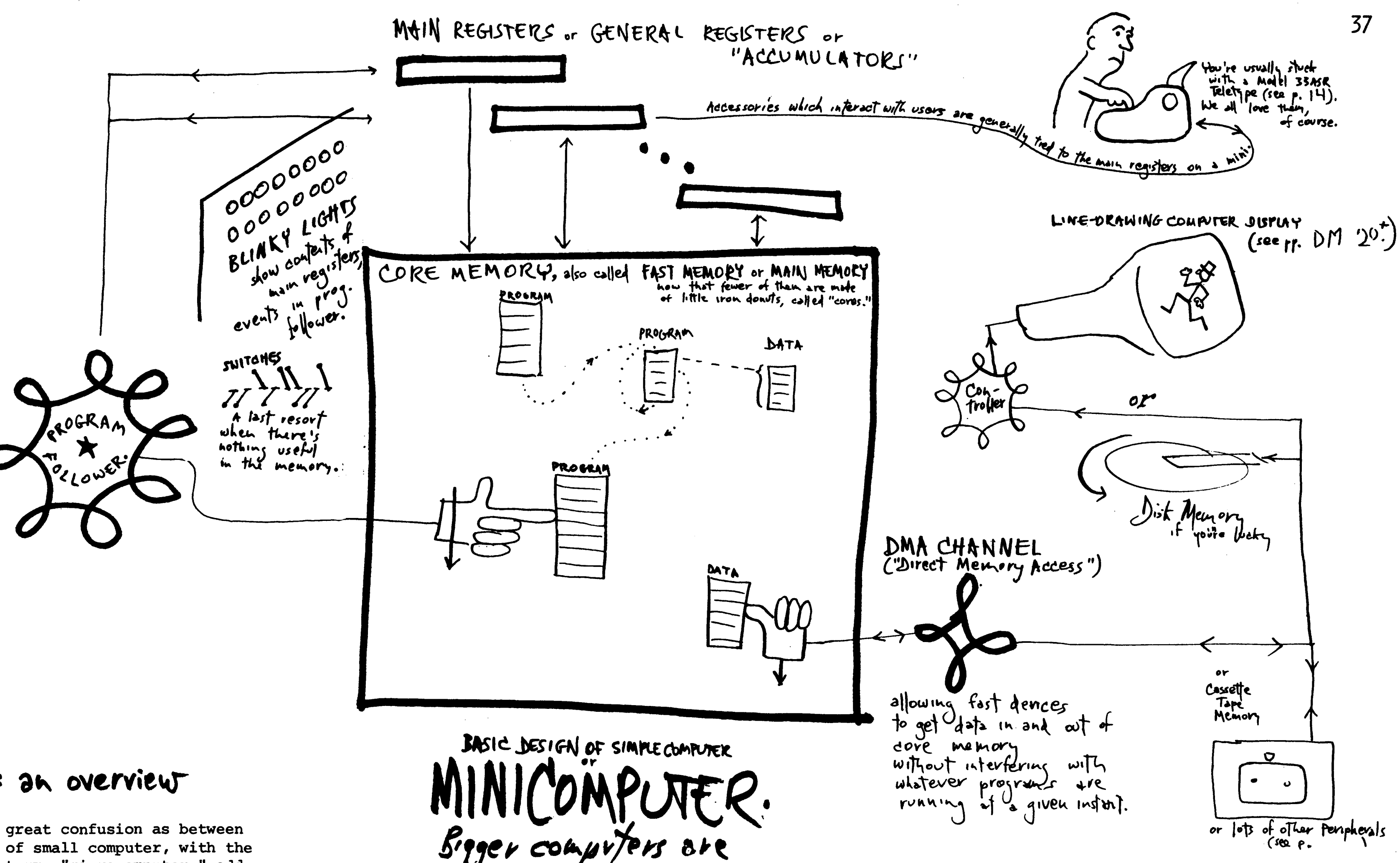


Here's that selfsame PDP-11 in its overall setting. With peripherals shown, plus the magnificent Vector General display (shown later on in book, p. DM31 & elsewhere), this setup cost well over a hundred grand. (This is the Circle Graphics Habitat, otherwise known as the Chemistry Department Computer, U. Illinois at Chicago Circle. Why do chemists need such things? See p. DM31.)



The good ol' PDP-8, perhaps the most popular minicomputer (12 bits). Full PDP-8s now cost about $3000, "kits" less. Shown here with a Sykes cassette tape deck-- a nice, rather reliable unit-- and a screen display (see pp. DM22-3). Courtesy Princeton University & R.E.S.I.S.T.O.R.S. (see p. 47).

Kids love computers. They belong together. This lad flips panel switches on a Nova, perhaps the third most popular mini after the 8 and 11 (16 bits; see p. 41).



BASIC DESIGN OF SIMPLE COMPUTER or

MINICOMPUTER.

Bigger computers are the same but more so.

DINKIES: an overview

There is great confusion as between various types of small computer, with the latest stupid term, "microcomputer," adding to the confusion. We have:

minicomputer or mini
: Traditionally, any computer having an architecture (memory and main registers) of 18 bits or less. Lately, unfortunately, some people have been advertising their 24-bit and even 32-bit computers as minis. This is just confusing.
  (They base this on the fact that "minicomputer" has also referred to a machine sold without a lot of programs. But that's really a separate issue.)

microprocessor
: Two-level computer (see p. 44).

microcomputer
: Crummy term apparently being used to mean any tiny computer, regardless of its structure. Thus all computers will be "microcomputers" in a few years. This clarifies nothing as to their structure or use.

midi computer
: Remember midi skirts? Well, this term has been used for computers larger than 16 bits or faster than usual, by people seeking to give the impression that their machines are bigger than minis and less than biggies. Even the PDP-10 (a genuwine biggie) has sometimes been called a midi.

A product called Cling Free -- comes scented in a spray can, for preventing static in your laundry-- is said to eliminate static electricity in carpeted computer rooms. Spray it all over the rug, especially near the computer, and you won't zapp the computer with sparks from your fingers.

HEY, SOME MINI RENTALS MAY BE REASONABLE

Nova minicomputers are leasable from:

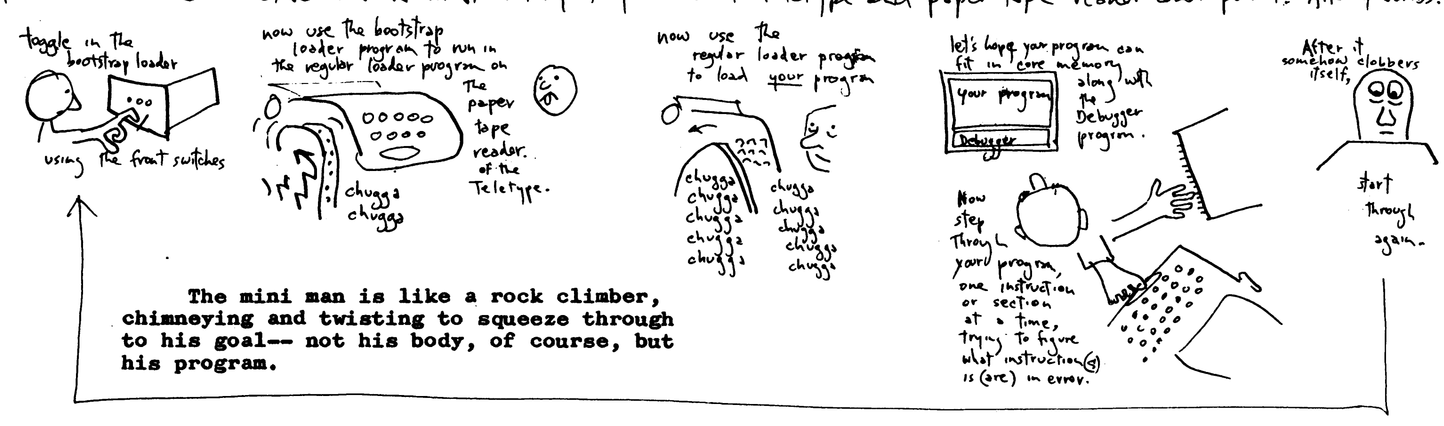
Rental Electronics, Inc.
(a subsidiary of Pepsico)
99 Hartwell Ave.
Lexington, MA 02173

for as little as $250/mo., long-term.

WHERE TO GET 'EM

A long but incomplete list of minicomputer manufacturers is at the bottom of p. 43.

THE FUN OF DEBUGGING ON A MINI with just your usual Teletype and paper tape reader and punch. After it bombs:

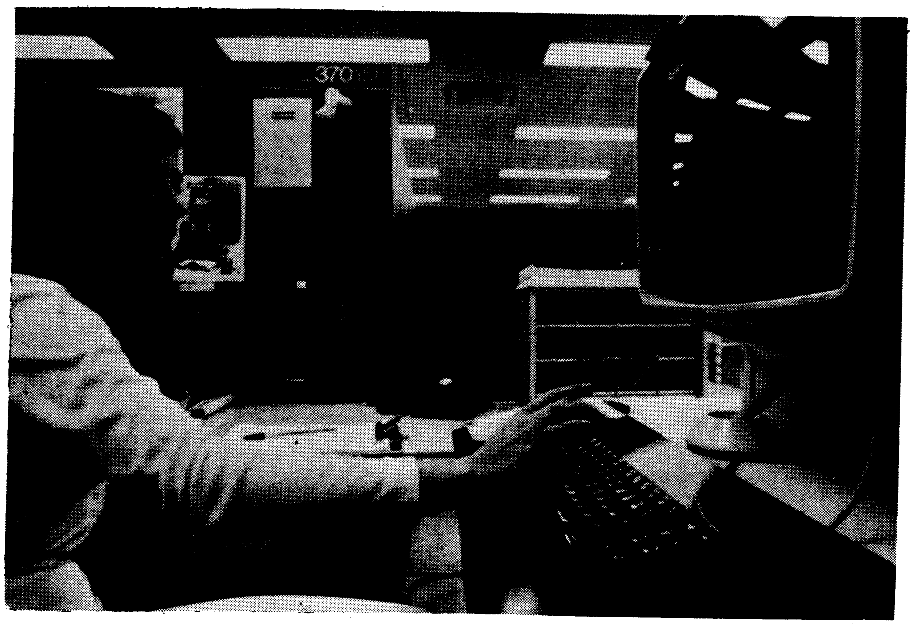


The mini man is like a rock climber, chimneying and twisting to squeeze through to his goal-- not his body, of course, but his program.

THE BIGGIE

Operator's console of this particular setup. The operator may use the keyboard or light-pen (see p. DM23) to select among waiting programs, submitted by various programmers and departments.

© Walt Disney Prod.



The operator muses at the console of the main computer at the University of Illinois at Chicago Circle. It is an IBM 370 model 158, which rents for about $50,000 a month, including all accessories and a dozen or so terminals -- in the parlance of big-computer people, a "medium-sized installation."

This is a big computer.

In principle it's no different from a small one; but it has bigger memories, more registers, more program followers. There are more specialized parts and more things happening at once. (Thus the term "digital computer complex" is sometimes used for a big computer.) It comes supplied with a monitor program or operating system (see p. 45) and a variety of other utility programs and language processors.

Biggies have many ominous and seemingly incomprehensible things to scare the layman.

For one thing, where is the computer? All you see is a lot of roaring cabinets. Which is it?

Answer: all of them. "The computer" is divided among the different cabinets (note diagram and cluster of pictures locating the operator among them, below). The external devices or peripherals (see p. 57) are usually in separate housings. Usually there is one single box or "mainframe" containing core memory, main registers, program-following circuitry, etc., as in the machine illustrated, but these things don't have to be in one box, and sometimes aren't.

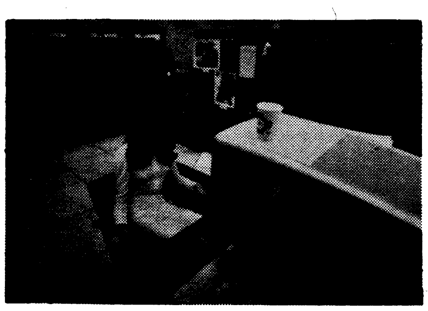
The parts of a computer are set up to be gotten at, to be refilled and repaired. Their innards swing open like refrigerators. Similarly, the wiring of computers is in separate sections or modules ("module" merely being today's stylish term for "unit"), having very orderly connections among them. Individual circuits are on circuit sheets or "cards" which plug in sideways and may be replaced easily. There's nothing really computerish about this, it's merely sensible construction; but it is traditional in other fields to build something as a tangle of wires. (When TV makers follow these rational practices, they call it "space age construction.")

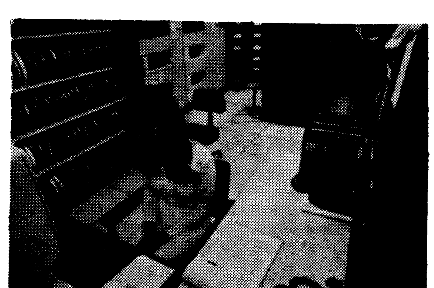
Why are the different parts so far apart? So there's room to swing them open, refill or change them, sit down and repair them. Refrigerators could, and perhaps should, also be built in separate sections, but it's not traditional. Automobiles can't be spread out because they have to endure the jostles of the road. But computers like this baby aren't going anywhere.

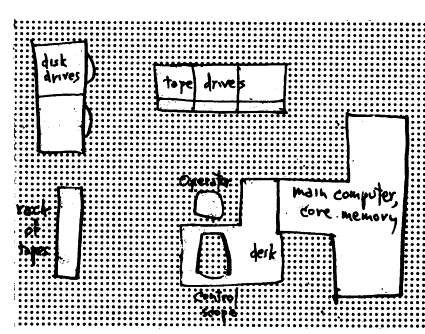
Also intimidating is the fact that you have to step up as you enter a computer room. That's because computer rooms ordinarily have raised floors, permitting cables to be run around among the pieces of equipment without your tripping.

Computer rooms are generally lit by millions of fluorescent bulbs, making them garishly bright. This is simply tradition.

Big computers can have millions of words of core memory. Moreover, there are usually several disk drives and tape drives, as seen in the pictures, used to hold data and programs. (Some of the programs are the system programs, especially the language processors and the operating system-- see p. 45-- but other programs and most of the data belong to the users.)



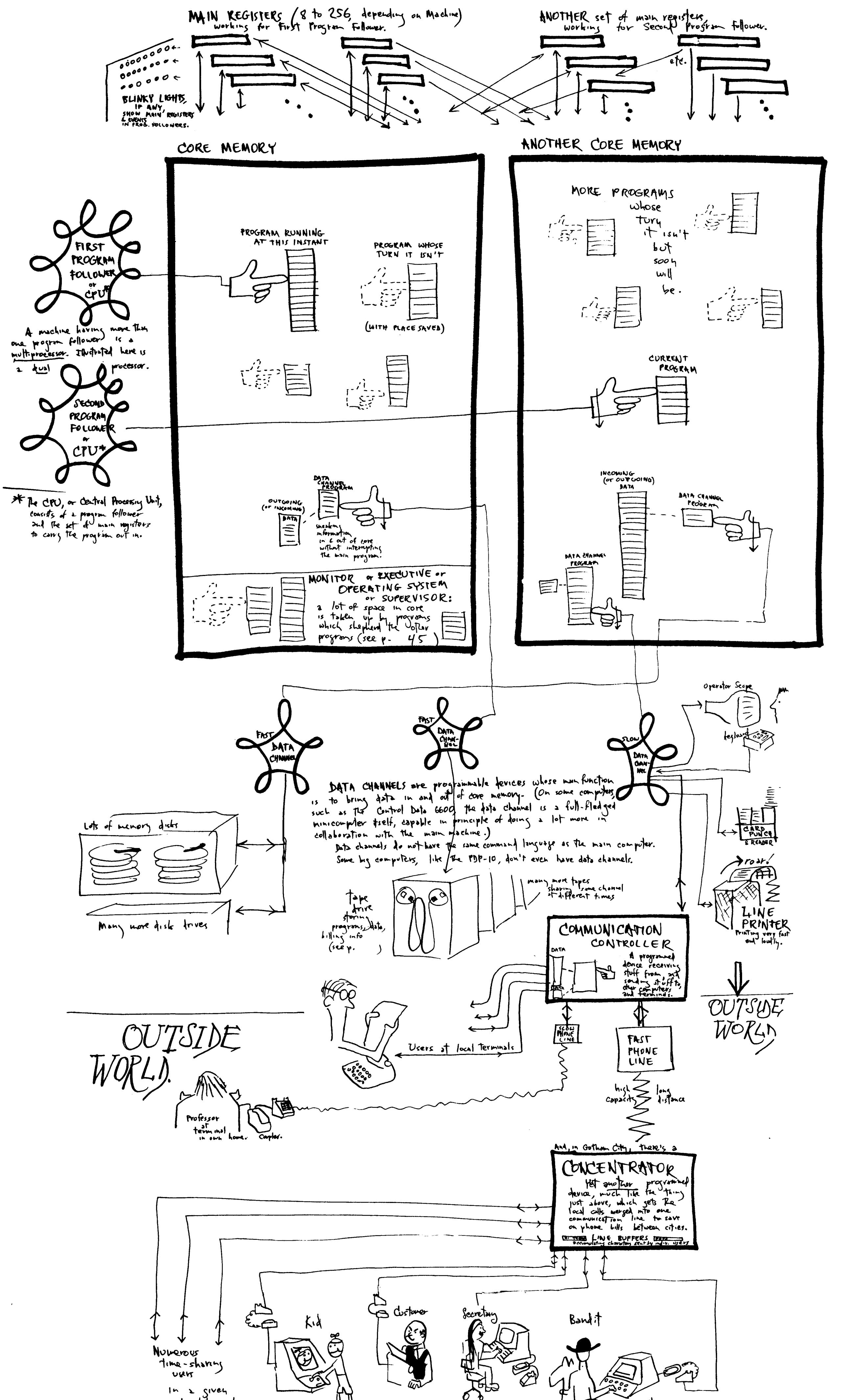




AN OPERATOR IS NOT A PROGRAMMER

Cindy Woelfer is the day-shift operator of Circle's big computer. The job mainly consists of changing disks and tapes, starting and stopping different jobs listed on the scope, and restarting the computer when the system crashes (gratuitously ceases operation).

Ms. Woelfer, a thoughtful person, says she does not find her job very stimulating. She can program, but the job doesn't involve programming. It's also a lonely job. Non-systems people, except Mayor Daley, aren't ordinarily allowed around. About the only people to talk to are the systems programmers who stop through to look at the scope and see whether their programs are up next.



TO BE MORE SPECIFIC,

descriptions of some prominent big computers will be found on the next four pages.

It used to be traditional for machines like this to have many many rows of blinking lights, showing what was in all the main registers at any fraction of a second. But there's really no point in seeing all that, since about all you can tell from it is whether the computer is going or not (if it's not, the lights are stopped) and other high-level impressions. For that reason some big computers, beginning with the CDC 6600, started doing away with the fancy lights and bringing written messages to the operator on a CRT scope instead (for lots more on the glories of CRTs, see the flip side, pp. DM 22.

Big computers can have multiple program followers and sets of registers (a program follower and its main registers are together called a CPU, Central Processing Unit). A computer with two CPUs, i.e., two sets of program followers and registers to carry the programs out, is called a dual processor; a computer with more than two CPUs is called a multi-processor.

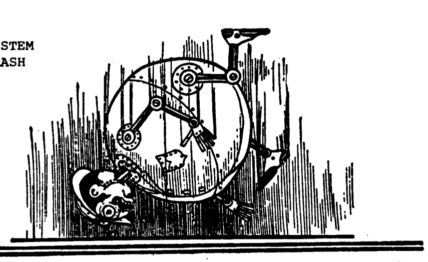
Separate independent sections of core memory may be put in one computer, allowing separate program followers and data channels to work at the same time. (Note: a "bank" of core memory is an independent section. Except in this sense of "core memory bank" or "core bank," there is no other correct usage of the layman's vague term "memory bank." Computer people only say "memories," and distinguish further among core, disk, tape, etc. Note that "data banks" are a separate issue-- see "Issues," p. 58.)

DINOSAURS?

Many computer people, the author included, entertain certain doubts about the long-term usefulness of big computers, since minicomputers are cheaper, especially in the long run, and can actually be in the offices and homes where people create and use the information. Big computers are necessary for time-sharing (see p. 45) and huge "number-crunching" jobs (see "Grosch's Law," nearby). However, it will soon be cheaper to put standardized number-crunching jobs in standalone or accessory hardware; see "Microprocessors," p. 44.

Fans of big computers also argue that they are necessary for business programming, but that only means traditional business programming-- non-interactive and batch-oriented. For tomorrow's friendly and clear business systems, networks of minis may be preferable. But makers of big computers may be unwilling to admit this possibility.

SYSTEM CRASH



Tends to happen several times a day.

GROSCH'S LAW

Minicomputers are so nifty that we may ask why have big computers at all. The answer is that there are considerable economies, especially in applications that require many repetitive operations and don't need interaction with users.

A hypothesis about the economy of big computers was formulated a long time ago by Herbert J.R. Grosch, onetime director of IBM's Watson Lab and now a heavy detractor of IBM. Thus it is called Grosch's Law. The idea is basically that there is a square-law relationship between a machine's size and its power (narrowly defined in terms of the cost of millions of operations, and without considering the advantages of interactive systems or other features which may be of more ultimate value). Anyway, when I asked him recently for his formulation of Grosch's Law, I got the following:

"Grosch's Law (formal): Economy in computing is as the square root of the speed.
(informal): If you want to do it ten times as cheap, you have to do it a hundred times as fast.
(interpretive): No matter how clever the hardware boys are, the software boys piss it away!"

SOME GREAT COMPUTERS

Here, then, are some thumbnail descriptions of some great, classic or popular computers, expanding our basic diagrams as needed.

Individual computers represent variations of the patterns shown so far.

The particular structure of registers, memories and pathways among them is called the architecture of a computer (see p. 32). The binary instructions available to the programmer are called the instruction-set of the particular computer (see p. 33). (The word "architecture" is often used to cover both, including the instruction-set as well.)

The principal variations among computers are the word length (in bits-- see "binary patterns," p. 33) and the number and arrangement of main registers. Then come the details of the instruction-set, especially the ways in which items are selected from core memory -- the addressing structure. Then the instruction-set, whose complications and subtleties can be considerable indeed.

The individual computer is the complex result of all of these. If they fit together well, it is a good design. If they fit together poorly, it is a bad design. A bad design is usually not so much a matter of overt stinky features as of ramifications which fit together disappointingly. (Glitch is a term often used for such stinky features or relationships.)

The possible ways of organizing computing hardware are vast, and only partly explored. (An aside to computer guys: on the Intel chip debugging consoles they have an address trap (trapping on a presettable effective address) and a pass counter (trapping after n passes). How come we haven't seen these sooner?)

The machines mentioned here are an arbitrary selection. Some of them are the Great Numbers, computers so important that folks use their numbers as proper nouns, with no brand name:

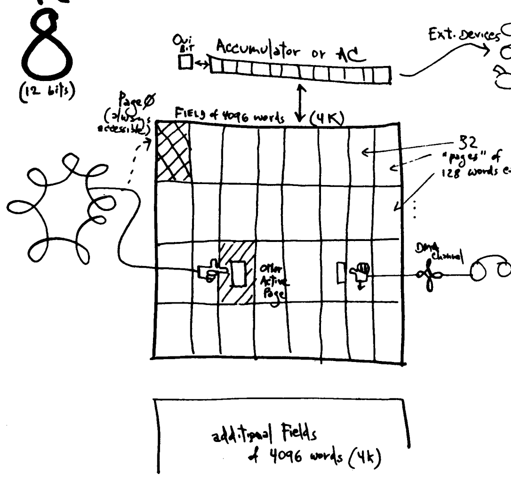
"Do you have a 360 up there?"

"No, but there's a 6600, a 10 and a bunch of 8s."

"Personally, I'd rather work on a 5500."

Here is what they are talking about.

the 8 (12 bits)



The PDP-8 was designed by Gordon Bell (in its original version, the PDP-5) about 1960. Originally it cost about $25,000; as of May 1974 that price is down to about $3000, or less than a thousand dollars if you want to buy the circuits and wire it all up yourself. Yup, here comes that Heathkit.

The PDP-8 has been DEC's hottest seller; you'll find them in industrial plants and museums, or even hidden in the weirdest equipment, from typesetting devices to big disk drives. At universities all over there are kids who know them inside out.

Today the PDP-8 seems archaic, with its one accumulator and awkward addressing schemas: you can only get to 256 different addresses in core memory directly, and it's chopped up into pages. But for its time it was a brilliant design, packed like a parachute, and even today there are people who swear by it. (But look at what Bell's done lately: the PDP-11.)

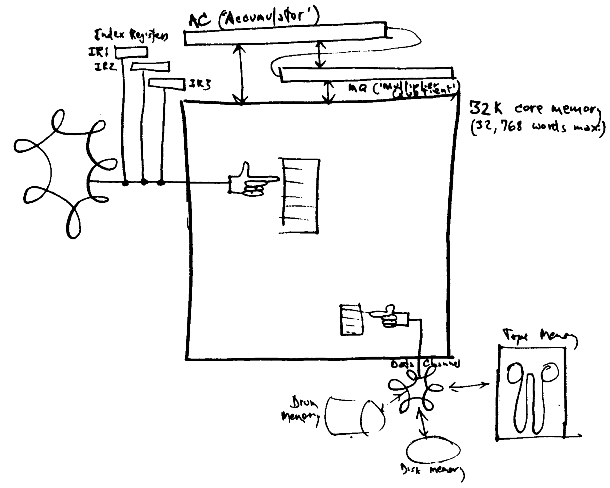
So many programs exist for the PDP-8, though, and so much sentimental fondness, that it will be with us for the foreseeable future. Thus the "Bucky's Wristwatch" example (see p. 34-5) is not totally frivolous: we may assume that a PDP-8 on one or two wristwatch-sized chips is only a year or so away. But let's hope they do the 11 first.

(Lookalikes available from Digital Computer Controls and Fabri-Tek.)

the 90 & 94 (36 bits)

The IBM 7090 was the classic computer. Introduced about 1960 and mostly gone by '66, it was simple and powerful, with clean and decent instructions. With its daughter the 7094, it became virtually standard at universities, research institutions and scientific establishments. At many installations that went on to 360s they long for those clearminded days.

The 90 had three index registers and fifteen bits to specify core addresses. (This meant, of course, that core memory could ordinarily be no longer than 32,768 words ("32K"-- see "Binary patterns," p. 33.) A later model, the 94, went up to 7 index registers, since there were three bits to select them with.



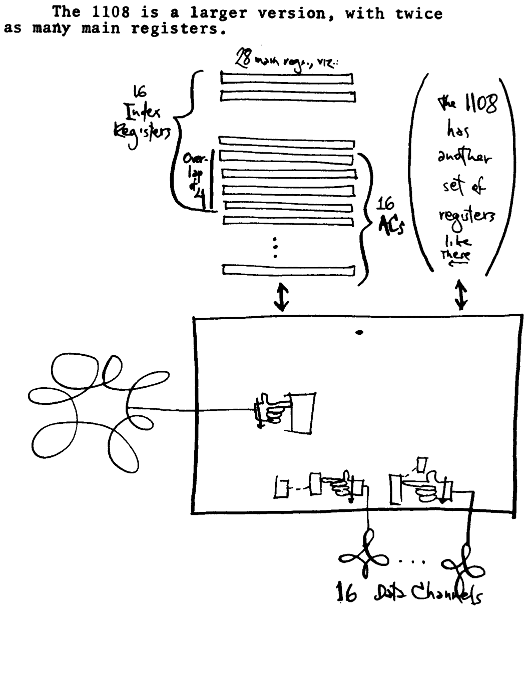
Though these were million-dollar machines ten years ago, you now hear of them being offered free to anyone who'll cart them away; partly because they needed a lot of power, airconditioning and oso on. But they were great number crunchers. (If you want a 90, I believe that 90 lookalikes are still available from Standard Machines in California.)

the 1106 "Eleven Oh Six" (36 bits) & 1108.

Univac's 1106 and 1108 are fast, highly regarded machines. In designing the computer Univac did a clever thing: they built an upgraded 7094. This meant (as I understand it) that all the programs from the old 7094 will run on it. But instead of two main registers they have 28.

(Where they found the bits in the instruction word to select among all those registers I can't tell you.)

The 1108 is a larger version, with twice as many main registers.



THE 10, formerly the 6 (36 bits)

DEC's PDP-10 is in some ways the standard scientific computer that the IBM 7094 was in the sixties.

The PDP-10 is excellent for making highly interactive systems, since it can respond to every input character typed by the user.

It is a favorite big computer among research people and the well-informed. The ARPANET, which connects big computers at some of the hottest research establishments, is largely built with PDP-10s. There are PDP-10s at MIT, U. of Utah, Stanford, Yale, Princeton and Engelbart's shop (see p. DM46). The Watkins Box (see p. DM33) hooks to a 10.

Digital Equipment Corporation, aware that its computer trademark "PDP" connotes minicomputers to the uninformed, now wants the 10 to be called DECsystem-10 rather than PDP. We'll see if that catches on.

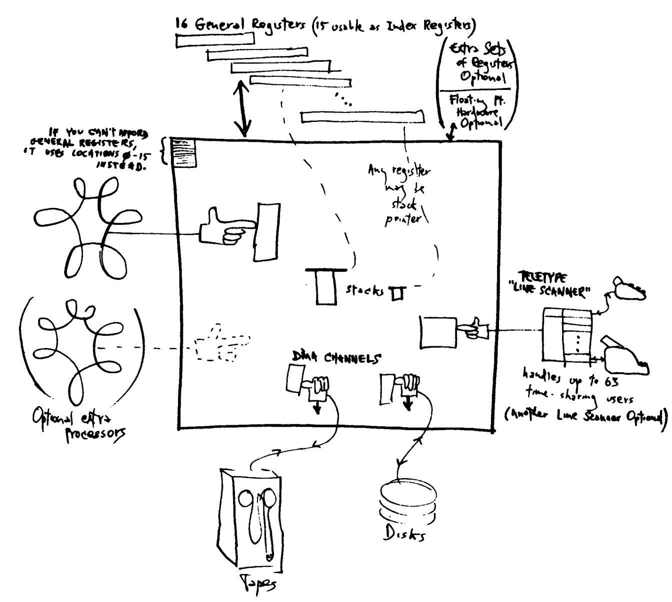
Who designed it is not entirely clear. I've heard people attribute it variously to the Model Railroading Club at MIT, to Gordon Bell, and one Alan Kotok.

Originally it was the PDP-6, which appeared about 1964, and was the first computer to be supplied with a time-sharing system, which worked from the beginning, if rockily. Now it's good and solid. DEC's operating system for it (see p. 45) is called TOPS, but BBN sells one called TENEX, also highly regarded. The 10 does time-sharing, real-time programming and batch processing simultaneously, swapping to changeable areas of core memory. (This feature should soon be available, at last, on IBM computers ("VS2-2").)

PDP-10 time-sharing works even if you don't have a disk, using DECtape (DEC's cute little tapes). Of course, without disk it's really hobbling, but this capacity is nevertheless noteworthy.

The PDP-10 has debugging commands which work under time-sharing and with all languages, and hugely simplify programming.

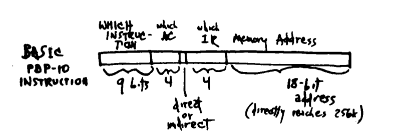
Unlike the IBM 360, whose hardware protection comes in options, the 10 has seven levels of protection: the user can specify who may read his files, run them, change them, and do four other things. The PDP-10 does have job control commands, but they are not even comparable in cumberosity to IBM's JCL Language (see p. 31), and they are the same for all three modes of operation: time-sharing, real-time and batch.



The PDP-10 has 36 bits but has instructions to operate on chunks, or bytes, of any length. It has sixteen main registers, as does the 360, but uses them more efficiently.

The PDP-10 also has unlimited indirect addressing: an instruction can take its effective address from another location, which can in turn say to take its effective address elsewhere, ad infinitum. For your heavy tight elegant stuff.

Perhaps most important, the 10 has a full set of stack instructions (see "The Magic of the Stack," p. 42), allowing programmers to use multiple stacks for purposes of their own. (The operating system's own stacks are protected.) Programmers do not have to save each other's registers, as on the 360. Programmers are relatively safe from each other.



Some think of the PDP-6 and 10 as a glorified 7094 (with 18 addressing bits, instead of 15). In this case we might consider the 360 a stripped-down version of the 6, since IBM threw out the stack and in most models the memory mapping.

PDP-10s are ordinarily sold where the views of scientists and engineers are considered important, and comptrollers do not have first choice. Nevertheless, some say that its business-programming facilities (i.e., COBOL, duh) are just as good as those of companies who claim to have designed computers "for all purposes." First National City Bank of New York has found that the PDP-10 makes a splendid banking computer for internal use, profitable at an internal charge of $3.75 an hour plus processing charges. Prices for a PDP-10 system with disk start start about $500,000, or $15 grand a month, and go up into the millions.

However, DEC salesmen are not like IBM's, who can reputedly sell Eskimos to iceboxes. For one thing, DEC salesmen are on salary. That fits DEC's demure, aw-shucks image, but it doesn't exactly sell big computers.

(For you Firesign Theater fans, the mutterings of the dying computer on the "Bozos" album are various PDP-10 system thingies, artistically juxtaposed.)

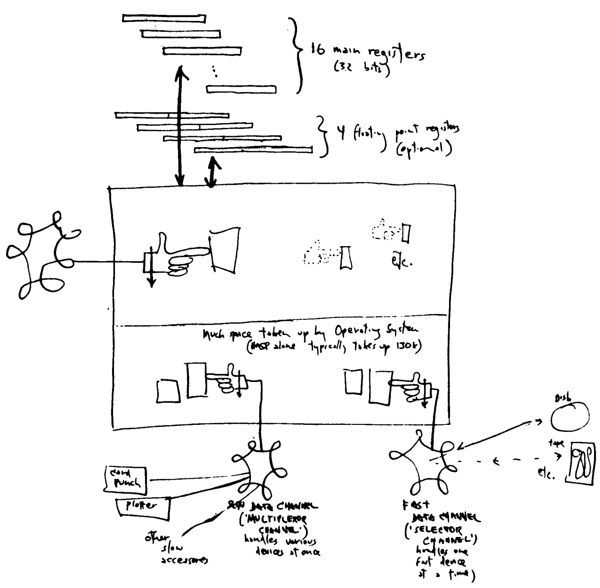
the 360 & 370 (32 bits, 16 bits, 8 bits, 4 bits)

"No corporation except IBM could sell a computer like this." -- A friend.

The IBM 360 (now called 370 because we're in the 70s) is the commonest and most successful line of computer in the world. This does not necessarily mean it is the best. There are those who appreciate IBM typewriters but not their computers.

360s are bought because the repair service is great; because IBM has very tough salesmen; and possibly for other reasons (see pp. 52-6).

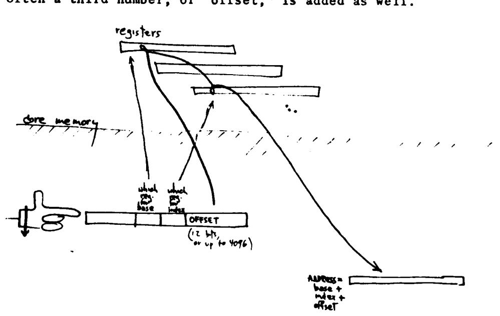
A strange unseen curse seems to haunt the 360 series; indeed, some cynics even think it results from deliberate policies of IBM! Yet the 360 (and its software) seem somehow organized to make programs inefficient and slow; to make programs big, needing lots of core memory (with numerous enticements for the programmer to take up more); to prevent the compatibilities that are so widely advertised, except through expensive options; to make things excessively complicated, thus locking in both its customers and the employees of its customers to practices and intricacies that are somehow unnecessary on other brands of computer.



The design of the 360, which was basically decent, is generally attributed to Amdahl, Blaauw and Brooks. Those who hate it, and there are many, base their complaints largely on the restrictions and complications associated with its operating system OS, which is notoriously inefficient (see p. 45).

The architecture of the 360 was quite similar to the PDP-6 (now the PDP-10), designed about the same time: sixteen main general-purpose registers of over thirty bits, and using the 16 main registers as either accumulators or index registers.

A curious form of addressing was adopted, called "base-register addressing." This had certain advantages for the operating system that was planned, and was thought to be sufficiently powerful that you wouldn't need Indirect Addressing. Two main registers were required, one holding a "base" more or less equal to the program's starting address, and an "index register," whose contents are added to the base to specify an address. Often a third number, or "offset," is added as well.



The idea of this technique is that programs can be "relocatable," operating anywhere in core memory. A few instructions at the beginning of each program can ascertain where it is running from, and establish the Base accordingly.

The basic idea of the 360 seems to have been doped out for multiprogramming, or the simultaneous running of several programs in core, a feature IBM has pushed heavily with this computer.

WHAT'S WRONG WITH THE 360?

The main differences between the 360 and the PDP-6 and 10 represent conscious and legitimate and arguable design decisions. To fans of the PDP-6 and 10, here are the 360's main drawbacks:

NO INDIRECT ADDRESSING. This was because, within the addressing scheme adopted, indirect addresses could not be adjusted automatically. (But it also makes programs more inefficient, thus more profitable to IBM.)

NO STACK. Why? Too expensive, said Amdahl, Blaauw and Brooks in the IBM Systems Journal. Funny, they have stacks on $5000 PDP-11s-- and it would have saved everybody a lot of money on programming.

NO MEMORY MAPPING (except on certain models). Where the PDP-6's successor, the PDP-10, automatically takes care of redistributing addresses in core to service every program as if it were operating from location zero on up, the 360 left this general problem to local programmers and (on certain levels) to operating systems.

Handling this automatically in the PDP-10's hardware obviates the complications of base-index addressing and makes possible the efficiencies of indirect addressing.

LOOKALIKES

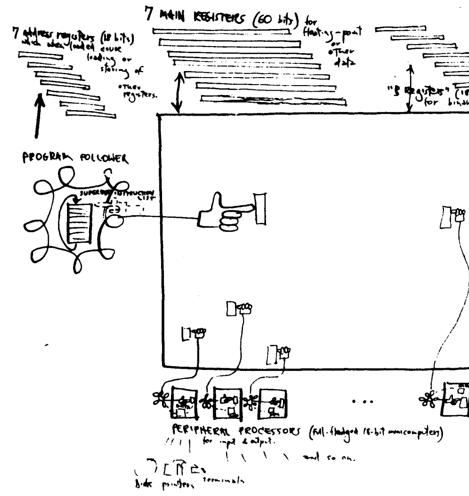
360 lookalikes were sold by RCA and Univac. Now that RCA no longer makes computers, Univac is servicing the ones they made.

And Amdahl, no longer with IBM and now head of the Amdahl Corp., is coming down the pike with a super-360 of his own, in part backed by Japanese money. It will be bigger than IBM's biggest-- and cheaper. (See Hesh Wiener, "Outdoing IBM: the Amdahl Challenge," Computer Decisions, March 73, 18-20.)

THE 6600 ("Sixty-six hundred") (60 bits) FIRST OF THE SUPERCOMPUTERS. Also 6400, 6800.

Control Data's 6600 computer was the first really big computer. The first one was delivered around 1965. The machine and its operating system, CHIPPEWA, were created by Seymour Cray and his team in hinterland Minnesota.

Extreme speed was designed into the computer in a number of ways. The main computer has no input or output at all; this is handled by data channels which have been built up into full-scale minicomputers or "peripheral processors" of 18 bits.

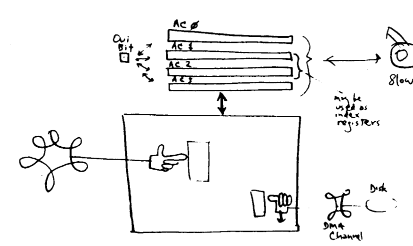


Instructions can be executed at lightning speed, much faster than the usual microsecond or so. However, since core memory is much slower than the main registers, a trick is used: program instructions are drawn from core into a superfast instruction list (often called a cache), and any jumps or loops within this seven-word cache can be executed at unthinkable speeds-- perhaps tens of millions of times per second.

The machine is especially geared for floating-point numbers (see p. 29). Because of the intense speed of the fast instruction cache, many instructions (such as multiplication and division of integers) can be accomplished faster by a short program than if they had actually been wired into the computer.

They 6600 became the start of a whole line, including the 6400, 6800 and others. The 6400 is used by PLATO (see p. DM 18-19).

the NOVA (16 bits)



The Nova came out in the late sixties. Basically the story was this: some of the higher people at DEC, perhaps dissatisfied with DEC's soft sell, perhaps out for their own personal share of things, broke out and started their own corporation. They had in hand the design for a hot, solid minicomputer -- some say it was the rejected design for the as-yet-nonexistent PDP-11-- and since then they have built it reliable and sold it hard.

The basic design of the Nova is sleek and simple: four main registers, no stack, well-designed instructions. Moreover, it was (I think) the first computer to be built around a Grand Bus (see 60X), a design which has caught on rather widely.

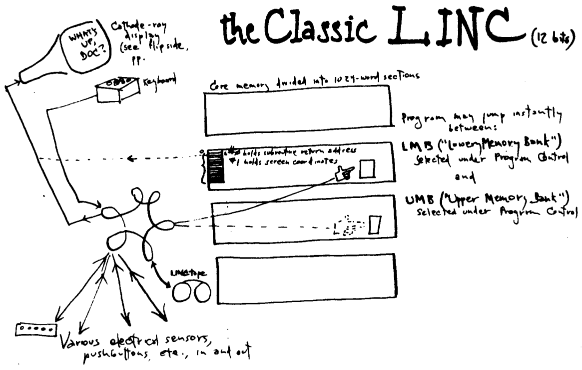
Data General (the company mentioned) has used a very interesting marketing strategy. Instead of bringing out a variety of new computers as time goes on, they concentrate on making the Nova faster and smaller. They began by competing against DEC-- especially in "the OEM market," purchasers who are burying minicomputers in larger equipment they in turn make-- but more recently they have actually started to market against IBM with business systems. In recent months, Data General ads have ridiculed the complexity and mystery of IBM systems, arguing quite rightly that minicomputers programmed in BASIC are a reasonable alternative for a wide variety of business applicatons.

The Nova's instruction-set is clean and straightforward. Key examples (first bits only):

00000	Jump (thus an all-zero instruction jumps to loc $\emptyset$)
0000X	Subroutine jump
000X0	Increment, skip if zero
000XX	Decrement, skip if zero
00X	Load AC
0X0	Store AC
X	Instructions among registers.

One competitor, Digital Computer Controls, sells a Nova lookalike. Whether Data General will sell you its programs to run on it is another question.

the Classic LINC (12 bits)



A computer named the LINC, now usually referred to as "the classic Linc," was perhaps the first minicomputer. It was an important forerunner of our highly interactive systems of today, notably including today's graphic displays with double program followers (see p. DM 23), which offer the highest interactive capabilities.

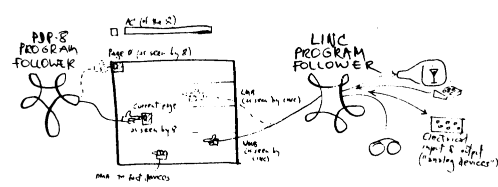
Perhaps most importantly, it was designed with none of the biases that creep in from the traditions of business computing.

It was called the Linc because it was designed at Lincoln Laboratories (about 1960), for "biomedical research"-- actually it was the sort of computer you'd want for hooking up to all sorts of inputs and outputs, to make music, to run your darkroom, but only medical scientists could afford it, so that's what they said it was for.

The LINC had two interesting innovations. It was probably the first computer to be designed with a built-in CRT display (see flip side). It also came with a funny little tape drive, designed for reliability and high response, that was supposed to perform almost as conveniently as a disk and be reliable even in dusty or messy environments. This was the LINCtape, still offered as an accessory by one company. DEC adapted it somewhat and made it the DECtape, handy pocket tape unit of the PDP computer line.

It was never sold commercially. A dozen or so were made up specially out of DEC modules and dealt out to various scientists, and the general hope was that DEC would take the machine up as part of its product line, but that's not what happened. DEC instead pushed its PDP-8 and gave us instead, by and by,

the LINC-8 (12 bits: a marriage of the PDP-8 with the LINC)



DEC was offered the option of building Lincoln Laboratories' classic LINC, but decided instead to combine it, in the mid-sixties, with the already-successful PDP-8. That way all the PDP-8 programs and most of the LINC programs would work on it. The result is kind of strange, but very popular in biomedical research: two computers in one, handing control back and forth as needed. You can write programs on the Linc with sections for the 8, and vice versa. Hmm. A more recent and slicker version is called the PDP-12.

While you might half-think that both sides of the computer could work simultaneously, giving you double speed, it doesn't work that way. There's only one core memory, and that sets the basic speed; either a PDP-8 instruction or a Linc instruction can be underway at once, but not both.

Nevertheless, we see here the double structure that plays such an important part in highly interactive computer displays (see p. DM 23). Indeed, Linc programmers often use the machine just that way: the PDP-8 running an actual program, the Linc part running the CRT display in conjunction with it.

A horrifying and weird picture of an experimental monkey sitting on a PDP-12 and making like the Creature from the Black Lagoon is to be seen in Time, 14 Jan 74, p. 54. It looks very scientific.

BIBLIOGRAPHY

The classic book: C. Gordon Bell and Allen Newell, Computer Structures: Readings and Examples. McGraw-Hill, 1971.

Note that Bell designed various of the PDPs, and Newell pioneered in list processing (see p. 26).

Computer Characteristics Review keeps you in touch with the traits of available computers and peripherals. $25/year (3 issues) GML Corp., 594 Marrett Rd., Lexington, MA 02173.

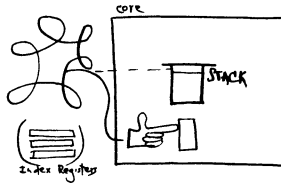
Other firms, such as Auerbach, offer more expensive services of the same nature.

B. Beizer, The Architecture and Engineering of Digital Computer Complexes. Plenum Press, 2 vols., $40.

Heavier than Bell and Newell. A catalog of thousands of structures and tricks, emphasizing the tradeoffs among them.

the Very Great 5000

& 5500.
(Bit length obscure)



I have heard no computer more widely praised among computer people than the Burroughs 5000 (replaced by the 5500). The 5000 was designed about 1960 by Edward Glaser and Bob Barton. It was designed to be used only with higher languages, not allowing programmers access to the binary instructions themselves. Indeed, it was particularly designed to be used with ALGOL, which would have been the standard language if IBM had allowed it (see p. 31) and is still the "international" language.

Because of this approach, its main registers were to be hidden from the programmer, and attention centered instead upon the stack, a high-level programming device (see box on Stacks). However, index registers were added to make it better for Fortran.

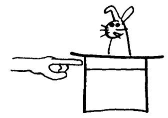
The 5000 was marketed as an "all-purpose" computer with an operating system, anticipating IBM's 360 of a few years later. Indeed, after the 360 was announced, Burroughs sales picked up, because IBM salesmen were at last promoting the concepts that customers hadn't understood when they heard about them from Burroughs salesmen years before.

Bigger machines in the line are now the 6500, 6700...

The Burroughs Corporation continues to be an acknowledged leader in computer design. Apparently their sales force is something else, unfortunately. I once spent some time with a Burroughs salesman who not only knew nothing about the magnificent structure of the machine he represented, but would not get me further information unless I demonstrated that the company I represented (a large corporation) was seriously interested. He wore very fancy clothes.

EVERYTHING IS DEEPLY INTERTWINGLED.

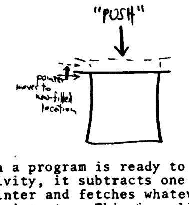
THE MAGIC OF THE STACK



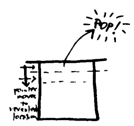
The Stack is a mechanism-- either built into the computer ("hardware") or incorporated in a program ("software") which allows a computer to keep track of a vast number of different activities, interruptions and complications at the same time.

Basically, it is a mechanism which allows a program to throw something over its shoulder in order to do something else, then reach back over its shoulder to get back what it was previously working on. But no matter how many things it throws over its shoulder, everything stays orderly and continues to work smoothly, till it has resumed everything and finished them all.

It goes like this: if the program has to set aside one thing, it puts that one thing in core memory at a place specified by a number called a stack pointer. Then it adds one to the stack pointer, to be ready in case something else has to go on the stack. This is called a PUSH.

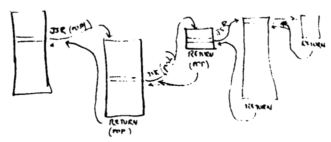


When a program is ready to resume a previous activity, it subtracts one from the stack pointer and fetches whatever that stack pointer points to. This is called a POP.



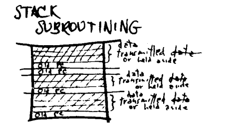
It may not be immediately obvious, but this trick has immense power. For instance, we may stack any number of things together-- the addresses of programs, data we are moving between programs, intermediate results, and codes that show what the computer was doing previously.

Using stacks, programs may use each other very freely. It is possible, for instance, to jump among subroutines-- independent little programs-- willy-nilly, using a stack to keep track of where you've been.

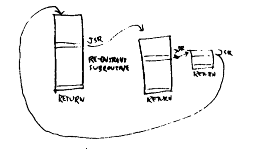


In this case the stack holds the previous locations and intermediate data, so that the program follower can go back where it came from at the end of each subroutine.

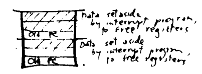
STACK SUBROUTINING



This even makes possible "re-entrant" programs, meaning subroutines that can be used simultaneously by different programs without mixup, and "recursive" programs, meaning programs that manage to call themselves when they themselves are in progress.



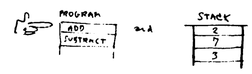
Stacks are also used for handling "interrupts" -- signals from outside that require the computer to set aside one job for another. Having a built-in hardware stack enables the interrupts to pile up without confusion:



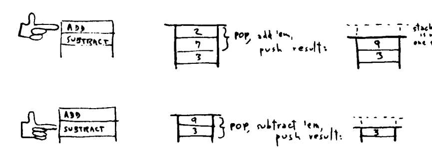
Finally, stack arithmetic, like that done on the Burroughs 5500, enables arithmetic (and other algebraic types of activity) to be handled without setting aside registers or space in core memory. As a simple-minded example on a hypothetical machine, suppose we wanted to handle

2 + 7 x 3

On this machine, let's say, this gets compiled to a program and a stack:



Then the operations are carried out on the stack itself:



Stack programming tends to be efficient, particularly in its use of core memory.

Some languages, such as Algol and TRAC Language, require stacks.

Some computer companies, such as IBM, resolutely ignore stack architecture, though hardware stacks have become widely adopted in the field.

THE GRAND BUS

In electronics, a "bus" is a common connector that supplies power or signals to and from several destinations. In computers, a "bus" is a common connection among several points, using carrying a complex parallel signal.

The Grand Bus, a new idea among computers, is catching on. (The term is used here because the colloquial term, "Unibus," is a DEC trademark.)

Basically the Grand Bus is a connector of multiple wires that goes among several pieces of equipment. So far that's just a bus. But a Grand Bus is one that allows the different pieces of equipment to be changed and replaced easily, because signals to any common piece of equipment just go out on the bus.

This means that the interface problem is deeply simplified, because any device with a proper bus interface can simply be plugged onto the bus.

It does mean a lot more complexity of signals. The Unibus, for example, has about fifty parallel strands. But that means various tricky electrical dialogues can rapidly give instructions to devices and consider replies about their status, in quick and standardized ways.

Prominent grand buses include:

- The Nova bus (nameless; the first?)
- PDP-11's Unibus
- Lockheed SUE's Infibus
- PDP-8's Omnibus.

The idea is great in general. For your home audio equipment, for instance, Grand Bus architecture would simplify everything.

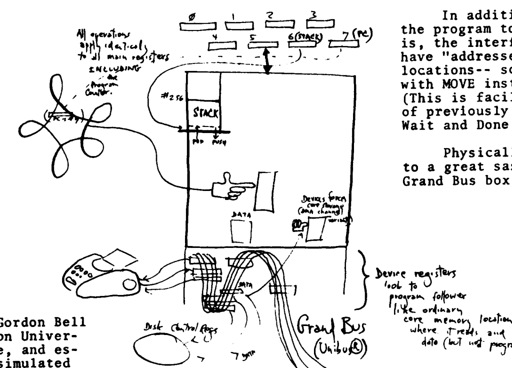
Not only that, but Detroit is supposedly going to put your car's electrical system on a Grand Bus. This will mean you can tell at once what is and isn't working, and hook up new goodies easily.

The 11

(16 bits)

The PDP-11 is not a beginner's computer. But the power and elegance of its architecture have established it, since its introduction in 1970, as perhaps the foremost small computer in the world.

Actually, though, we can't be too sure about the word "small." Because as successive parts of the line are unveiled, it becomes increasingly clear that this line of "small" computers has been designed to include some very powerful machines and coupling techniques among them; and it would seem that we haven't seen everything yet.



In other words, DEC's PDP-11, which has already cut into sales of their PDP-8 12-bit series and PDP-15 18-bit series, may soon cut into its PDP-10 36-bit series-- as designer Bell unveils (perhaps) monster PDP-11s in arrays or double word-length or whatever.

The PDP-11 was designed by C. Gordon Bell and his associates at Carnegie-Mellon University. In designing the architecture, and especially the instruction-set, they simulated a wide variety of possibilities before the final design was decided. The resulting architecture is extremely efficient and powerful (see box, "The 11's Modes").

Basically it is a 16-bit machine, with most instructions operating on 8-bit data as well.

There are eight main registers. Two, though, function specially: the program counter (that part of the program follower that holds the number of the next instruction), and the hardware stack pointer, both follow the same programming rules as the main registers-- an unusual technique. Thus a jump in the program is simply a "move" instruction, in which the next program address is "moved" into main register #7, the program counter.

In addition, all external devices seem to the program to be stored in core memory. That is, the interface registers of accessories have "addresses" numerically similar to core locations-- so the program just "moves" data, with MOVE instructions, to doorways in core. (This is facilitated by the automatic handling of previously bothersome stuff, like Ready, Wait and Done bits.)

Physically all devices are simply attached to a great sash of wires called a Unibus. (See Grand Bus box.)

BIBLIOBRAPHY

R.W. Southern, PDP-11 Programming Fundamentals. (Programmed workbook. No price listed.) Algonquin College Bookstore, 1385 Woodroffe Avenue, Ottawa, Ontario, Canada K2G-1V8.

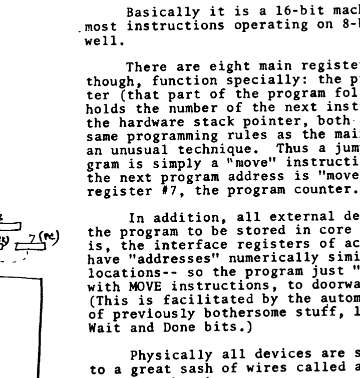
PDP-11 lookalikes are sold by Cal Data. Other firms have been scared off by DEC's patent, but Cal Data say they have a patent too.

Bell peels out: THE 11's MAGIC MODES

Minicomputers are cramped, and so the basic problem in mini architecture is how to cram into the instruction enough choices for getting around in core memory.

In designing the PDP-11, Gordon Bell and his co-workers systematically sought a powerful solution, simulating various possible structures by computer program, trying out a variety of different combinations and structures.

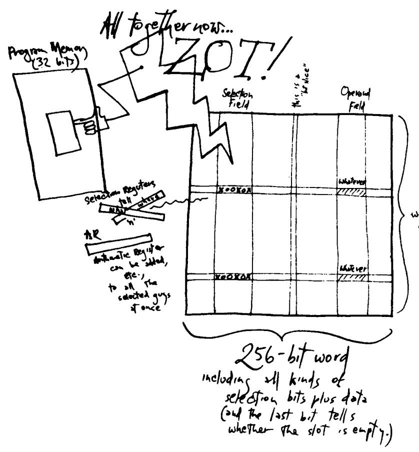
The elegance and power of the solution are little short of breathtaking. Basically the PDP-11, the final design, provides seven different types of indirect addressing. The computer's main registers may be used both to operate on information (the usual technique, here called mode zero), or to point to locations to be operated on (indirect modes 1 through 7). These provide extremely efficient means for stepping through tables, PUSH and POP, dispatch tables, and various other programming techniques. The following diagram is meant for handy reference.



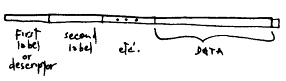
An Exciting New Weirdie the STARAN

There are a lot of strange computers being designed-- it's a traditional occupation of electronics professors and a great way to soak the Defense Department-- but this one is commercially available. Now if we just knew what to do with it.

Goodyear's STARAN is the first available computer with a Content-Addressable Memory, which is actually very hot stuff. Instead of having to search for a particular item of information in core, or having to make lists of where in core things are being put, or creating linked data structures (see p. 26), the program can simply ask all items of data having particular properties to step forward.



It works like this. Having an immense 256-bit word to play with, the programmer uses different parts or "fields" of the word (see p. 28, col. 2) to specify what other information is in it:

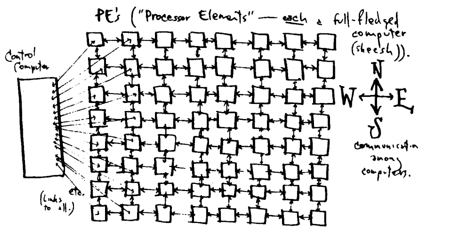


With a single command, the program may ask all words in memory to clear a particular field, or set a particular bit. Then with another command it can tell all memory locations with particular identifiers to add a certain number to their data, and this occurs in a couple of microseconds. Or it can direct all memory locations having particular identifiers to multiply one section of their data by another-- which takes rather longer.

This is an entirely different kind of programming, and considering how much thought computer people have given to doing things one at a time, it kind of sets you back a little. The brochure lists these possible applications: "ballistic missile defense," "intelligence data processing," "electronic warfare," "airborne command and control," as well as more peaceful applications like weather prediction, data management, transportation reservations, air traffic control. Truth is, most computer people would have to scratch their heads quite a while to figure out how to start using this fascinating machine for any of these things; the reason the military applications seem to be so many is simply that the military computer types have been scratching their heads longer. We might as well start too, and find some of the nicer things to do for humanity with it.

Bibliography: Jack A. Rudolph, "A Production Implementation of an Associative Array Processor-- STARAN," Proc. FJCC 72, 229-241.
Contact: Computer Division Marketing, Goodyear Aerospace Corp. Akron, O. 44315.

the ILLIAC 4



The Illiac IV is the biggest and most extraordinary computer in the world, knock wood. To most computer people it's as big as anything they want to think about.

The Illiac 4 consists of sixty-four biggish computers, all going at once under the supervision of yet another big computer, typically all working on a single problem. It is the brainchild of Daniel Slotnick, who worked on the theory of array computers and pressed for its creation for years; eventually built by Burroughs, it sits at an airbase but is available to outside users through the ARPA network.

In principle the idea is this: certain classes of problems, especially those involving very large arrays and matrices, can be run only rather slowly on ordinary computers. If, however, a computer is built which itself is an array, certain operations can take place very much faster because they happen in parallel units simultaneously. Matrices, particular formal kinds of array, are used in a great variety of mathematical-type applications. For instance, weather prediction. It seems that the theory of weather prediction has been well worked out for decades, but because the swirly behavior of the atmosphere is so intricate, actually calculating out everything involves billions of operations. At one conference session I believe it was explained that it used to take twenty-five hours to predict the weather twenty-four hours in advance, whi which means you get the answer an hour after it's happened already; now it is possible, using Illiac IV, to do the whole planet's weather in an hour and a half, said the speaker.

Some say that may be its only use and the whole project was inadequately thought out. Others suspect it's really intended as a radar-watcher for the ABM system.

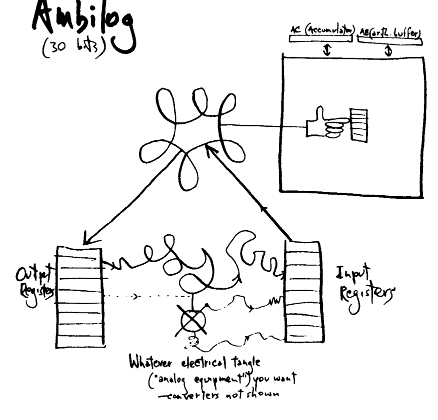
Anyway, there it is. And the individual briefcase-sized Burroughs machines, if they're ever marketed, may provide a new price breakthrough for small highpower systems.

Incidentally, "Illiac" is the traditional name for computers built at the University of Illinois. Will the series end with this one?

BIBLIOGRAPHY

Daniel J. Slotnick, "Unconventional Systems." Proc. SJCC 1967, 477-481.

the Ambilog (30 bits)



An interesting but little-known computer was the Ambilog, made by Adage, Inc. of Boston, a most innovative machine first marketed in the mid-sixties.

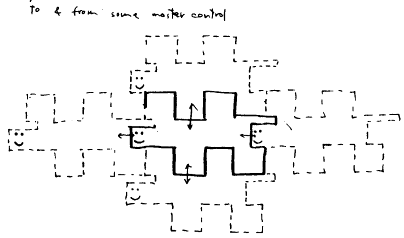
The Ambilog is a hybrid computer, i.e., both digital and analog (note Analog Computers, p. 11), it was mentioned that "analog computers" are any electrical circuits set up to produce a result according to some formula). For certain types of repetitive functions, analog makes a lot of sense. Thus the Adage people put this machine together for highly efficient hybrid computing.

The essential idea was to have a highly ventilated machine that could take in and put out measurable electric signals at high rates. What they created was a rather straightforward digital computer with a lot of registers and converters to send analog information out and bring it back in. This meant that problems suited to repetitive electrical twisting and measurement could gush out through special analog circuits, and the "answers" or doctored signals could gush back in.

The instruction-set was designed for this high-speed management of input and output.

The principal applications this equipment has been used for are three-dimensional display (see Adage Display, p. DM30) and Fourier analysis for sound and other applications (see p. DM24 & DM32).

CELLULAR SYSTEMS



Now that integrated circuits are getting cheap, the distinction between registers (where things happen to information) and memory (where nothing happens to information) can be reconsidered. Storing information in cells that can themselves perform actions, or having numerous subsystems in which computation takes place, leads to a fascinating variety of possible architectures. These are generically called "cellular" computers; this is slightly ironic considering that the living cell itself is now known to be at least a digital memory, and probably more (see p. 60).

Examples of cellular computers more or less include STARAN, ILLIAC IV and the author's own hypothetical FANTASM™ (see p. DM38). But this type of architecture has barely begun.

WHERE TO GET MINICOMPUTERS.

being an incomplete, but not-bad, list of manufacturers.

Company	Some Popular Minis	Comments
Digital Equipment Corp. Maynard, Mass.	PDP-11 (16 bits)	Beautiful, splendid architecture. Current world favorite among sophisticates.
	PDP-8 (12 bits) PDP-15 (18 bits) PDP-12 (12 bits)	Beloved family dog of computer world. Lots of goodies for measuring, dispensing info, switching.
Data General Corp. Southboro, Mass.	Nova (16 bits) & variants: Supernova, Nova 1200, Nova 800.	Reliable. All on one large circuit card. Sleek.
Hewlett-Packard 1100 Wolfe Rd. Cupertino, Cal.	2100 series (16 bits) Variety of other machines, variety of architectures.	
Varian Data Machines 2722 Michelson Drive Irvine, Cal. 92664.	Varian 620 (16 bits) Varian 520 (16 bits) Varian 73.	
General Automation, Inc. 705 West Katella Orange, Cal. 92667.	SPC-16 (16 bits) 18/30 (16 bits)	They claim marvelous architecture, purport to show this in free book, The Value of Power. Copy of IBM 1800/1130.
Texas Instruments, Inc. P.O. Box 1444 Houston, Texas 77001	TI 960, 960A, 980 (16 bits)	Model 960A is fast (750 ns) and under $3000.
Interdata, Inc. 2 Crescent Place Oceanport, NJ 07757	Model 70 (variable word length); line of others	Instructions vaguely copy IBM 360.
Digital Computer Controls 12 Industrial Road Fairfield, NJ 07006	DCC-116 (16 bits) DCC-112 (12 bits)	Nova copy PDP-8 copy
Systems Engineering Laboratories 6901 W. Sunrise Blvd. Ft. Lauderdale, Fla. 33313	810 "Systems 72" (16 bits)	Offers virtual memory (making disk seemingly an extension of core memory). 60-ns mapping registers.
Lockheed Electronics Co., Inc. Data Products Division 6201 East Randolph St. Los Angeles, Cal. 90040	MAC 16 (16 bits) SUE	(See "Microprocessors," p.)
Electronic Processors, Inc. 5050 S. Federal Blvd. Englewood, Colorado 80110	18-bitter	
GRI Computer Corp. 320 Needham St. Newton, Mass. 02164	GRI-99 (16 bits)	
Computer Automation, Inc. 18651 von Karman Irvine, Cal. 92664	Naked/Mini LSI	$2400 w/cabinet! (But no accessories, and roughly 7 μsec cycle time.)
Xerox Data Systems, Inc. Los Angeles, California.	Sigma 2 (16 bits)	
Modular Computer Systems 2709 North Dixie Highway Ft. Lauderdale, Fla. 33308	MODCOMP line (16 bits)	"240" general-purpose registers (presumably in core). Multi-programming.
Computer Terminal Corp. 9725 Datapoint Drive San Antonio, Texas 78284	Datapoint 2200 (8 bits)	Built-in keyboard, video character scope, cassettes. Model 1: 8 μsec, Model 2: 1.6 μsec. No standard languages.
Standard Logic, Inc. 2215 S. Standard Ave. Santa Ana, CA 92707	CASH-8 (16 bits)	600 nsec, starts at $500 (Whee!) and offer a whole setup (4K, TTY, Floppy Disk) for $5000.
Datacraft 1200 Northwest 70th St. Fort Lauderdale, Fla. 33307	Model 6024 (24 bits-- not really a mini)	$11,000.
International Business Machines Armonk, NY	IBM 1130 (16 bits) IBM 1800 (16 bits; beefed-up 1130) System 3 System 7	Both very expensive; hardly minis in price. Sold principally for canned business systems.
Honeywell	H-316 (16 bits) DDP-516 (16 bits) DDP-716 (16 bits)	
General Electric	GEPAC 4010 GEPAC 4020	
Westinghouse	W2500	
Raytheon, Inc.	704 706 707	

TWO USED-COMPUTER COMPANIES

American Used Computer Corporation
P.O. Box 68, Kenmore Station
Boston, MA 02215

Computer Marketing Corp.
467 Sylvan Ave.
Englewood Cliffs, NJ 07632

HERE THEY COME — the MICROPROCESSORS!

COMPUTERS INSIDE COMPUTERS

"Big fleas have little fleas that bite 'em;
And so forth, ad infinitum."
Proverb

Microprocessors are what's happening.

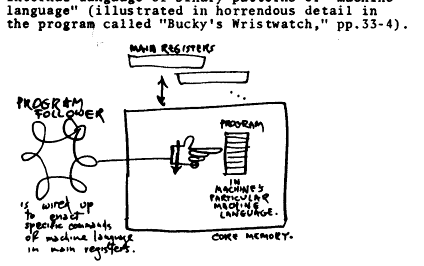
Computers cost several thousand bucks on up. Microprocessors cost several hundred on up, and that price range is falling fast.

Some microprocessors are already on integrated circuits, postage stamp-sized electronic tangles that are simply printed and baked, rather than wired up; this means there is effectively no bottom limit to the price of microprocessors. Mark this well. It means that in a few years there will be a microprocessor in your refrigerator, your typewriter, your lawnmower, your car, and possibly your wallet. (If you don't believe this, look what happened to pocket calculators in the last couple of years. The chip those are built around costs five bucks. But next come the programmable chips, the microprocessors.)

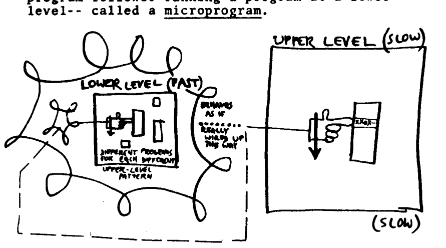
Microprocessors should not be called microcomputers, a term that seems to have captivated Wall Street lately. "Microcomputer" just means any teeny computer; but there is an exact and crucial difference between an ordinary computer (whatever its size) and a microprocessor (whatever its size).

A microprocessor is a two-level computer.

You will remember from the "Rock Bottom" section (pp. 32-3) that every computer has an internal language of binary patterns or "machine language" (illustrated in horrendous detail in the program called "Bucky's Wristwatch," pp.33-4).



Well, a microprocessor has two levels. It has an upper-level program follower with its own binary program; but each instruction of this upper-level program is in turn carried out by a program follower running a program at a lower level-- called a microprogram.



This has some extraordinary ramifications.

First of all, it means that the upper-level binary language can be anything you want-- that is, any feasible computer language-- because each of its instructions, in turn, will be carried out by program.

This means, for instance, that machines can be created which may be programmed directly in some higher-level language, such as APL (note Canadian machine described on p. 23) or BASIC (note one of the Hewlett-Packard machines described on p. 17). The characters in the upper-level program (APL or BASIC), stepped through by the upper-level program follower, cause the lower-level program follower to carry out the operations of the language.

Second, the machine costs less to make than an ordinary computer. The reason is that the architecture of ordinary computers is designed now (at last) for programmer convenience. Thus a machine like the PDP-11, which in principle does nothing any other computer doesn't do, is still more desirable than most, because its instructions are so well designed. It is clear and sensible to the programmer, with the result that programming it takes less time and costs less money.

Microprocessors reverse this trend. The lower-level structure of registers and instructions can be anything that is convenient to manufacture, whether or not programmers like it. Low manufacturing cost is one of the main design criteria.

The purpose of microprocessors, you see, is generally to be hidden in other equipment and do some simple thing over and over; not to have their programs changed around all the time as on an ordinary computer.

There are exceptions, computers which have a second level down where you can put microprograms; and these are called, sensibly enough, microprogrammable computers. They are bought and set up with regular computer accessories, plus facilities to change the microprograms. Thus they cost a lot more; but oh, they do so much more for you. You can design your own computer-- i.e., its instruction-set-- and then create it, with a microprogram. (See the Standard Computer and the Meta-4, nearby.)

HARDWARE: equipment itself.

SOFTWARE: computer programs

FIRMWARE: underprograms for microprocessors. (Also called Microprograms. Should be called Underware.)

A microprocessor simply means

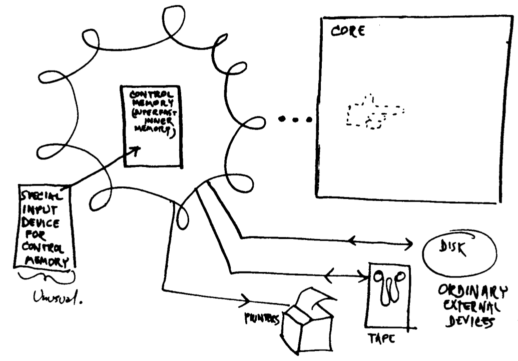
a computer which has,

under the binary language

you want to use,

another binary language

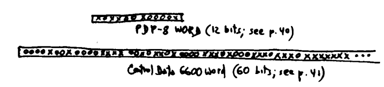
that's cheaper to wire up.



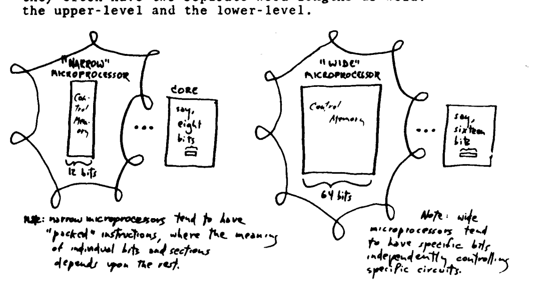
TWO LEVELS, TWO SPEEDS

The trick that makes this all work-- whether for the hidden-away type or the computer type of microprocessor-- is that the lower level has a much faster memory than the upper level. This means that an upper-level word can be taken, and looked up in the lower level, and all the lower-level steps carried out, very fast compared to the upper-level memory. Many such machines, for instance, have lower-level speeds in the nanoseconds (billionths of a second), while the upper-level speeds are merely in the microseconds (millionths of a second).

A last point. One of the most important characteristics of an ordinary computer is its word length, that is, the number of binary positions in a usual chunk of its information.



But since microprocessors have two separate levels, they often have two separate word lengths as well: the upper-level and the lower-level.



Microprocessors are usually sold in quantity, to people who are building super-cash-registers or pinball machines or the like. So their memories come in many sizes and speeds, to be tailored to an application. You should know the differences between--

ROM-- Read-Only Memory. Contents can't be changed, costs less than changeable (at any given speed).

RAM-- Rapid-Access Memory. Also called read-write memory. Same as core memory: May have its contents changed. NOTE: if you simulate some computer with a microprogram, its simulated "registers" are usually locations in the lower-level RAM.

RMM-- Read-Mostly Memory. You can get out its contents fast, but change them only very slowly.

(The lower-level memory is sometimes called "program memory" and the upper-level memory is often called "data memory," but this is a confusion resulting from certain typical applications of the devices, rather than their inherent nature. You can have programs at both levels.)

BIBLIOGRAPHY

Raymond M. Holt and Manuel R. Lemas, "Current Microcomputer Architecture." Computer Design, Feb 74, 65-73.

Summarizes nine teeny machines now on the market (some 1-level). Good bibliography also.

SOME MICROPROGRAMMABLE COMPUTERS.*

Standard Computer	18 bits	36 bits	Big & Expensive.
Meta 4	16 bits 90 or 35 nsec	16 bits 900 nsec	Up to 32 hardware registers.
Burroughs 1700	16 bits 60 nsec	24 bits 666 nsec	Comes with cassette holding various emulators.
Lockheed SUE	36 bits	16 bits	$650 stripped.
Hewlett-Packard 2100	?	16 bits	Already microprogrammed to be like other HP computers -- but there's space for yours. as well. $7500.
Microdata 3200	32 bits 135 nsec	16 bits	$8000 up ($10,000 for model 32/S, stack-oriented).
Varian 73	64 bits 165 nsec (190 read-write)	16 bits 660 nsec	$15,000 to $100,000 (heavy upgrade of Varian 620).
IBM 360 model 25	?	16?	
Prime 200	64 bits 160 nsec	16 bits 750 nsec	
Interdata 85	32 bits 160 nsec	16 bits 320 nsec	$23,000

SOME MICROPROCESSORS TO BE BUILT INTO THINGS.*
(most offer consoles, etc., for debugging)

Intel MCS-8	8 to 24 bits 900 nsec	8 bits 12.5 usec	Stack-oriented (now faster model).
Intel MCS-4	8 or 16 bits 900 nsec	4 bits 10.8 usec	Basic chip $60.
SYS 500	(Weird but interesting wide microprocessor-- circulates among many separate activites, rather than branching.)		
Microdata Micro 800 Micro 1600	16 bits 220 nsec 200 nsec (read-write)	8 bits 1.1 usec 1 usec	
AES-80 (Auto. Electric Systems, Montreal)	12 bits 240 nsec	8 bits 240 nsec or 1 usec	$950 w/o memory
National Semiconductor IMP-16C (8 1/2 x 11-- odd size for computer, convenient for notebook.)			$1380 stripped
DEC PDP-16M	8 bits	16 bits	$2000. (Compatible w. PDP-11 Unibus.)
Atron 601	16 bits 260 nsec	16 bits 1 usec	

*(Abbreviations: nsec (nanoseconds, or billionths); usec (microseconds, millionths; usual weird abbreviation).)

SUPERCHIP!

The history books ten years from now, if any, will note that the first computer-on-a-chip was produced by Intel. Intel, an astutely managed company, chose to make a microprocessor that would be suited to placement in others' machines at low cost. This means that if you make a fancy bulldozer or bake-oven, and want it to have some form of intricate pre-planned behavior, you'll put "the Intel chip" in it.

Actually the Intel chip is a number of separate chips, which start low in cost-- a fairly complete set can be had for under $500-- and can be assembled into a full computer. (Indeed, various firms do offer complete computers built out of Intel chips. Including one the size of an Oreo cookie, guaranteed for 25 years.)

The original Intel chips are the MCS-4 and MCS-8, viz.:

	Upper level	Lower level
MCS-4	4 bits (10.8 microseconds)	8 or 16 bits (900 nanoseconds)
MCS-8	8 bits (12.5 microseconds)	8 to 24 bits (900 nanoseconds)

While these individual chips cost under a hundred dollars each, memories and other necessary sections cost extra. For people who want to develop systems around these chips, Intel has cannily prepared a number of setups. If you want to go 4-bit, you get the "Intellec 4," $2200, which also needs a Teletype. This gives you various display lights and debugging features. Meanwhile, you can assemble and simulate on simulation programs offered on national time-sharing. If you want to go 8-bit, you get the "Intellec 8" for $2400 (also without Teletype), and benefit additionally from the fact that you can prepare the underware in PL/I, and compile it on national time-sharing.

Crafty and clever Intel, which has captured much of the overall market already, has now brought out much faster versions of these chips. Rah.

the META 4

A computer wittily called the Meta 4 (heh heh) is a fairly neat machine made by Digital Scientific Corp., 11455 Sorrento Valley Rd., San Diego CA 92121.

Lower memory: 16 bits, 90 nanoseconds (or 35 nanoseconds, programmed by a card (on which you darken the squares.)

Upper memory: 16 bits, 900 nanoseconds.

What this is is a very high-power minicomputer: it can be turned into a lookalike for any other 16-bit minicomputer. For instance, they can sell it to you with an imitative microprogram that turns it effectively into an IBM 1130. From a marketing point of view, this effectively means a firm owning an IBM 1130 can replace it with a Meta 4 which runs the same programs, saves money and gives you in addition the bottom-level features of a far more powerful computer. (Such an under-level program that makes one machine effectively imitate another computer is called an emulator.) This capacity to emulate other computers is the "metaphor" alluded to in the machine's name.

the LOCKHEED SUE

The Lockheed SUE ("System User-Engineered Computer") is a very interesting and desirable machine. The central processing unit costs a little over six hundred and forty dollars! (That's without memory, power supply or card cage.) It uses a Grand Bus system of interconnection (see p. 42).

It's a microprocessor. The lower-level cycle time is 50 nanoseconds, so it can be programmed to imitate any microsecond mini.

One nice thing is that you can put together several cpu's and different memories-- core, semiconductor and ROM-- selecting with switches which cpus have what priorities in what memories, as well as interrupts, etc. Darn nice-- especially considering the upper-level instruction-set.

The microprogram it comes with makes the Lockheed SUE into a sort of copy (??) of the PDP-11, including its eight registers and similar address modes (see p. 42).

Was the name SUE actually Lockheed's impudent challenge to DEC? DEC did sue, but no outcome has been publicized.

THE STANDARD COMPUTER

A microprogrammable biggie has been available for some time. It's a 36-bit computer manufactured by Standard Computer Corporation, 1411 W. Olympic Boulevard, Los Angeles, CA 90015.

This computer is a serious machine, in the many-hundred-thousand-dollar class, which can be set up to mimic any other 36-bit machine. It has been sold in two versions: one a pure FORTRAN machine (that's right, its upper language is pure Fortran!) and a lookalike for the IBM 7094. Lower-level word length is 18 bits.

(An interesting puzzle is why this outfit has not gotten together with Lincoln Laboratories. Lincoln Laboratories, outside Boston, has a 36-bit experimental machine called the TX-2 which has been used for computer graphics, such as Sutherland's SKETCHPAD system (see p. DM 23) and Baecker's GENESYS (see p. DM 25). Now, presumably Lincoln Labs, like most other research outfits, is hurting for money. Why couldn't they make an arrangement for Standard to sell its machine with a TX-2 emulator, thus making available such programs as Sketchpad (which has never been equalled) to a wider public?

ADVANCED PROGRAMS

In the early throes of computer enthusiasm, it is easy to suppose that anything can be done by computer-- that is, anything involving the chewing or diddling of information. This is decidedly not so.

For instance, it is easy enough, and often practical, to have a computer do something a few million times. But it is almost never practical to have a computer do something a trillion times. Why? Well, let's say (for the sake of simplicity) that a certain program loop takes 1/1000 of a second. To do it a thousand times, then, would take one second, and to do it a million times would take a thousand seconds, or about seventeen minutes. But to do it a trillion times, now, would mean doing it 17,000,000 minutes, or over thirty years.

Now, you will note that even if you speed up that loop to 1/1,000,000 of a second, a trillion repetitions will take almost twelve days, which is obviously going to need some justifying, even assuming that it is otherwise feasible.

(For problems of this type people begin thinking about building special hardware, anyway. It will be noted, for instance, that the PDP-16-- see p. 57-- lets you compile your own special equipment for problems that need eternal repetitions.

COMBINATORIAL EXPLOSIONS

One kind of thing that's too much to do is generally called a combinatorial explosion-- that is, a problem that "explodes" into too many things to do. For instance, consider the game of chess. Just because you can write a program to look ahead at all the possible outcomes of, say, tic-tac-toe, that doesn't mean you can consider all the possibilities of chess. To look at "all" the possibilities just a few moves ahead involves you in trillions of calculations. Remember about trillions? And it turns out that there are a lot of problems like that.

METHODS FOR DOING THINGS

There are really no clear bounds on "what computers can do."

The problem is always to think up methods for doing things by computer. (Also called algorithms.)

Basically what can be done by computer is what can be done on a tabletop with slips of paper-- comparing, copying, sorting, marking, doing arithmetic-- and handing slips of paper out to users.

So the question should never be, "How would you do that by computer?" -- but "Can you think of a method for accomplishing that?" The "computer" is really irrelevant, for it has no nature and merely twiddles information on demand.

Then there is the problem of "Turing impossibility." Turing was a mathematician who discovered that some things can be done sequentially in a finite amount of time, and some things can't, such as proving certain types of mathematical theorem. In other words, anything that has to do things in sequence-- whether a computer or a mind of God, if any-- cannot possibly know anything which is not Turing-computable. Another important limitation.

On a more practical level, though, there are just lots of things which nobody has figured out how to do in any feasible way, or are just now figuring out different systematic ways of doing. (For a favorite such area of mine, compare the different computer half-tone image synthesis systems described on pp. DM 32 to DM 39.)

Thus you see that figgering out ways of doing stuff is still one of the principal aspects of the computer field. (Whole journals are devoted to it, such as CACM, JACM and so on.)

But then of course, every few years there comes a new movement in the field that bodes to make us start all over.

One such trend is called structured programming, being promulgated by a Dutch researcher named Dijkstra, among others. The idea of structured programming is to restrict computing languages in certain ways and "eliminate the GO TO," i.e., no longer have jumps to labeled places in programs. By dividing computer programs up only in certain ways, goes this school of thought, the programs can perhaps be proven workable, in the mathematical sense, rather than just demonstrated to work, as they are now-- a notoriously error-prone situation. If the Dijkstra school is correct, we may have to start all over again with a new bunch of programming languages.

These remarks give you the flavor of some restrictions and lines of development. The rest of this page is devoted to The Great Software Problem-- the Operating System.

OPERATING SYSTEM/360

or OS/360, or OS

We have no space here to discuss OS, the operating system of the IBM 360 and 370, which is just as well: it is a notoriously heavy-handed system, elaborated with what some would call devastating messiness. Kinds of convenience taken for granted by users of such computer systems as the Burroughs 5000, the PDP-10, DTSS and others aren't there.

The programmer has to concern himself with intricacies having names like ACONs, VCONs, TCBs, ECBs, and the complications of JCL. (While these other systems may have equivalent complications, the programmer need not mess with them to create efficient programs, as the 360 demands.) The programmer must even set aside the previous programmer's information in "SAVE AREAS," which is like a restaurant guest having to clear the dirty dishes on sitting down-- and return them when he leaves. Several of the 360's sixteen general registers are confiscated. Time-sharing requires its own JCL-type language. And so on.

IBM says its forthcoming operating system, OS/VS2-2, will be better.

BIBLIOGRAPHY

A.L. Scherr, "The Design of IBM OS/VS2 Release 2." Proc. NCC 73, 387-394.

OPERATING SYSTEMS & TIME-SHARING

Basically, an operating system is a program that supervises all the other programs in a computer. For this reason it is also called a supervisor or a monitor. Because the operating system is supposed to be in charge, many computers now offer special wired-in instructions that only the operating system can use. This prevents other programs from taking complete control of the machine.

Operating systems come in all sizes. The bigger ones take up a lot of computer time because they have to do a lot. The smallest kind, which are really kind of different, are just to help a single programmer move quickly between his basic programs. (A typical such system is DEC's DOS, or Disk Operating System, which you can get with the PDP-11.) This system is really a kind of butler that keeps track of where your basic programs are stored on disk and brings them in for you quickly.

A step up is the Batch Monitor, or operating system set up for Batch Processing (see p. 8). In batch processing, programs go through the computer as if on a conveyer belt, one at a time (or in some systems several at a time). The operating system shepherds them.

Batch processing is used when programs don't need any interaction with human users. (Or, and this is more common, when human users want time-sharing but can't get it; see below.) A multiprogramming operating system is one that allows several different programs (or conveyor-belt sequences of batch programs) to operate at one time. (This is how most IBM 360s are used.)

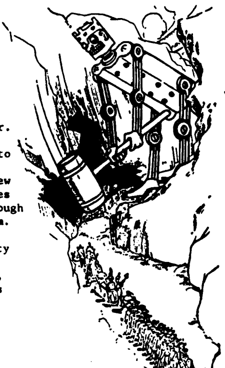
SYSTEMS PEOPLE
are the folks who bring you the computer.

That is, they're the ones who try to keep the operating system working. And make the changes it needs to adapt to new equipment and working rules and schedules and software. And change the parts through which mischievous users crash the system.

Systems people often look like dirty rats to users of computer systems. To each other they often look like harried, overworked, unsung heroes, their fingers (and whatever else) in the dike, trying to hold back the tide of Disorder.

Systems people deserve more thanks than they get.

Thank you, systems people.



BATCH SYSTEM

Then there is time-sharing.

Time-sharing means the simultaneous use of one computer by several different users at once. It's basically a complex form of multiprogramming.

In principle this is like a lazy susan. The central computer works on one user's program for a while, then on another's... until it is back to the first user.

There are basically two kinds of time-sharing: time-sharing where you can only use certain facilities or languages, and time-sharing where you can use all the facilities of the computer (including programming in the computer's assembly language).

Examples of restricted time-sharing are the various minicomputer systems that are available which time-share the BASIC language. (Nova and PDP-11 and Hewlett-Packard, for instance.)

Some examples of unrestricted time-sharing are the PDP-10 (see p. 40), Dartmouth's DTSS, Honeywell's MULTICS, IBM's TSO, and General Electric's MARK III.

Bigger is not necessarily better. For instance, there are time-shared versions of BASIC that run on big IBM computers. However, it may very well be that big IBM installations can save money by eliminating this function and buying instead a small Hewlett-Packard minicomputer to run their BASIC on, thereby supplying BASIC to more users at less cost and freeing the 360 for whatever it is IBM systems do better.

Restricted time-sharing, with only one or a few languages offered, is much easier to provide for than full time-sharing.

Full time-sharing is always shared with batch. In other words, the computer, darting among users, still finds some time to devote to the batch stream.

Time-sharing is self-limiting. That is, the more users are signed onto a time-sharing system at a given moment, the more slowly the system responds to all of them.

Operating systems are big and hard to program. They take a lot of the computer's time: for instance, Dartmouth's time-sharing operating system, taking as much as 23% of the computer's time, is considered efficient.

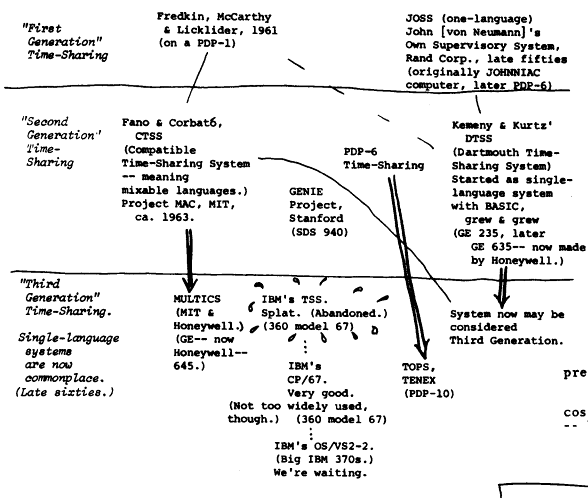
The importance of time-sharing is not in terms of "raw" efficiency, that is, the cost of each million operations, but in terms of human efficiency, the ability of each user to get so much more out of the computer by using interactive programs and languages.

OPERATING SYSTEMS TRICKERY

Swapping means transferring one user's program out of core memory in order to move in somebody else's program. This can happen very rapidly, and even when it's done to you every turn, your terminal may seem to respond as though you are in continuous possession of the entire computer

Paging is one of the Great Abstruse Problems of modern operating systems. The problem is this: you've always got fast expensive memory and cheap slow memory. How can the operating system store most of your program in cheap slow memory and still predict which parts you'll need soon enough to get them in there for you? In the hotter systems, indeed, the operating system tries to predict what's most important and move it to a fast little memory called a cache. This area is so bizarre and complicated I prefer not to think about it. "Minis for me," says Mr. Natural.

SOME IMPORTANT TIME-SHARING SYSTEMS (very incomplete)



BIBLIOGRAPHY

M.V. Wilkes, Time-Sharing Computer Systems, MacDonald/American Elsevier Publishing Co.

All About Timesharing Service Companies. Datapro Research (1 Corporate Center, Moorestown, NJ 08057), $10.

★DTSS★

DTSS is the Dartmouth Time-Sharing System, and let it be an example to us all.

It was created by Kemeny and Kurtz, who created the BASIC language to be used on it (see p. 16).

Their computer arrived in fall '63. Their time-sharing system went into operation in spring '64, programmed mostly by Dartmouth students, and has grown and improved continuously since then. On that basis: programmed by students.

It's great.

The Dartmouth computer philosophy-- i.e., the idea carried through by John Kemeny and Tom Kurtz--was that a computer is like a library: its services should be free to all in a community, paid for through some general fund.

Students and faculty at Dartmouth use it free. (Unless they have grants.) You can use it too, if you pay.

The result: everybody at Dartmouth uses the computer. It's always running, (ahem) six days a week. There are almost two hundred terminals around the campus; peak afternoon usage is about a hundred and fifty. Freshmen learn BASIC first thing, after which the computer is a standing facility, to be used in courses in music, sociology, literature, history, engineering or whatever; for independent research; or just for fun and games and showing off to visitors.

The entire Dartmouth system is built for simplicity and clarity, with explanations of all the facilities available at terminals. (The command explain JGK causes the terminal to type out a picture of Kemeny.)

Many fuddy-duddies insist that computer usage should be billed, as it is on most college campuses. That is essentially the Calvinist view. But what if we treated libraries like that? It would probably cost $10 just to borrow any book. The point is that if we believe that certain conditions are a social good, then we should be flexible about how to implement them. (See Arthur W. Luehrmann and John M. Nevison, "Computer Use under a Free-Access Policy, Science, 31 May 74, 957-961. This article continues this line of argument and further describes the Dartmouth billing system.)

Anyway, Dartmouth will sell you its time-sharing system for about $7500 a month (and you'll need a computer setup that begins at $17,500 a month). That'll run 50 terminals. A bigger setup will cost more. But that gets you Fortran, COBOL, SNOBOL, etc., the best BASIC in the whole world, games, financial systems, and myriad other programs they've built at Dartmouth. Furthermore, Mr. Administrator, it means the system will be available to users with a minimum of complication and bother.

A number of companies have bought. Including the U.S. Naval Academy at Annapolis, which offers Dartmouth-style computing to its midshipmen.

Connect charge is $2 to $9 an hour depending on your terminal speed, plus processing charges.
Contact: DTSS, INC., Hanover NH 03755. (Several commercial firms also offer DTSS to users, including Computer Sharing Services, Inc. Denver; Grumman Data Systems, Woodbury, NY; PolyCom Systems Ltd., Toronto.)

The most enjoyable session at the 1974 National Computer Conference was the Nostalgia session on the Dartmouth System, DTSS. The Old Hands were there-- guys who as kids worked on the original time-sharing system, and have now become grownups of one sort or another.

An alarming statement was made at that session by Jerome B. Wiener, who said he had been the liaison man between the Dartmouth effort and the computer manufacturer (not IBM). He stated that he had been ordered by his company to stop the Dartmouth "experiment" any way he could, or lose his job in three months. He did no such thing, and (he said) after being fired continued to help the Dartmouth effort, holding weekend meetings with others from that company in his home. He deserves the Frances O. Kelsey we-do-our-real-job medal.

Time-sharing prices are a mix of lotsa stuff:

1. Connect time. This you pay by the hour. Good prices: $2 (DTSS), $1.50 (Monmouth County [NJ] Community College -- but they have no concentrators and want no beginners).
2. "Core charges"-- essentially the price of processing itself; depends on amount of number crunching. PDP-10 bills this in kilocore-seconds, i.e., how many thousand words of core memory your program really turns out to need, for how many seconds.
3. Storage, which costs much. Example: 1000 characters for a month for a buck. (Typical.) That adds up fast. You might do better with a cassette memory on your terminal, such as the Techtran (see p. 14 43).

Five bucks an hour overall is a pretty good rate.

Note that time-sharing usually costs less in non-business hours -- but some exceptions charge more.

WHERE TO GET IT

No way can we here get into the prose and cons (both senses) of the myriad time-sharing services that are available. An excellent summary of fifty-six different time-sharing services (variously using computers by Honeywell, IBM, DEC, Univac, CDC, Xerox and Burroughs) appeared in the February, 1973 Computer Decisions ("Piecing Out the Timesharing Puzzle" by John R. Hillegass, pp. 24-32). This summarizes information available from Datapro Research Corp., Moorestown, NJ. The article cautions against the potential high cost of time-sharing services, and urges you to get all the advice you can before committing to a time-sharing service.

MULTICS!

MULTICS was announced in 1965 as the Time-Sharing System of All Time, to be created jointly by MIT, General Electric and Bell Labs.

It took a lot longer to get going than they expected-- I have a 1968 (?) button that says, YOU NEVER OUTGROW YOUR NEED FOR MULTICS-- but now it's available from Honeywell. People say it's the greatest, all right-- its fascinating facilities include the ability to execute parts of other people's programs, if you have permission-- but it's also said to be awfully expensive.

Interestingly, the MULTICS operating system is largely programmed in the PL/I language (see p. 31).

Contact: Honeywell Information Systems, 200 Smith Street, MS 061, Waltham, Mass. 02154.

THE LOGIN HEARD 'ROUND THE WORLD: GE's MARK III

Some time-sharing systems are local, others have "concentrators" allowing users in other cities to log into them with local telephone calls.

Perhaps the most far-reaching time-sharing system, though, is General Electric's MARK III, with concentrators in many of the cities of the world (mostly Europe). The main computer is in Ohio, but the overall system may be thought of as an octopus around the globe. Besides hundreds of cities in the USA, The GE system offers local access in Australia, Austria, Belgium, Canada, Denmark, Finland, France, Italy, Japan, Netherlands, Norway, Puerto Rico, Sweden, Switzerland, United Kingdom and West Germany.

What this basically means is that if a company has offices in these places, it can do its internal communication through General Electric's computer system.

This presents obvious merits and difficulties, which there is no room to discuss here. The service is said to be expensive.

They also offer a toll-free number for program consultation.

Contact:

General Electric Information Services
Business Division,
401 North Washington St.,
Rockville, Md. 20850.

TSO

IBM's "TSO", for Time-Shared Operating System, is an odd sort of time-sharing they have come up with for the 370.

It is a sort of interactive batch programming. That is, it allows the user at a terminal to communicate with programs running in batch mode.

While this is a form of true time-sharing, (though its detractors tend to compare it with what they call "true" time-sharing, such as that on the PDP-10), it has a number of drawbacks.

For instance, on the model 158, a fairly large machine (ca. $50,000 a month-- see p. 38), TSO normally allows only twenty users.

The bad feature of TSO most often mentioned is its slow response time. That is, response may be sometimes good, sometimes execrable.

IBM is urging its fans to believe that its next operating system, called OS/VS2-2, will be much better.

THE HEARTS AND MINDS OF COMPUTER PEOPLE

Computer people are a mystery to others, who see them as somewhat frightening, somewhat ridiculous. Their concerns seem so peculiar, their hours so bizarre, their language so incomprehensible.

Computer people may best be thought of as a new ethnic group, very much unto themselves. Now, it is very hard to characterize ethnic groups in words, and certain to give offense, but if I had to choose one word for them it would be elfin. We are like those little people down among the mushrooms, skittering around completely preoccupied with unfathomable concerns and seemingly indifferent to normal humanity. In the moonlight (i.e., pretty late, with snacks around the equipment) you may hear our music.

Most importantly, the first rule in dealing with leprechauns applies *ex hypothesi* to computer people: when one promises to do you a magical favor, keep your eyes fixed on him until he has delivered. Or you will get what you deserve. Programmers' promises are notoriously unkept.

But the dippy glories of this world, the earnestness and whimsy, are something else. A real computer freak, if you ask him for a program to print calendars, will write a program that gives you your choice of Gregorian, Julian, Old Russian and French Revolutionary, in either small reference printouts or big ones you can write in.

Computer people have many ordinary traits that show up in extraordinary ways-- loyalty, pride, temper, vengefulness and so on. They have particular qualities, as well, of doggedness and constrained fantasy that enable them to produce in their work. (Once at lunch I asked a tableful of programmers what plane figures they could get out of one cut through a cube. I got about three times as many answers as I thought there were.)

Unfortunately there is no room or time to go on about all these things-- see Bibliography-- but in this particular area of fantasy and emotion I have observed some interesting things.

One common trait of our times-- the technique of obscuring oneself-- may be more common among computer people than others (see "The Myth of the Machine," p. 9 , and also "Cybercrud," p. 8). Perhaps a certain disgruntlement with the world of people fuses with fascination for (and envy of?) machines. Anyway, many of us who have gotten along badly with people find here a realm of abstractions to invent and choreograph, privately and with continuing control. A strange house for the emotions, this. Like Hegel, who became most eloquent and ardent when he was lecturing at his most theoretical, it is interesting to be among computer freaks boisterously explaining the cross-tangled ramifications of some system they have seen or would like to build.

(A syndrome to ponder. I have seen it more than once: the technical person who, with someone he cares about, cannot stop talking about his ideas for a project. A poignant type of Freudian displacement.)

A sad aspect of this, incidentally, is by no means obvious. This is that the same computer folks who chatter eloquently about systems that fascinate them tend to fall dark and silent while someone else is expounding his own fascinations. You would expect that the person with effulgent technical enthusiasms would really click with kindred spirits. In my experience this only happens briefly: hostilities and disagreements boil out of nowhere to cut the good mood. My only conclusion is that the same spirit that originally drives us muttering into the clockwork feels threatened when others start monkeying with what has been controlled and private fantasy.

This can be summed up as follows: NOBODY WANTS TO HEAR ABOUT ANOTHER GUY'S SYSTEM. Here as elsewhere, things fuse to block human communication: envy, dislike of being dominated, refusal to relate emotionally, and whatever else. Whatever computer people hear about, it seems they immediately try to top.

Which is not to say that computer people are mere clockwork lemons or Bettelheimian robot-children. But the tendencies are there.

BIBLIOGRAPHY

Gerald M. Weinberg, The Psychology of Computer Programming. Van Nostrand Reinhold.

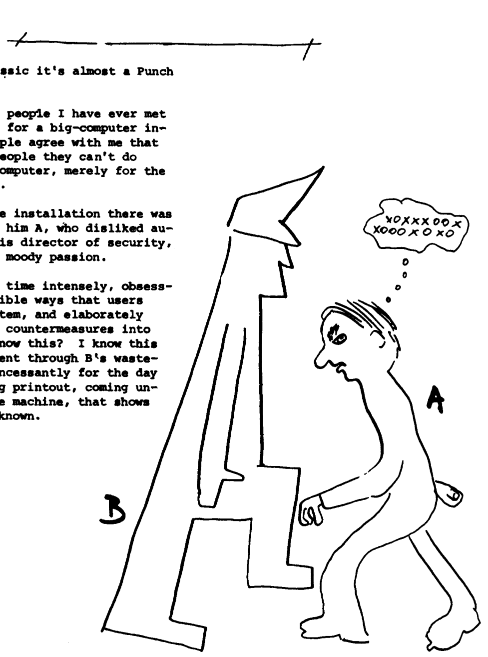
Systematic treatment in a related vein.

This case is so classic it's almost a Punch and Judy show.

One of the nastiest people I have ever met was the head of security for a big-computer installation. Several people agree with me that he delights in telling people they can't do specific things on the computer, merely for the sake of restricting them.

Anyway, at this same installation there was a programmer, let's call him A, who disliked authority, and disliked this director of security, let's call him B, with a moody passion.

B spent much of his time intensely, obsessively contemplating possible ways that users might break into the system, and elaborately programming defenses and countermeasures into the monitor. How do I know this? I know this from A, who constantly went through B's wastebasket. A still plans incessantly for the day B will get a big taunting printout, coming unexpectedly to him off the machine, that shows him all his secrets are known.



COMPUTER PUTDOWNS

Practice saying them loudly and firmly to yourself. That way you won't freeze when they're pulled on you.

THAT'S NOT HOW YOU DO IT
THAT'S NOT HOW YOU USE COMPUTERS
THAT'S NOT WHAT YOU DO WITH COMPUTERS
THAT'S NOT HOW IT'S DONE
THAT'S NOT PRACTICAL
HOW MUCH DO YOU KNOW ABOUT COMPUTERS?
WITH YOUR BACKGROUND,
YOU COULDN'T UNDERSTAND IT
LET'S CALL IN SOMEONE WHO KNOWS THIS
APPLICATION (generally a shill)
IT ISN'T DONE
(you know the answer to that one)
and the one I've been waiting to hear,
IF GOD HAD INTENDED COMPUTERS TO BE USED
THAT WAY, HE WOULD HAVE DESIGNED
THEM DIFFERENTLY.

Unfortunately there is no room here to coach you on how to reply to all these. Be assured that there is always a reply. The brute-force brazen comeback, equally dirty, is just to say something like

DIDN'T YOU SEE THE LAST JOINT PROCEEDINGS?
or
OH YEAH? WHAT ABOUT THE x WORK
USING A y?

(where x is anyplace on the map on p. 5 , and y is any current computer, such as a PDP-10.)

" ... programmers, in my experience, tend to be painstaking, logical, inhibited, cautious, restrained, defensive, methodical, and ritualistic."

Ken Knowlton, "Collaborations with Artists-- A Programmer's Reflections," in Nake & Rosenfeld (eds.), Graphic Languages (North-Holland Pub. Co.), p. 399.

USEFUL, AND POSSIBLY EMBARRASSING QUESTIONS

If the Computer Priests start to pick on you, here are some helpful phrases that will give you strength.

I do not want to give the impression that the Guardians of the Machine are always bad guys. Nevertheless, sad to relate, they are not always good guys. Like everyone out to bolster his position, including the plumber and the electrician, the computerman has learned how easy it is to intimidate the layman.

Now, these people are often right. But if you have reason to question the way things are done-- whether you're a member of the same corporation, a consumer advocate or whatever-- you are probably entitled to straight answers that will help settle the matter honestly, without putdowns. Any honest man will agree.

Now, these helpful questions, honestly answered, may elicit long mysterious answers. Be patient and confident. Write down what's said and sit down with the glossary in this book until you understand the answer. Then you can ask more questions.

I am not inviting the reader to make trouble flippantly. I am suggesting that many people have a right to know which has not been exercised, and there may be some discomfort at first.

HOW DOES IT WORK?
(This question may have to be backed up as follows: "There are no computer systems whose workings cannot be clearly described to someone who understands the basics. I INSIST THAT YOU MAKE A SINCERE ATTEMPT.")
WHY DO YOU CLAIM IT HAS TO BE THIS WAY?
(SPEAK MORE SLOWLY, PLEASE.)
WHAT IS THE DATA STRUCTURE?
COULD YOU EXPLAIN THAT IN TERMS OF THE DATA STRUCTURE?
WHO DESIGNED THIS DATA STRUCTURE?
And can I talk to him?
WHAT IS THE ALGORITHM?
WHO IS THE PROGRAMMER?
And can I talk to him?
WHY DO WE HAVE TO USE A CANNED PROGRAM FOR THIS?
WHY IS THE INPUT LANGUAGE SO COMPLICATED?
WHY DO WE NEED CARDS? WHY CAN'T PEOPLE TYPE IN THEIR OWN INPUT?
WHY NOT HAVE A SIMPLE-MINDED FRONT END THAT LETS USERS CONTROL IT THEMSELVES?
WHY HAVE FORMS TO FILL OUT? WHY NOT HAVE A DIALOGUE FRONT-END ON A MINI?
WHY CAN'T IT BE ON-LINE? And on CRT terminals (see pp. 20-21)?
WHY DOES IT HAVE TO BE THAT BRAND OF COMPUTER?
WHY NOT GET A SYSTEM WITH LESS OVERHEAD?
WHY SHOULD ALL COMPUTER OPERATIONS BE CENTRALIZED?
DON'T THEY GET IN EACH OTHER'S WAY?
WHY DOES IT ALL HAVE TO BE ON ONE COMPUTER?
WHY NOT PUT PART OF IT ON A DEDICATED MINI?
WHY CAN'T WE DO THIS PARTICULAR THING ALL ON A MINI?
WOULDN'T IT COST LESS IF WE GOT A MINICOMPUTER FOR THIS TASK?
WHY CAN'T THIS BE PROGRAMMED IN SOME LANGUAGE LIKE BASIC?
YOU KNOW AND I KNOW THAT COMPUTERS DON'T HAVE TO WORK THAT WAY. WHY DO YOU CHOOSE TO DO IT THAT WAY?

If these suggestions seem unnecessarily contentious, it is because some of these guys like to pick on people, and you may have to be ready. And you may need all the support you can get, if, say, you take a stand like one of these:

"If the information is in there, I don't see why we can't get it out."

"You have no right to ask questions like this, and if the program requires it, change the program."

Remember, ILLEGITIMIS NON CARBORUNDUM (don't let the bastards grind you down)

"For me it always comes down to a personal challenge: not just to create a program that meets the specifications, but to do it in a way that I find aesthetically pleasing."

Robert H. Jones IV, a heavy programmer at Chrysler

PROGRAM NEGOTIATION

A very important kind of discussion takes place between people who want computer programs, but can't write them, and people who can write them, but don't want to. Or, that is, who don't want to get caught having to do a lot of unnecessary work if it could be done more simply.

Program negotiation, then, is where the "customer"-- he may actually be the boss-- says, "I want a program that will do so-and-so," and the programmer says, "I'd rather do it this way."

In a series of requests and counter-offers the customer explains what he wants and the programmer explains why he would rather do it a different way. It is essential for both sides to make themselves completely clear. Often the customer thinks he wants one thing but would be quite satisfied with another that is much easier to program. Often the programmer can make helpful suggestions of better ways to do it that will be easier for him.

Very bad things can happen if program negotiation is not done carefully and honestly enough. The programmer can misunderstand and create something that was not wanted. Or the customer can carelessly misstate himself and ask for the wrong thing. Or worst of all-- the programmer can deliberately mishear and do something different, saying, "There, that's what you wanted," as he hands over something that isn't what was really asked for. And the poor customer may even believe it (see "Cybercrud," p. 8).

Program negotiation should be more widely acknowledged as a difficult and painful business. It is exhausting and fraught with stress; people (on both sides) get all kinds of psychosomatic symptoms (like abdominal pains, tics and chills). The fact that people's careers often depend on the outcome makes the atmosphere worse, rather than fostering the thorough and sympathetic cooperation which is essential.

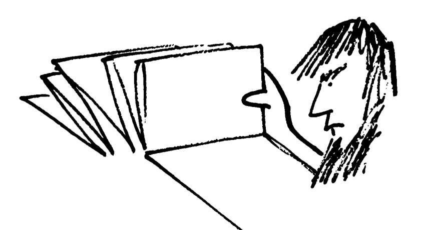
If there is one thing that laymen in business should be taught about computing, this is it.

"I CAN'T BEAR HEAT," REMARKED LANGWIDERE

THE MEETING OF THE MINDS

The Customer, Naive Advocate or Chump	The "Expert"
I don't see why since it's a computer... These are not details that concern me... These are just technical issues... I mean a computer can do all these things, can't it?	What you've gotta understand is that there are problems involved... It can't be that way... Leave it to me, it'll be just what you want...

Comeuppance: the customer will get what he deserves.
Moral: if you want something, you'd better damn well negotiate it at the detailed level.



The strange language of computer people makes more sense than laymen necessarily realize. It's a generalized analytical way of looking at time, space and activity. Consider the following.

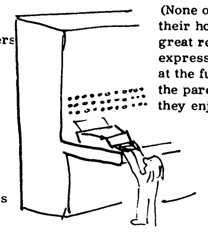
"THERE IS INSIGNIFICANT BUFFER SPACE IN THE FRONT HALL." (Buffer: place to put something temporarily.)

"BEFORE I ACKNOWLEDGE YOUR INTERRUPT, LET ME TAKE THIS PROCESS TO TERMINATION."

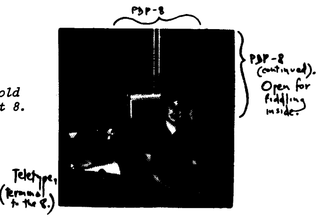
"COOKING IS AN ART OF INTERLEAVING TIME-BOUND OPERATIONS." (I.e., doing parts of separate jobs in the right order with an eye on the clock.)

THOSE ADORABLE INFURIATING R.E.S.I.S.T.O.R.S.

Their name makes people think they're a war protest group, but actually the R.E.S.I.S.T.O.R.S. of Princeton, N.J. are a bunch of kids who play with computers. They're all young; members are purged when they finish high school. Their clubroom is at Princeton University, but the initiative is strictly theirs.



The name stands for "Radically Emphatic Students Interested in Science, Technology and Other Research Subjects." Computers are not all they do--they've also gotten into slot racing and the game of Diplomacy-- but computers are what they're known for. The Resistors (let's spell it the short way) exhibit regularly at the computer conferences, and have startled numerous people with the high quality of their work. They've been invited to various conferences abroad. They have built various language processors and done graphics; lately their fad is working with the LDS-1 in Princeton's Chemistry Department.



Steve at the old straight 8.

Where do they learn it all? They teach each other, of course. Newcomers hang around, learn computer talk, work on projects, and tease each other. They also use the informal trade channels, subscribing to magazines and filling out information request cards under such company names as Plebney International Signal Division and Excalibur Wax Fruit.

The great thing about these kids is their zany flippancy. They've never failed, they've never been afraid for their jobs, and so they combine the zest of the young with their expertise. Their forms of expression are as startling to professionals as they are to outsiders: don't say anything ponderously if it can be said playfully. Don't say "bit field" if you can say "funny bits;" don't say "alphanumeric buffer" if you can say "quick brown fox box;" don't say "interrupt signal" if you can call it a "Hey Charlie;" don't say "readdressing logic" if you can say "whoopee box."

What's a group like you doing at a Joint like this?

Now here's my plan...

A coven of R.E.S.I.S.T.O.R.S. in executive session, Atlantic City

They have varied backgrounds. The father of one is a butcher, the father of another is one of the country's foremost intellectuals. (None of that matters to the kids.) I have dined in a number of their homes, and find this in common: their parents show them great respect, love and trust. Indeed, Resistor parents have expressed some surprise to learn that their children's work is at the full-fledged professional level. The important thing, to the parents, is that the kids are working on constructing things they enjoy.

R.E.S.I.S.T.O.R.S. after infamous Omega ceremony.

The trade press is ambivalent toward the Resistors. On the one hand they make good copy. (At one Spring Joint they had the only working time-sharing demo-- on a carpet next to a phone booth.) On the other, they sometimes seem bratty and publicity-hungry, like many celebrities. (At another Spring Joint they dug up an IBM Songbook and serenaded the guys at the IBM pavilion, who had to act nice about it.) So they don't get written up in computer magazines so much anymore.

I first met the Resistors in 1970, and started hanging around with them for two reasons. First, they are perfectly delightful: enthusiastic in the way that most adults forego, and very witty. To them computer talk was not a thing apart, as it is for both outsiders and many professionals.

Secondly, and this was the self-seeking aspect, I noted that these kids were quite expert, and interested in giving me advice where computer professionals would not. They got interested in helping me with my (perhaps quixotic) Xanadu™ project (see flip side). This was enough to keep me visiting for a couple of years. Now, some people are too proud to ask children for information. This is dumb. Information is where you find it.

The last I heard, the Resistors were at work in a COBOL compiler for the PDP-11, hoping it would save the local high school from the disastrous (to them) purchase of an IBM 1130. (Since the school's intent was to teach business programming, they hoped that the availability of COBOL would encourage the school to buy the more powerful and less expensive PDP-11.)

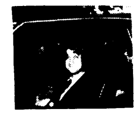
The Resistors are few, but I think they are very important in principle, an existence proof. They show how silly and artificial is our edifice of pedagogy, with all its sequences and sterilizations, and how anybody can learn anything in the right atmosphere, stripped of its pomposities. The Resistors are not obsessed with computers; their love of computers is part of their love of everything, and everything is what computers are for.

R.E.S.I.S.T.O.R.S. Anecdotes.

Lauren, 14, was talking to another girl at the ACM 70 conference. A passerby heard her explaining the differences among the languages BASIC, FORTRAN, COBOL and TRAC. "How long have you been programming?" he asked in surprise. "Oh, almost a month," she said.

* * *

I was driving some Resistors around Princeton; they were yelling contradictory driving instructions. "I demand triple redundancy in the directions," I said. "Right up ahead you turn right right away," said a spokesman.



triple redundancy

* * *

Since there was a lot of excess capacity, the Resistors got a free account on a national time-sharing system. Though they didn't have to pay, the system kept them informed on what they would have owed. In a year or so they ran up funny-money bills of several hundred thousand dollars.

* * *

Did they rate free subscriptions to computer magazines? I asked. Could they claim they really "make decisions affecting the purchase of computers"?

"Of course we do!" was the reply. "All together: shall we buy a computer?"

Resistors (in unison) "NO!"

* * *

Their original advisor, whom we shall call Gaston, is mischievous in his own right. It was meeting-time at Gaston's place on a bright Saturday, and I was on the lawn working on Xanadu with Nat and Elliott when Gaston interrupted to say that an unwelcome salesman of burglar alarms was about to arrive. "Let's have a little fun with him," said Gaston. The kids were to be introduced as Gaston's children, I was an uncle. We took our stations.

The salesman may have realized he was walking into a trap from all the strangely beaming adolescents that stood in the living room. He got out his wares and started to demonstrate the burglar alarm, but it didn't go right. Peter, standing in front of the equipment with a demonically vacuous grin, had reversed a diode behind his back so that the alarm rang continuously unless you broke the light beam.

"Humpf," said Gaston, "you want to see a real security system?" We trooped into the kitchen, where Gaston kept a Teletype running.

ANY NEWS? typed Gaston.

CREAM YELLOW BUICK PULLED INTO DRIVEWAY, replied the Teletype. JERSEY LICENSE PLATE . . . (and the salesman's license number), and finally, OWNER OF RECORD NOT KNOWN. John was typing this from the other Teletype in the barn.

The salesman stared at the Teletype. He looked around at our cherubic smiling faces. He looked at the Teletype. "That's all right," said the salesman. "But now I'd like to show you a real security system. . ." And it was back to the old burglar alarm.

GUIDELINES FOR WRITERS AND SPOKESMEN

The public is thoroughly confused about computers, and the press and publicists are scarcely free from blame. IT'S TIME FOR EXPLANATIONS. People want to know what computer systems really do-- no more of this "latest space-age technology" garbage. Mr. Businessman, Mr. Writer, are you man enough to start telling it straight?

The computer priesthood, unfortunately, often wants to awe people with, or unduly stress, the notion of the computer being involved in a particular thing at all. It is time for everybody to stop being impressed by this and get on with things. Don't just copy-edit what they give you. Nose around and really find out, then write it loud and clear.

These simple rules are my suggestions for bringing on more intelligent descriptions that will help enlighten the public by osmosis.

1. FIND OUT AND DESCRIBE THE FUNDAMENTAL APPROACH AND PHILOSOPHY OF THE PROGRAM. This can invariably be stated in three clear English sentences or less, but not necessarily by the person who created it. THIS IS WHAT WRITERS ARE FOR: it is your duty to probe until the matter has become clear.

Examples.

"This chess-playing program evaluates possible moves in terms of various criteria for partial success, and makes the move which has the highest merit according to these ratings."

"This music-composing program operates on a semi-random basis, screening possible notes for various kinds of attractiveness..."

"This archaeological cataloguing system keeps track of a variety of objective features of each artifact, plus information on where it was, including linkages indicating what other artifacts were near it."

What or whose computer is used to do a thing is of almost no concern (unless it is one of unusual design, of which there are comparatively few). Not the make of the computer, but the GENERAL IDEA OF HOW THE PROGRAM OPERATES, is the most important thing.

Of course, if you are being paid by a hardware manufacturer, you'll have to name the equipment over and over; but recognize that your real duty is public understanding, and put the facts across. (If you think it can't be done, read the splendid Kodak ads in the Scientific American.)

2. Keep gee-whizzing restricted to the description of a system's psychological effect on real people. (What impresses you may turn out to be old hat.)

3. Look for angles special to what you're reporting. Pursuing details is likely to bring up better story pegs and more human interest. Instead of saying "computer scientists" have done something, you might find something more interesting for your lead; how about "The unlikely team of a biophysicist and a teen-age art student..." or-- finding what's special-- "Never before has this been done on a computer so small, the size of a portable typewriter (and having only some 4000 words of memory)..."

4. Attempt to find out how else computers are used in the particular area, and mention these to help orient the reader.

This goes against the exclusivist tendencies we all have when we want to ballyhoo something. It is a matter of conscience, an important one.

5. Questions to ask:

What are the premises of your program?

What if people turn out to need something else?

What could go wrong?

And most important: What is that?

IMPORTANT DISTINCTIONS

It is only by clarifying distinctions that people are ever going to get anything straight.

6. Do not say "the computer" when you mean "the system" or "the program."

7. Don't say "a malfunctioning computer" (hardware error) if the computer functioned as it was directed on an incorrect program (software error). (And remember that the best programmers make mistakes, so that a catastrophic bug in a system is no sign that it was programmed by an incompetent, only that it isn't finished.)

8. (A particular point about graphics. See flip side.) Don't say "TV screen" if a computer screen is not TV, i.e., 525 horizontal lines that you can see on the screen if you look for them. (See p. DM6 versus p. DM23.) HOW ABOUT: "visual display screen"? -- you can add, "on which the computer can draw moving lines," or whatever else the particular system does.

9. Don't assume that your audience is computer-illiterate.

10. Don't assume that it can't all be said simply. Only lazy or hard-pressed writers are unclear.

11. Do not use cutesy-talk, particular that which suggests that computers have an intrinsic character. By "cutesy" I mean sentences like "Scientists have recently taught a computer to play chess," Mis-Leads like "What does a computer sound like?" (when talking about music constructed by a particular program in a particular way), and awe-struck descriptions like, "At last the Space Age has come to the real estate business..."

12. Do not use the garbage term "computerized," unless there is a clear statement of where the computer is in the system, what the computer is doing and how. A "computerized traffic system," for instance, could be any damn thing, but a "system of traffic lights under computer control, using various timing techniques still under development," says something.

13. Don't put in clichés as fact, for example by the use of such terms as "mathematical" or "computer scientist" unless they really apply. Do not imply any mathematical character unless you know the system possesses it: many programs contain no operations that can fairly be called mathematical. Similarly, a "computer scientist" is someone widely or deeply versed in computers or software, not just a programmer. (Anyway, if something has been programmed by an entomologist, it is probably more interesting to refer to him as an entomologist than as a "computer scientist.")

14. Do not refer to apparent intelligence of the computer (unless that is an intended feature of the program). Credit rather the ingenuity of the system's creator. Do not say "the clever computer." If anybody is clever it is the programmer or program designer, and if you think so, say so. These guys don't get the recognition they deserve.

15. Never, never say "teach the computer" as an elliptical way of saying "write computer programs." Programming means creating exact and specific plans that can be automatically followed by the equipment. To say "teach" when you mean "program" is like "persuading" a car instead of driving it, or making a toilet "cry" instead of flushing it.

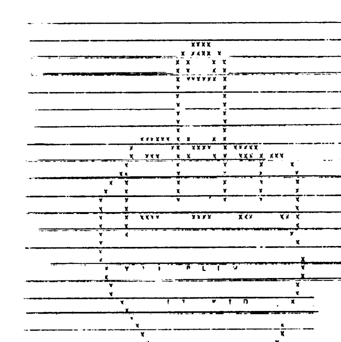
(There are systems, described on the flip side, which simulate intelligent processes and may thus be said to "learn" or "be taught." But neither programming nor simulated learning should be described in a slipshod fashion that suggests the computer is some sort of trainable baby, puppy or demon.)

16. Do not imply that something is "the last word," unless you have checked that it is.

BIBLIOGRAPHY

Ernest Gowers, Plain Words.

This wonderful little book showed English civil servants "bureaucratic writing" was totally unnecessary. Its precepts-- mainly concerned with calling a spade a spade (see p. 12)-- transpose exactly to the computer world.



"You Blew It, Kid"--

bad news for student programmers in their unsuccessful printouts.

U. Illinois at Urbana.

COMPUTER FUN & MISCHIEF

All kinds of dumb jokes and cartoons circulate among the public about computers. Then our friends regale us computerfolk with these jokes and cartoons, and because we don't laugh they say we have no sense of humor.

Oh we do, we do. But what we laugh at is rather more complicated, and relates to what we think of as the real structure of things.

Some of the best humor in the field is run in Datamation; an anthology called Faith, Hope and Parity reran a lot of their best pieces from the early sixties. Classic was the Kludge series, a romp describing various activities and products of the Kludge Komputer Korporation, whose foibles distilled many of the more idiotic things that have been done in the field. ("Kludge," pronounced "klooj," is a computerman's term for a ridiculous machine.) Datamation's humorous tradition has continued in a ponderous but extremely funny serial that ran in '72 called Also Sprach von Neumann, which in mellifluous and elliptical euphemisms described the author's adventures at the "airship foundry" and other confused companies that had him doing one preposterous thing with computers after another.

COMPUTER PRANKS

Pranks are an important branch of humor in the field. Here are some that will give you a sense of it.

ZAP THE 94

One of the meaner pranks was a program that ran orr the old 7094. It could fit on one card (in binary), and put the computer in an inescapable loop. Unfortunately the usual "STOP" button was disabled by this program, so to stop the program one would eventually have to pull the big emergency button. This burnt out all the main registers.

TIMES SQUARE LIGHTS

One of the weirder programs was the operator-waker-upper somebody wrote for the 7094. It was a big program, and what it did was DISPLAY ALPHABETICAL MESSAGES ON THE CONSOLE LIGHTS, sliding past like the news in Times Square. You put in this program and followed it with the message; the computer's console board would light up and the news would go by. Since the lights usually blink in uninteresting patterns, this was very startling.

This program was extremely complex. Since the 94 displayed the contents of all main registers and trap, arithmetic and overflow lights, it was necessary to do very weird things in the program to turn these lights on and off at the right times.

THE TIME-WASTER

In one company, for some reason, it was arranged that large and long-running programs had priority over short quick ones. Very well: someone wrote a counterattack program occuping several boxes of punch cards, to which you added the short program you really wanted run, and a card specifying how long you wanted the first part of the program to grind before your real one actually started.

This would blink lights and spin tapes impressively and lengthen the run of your program to whatever you wanted.

BOMBING THE TIME-SHARE

One of the classic bad-boy pranks is to bomb time-sharing systems-- that is, foul them up and bring them to a halt. Many programmers have done this; one has told me it's a wonderful way to get rid of your aggressions.

Of course, it can damage other people's work (especially if disks are bombed); and it always gets the system programmers hopping mad, because it means you've defied their authority and maybe found a hole they don't know about. Here are a couple of examples.

1. THE PHANTOM STRIKES

The way this story is told, one of the time-sharing systems at MIT would go down at completely mysterious times, with all of core and disk being wiped out, and the lineprinter printing out THE PHANTOM STRIKES.

For a long time the guilty program could not be found. Finally it was discovered that the bomb was hidden in an old and venerable statistics program previously believed to be completely reliable. The reason the phantom didn't always strike was that the Bomb part queried the system clock and made a pseudo-random decision whether to bomb the system depending on the instantaneous setting of the clock. This is why it took so long to discover; the program usually bided its time and behaved properly.

Apparently this was the revenge of a disgruntled programmer, long since departed. Not only that, but his revenge was thorough: the Bomb part of the program was totally knitted into the rest of it, it was a very important program that had to be run a lot with different data, and no documentation existed, making it for practical purposes impossible to change.

The final solution, so the story goes, was this: whenever the rowdy program had to be run, the rest of the machine was cleared or put on protect, so it ran and had its fits in majestic solitude.

2. RHBOMB

The time-share at the Labs, never mind which Labs, kept going down. Mischief was suspected. Mischief was verified: a program called RHBOMB, submitted by a certain programmer with the initials R.H., was responsible, and turned out always to be present when the terminals printed TSS HAS GONE DOWN. It was verified by the systems people that the program called RHBOMB was in fact a Bomb program, with no other purpose than to take down the time-sharing system.

R.H. was spoken to sternly and it did not happen again.

However, some months later a snoopy systems programmer noted that a file called RHBOMB had been stored on disk. Rather than have R.H. scalped prematurely, he thought he would check the contents.

He sat down at the terminal and typed in the command, PRINT RHBOMB. But before he could see its contents, the terminal typed instead

TSS HAS GONE DOWN

But this was incredible! A program so virulent that if you just tried to read its contents, without running it, it still bombed the system! The systems man rushed from the room to see what had gone wrong.

He did so prematurely. The contents of the new file RHBOMB were simply

TSS HAS GONE DOWN

followed by thousands of null codes, which were silently being fed to the Teletype, 10 per second, preventing it from signalling that it was ready for the next thing.

GAMES

Games with computer programs are universally enjoyed in the computer community. Wherever there are graphic displays there is usually a version of the game Spacewar. (see Steward Brand's Spacewar piece in Rolling Stone, mentioned elsewhere.) Spacewar, like many other computer-based games, is played between people, using the computer as an animated board which can work out the results of complex rules.

Some installations have computer games you can play against; you are effectively "playing against the house," trying to outfox a program. This is rarely easy. A variety of techniques, hidden from you, can be used.

When "a computer" plays a game, actually somebody's program is carrying out a set of rules that the programmer has laid out in advance. The program has a natural edge: it can check a much longer series of possibilities in looking for the best move (according to the criteria in the program).

There is a more complicated approach: the computer can be programmed to test for the best strategy in a game. This is much more complicated, and is ordinarily considered an example of "artificial intelligence" (see "The God-Builders," elsewhere in this book).

CONWAY'S GAME OF LIFE

A Grand Fad among computerfolk in the last couple of years has been the game of "Life," invented by John Horton Conway.

The rules appeared in the Scientific American in October 1970, in Martin Gardner's games column, and the whole country went wild. Gardner was swamped with results (many published in Feb. 71); after a couple more issues Gardner washed his hands of it, and it goes on in its own magazine.

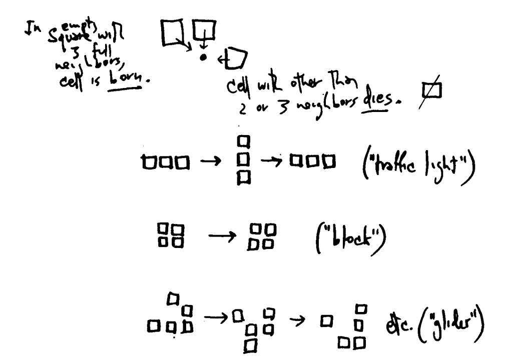
The game is a strange model of evolution, natural selection, quantum mechanics or pretty much whatever else you want to see in it. Part of its initial fascination was that Conway didn't know its long-term outcomes, and held a contest (eventually won by a group from MIT).

The rules are deceptively simple: suppose you have a big checkerboard. Each cell has eight neighbors: the cells next to it up, down and diagonally.

Time flows in the game by "generations." The pattern on the board in each generation determines the pattern on the board in the next generation. The game part simply consists of trying out new patterns and seeing what things result in the generations after it. Each cell is either OCCUPIED or EMPTY. A cell becomes occupied (or "is born") if exactly three of its neighbors were full in the previous generation. A cell stays occupied if either two or three of its neighbors were occupied in the previous generation. All other cells become empty ("die").

These rules have the following general effect: patterns you make will change, repeat, grow, disappear in wild combinations. Some patterns move across the screen in succeeding generations ("gliders"). Other patterns pulsate strangely and eject gliders repetitively (glider guns). Some patterns crash together in ways that produce moving glider guns. Weird.

While the game of Life, as you can see from the rules, has nothing to do with computers intrinsically, obviously computers are the only way to try out complex patterns in a reasonable length of time.



NON-OBVIOUS RESULTS OF SOME SIMPLE PATTERNS: some die, one blinks back and forth, others become stable. (Conway's Game of Life programmed for PLATO by Danny Sleator.)

BIBLIOGRAPHY

Donald D. Spencer, Game Playing with Computers. (Spartan/Hayden, $13.) This includes flow-charts, programs and what-have-you for some 25 games, and suggestions for more.

A continuing series of game programs (mostly or all in BASIC) appears in PCC, a newspaper mentioned earlier.

Stewart Brand's marvelous Spacewar piece, also mentioned earlier, is highly recommended.

Robert C. Gammill, "An Examination of Tic-Tac-Toe-like Games." Proc. NCC 74, 349-355. Examines structure of simple games (esp. 3D tic-tac-toe or QUBIC) where forced wins are possible; and program structures to play them.

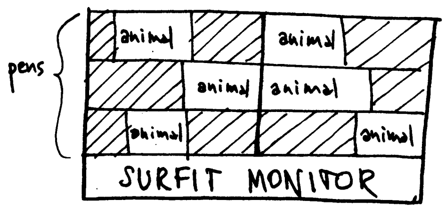
"The Game of Life," Time, 21 Jan 74, 66-7.

(Lifeline, said to be published by Robert T. Wainwright of Wilton, Connecticut.)

SURVIVAL OF THE FITTEST

One of the stranger projects of the sixties was a game played by the most illustrious programmers at a well-known place of research; the place cannot be named here, nor the true name of the project, because funds were obtained through sober channels, and those who approved were unaware of the true nature of the project, a game we shall call SURFIT ("SURvival of the FITtest".) Every day after lunch the guys would solemnly deliver their programs and see who won. It was a sort of analogy to biological evolution. The programs would attack each other, and the survivors would multiply until only one was left.

It worked like this. Core memory was divided up into "pens," one for each programmer, plus an area for the monitor.



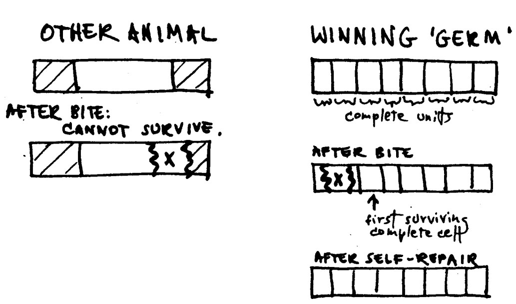
Each program, or "animal," could be loaded anywhere in its pen. The other programs knew the size of the pen but not where the animal was in it. Under supervision of the special monitor, the animals could by turns bite into the other pens, meaning that the contents of core at several consecutive locations in the other pen was brought back, and changed to zero in its original pen.

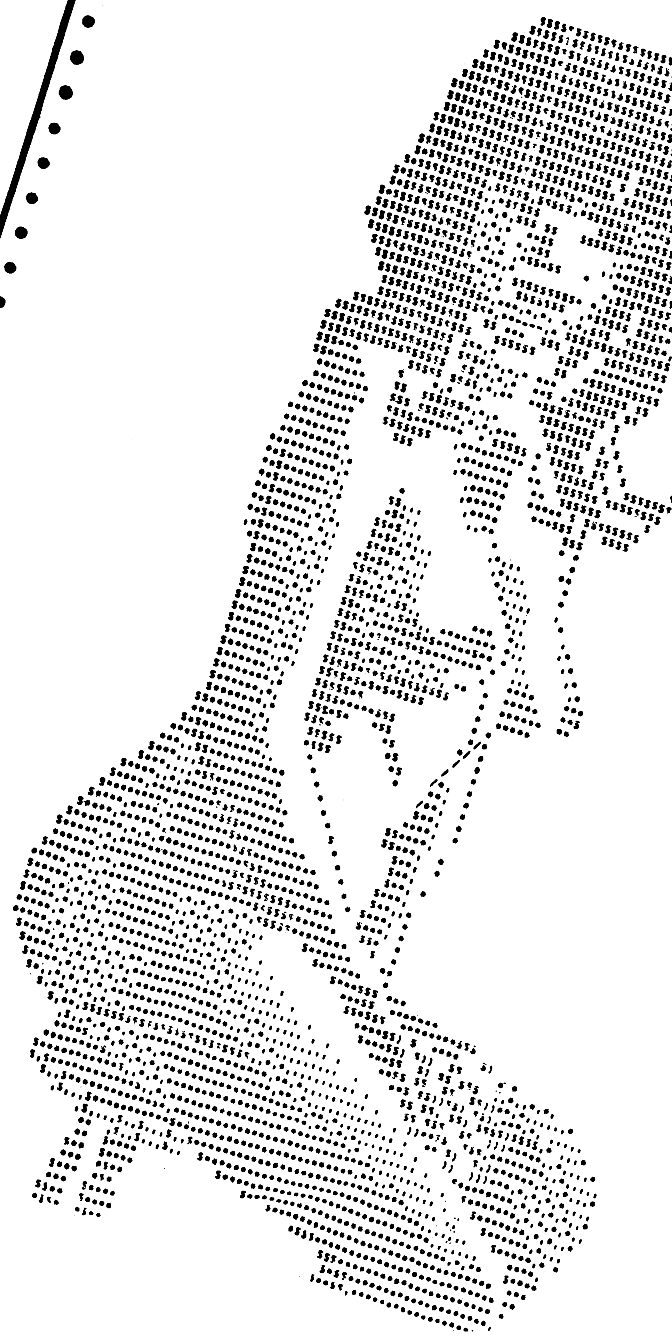
Your animal could then "digest"-- that is, analyze-- the contents bitten. Then the other animal got his turn. If he was still alive-- that is, if the program could still function-- it could stay in play; otherwise the animal who had bitten it to death could multiply itself into the other pen.

The winner was the guy whose animal occupied all pens at the end of the run. If he won several times in a row he had to reveal how his program worked.

As the game went on, more and more sophistication was poured into the analytic routines, whereby the animal analyzed the program that was its victim; so the programmer could attack better next time. The programs got bigger and bigger.

Finally the game came to a close. A creature emerged who could not be beaten. The programmer had reinvented the germ. His winning creature was all teeth, with no diagnostic routines; and the first thing it did was multiply itself through the entirety of its own pen, assuring that no matter where it might just have been bitten, it would survive.





When word got around that this nude was in a public file on the time-sharing system, my office-mates scrambled to get printouts of her. The cleverest, though, had a deck punched. As he predicted, she was thrown off by the systems people within an hour or so-- leaving the other guys with their printouts, but he had the deck. Now he can put her back in the computer any time, but they can't.

Twitting a program within its own premises is a jolly aspect of computer fun. This game of three-dimensional tic-tac-toe was played with a program running on a minicomputer at the Spring Joint, 1969. CAUTION-- ADULTS ONLY. (X) While this example may offend some people, it vividly shows how programs may be toyed with -- in this case, by the mischievous sign-on-- to make them behave humorously.

```
NAME?MOTHERFUCKER

PLEASE PRINT YOUR NAME IN THREE PARTS SEPARATED BY BLANKS.

EXAMPLE: MR. JOHN JACKSON

NAME?M. OTHER FUCKER

PLEASE PRINT YOUR NAME IN THREE PARTS SEPARATED BY BLANKS.

EXAMPLE: MR. JOHN JACKSON

NAME?MR. MOTHER FUCKER

THANK YOU, MOTHER.  YOU MAY TERMINATE THE FOLLOWING PRINTOUT
AT ANY TIME BY PRESSING THE 'BREAK' KEY.

 . . .

DO YOU WANT INSTRUCTIONS?NO.

PLEASE TYPE 'YES' OR 'NO'?NO

DO YOU WANT TO MOVE FIRST?YES
YOUR MOVE?141
MACHINE MOVES TO 111

X... .... .... ....
.... .... .... ....
.... .... .... ....
O... .... .... ....

YOUR MOVE?411
MACHINE MOVES TO 414

X... .... .... O..X
.... .... .... ....
.... .... .... ....
O... .... .... ....

YOUR MOVE?144
MACHINE MOVES TO 114

X..X .... .... O..X
.... .... .... ....
.... .... .... ....
O..O .... .... ....

YOUR MOVE?441
MACHINE MOVES TO 444

X..X .... .... O..X
.... .... .... ....
.... .... .... ....
O..O .... .... O..X

YOUR MOVE?
     434
MACHINE MOVES TO 222

X..X .... .... O..X
.... .X.. .... ....
.... .... .... ...O
O..O .... .... O..X

YOUR MOVE?
     333
MACHINE MOVES TO 323

X..X .... .... O..X
.... .X.. ..X. ....
.... .... ..O. ...O
O..O .... .... O..X

YOUR MOVE?
     233
NICE TRY MOTHER--MACHINE MOVES TO 322

X..X .... .... O..X
.... .X.. .XX. ....
.... ..O. ..O. ...O
O..O .... .... O..X

YOUR MOVE?
     433
NICE TRY MOTHER--MACHINE MOVES TO 133

X..X .... .... O..X
.... .X.. .XX. ....
..X. ..O. ..O. ..OO
O..O .... .... O..X

YOUR MOVE?
     431
NICE TRY MOTHER--MACHINE MOVES TO 421

X..X .... .... O..X
.... .X.. .XX. X...
..X. ..O. ..O. O.OO
O..O .... .... O..X

YOUR MOVE?
     432
CONGRATULATIONS MOTHER
YOU WIN AS FOLLOWS 431    432    433    434
FINAL POSITION:

X..X .... .... O..X
.... .X.. .XX. X...
..X. ..O. ..O. OOOO
O..O .... .... O..X

YOU CAN TRY SOME OTHER PROGRAMS OR PLAY ANOTHER GAME
WOULD YOU LIKE ANOTHER GAME MR. FUCKER?
```

HOW COMPUTER STUFF IS BOUGHT AND SOLD

For the most part, big computers have always been rented or leased, rather than bought outright. There are various reasons for this. From the customer's point of view, it makes the whole thing tax-deductible without amortization problems, and means that it's possible to change part of the package-- the model of computer or the accessories-- more easily. And big amounts of money don't have to be shelled out at once.

From the manufacturer's point of view (and of course we are speaking mostly of IBM), it is advantageous to work the leasing game for several reasons. Cash inflow is steady. The manufacturer is in continuous communication with the customer, and has his ear for changes and improvements costing more. Competitors are at a disadvantage because the immense capital base needed to get into the selling-and-leasing game makes competitition impossible.

Basically, leasing really may be thought of as having two parts: the sale of the computer, and banking a loan on it; essentially the lease payments are installment payments, and the real profits come after the customer has effectively paid the real purchase price and is still forking over.

Many firms other than IBM prefer to sell their computers outright. Minicomputers are almost always sold rather than rented. However, for those who believe in renting or leasing, the so-called "leasing firms" have appeared, effectively performing a banking function. They buy the computer, you rent or lease it from them, and they make the money you would've saved if you'd bought.

IBM, now required to sell its computers as well as lease them, keeps making changes in its systems which cynics think are done partly to scare companies away from leasing, since if you've bought the computer you can't catch up. (Large computers bought from companies that like to sell them, such as DEC and CDC, do not seem to have this problem.)

UH OH, MAINTENANCE

A practical problem of immense importance is "maintenance," meaning repair and upkeep of computers and their accessories. Lots of guys in Boston and L.A. are having fun making computers, but here you are stuck in Squeedunk and it doesn't work anymore.

Trying to find people who will fix these things on a stable basis is a great problem.

You can sign a "maintenance contract" with the manufacturer, which is sort of like breakdown insurance: whatever happens he'll fix. Eventually. If you own equipment from different manufacturers, though, it's worse: each manufacturer will only contract to fix his own equipment. (And remember, interfaces have to be maintained too.)

This is the biggest point in favor of IBM. Their maintenance is superb.

There's also something called third-party maintenance: companies who'll contract to keep all your hardware working. RCA and Raytheon are into that.

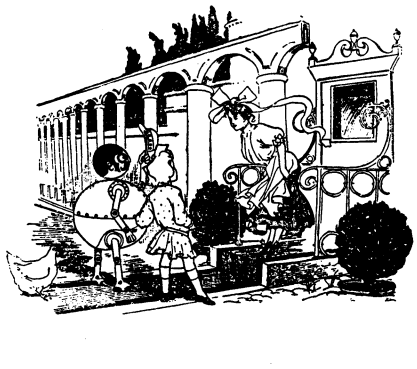
THE SEVEN DWARVES AND THEIR FRIENDS

The computer companies are often called "Snow White and the Seven Dwarves," even though the seven keep changing. Here are some main ones beside IBM. I hope I haven't left anyone out.

- Sperry Rand Univac
- Honeywell
- Burroughs
- Control Data Corporation (CDC)
- National Cash Register (NCR)
- Digital Equipment Corporation (DEC)
- Xerox Data Systems (XDS; formerly Scientific Data Systems (SDS))
- Hewlett-Packard (HP)
- Data General
- Interdata, Inc.
- Varian Data Machines
- Lockheed

Requiescant in Pace:

- General Electric (sold out to Honeywell)
- RCA (sold out to Univac)
- Philco
- General Foods
- & others beyond recollection.



SOFTWARE

Computer programs, or "software," used to come free with the computer. But IBM turned around and "unbundled," meaning you had to buy it separately, and there has been some following of this example. However, for users who are buying a computer with some canned program for a particular purpose, prices are obviously for the whole package; it's people who use the same computer for a lot of different things that have to pay for individual programs.

There are many small software companies. For the cost of a letterhead anyone can start one; the question is whether he has anything special to sell. Some people whomp up programs on their own which turn out to be quite useful. (For instance, one Benjamin Pitman offers a magnificent program in Fortran to generate textual garbage. It's so good it can be used to expand proposals by hundreds of pages. He calls it Simplified Integrated Modular Prose (SIMP) and it sells for $10. His address is Computer Center, University of Georgia, Athens GA 30602*.)

Obviously, to create big systems for intricate management purposes requires a great deal more effort. Traditionally these are done by vast programmer teams working in COBOL or the like, constantly fighting with monitor programs and chewing up millions of dollars. However, the new Quickie Languages (three shown pp. 16-25) may offer great simplification of such programming tasks.

Programs are protected by copyright-- that's the only way there can be a software industry at all-- but since there has been no court litigation in the field, nobody knows what the law really is or what it covers. Everybody agrees that traditional copyright precedent covers a lot of ground-- "derivative works" definitely violate copyright, even study guides to textbooks-- -- but no one knows how far this goes.

Same for patents. The Patent Office has granted program patents, notably the one on the sorting program of Applied Data Research, Inc., but The Patent Office has a profound distaste for this potential extension of its duties, and is telling everyone that programs aren't patentable, even though they clearly fall within its mandate as unique, original processes.

People who only read the headlines think that the Supreme Court struck down the patentability of programs. No such thing.

In this light the patents that the University of Utah has gotten on the halftone image synthesis programs of Warnock and Wylie and Romney (see p.) are of considerable interest. These patents use the "software-as-hardware" ruse: the program is described in detail as taking place in a fictitious machine shown in many detailed drawings whose nebulous character is not readily seen by the uninitiated: events vaguely taking place in "microprogrammable microprocessors" have been neatly foisted on the Patent Office as detailed technical disclosure. It's a great game. The idea is that the claims are so drawn as to cover not just the fictitious machine, but any program that should happen to work the same way. But such approaches, though common to previous patent practices, have not yet been litigated in this field.

USED COMPUTERS

While in principle there would seem to be every advantage in buying used computers, there are certain drawbacks. Service is the main one: the manufacturer is not very helpful about fixing discontinued machines, and you may have to know how to do it yourself. Even with machines still available, you may have trouble getting onto a service contract from the manufacturer, since it "may have been mistreated." (American Used Computer, in Boston, will usually guarantee that its merchandise will be accepted back into manufacturer's contract service.) A final drawback is price: a popular machine may cost as much used as new, since they're saving you the waiting period.

It's kind of unfortunate: otherwise usable machines get wasted. (But here's waste for you: certain well-known laboratories, owned by a profit-making monopoly, smash their used computers if nobody wants them within the lab. They claim they can't resell them beceuse they would then be "competing" with the manufacturers. I wish the conservationists would get on that one.)

(Notes from all over: it seems that all the surviving numbers of the Philco computer, a nice machine but very much discontinued, have either gone to the state of Israel or to Pratt Institute in Brooklyn. When I spoke at Pratt they showed me their Philco machines, chugging healthily, and said they had (I think) some four more Philcos in crates, donated by their original owners.)

ANNOUNCEMENTS

An eccentric aspect of the computer field is the Announcement, the statement by a company (or even individual) that he is planning to make or sell a certain computer or program. Some very odd things happen with announcements in this field. (None of this is unique to computerdom, but it goes to unusual extremes here.)

Under our system it is permissible for any person or firm to announce that he will make or sell any particular thing, and even if he's lying through his teeth, it's not ordinarily considered fraud unless money changes hands. Talk is cheap. Thus it is common practice in American industry for people to say that they will soon be selling hundred-mile-an-hour automobiles, tapioca-powered rocketships, antigravity belts.

Okay. In the computer world the same thing happens. The strategy depends on the announcer's market position. The little guys are often bluffing wistfully, hoping someone will get interested enough to put up the money to finish the project, or the like; the big companies are often "testing the water," looking to see whether there are potential customers for what they haven't even attempted to develop. Announcements by big companies also have strategic value: if they announce something a smaller guy has already announced, they may cut him off at the pass, even though they have no intention of delivering. That's just one example. The analysis of IBM's announcements is a parlor game in the field. It has been alleged, for instance, that IBM announced its 360 computer long before it was ready to cut off incursions on its customers by other firms; Control Data, in a recent suit, alleged that the Model 90 numbers of the 360 were announced, and then developed, simply to destroy Control Data and its own big fast machines. These are just examples.

In other words, caveat auditor.

Datamation ran several good articles on buying computer stuff in its September, 15, 1970 issue.

"Software Buying" by Howard Bromberg (35-40) and "Contract Caveats" by Robert P. Bigelow (41-44) are very helpful warnings about not getting burned.

Another, "Project Management Games," by Werner W. Leutert (24-34) is an absolutely brilliant, bloodcurdling strategic analysis of the ploys and dangers involved in buying and selling very expensive things, such as computers and software. ANYONE INVOLVED IN COMPUTER MANAGEMENT SHOULD READ THIS MACHIAVELLIAN PIECE WITH THE GREATEST CARE. Anyone interested in the theory of showdown and negotiation can read it with a different slant.

Some novelty programs offered by Benj. Pitman, Esq.* (see text)

Additional Programs

Calendar
This FORTRAN program can produce a calendar for any year from 1968 through 2100. The calendars are printed at the bottom of a Playboy bunny who is perched atop a bar stool. $20.

Birth Announcement
This PL/1 program produces a picture of a baby in the fetal position with the details of a birth announcement in the interior. The details may occupy up to 160 characters. See reduced copy attached. $15.

Simplified Integrated Modular Prose (SIMP)
This FORTRAN program produces page after page of double talk. The input is a permanent data deck of phrases followed by one or more user-written title cards. Each title card is placed at the top of a one page listing. The program uses a letter from the title to prime a random number generator and selects phrases from the table and combines them into sentences and paragraphs to produce the 'report'. The program may easily be expanded by increasing the size of the arrays and the data in the phrase deck. See attached example. $10.

Santa and Reindeer
The deck when listed produces a Santa in his sleigh being pulled over several tree tops and houses by 6 reindeer. Above are the words MERRY CHRISTMAS and HAPPY NEW YEAR. $5.

Snoopy
The deck when listed produces a Snoopy. $5.

* More recent address:
c/o Computech Systems Inc.,
1819 Peachtree Rd.,
Atlanta GA 30309.

STATUS REPORT NUMBER 5

THE PRODUCT CONFIGURATION BASELINE REQUIRES CONSIDERABLE ANALYSIS AND TRADE-OFF STUDIES TO ARRIVE AT THE TOTAL SYSTEM RATIONALE. THEREFORE, IT IS INTUITIVELY OBVIOUS THAT A CONSTANT FLOW OF EFFECTIVE INFORMATION MAXIMIZES THE PROBABILITY OF SUCCESS AND MINIMIZES THE COST REQUIRED FOR THE ANTICIPATED THIRD GENERATION EQUIPMENT.

AS A RESULTANT IMPLICATION, PARALLEL ORGANIZATIONAL CAPABILITY BALANCES MANAGEMENT FLEXIBILITY FOR THE STRUCTURAL DESIGN, BASED ON SYSTEM ENGINEERING CONCEPTS. ACCORDINGLY, INITIATION OF CRITICAL SUBSYSTEM DEVELOPMENT NECESSITATES THAT URGENT CONSIDERATION BE APPLIED TO THE PRELIMINARY QUALIFICATION LIMIT. IN THIS REGARD, THE INDEPENDENT FUNCTIONAL PRINCIPLE IS FURTHER COMPOUNDED, WHEN TAKING INTO ACCOUNT THE PHILOSOPHY OF COMMONALITY AND STANDARDIZATION.

ON THE OTHER HAND, THE FULLY INTEGRATED TEST PROGRAM ADDS EXPLICIT PERFORMANCE LIMITS TO THE SUBSYSTEM COMPATIBILITY TESTING.

THUS, IT CAN EASILY BE SEEN THAT ANY ASSOCIATED SUPPORTING ELEMENT EFFECTS THE SIGNIFICANT IMPLEMENTATION TO SYNCRONIZED DIGITAL PROGRAMMING.

BASED ON INTEGRAL SUBSYSTEM CONSIDERATIONS, THE INCORPORATION OF ADDITIONAL SYSTEM CONSTRAINTS ADDS OVERRIDING PERFORMANCE CONSTRAINTS TO THE EVOLUTION OF SPECIFICATIONS OVER A GIVEN TIME PERIOD.

IN RESPECT TO SPECIFIC GOALS, A PRIMARY INTERRELATIONSHIP BETWEEN SYSTEM AND/OR SUBSYSTEM TECHNOLOGIES IS MONITORED TO ENSURE CONTINUOUS INTERFACING WITH THE MINIMUM THRU-PUT TIME CONCEPT. TO ACHIEVE THIS, ANY TRANSITIONAL INPUT OR OUTPUT STATE PRESENTS EXTREMELY INTERESTING CHALLENGES TO ANY DISCRETE CONFIGURATION MODE.

SIMILARLY, THE CHARACTERIZATION OF SPECIFIC CRITERIA MUST UTILIZE AND BE FUNCTIONALLY INTERWOVEN WITH THE SOPHISTICATED HARDWARE. IN PARTICULAR, DIRECT ACCESS MODULAR I/O DEVICES WILL GENERATE THE MULTI-RESPONSE PARAMETERS FOR ANY DESIGNATED AUTOMATIC PERIPHERAL DEVICE.

HOWEVER, A LARGE PORTION OF THE INTERFACE COORDINATION COMMUNICATION RECOGNIZES THE IMPORTANCE OF OTHER SYSTEMS AND THE NECESSITY FOR TIME-PHASED INTERRUPT HANDLING.

SIMP EXAMPLE

HOW (SOME) COMPUTER COMPANIES ARE FINANCED — A PERSPECTIVE

Those of us who were around will never forget the Days of Madness (1968-9). Computer stocks were booming, and their buyers didn't know what it was about; but everywhere there were financial people trying to back new computer companies, and everywhere the smart computer people who'd missed out on Getting Theirs were looking for a deal.

Datamation for November 1969 was an inch thick, there were that many ads for computers and accessories.

At the Fall Joint Computer Conference that year in Las Vegas, I had to cover the highlights of the exhibits in a hurry, and it took me all afternoon, much of it practically at a trot. Then, after closing time, I found out there had been a whole other building.

It is important to look at how a lot of these companies were backed, the better to understand how irrationality bloomed in the system, and made the collapse of the speculative stocks in 1970 quite inevitable.

A number of companies were started at the initiative of people who knew what they were doing and had a clear idea, a new technique or a good marketing slant. These were in the minority, I fear.

More common were companies started at the initiative of somebody who wanted to start "another X"-- another minicomputer company, another terminal company, expecting the product somehow to be satisfactory when thrown together by hired help. Perhaps these people saw computer companies as something like gold mines, putting out a common product with interchangeable commodity value.

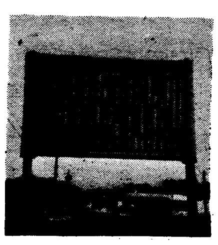
The deal, as some of these Wall St. hangers-on would explain it, was most intriguing. Their idea was to create a computer company on low capital, "bring it public" (get clearance from the SEC to sell stock publicly), and then make a killing as the sheep bought it and the price went up. Then, if you could get a "track record" based on a few fast sales, the increasing price of your stock (these are the days of madness, remember) makes it possible to buy up other companies and become a conglomerate.

The Editor, TIME
Rockefeller Center
New York, New York 10020

Sir:

Even if you missed out on all the skyrocketing new stock issues, such as Educational Computer Corp., Frigitronics, Nathan's Famous Inc., and Minnie Pearl's Chicken System (June 14), there's still hope. You can buy stock in my new company. The price has already tripled (from 5¢ a share to 15¢) and nothing stands in the way of further rises: the company has no assets no product, no employees, and no plans. Thus it is a much safer investment than other hot issues--the company can't possibly lose money. The one thing we have going for us is a sure-fire name.

Sincerely yours,
Charles S. Harris
Charles S. Harris, President
Nathan's Chickentronic Computer Associates



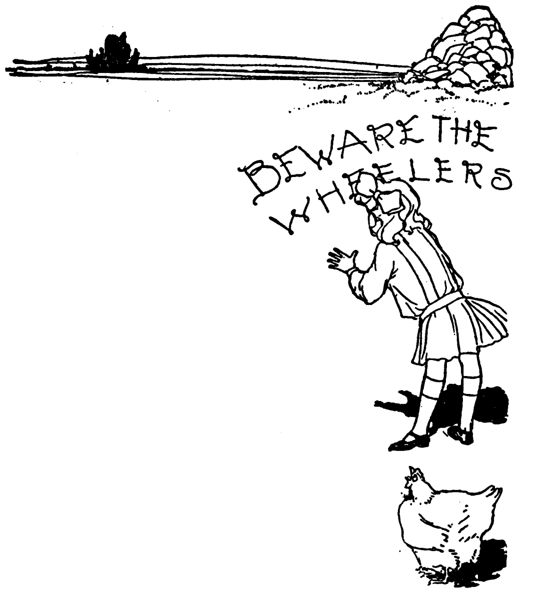
Yes, it's real.
Life imitates art
on Route 46, N.J.

It was very difficult to talk to these people, particularly if you were trying to get support for a legitimate enterprise built around unusual ideas. (Everybody wants to be second.) And what's worse, they tended to have that most reprehensible quality: they wouldn't listen. Did they want to hear what your idea actually was? "I'll get my technical people to evaluate it"-- and they send over Joe who once took COBOL. I finally figured out that such people are impossible to talk to if you're sincere-- it's a quality they find unfamiliar and threatening. I don't think there's any way a person with a genuine idea can communicate with such Wheeler-Dealers; they just fix you with a piercing glance and say "Yeah, but are we talking about hardware or software?" (the two words they know in the field).

"IT'S A WHEELER!"

The joker is that if you missed out on all this you were much better off. Anyone with a genuine idea is being set up for two fleecings: the first big one, when they tell you your ideas, skills and long-term indenture are worth 2½% (if you're lucky) compared to their immense contributions of "business knowhow," and the second, when you go public and the underwriter gets vast rakeoffs for his incomparable services. What is most likely to get lost in all this is any original or structured contribution to the world that the company was intended, in your mind, to achieve.

In part this is because anyone with technical knowledge is apparently labelled Silly Technician in the financial community, or Impossible Dreamer; it is entrenched doctrine among many people there that the man with the original idea cannot be allowed to control the direction of the resulting company. In one case known to me, a man had a beautiful invention (not electronic) that could have deeply improved American industry. It was inexpensive, simple to manufacture, profoundly effective. He made his deal and the company was started, under his direction. But it was a trick. When the second installment of financing came due (not the second round, mind you), the backers called for a new deal, and he was skewered. Result: no sales, no effect on the world, no nothing to speak of.

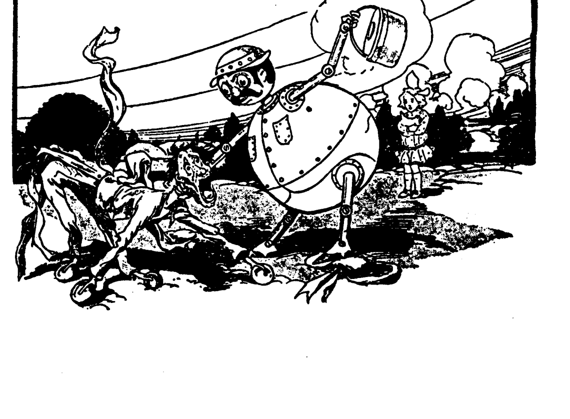


This is all the sadder because the companies that achieve important things in this field, as far as I can see, are those with a unifying idea, carried out unstintingly by the man or men who believe in it. I think of Olsen's Digital Equipment Corporation, Data General, Evans and Sutherland Computer Corporation, Vector General. This is not to say that a good idea succeeds without good management or good breaks: for instance, Viatron, a firm which was the darling of the computer high-flying stocks, had a perfectly sound idea, if not a deep one: to produce a video terminal that could be sold for as little as $100 a month. But they got overextended, and had manufacturing troubles, and that was that. (You can now get a video terminal for $49 a month, the Hazeltine.) Of course, a lot of ideas are hard to evaluate. A man named Ovshinsky, for instance, named a whole new branch of electronics after himself ("ovonics"), and claimed it would make integrated circuits cheaper or better than anybody else's. Scoff, scoff. Now Ovshinsky has had the last laugh: what he discovered some now call "amorphous semiconductor technology," and his circuits are being used by manufacturers of computer equipment. Another example is one Frank Marchuk, whose "laser computer" was announced several years ago but hasn't been seen yet. Many computer people are understandably skeptical.

This is still a field where individuals can have a profound influence. But the wrong way to try it is through conventional corporate financing. Get your own computer, do it in a garret, and then talk about ways of getting it out to the world.

BIBLIOGRAPHY

John Brooks, The Go-Go Years. Weybright & Talley. $10.



THE BEHEMOTH IBM

also known affectionately in the field as

International Big Mother
Itty-Bitty Machine Co.
International Brotherhood of Magicians
"I'm Being Moved"
Institute of Black Magic
In Bleakest Mordor
It's Better Manually

as well as

Mother of Us All
The Grim Gray Giant
Big Mama Crass
Security Blanket
Snow White
Grey Menace
and
Big Brother.

"IBM," as everyone knows, is the trade mark of the International Business Machines Corporation, an immense company centered in Armonk, N.Y., but extending to over a hundred countries and employing well over a quarter of a million people.

IBM dominates two industries, computers and electric typewriters.

To many people, IBM is synonymous with computers. Some of the public, indeed, believes them to be the only computer manufacturer.

In cameras and film, there is Kodak. In automobiles, there is General Motors. And in the computer field there is IBM.

IBM sells some 65 to 70% of all the computers and programs that are sold. In this respect, the balanced near-monopoly, they are like Kodak and GM.

But there are important differences. Everybody knows what a camera is, or an automobile. But to many, if not most, people, a computer is what IBM says it is.

The importance of this firm, for good or ill, cannot be overstated: whose legend is so thick, whose stock prices have doubled and redoubled, ten times over, to its multibillion-dollar mass; whose seeming infallibility-- at least, as seen by outsiders-- have been the stuff of legend, whose style has proliferated across the world, a style which has in a way itself become synonymous with "computers;" whose name symbolizes for many people-- remarkably, both those who love it and those who hate it-- the New Age.

The rigidity associated in the public mind with "the computer" may be related in some deep way to this organization. As a corporation they are used to designing systems that people have to use in their jobs by fiat, and thus there are few external limitations on the complications to our lives that IBM can create.

Many people mistake IBM for "just another big company," and here lies the danger. IBM's position in the world is so extraordinary, so carefully poised (as a result of various anti-trust proceedings and precautions) just outside of total monopoly of a vitally important and all-penetrating field, that much of what they do has implications for all of us. Ralph Nader's contention that General Motors is too powerful to function as an independent government surely applies even more to IBM. General Motors is not in a position to persuade the public that every car has to have ten wheels and a snowplow. IBM seems in some ways to have molded computers in its own image, and then persuaded the world that this is the way they have to be.

But IBM is deeply sensitive, in its way, to public relations, and has woven an extensive system of political ties and legends (if not mythology) which have kept it almost completely exempt from the critical attention of concerned citizens.

Thus it is necessary here, simply as a matter of covering the field at an introductory level, to raise some questions and criticisms that occur to people who are concerned about IBM. IBM presumably will not mind having these matters raised; their public-spirited concern in so many areas assures that when something so publicly important as the character of their own power is concerned, occasional scrutiny should be welcome.

A FINE PROGRESSIVE CORPORATE CITIZEN AND A WONDERFUL EMPLOYER

It is important to note first of all that IBM is in many respects the very model of a generous and dutiful corporate citizen. In "community relations," in donations to colleges and universities, in generous release of the time of its employees for charitable and civic undertakings, it is almost certainly the most public-spirited corporation in America, and perhaps on the face of the earth.

They have been generous about many public interest projects, from Braille transcription to donating photographers and facilities for films on child development.

The corporation sponsors worthwhile cultural events. "Don Quixote" with Rex Harrison on TV was terrific, Katherine Hepburn's "Glass Menagerie" was marvelous.

They treat their small suppliers honorably and with great solicitude.

IBM's enlightenment and benevolence toward its employees is perhaps beyond that of any company anywhere. They have rigorously upgraded the position of women and other minority employees; the opportunities for women may be greater there than anywhere else. They have upgraded repair of their systems, at any level, to white-collar status, and tool kits are disguised as briefcases. This innovation, making a repairman into a "field engineer," is one of the cleverest public-relations and employment policies ever instituted.

They are openhanded to employees who want to run for office, evidently regardless of platform. In the sixties there were peace candidates who worked for IBM, and evidently got time off for it. More recently, Fran Youngstein, an IBM marketing instructor, was a 1973 candidate for Mayor of New York on the ticket of the Free Libertarian Party, opposing all laws against victimless crimes (e.g. prostitution and odd sex), as well as Day Care and welfare.

They also rarely fire people. Once you're in, and within certain broad outlines, it's extremely safe employment. For those who turn out not to fit in well, they have a tradition of certain gentle pressure-practices like moving you around the country repeatedly at IBM expense. This encourages leaving, but also exposes the less-wanted employee to a variety of opportunities he might not otherwise see, without the trauma and anxiety of dismissal.

(It is said that there are IBM firings, but they are rare and formidable. Heywood Gould's description of an IBM firing (Corporation Freak, pp. 113-115), for which he does not claim authenticity, is nevertheless bloodcurdling.)

IBM's international manners (in its 115 countries) are likewise praiseworthy. Compared to the perfidious behavior of some of our other multinational corporations, they are sweetness and light and highschool civics. Sensitive to the feelings of people abroad, they are said to operate carefully within arrangements made to satisfy each country. They train nationals for real corporate responsibility rather than bringing in only outside people. And they are sensitive to issues: for instance, they recently refused to set up an Apartheid computer in South Africa.

> ONE THING IS PERFECTLY CLEAR:
>
> IBM has no monopoly on understanding or sophistication.

THEN WHY SUCH A RANGE OF FEELINGS TOWARD IBM?

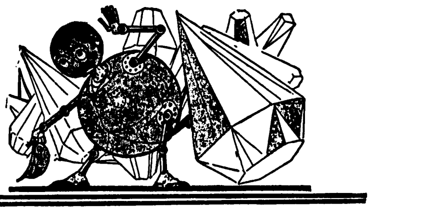
Among computer people, feelings toward IBM range from worship to furious hate (depending only in part on whether you work there).

Many, many are of course employed by IBM, and the devotion with which they embrace the corporation and its spirit is a wonder of the world.

But the spiritual community of IBM extends further. Upper-management types, especially Chairmen of Boards and comptrollers, seem to have a reverence for IBM that is not of this world, some amalgamated vision which entwines images of eternal stock and dividend growth with an idealized notion of management efficiency. Many others use and live with IBM's equipment, and view IBM as anything from "the greatest company in the world" to "a fact of life" or even "a necessary evil." In some places whole colonies of users mold themselves in its image, so that around IBM computers there are many "little IBMs," full of people who imitate the personalities and style of IBM people. (RCA, before its computer operation fell to pieces, imitated not just the design of IBM's 360 computer, but a whole range of titles and departmental names from out of IBM. The sincerest form of flattery.)

But outside this pale-- beyond the spiritual community of IBM-- there are quite a few other computer people. Some simply ignore IBM, being concerned with their own stuff. Some like IBM but happen to be elsewhere. Others dislike or hate IBM for a variety of reasons, business and social. And this smoldering hatred is surely far different in character from anybody's attitude toward Kodak or GM.

While it is not the intent here to do any kind of an anti-IBM number, it is nevertheless necessary to attempt to round out the one-sided picture that is projected outside the computer world. In what follows there is no room to try to give a balanced picture. Because IBM can speak for itself, and does so with many voices, it is more important to indicate here the kinds of criticisms which are commonly made of IBM by sophisticated people within the industry, so that IBM-worshipers will have some idea of what bothers people. But of course no attempt can be made here to judge these matters; this is just intended as source material for concerned citizens.



THE GOOD NEWS AND BAD NEWS ABOUT IBM

First, the good news	Now for the bad news...
They offer many computer programs for a variety of purposes.	These programs are not necessarily set up the way you would want them. (But if you take the trouble to adapt to them, you'll probably never get back.) The programs favor card or card-like input and; to date, strongly discourage time-sharing and widespread convenient terminal use by untrained people. IBM programs are also notoriously inefficient. (That way you have to use bigger machines for longer.)
A company or governmental agency can get immense amounts of "help" and "information" from IBM, which offers free courses, even IBM people on "released time" to look over the problems on the premises.	The courses indoctrinate with the IBM outlook, and the planted people spread it. Moreover, both mechanisms help IBM spot the people they can work with to make a big sale-- and (it is alleged by some) those who stand in the way.
IBM offers various kinds of compatibility among its systems.	It always seems to cost extra.
IBM equipment is rugged and durable, and their repairmen or "field engineers" struggle with great diligence and alacrity to keep it running.	You may not like the way it runs.

1. SOCIAL ASPECTS OF IBM.

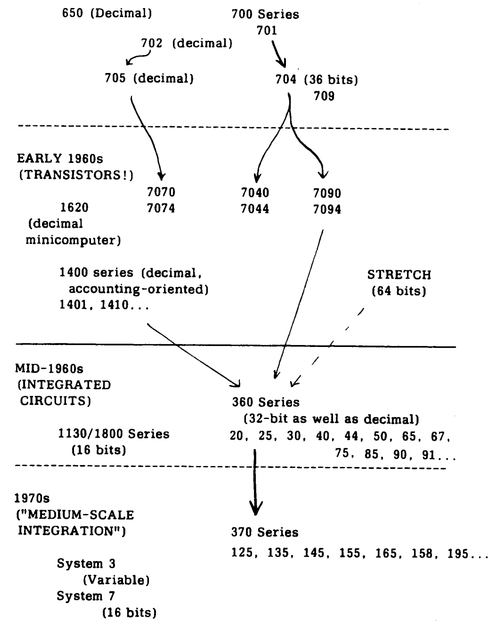
It is perhaps in the social realm, including its ideological character, that a lot of people are turned off by IBM.

IBM has traditionally been the paternalistic corporation. (Paternalistic corporations were some kind of big philosophical issue to people in the fifties, but nobody cares anymore. Anyway, the rest were perhaps inconsequential compared to IBM.) Big IBM towns not only have a Country Club (no booze), but a Homestead for the comfort of important corporate guests. There are dress codes (although non-white shirts and below-the-collar hair are now allowed), and yes, codes of private behavior (now subdued). These irritate people with libertarian concerns. They do not bother employees, evidently, because employees knew what they were getting into.

Generalizations about IBM people obviously cannot be very strong. Obviously there is going to be immense variation among 265,000 people, half of whom have college degrees; but of course one of the great truths of sociology is that any non-random group has tendencies.

More than that in this case. In a way IBM people are an ethnic group. Impressive indeed are the general energy and singlemindedness of the people, galvanized by their certainty that IBM is true, good and right, and that the IBM way is the way. This righteousness is of course a big turn-off for a lot of people. Perhaps it leads in turn to the most-heard slurs about IBM people, that they are brainwashed or provincial.

MAJOR IBM COMPUTERS AT A GLANCE



The same slick marketing could be applied to any other industry. But it wouldn't be IBM. Nowhere else could the mystery of the subject be met and enhanced with so many more mysteries.

PROVINCIAL?

There would seem to be no question that IBM people are comparatively conservative and conventional. This partly because that's who IBM hires (though they reportedly urge tolerance of the unusual employee in a training film, "The Wild Duck"). A huge number of IBM people never worked for anybody else; obviously this affects the perspective, like staying at one university all your life, or in one city.

It may also be that because IBM places such a premium on dependability and obedience, new ideas (and the abilities needed to generate them) naturally run into a little trouble. Some critics find among IBM people a heavy concern with conventional symbols of achievement, and (unfortunately) seeing the world stuck all over with conventional labels and Middle American stereotypes.

Some of the most amusing material on this comes from an odd source: a writer named Heywood Gould who, all unprepared, became a consultant to IBM, earned unconscionable amounts of money ($40,000 in six months), and lived to write a very funny and observant book about it (see Bibliography).

But it is necessary on these matters to see how difficult things can be for IBM people. To be identified as an IBM person is something like wearing a ring in your nose, a yarmulka or a halo: an entrapment in a social role that makes the individual's position awkward among outsiders. IBM people often have to take guff at parties, unless they are IBM parties. Defensiveness may account for some of the Overdo, and some of the clannishness.

BRAINWASHED?

It is true that IBM people are essentially in their own world. One theory is that compartmentalization within the firm (rather visible in their designs) may tend to stifle. Indeed, because IBM people can expect to be briefed and schooled in every technical matter they will need to know for a given assignment, the incentive to follow technical developments through outside magazines and societies may be reduced. Between Think magazine and corporate briefings, it is possible for IBM people to be comparatively (or even completely) unaware of innovations elsewhere in the field, except as these new developments are presented to them within the organization. In this light it is easy to understand the Ibmers' sense of certainty that their firm invented everything and is at the forefront.

Of course many fine research efforts do go on there, in considerable awareness of what's happening elsewhere. Particular individuals at IBM have done excellent research on everything from computer hidden-line imaging to the structure of the genetic code and computer-synthesized holograms. APL itself (see pp. 22-3), as developed by Iverson at Harvard and later programmed by him at IBM, is another example of sophisticated individual creativity there. So it's entirely possible. But IBM certainly has no monopoly on understanding or creativity, and IBM-haters sometimes talk as if the reverse is true.

I hope to be able to report in future editions of this book that IBM has moved firmly and credibly toward making its systems clear and simple to use, without re-requiring laborious attention to needless complications and oppressive rituals.

It's still possible.

One of the things we often forget is that public-spirited corporations can be reached, they do listen; and IBM is nothing if not public-spirited-- except when it comes to the design of its systems.

I hope that this book will help people who are inconvenienced by computer systems to understand and pinpoint what they think is wrong with the systems-- in their data structure, interactive properties, or other design features-- and that they will try to express their discontents intelligently and constructively to those responsible. Including, where appropriate, International Business Machines Corporation, Armonk, NY.

2. SALES TECHNIQUES.

It is IBM's alleged misbehavior in pursuit of sales that has drawn some of the strongest criticism within the industry, as well as considerable litigation. Their "predatory pricing" (a term used by the judge in the recent Telex decision), and other mean practices, are (whether true or false) folklore within the industry.

These accusations are well summarized by "Anonymous" in a recent article (see Bibliography). Basically the accusations against IBM's sales practices are that they play dirty: if you, say, the computer manager in a business firm, want to buy equipment from another outfit, IBM (so the story goes) will go over your head to your boss, accuse you of incompetence, try to get you fired if you oppose them, and Heaven knows what else. Anonymous claims that various forms of threat, intimidation, "hard-sell scare tactics" and "behind-the-scenes manipulation" are actually standard practice in IBM sales; he or she alleges various instances in certain municipalities.

Such behavior is emphatically denied, though not in relation to that article, by Board Chairman Cary, in a recent letter to Newsweek (see Bibliography). Cary emphasizes the importance of IBM's 76-page Business Conduct Guidelines. Whether these are publicly examinable is not stated.

These charges were also taken up concretely in a recent survey of computing managers done by Datamation (summarized by McLaughlin in "Monopoly Is Not a Game;" see Bibliography). In Datamation's analysis of this survey, the managers did not seem to agree with these charges against IBM. However, it must be noted-- and this seriously calls into question the entire survey as analyzed-- that out of 1100 panelists to the questionnaire, Datamation only considered 389 responses "usable," partly (it is stated) because many did not give data allowing themselves to be identified. Considering the widespread fear of IBM in the field, this may have strongly biased the poll in favor of IBM.

"When we went from IBM to National Cash Register, it was like the difference between night and day."

Retired hardware executive, talking about inventory programs

(Incidentally, it is amusing to note that even in this remaining company, in terms of "performance per dollar," the managers surveyed (and surviving the weedout) ranked the top three companies as DEC, Burroughs and Control Data. IBM was worst out of 8. Obviously service counts for a lot.)

An interesting view on IBM's sales ethics was expressed recently by Ryal R. Poppa, president of Pertec Corp.

"In the past, when there have been sales situations where 'you can't honor the policy and win the deal,' IBM has violated the policy with the practice, he said."

However, he believes that situation is changing under IBM's new management, so that the guidelines will be observed in the future. ("Poppa Sees Several IBM Changes," Computerworld, 21 Nov 73, 29.)

The people who take these matters of IBM sales practices most seriously-- IBM's competitors-- now have their own organization, the Computer Industry Association. This is an association of computer companies, which has as its intention the "establishment and preservation of a sound and viable U.S. computer industry, based on... free and open competition." Emphasis theirs. Translation: they're out to get IBM. President Dan L. McGurk, formerly of Xerox Data Systems, has blood in his eye. Membership is open only to computer companies, but their newsletter On Line is available to individuals (see Bibliography). Anyone seriously interested in these matters is referred to them.

3. TECHNICAL DECISIONS AND DESIGNS

A. Prologue.

Part of the myth of IBM's corporate perfection is based on the notion that technical matters somehow predominate in IBM's decisions, and that IBM's product offerings and designs thus emerge naturally and necessarily and inevitably from these considerations. This is rather far from the truth.

IBM presents many of their actions as technical, even as technical breakthroughs, when in fact they are strategic maneuvers. The announcement of a new computer, for example, such as the 360 or 370, is usually made to sound as if they have invented something special, while in fact they have simply made certain decisions as to "which way they intend to go" and how they plan to market things in the next few years.

IBM'S CONTROL

THE VIRTUAL MECHANICS

IBM controls the industry principally by controlling its customers. Through various mechanisms, it seems to enforce the principle that "Once an IBM customer, always an IBM customer." With an extraordinary degree of control, surely possessed in no other field by any other organization in the free world, it dictates what its customers may buy, and what they may do with what they get. More than this: the exactions of loyalty levied upon IBM's customers are similar, in kind and degree, to what it demands of its own employees. IBM makes the customer's employees more and more like its own employees, committing them as individuals, and effectively committing the company that buys from it, to IBM service in perpetuity.

Here are some of the ways this system of control seems to work. We are not saying here that this is necessarily how IBM plans it; rather, these are the virtual mechanics, virtual in the old sense; this is how it might as well be working. In the anthropological sense this is a "functional" analysis, showing the tie-ins rather than the actual detailed thought processes that occur. And even if these are really the mechanics, perhaps IBM doesn't mean them to be. It might just somehow be a continuous accident.

A. Interconnection and compatibilities.

IBM acts as if it does not want competitors to be able to connect their accessories to its computers. It's as though GM could design the roads so as to prevent the passage of other vehicles than its own.

This is done several ways. First, IBM has sometimes used contractual techniques to prevent such interconnections to its systems, either forbidding other things to be attached (or at least slapping on extra service charges if they are), or declaring that it would not be responsible for overall performance of such a setup, effectively withdrawing the hardware guarantee that is such a strong selling point.

Secondly, IBM does not tell all that needs to be known in order to make these interconnections-- the details of the hardware interfaces.

Finally, IBM can simply decree, perhaps claiming technical necessity, that interconnection is impossible. For instance, IBM said for a time that their latest big program, "VS," or Virtual System, wouldn't work (translation: would not be allowed) if competitive memories were used on the computer.

Now, there are many manufacturers who think this is very wrong of IBM; who believe they should have the right to sell accessories and parts-- especially core and disk memories-- to plug onto IBM's computers. It has been generally possible for these other manufacturers to work these interconnections out awhile after the computer comes out on the market, but it's getting more difficult.

Thus the Telex Decision of September 17, 1973, in which it was decreed by the judge that IBM would have to supply complete interface information promptly when introducing a new computer, was a source of great jubilation in the computer field. However, that part of the judgment has since been cancelled.

Much the same problem exists in the software area. IBM is less than interested in helping its competitors write programs that hook up to IBM programs, so the details of program hookup are not always made clear. Here, too, many smaller companies insist they should be made to do it.

B. Control and guidance of what the customer can get.

To a remarkable degree, if you are an IBM customer, you practically have to buy what they tell you. This IBM manages by an intricate system of fluctuating degrees of sales and support and contractual dealing. The IBM customer always has several options; but these are like forced cards. IBM is always introducing and discontinuing products, and changing prices and contractual arrangements and software options in an elaborate choreography, which applies calculated pressures on the customer. IBM has a finely-tuned system of customer incentives by which it controls product phasing, to use the polite term, or planned obsolescence, as some people call it.

(Ryal R. Poppa, president of Pertec Corp., predicts that IBM customers will now be required to switch over to new products every five or six years, rather than every seven, which Poppa contends has been the figure. ("Poppa Sees Several IBM Changes," Computerworld, 21 Nov 73, 29.)

Programs, especially, are available with different degrees of approval from IBM. The technique of "support" is the concrete manifestation of approval. A supported program is one which IBM promises to fix when bugs turn up. With an unsupported program, you're on your own, God has forgotten you. Because so much of IBM's virtue lies in the strength and fervor of its support, the use of unsupported programs, or unsupported features of supported programs, is a difficult and risky matter, like driving without a map and a spare tire, or even going into the Himalayas without gloves. Effectively the withdrawal of support is the death knell of any big program, such as TSS/360, even though customers may want to go on using them.

Availability of products is in general a matter of exquisite degree. It's not so much that you can or can't get a particular thing, but that the pricing and available contracts at a given time exert strong pressure to put you where they have chosen within their currently featured product line. Moreover, extremely strong hints are always available; the salesman will tell you what model of their computers is likely to be a dead end, or, on the other hand, what model is likely to offer various options and progressive developments in the near future.

Some things are half-available, either as "RPQs" (an IBM term for special orders-- Request Price Quotation), or available to sophisticated customers at IBM's discretion.

With all the degrees of availability, it is easy for IBM to open or close by degrees various avenues in which customers are interested.

Also, different sizes of computer will or won't allow given programs or desirable program features. Many IBM customers have to get bigger computers than they would otherwise want because a given program-- for instance, a COBOL compiler with certain capabilities-- is not offered by IBM for the smaller machine. Indeed, an elaborate sizing scheme exists for matching the machine to the customer-- or, a cynic might say, assuring that you can't get the program features you ought to be able to get unless you get a larger computer than you wanted.

What it boils down to is that you, the customer, have few genuine options, especially if your firm is already committed to doing certain things with a computer. And when IBM brings out a new computer, the prices and other influences are exactingly calculated to make mandatory the jump they have in mind to the new model.

(This planning of customer transitions does not always work. When the 370 was introduced, for instance, IBM had in mind that companies with a certain size of 360 would trade up to a bigger 370. In some cases users traded down to a smaller 370, which was able to do the same work for less money, to the acute bother of IBM.)

C. Having to do things just their way.

IBM systems and programs are set up to do things in particular ways. To a remarkable degree, it is difficult to use them in ways not planned or approved by IBM, and difficult to tie systems and programs together. Programs and features which the casual observer would suppose ought to be compatible, tend not to be. For some reason compatibility always tends to cost extra. It is as though the compatibility of equipment and programs were planned by IBM as much as their product line.

Effectively the IBM customer tends to be frequently trapped in a cage of restrictions, whether this cage is intentionally created by IBM or not. One is reminded of the motto of T.H. White's anthill in The Once and Future King:

THAT WHICH IS NOT FORBIDDEN IS COMPULSORY.

The degree to which these restrictions are manipulated or intentional is, of course, a matter of debate.

D. Captive bureaucracies running in place?

Perhaps the most unfortunate thing about IBM (from an outsider's point of view) is that effectively their systems can only be used by bureaucracies whom they have trained. From keypunch operator up to installation manager, all are effectively enslaved to curious complexities that keep changing. The ever-changing structure of OS, and its quaint access methods, is just one example. It might even seem to the outside observer that IBM's game, intentional or not, is to keep things difficult and intricately fluid to retain utter control. In other words, it is as though they fostered a continual turnover of unnecessary complications to keep a captive bureaucracy running in place. People who they have indoctrinated tend not to buy opponents' computers. People who are immersed in the peculiarities of IBM systems, and busy keeping up with mandatory changes, do not get uppity. They are too busy, and the investment of their time and effort is too high for them to want to change.

Anti-IBM cynics say that a lot of the work involved in working with IBM computers is self-generated, arising from the unnecessary complexities of OS/360, JCL, TCAM and so on. But of course that cannot be evaluated here.

PROSPECTS

These remarks should clarify the bleakness of the prospect for man's future among computers if IBM's system of control really does work this way, and if it is going to go on doing so. Because it means the future that some of us hope for-- the simple and casual availability to individuals of clear and simple computer systems with extraneous complications edited away-- may be foreclosed if they can help it.

Let's all hope, then, that these things turn out not to be really true.

"... IBM in its infinite wisdom has decreed that this is the way we must go."

Cynical computer installation manager, quoted in Computerworld, 22 Aug 73, p. 4.

An interesting example of an IBM non-breakthrough was the dramatic announcement in 1964 of the 360 computer, portrayed as a machine which would at last combine the functions of both "business" computers and "scientific" computers. But other companies, such as Burroughs (with the 5500) had been doing this for some time. The quaint separation of powers between scientific computers (with all-binary storage of numbers) and business computers (decimal storage) was based only on tradition and marketing considerations, and was otherwise undesirable. In amalgamating the "two types," IBM was only rescinding their own previous unnecessary distinction. The drama of the announcement derived in large measure from the stress they had previously laid on the division. (Fortune ran an interesting piece on the decision struggles preceding the introduction of the 360 computer, and the internal arguments as to whether there should be one line of computers or two. See the five-billion-dollar gamble piece, Bibliography.)

This ties in closely with another interesting aspect of the IBM image, the public notion that IBM is a great innovator, bringing out novel technologies all the time. It is well known in the field that they are not: IBM usually does not bring out a new type of product until some other company has pioneered it. (Again remember the earlier point, that the product offering is a strategic maneuver.) But of course such facts do not appear in the promotional literature, nor are they volunteered by the salesman.

The expression for this in the field is that IBM "makes things respectable." That is, customers get that reassured feeling, when IBM adds other people's innovations to their product line, and decide it's okay to go ahead and rent or buy such a product. (This also sometimes kicks business back to the original manufacturer.)

A few examples of things that were already on the market when IBM brought them out, often making them sound completely new: transistorized computers (first offered by Philco), virtual memory (Burroughs), microprogramming (introduced commercially by Bunker-Ramo).

This is not to say that IBM is incapable of innovation: merely that they are never in a hurry about it. The introduction of IBM products is orchestrated like a military campaign, and what IBM brings out is always a carefully-planned, profit-oriented step intended not to dislocate its product line. This is not to say that they don't have new stuff in the back room, a potential arsenal of surprises of many types. But it is probable that most of them will never be seen. This is because of IBM's "impact" problem.

Unique in IBM's position is the problem of fitting new products into the market alongside its old ones. Its problem is much worse, say, than that of Procter & Gamble. The problem is not merely its size and the diversity of its products, but the fact that they may interfere with each other ("impact" each other, they say) in very complicated ways. A program like their Datatext, for example, which allows certain kinds of text input and revision from terminals, may affect its typewriter line. These are no small matters: the danger is that some new combination of products will save the customers money IBM would otherwise be getting. Innovations must expand the amount IBM is taking in, or IBM loses by making them.

These complications of the product line in a way provide a counterbalance to IBM's fearsome power. The corporation has an immense inertia based on its existing product line and customer base, and on ways of thinking which have been carefully promulgated and explained throughout its huge ranks, that cannot be revised quickly or flippantly.

Nevertheless it is remarkable how at every turn-- notably when people think IBM will be set back-- they manage to make policy decisions or strategic moves which further consolidate their position. Often these seem to involve restricting the way their computers will be used (see box, "IBM's Control.")

(The most ironic such countermove by IBM occurred a few years ago with the so-called "unbundling" decision. IBM at last agreed (on complaint from other software firms) to stop giving its programs away to people renting the hardware. Glee was widespread in the industry, which expected IBM to lower computer prices in proportion to what it would now charge for the software. Not at all. IBM lowered its computer prices by a minuscule amount and slapped heavy new prices on the software-- often charges of thousands of dollars per month.)

A persistent rumor is that IBM fires all its salesmen in a geographic area if a key or prestige sale is "lost," as when M.I.T.'s Project MAC switched over to General Electric computers in the sixties, or when Western Electric Engineering Research Center passed over IBM computers to get a big PDP-10.

Much as some people would like to believe these stories, there seems to be no documentation. You would think one such victim would write an article about it if it were true.

Finally, there is the popular doctrine of IBM's infallibility. This, too, is a ways from the truth. The most conspicuous example was something called TSS/360.

TSS/360 was a time-sharing system-- that is, the control program to govern one model of the 360 as a time-sharing computer. According to Datamation ("IBM Phases Out Work on Showcase TSS Effort," Sept. 1, 1971, 58-9), over 400 people worked on it at once for a total of some 2000 man-years of effort. And it was scrapped, a writeoff of some 100 million dollars in lost development costs. The system never worked well enough. Reputedly users had to wait much too long for the computer's responses, and the system could not really compete with those offered elsewhere.

The failure and abandonment of this program is thus responsible for IBM's present noncompetitive position in time-sharing; customers are now assured by IBM that other things are more important. IBM-haters thank their stars that this happened. Cynics think it conceivable that high-power time-sharing was dropped by IBM in order to shoo its customer base toward areas it controlled more completely.

Two other conspicuous IBM catastrophes have been specific computers: the 360 model 90 in the late sixties, and a machine called the STRETCH somewhat earlier. Both of these machines worked and were delivered to customers. (Indeed, the STRETCH is said by some to have been one of the best machines ever.) But they were discontinued by IBM as not sufficiently profitable. Therein is said to have been the "failure." (However, it has been alleged in court cases that these were "knockout" machines designed to clobber the competition at a planned loss.)

B. Negative views of IBM systems.

In the technical realm, IBM is widely unloved because many people think some or all of their computers and programs are either poor, or far from what they should be. The reasons vary.

Some of the people feeling this way are IBM customers, and for a time they had an organized lobby, called SHARE (which also facilitated sharing of programs). Recently, however, SHARE has become IBM-dominated, a sort of company union, according to my sources.

The design of the 360, while widely accepted as a fact of life, is sharply criticized by many. (See "What's wrong with the 360?", p. 41.)

IBM's programs, while they are available for a broad variety of purposes, are often notoriously cumbersome, awkward and inefficient, and sometimes dovetail very badly. However, the less efficient a program is, the more money they make from it. A program that has to be run for an hour generates twice as much revenue than if it did its work in thirty minutes; a program that has to be run on a computer with, say, a million spaces of core memory generates ten times the revenue it would in two hundred thousand.

IBM programs are often thought to be rigid and restrictive.

The complex training and restrictions that go with IBM programs seem to have interesting functions. (See box, "IBM's Control.")

C. Theories of IBM design.

The question is, how could a company like IBM create anything like the 360 (with its severe deficiencies) and its operating system or control program OS (with its sprawling complications, not present in competitors' systems)? Three answers are widely proposed: On Purpose (the conspiracy theory), By Accident (the blunder theory), and That's How They're Set Up (the Management Science theory). These views are by no means mutually exclusive.

The Management Science theory of IBM design is the only one of these we need take up.

The extensive use of group discussion and committee decisions may tend to create awkward design compromises with a certain intrinsic aimlessness, rather than incisively distinct and simple structures. (See Gould's marvelous chapter, "The Meeting," 58-80.)

Their use of immense teams to do big programming jobs, rather than highly motivated and especially talented groups, is widely viewed as counterproductive. For instance, Barnet A. Wolff, in a letter to Datamation (Sept. 1, 1971, p. 13) says a particular program

> "remains inefffficient, probably because of IBM's unfortunate habit of using trainees fresh out of school to write their systems code."

There may also be something in the way that projects are initiated and laid out from the top down, rather than acquiring direction from knowledgeable people at the technical level, that creates a tendency toward perfunctoriness and clunky structure.

Thus there may very well be no intentional policy of unnecessary complication (see Box, "IBM's Control"). But the way in which goals are set and technical decisions delegated may generate this unnecessary complication.

THE UNTOLD INSIDE STORY

It is unfortunate that Rodgers' remarkable book does not follow the details of IBM's computer designs and politics in the computer age, i.e., since 1955. Later work, perhaps helped by some Pentagon Papers, will have to relate the decision processes that occurred in this unique national institution to the systems it has produced and the stamp it has put on the world.

QUICKIE HISTORY OF IBM

IBM appeared in 1911 as the consolidation of a number of small companies making light equipment, under the name C-T-R Company (Computer-Tabulating-Record). This was prophetic, considering how aptly it described the company's future business, and especially prophetic considering that today's stored-program computer was undreamed of at that time.

According to William Rodgers' definitive company biography Think, the company's creator was a shrewd operator named Charles R. Flint, dashing entrepreneur and former gun runner to the South American republics, who in his shrewdness brought in to run the company an incredibly talented, fire-breathing and self-righteous individual named Thomas J. Watson, even though Watson at that time was under prison sentence for his sales practices at another well-known company. The sentence was never served, and Watson went on to preside for many years over a corporation to which he gave his unique stamp.

Watson arises from the pages of Think as a sanctimonious tyrant, hard as nails yet reverently principled in his words; the pillar of fervid, aggressive corporate piety.

IBM was totally Watson's creation. The company became what he admired in others, a mechanism totally obedient to his will and implementing his forceful and inspiringly rationalized convictions with alacrity. As the Church is said to be the bride of Christ, IBM might be characterized as the Bride of Watson, molded to the styles of demandingness, pressure, efficiency and pietism which so characterized that man. But the ideas flowed from Watson alone, except for a few confidantes who received his nod. The company is vastly bigger now, and slightly more colorful, in a muted sort of way; but it is still the stiff and deadly earnest battalion of his dream.

Because of Watson's background as salesman, he made Sales the apex of the corporation. The salesmen had the most prestige within the company and could make the most money; below that was administration, below that, technical staff.

Watson eliminated the meat-slicing machines, and pushed the product line based on punched cards developed by IBM's first chief engineer, Herman Hollerith. According to Rodgers, it was impetus from the Depression, and the new bookkeeping requirements of Roosevelt's remedies, that skyrocketed the firm uniquely during the depths of general economic catastrophe, till Watson came to draw the highest salary of any man in the nation. In 1934 his income was $364,432 (Will Rogers, not the author of Think, was second with $324,314). Watson had neatly arranged to get 5% of IBM's net profit.

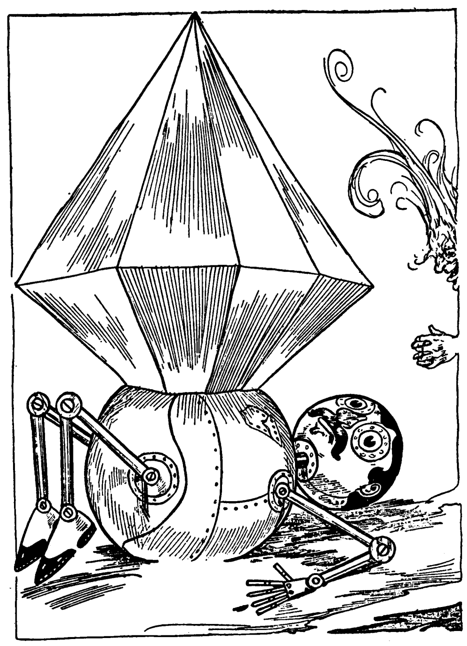
While IBM participated in the creation of certain early computers, it is interesting that Watson dismissed Eckert and Mauchly when they came around after World War II tring to get backing for their ENIAC design, in certain ways the first true electronic computer. Eckert and Mauchly went to Remington Rand, and the resulting Univac was the first commercial computer.

However, IBM bounced back very well. If there was one thing they knew how to do it was sell, and when they brought out their computers it was practically clear sailing. (The Univac I was the first of many computers to be delayed and boggled in the completion of its software, and this considerable setback helped IBM get the lead very quickly; they have never lost it since.)

In the early sixties the IBM 7090 and 7094 were virtually unchallenged as the leading scientific computers of the country. But IBM in the late sixties almost relinquished the fields of very big computers and time-sharing to other companies, and their computers are not regarded as innovative. Nevertheless, IBM's Systems 360 and 370, despite various criticisms, have been very successful; thousands of them are in operation around the globe, far more than all their rivals' big computers all put together. This despite the fact that some of these systems have failed, including the big Model 91 (an economic failure) and the TSS/360 time-sharing program, a technical catastrophe.

They have from time to time been accused of unfair tactics, and various antitrust and other actions (see "Legal Milestones" box) have required IBM to change its arrangements in various ways. One decree required them to sell the computers that before they had only rented; another decision, to "unbundle," or sell computers separately from their programs (previously "given" away with the computers they ran on), is widely believed to have prevented government action on the same matter. Showing characteristic finesse, IBM thereupon lowered the computer prices almost imperceptibly, then slapped heavy price-tags on the programs that had previously been free.

Recent moves by the government have suggested an especially serious and far-reaching anti-trust suit against IBM, possibly one that might break the company up, with its separate divisions going various ways. However, in today's climate of cozy relations between business and government, it is hard to imagine that such matters would not be settled to IBM's liking. This lends a curious tint to a remark one IBM person has made to the author, to wit, that maybe IBM wants to be broken up. That might be one way of reducing the unwieldiness and interdependency of its product line; in addition to reducing its vast, underutilized personnel base. (Another angle: Acting Attorney General Bork has expressed the view that IBM is big only because its products and management are wonderful, so the antitrust case may simply evaporate during the rump days of the Nixon incumbency.)



An interesting aspect of IBM publicity is its stress on status. Publicity photographs often show a subordinate seeking advice from a superior. IBM ads appeal to the corporation president in all of us-- either Going It Alone (taking a long walk over an Executive Decision) or soberly directing a lesser employee. In one extraordinary case, we saw worshipful convicts at the feet of a Teacher implausibly situated in the corner of a prison yard.

The 360 and Standardization

IBM announced a number of worthy objectives when the 360 line was announced in 1964. IBM should certainly be thanked for at least their lip service to these noble goals.

1. 'One machine for all purposes, business and scientific.' (Thus the name "360," for the "full circle" of applications.) By "business" this mainly meant decimal, at four bits a digit. Actually this meant grafting 4-bit decimal hardware to an otherwise normal binary computer, and making both types of users share the same facility.

2. 'Information storage and transmission will be standardized.' The 360 was set up to handle information 4 bits at a time, 8 bits at a time, 16, 32, and 64 bits at a time. (The preceding standard had been 6, 18 and 36 bits at a time.)

In their 360 line, IBM also replaced the industry's standard ASCII code with a strange alphabetical code called EBCDIC ("Extended Binary Coded Decimal Information Code"), ostensibly built up from the 4-bit decimal code (BCD), but believed by cynics to have been created chiefly to make the 360 incompatible with other systems and terminals.

3. '360s will all look alike to the program; thus programs can be moved freely from machine to machine.'

Unfortunately this compatibility has been undermined by numerous factors, especially the variety of operating systems, including half a dozen major types, and the language processors, intricately graded according to computer size. Both these factors tend to make changes necessary to move programs between computers. While one effect of this "standardization" has indeed been to facilitate the moving of programs from small computers to big ones, a more important effect has perhaps been to make it hard to move from a big computer to a smaller one. Note the usefulness of this apparent paradox to IBM's marketing.

The secret of it all, of course, lies in IBM's keen understanding of how to sell big computers. The comptroller, or somebody like him, generally makes the final decision; and if he is told that the one computer will run "all kinds" of programs, that naturally sounds like a saving. Shades of the F-111. (Businessmen's trust and respect for IBM is discussed elsewhere in this article.)

THE BIG QUESTIONS

Between the trade press and dozens of acquaintances in the field, almost everything I hear about IBM and its products is negative (say five or ten to one)-- except from people who work or have relatives there.

Perhaps it's just sour grapes. Or the authority-hating character of research types. Or selective reading.

Or perhaps there really is something sinister.

The major questions are these.

1. How clean is their salesmanship?

2. Are their systems unnecessarily difficult or cumbersome on purpose?

3. How deep is their system of entrapment and forced commitment of the customer? How necessary are the de-standardizations and the constant changes?

4. Do they have a final liberating vision? Do they really, after all, intend to bring about a day when life is easier for people? When the difficulties of present-day computer systems, especially theirs, wither away? I think that history's judgment on IBM in our time may narrow down to that simple question.

(In this light it is not hard to understand IBM's stand on software copyrights vs. patents. IBM is against programs being patentable, which would cover abstracted properties, but argues in favor of copyright, whose protection is probably more limited to the particulars of a given program. If they have their way, it would be assured that IBM could use any ingenious new programming tricks without compensation, whereas all unnecessary complications of bulky, cumbersome software would be covered in entirety by copyright.)

Finally, it has not been demonstrated that IBM has any general ability to make systems conceptually simple and easy to use. (Two good examples of hard systems are the Mag Tape Selectric and Datatext-- easy for programmers, but hardly for secretaries.) There seems to be no emphasis on elegance or conceptual simplicity at IBM. Those who adopt such a philosophy (such as Kenneth Iverson) do so on their own.

As mentioned earlier, this has something to do with the fact that individuals generally use IBM's systems because they have to, being employees or clients of the firms that rent IBM equipment, so there is no impetus to design programs or systems to run on simple or clear-minded principles, or dress out intricate systems so they can be used easily.

4. THE IMAGE.

It is hard to analyze images, corporate or personal. They are often received in such different ways by different populations. But there may be a commonality to the IBM image as generally seen. The Image of IBM involves some kind of cold magic, a brooding sense of sterile efficiency. But other things are percolating in there. If we slide that connotation of efficiency aside, the IBM image seems to have two other principal components: authoritarianism and complacency. It is this mixture that longhairs will naturally find revolting. This same combination, however, may be exactly what it is that appeals to business-management types.

IF YOU REALLY WANT IT...

you can get character-by-character responding systems on IBM computers. The new Stock Exchange system uses a "Telecommunications Access Method" permitting non-IBM terminals to respond character-by-character, just as systems for non-computer-people should.

Trying to use this input-output program on your local IBM computer is another problem, though. Aside from program rental costs, there is the problem of its compatibility with the whole line of IBM software. Adaptations and reprogramming would probably be necessary up and down the line.

THE FUTURE

What will IBM do next?

Speculation is almost futile, but necessary anyhow. The prospects are fascinating if not terrifying.

No one can ever predict what IBM will do; but trying to predict IBM's actions-- IBM-watching is something like Kremlin-watching-- is everybody's hobby in the field. And its consequences affect everybody. With so many things possible, and determined only in the vaguest way by technical considerations, the question of what IBM chooses to do next is pretty scary. Because whatever they do we'll be stuck with. They can design our lives for the foreseeable future.

We know that in the future IBM will announce new machines and systems, price changes (both up and down) in fascinating patterns, rearrangements of what they will "support," and changes in the contracts they offer (see box, "IBM's Control"). Occasional high-publicity speeches by IBM high officers will continue to be watched with great care. But mainly we don't know.

IBM's slick manufacturing capabilities mean that practically any machine they wanted to make, and put on a single chip, they could, and in a very short time. (The grapevine has it that the Components Division, which makes the computer parts, has bragged within the company that it doesn't really need the other divisions any more -- it could just put whole computers on teeny chips if it wanted to.)

In this time of the 370, things are for the moment stable. The 370 computer line is still their main marketing thrust. Having sold a lot of 370 computers (basically sped-up 360s), their idea is at the moment to sell conversion jobs to adapt the 370 to run the new "Virtual System" control program (VS or OS/VS or various other names). This system (which is, incidentally, widely respected) makes core memory effectively much larger to programs that run on it. This effectively encourages programmers to use tons of core, by means of virtual memory; essentially getting people in the habit of programming as if core were infinite. This extension of apparent memory size distracts from any inefficiencies of both locally written programs and IBM programs, thus tending to increase use and rental charges.

When that marketing impetus runs out we'll see the next thing.

The other new IBM initiative is with smaller machines, the System 3 and System 7, being pushed for relatively small businesses. That is where they see another new market. How easy and useful their programs are in this area will be an important question.

With the System 7, a 16-bit minicomputer for $17,000, IBM has at last genuinely entered the minicomputer market. (Balancing its speed and cost against comparable machines, we can figure the IBM markup as being about 50%, which is typical.)

In addition, it is rumored that IBM might put out a tiny business mini, to sell out of OPD. (Datamation, Dec 72, 139.) But really, who knows.

In addition to this huge-memory strategy for its big machines, and the starting foray into specialized mini systems, there is the office strategy and "word processing."

IBM has conceptually consolidated its various magic-typewriter and text services under the name of "word processing," which means any handling of text that goes through their machines. This superficially unites their OPD efforts (typewriters and dictation machines) with things going on in DPD, such as Datatext, and allays interdivisional rivalries for awhile. Also, by stressing the unity of the subject matter, it leaves the door open for later and more glamorous initiatives, such as hypertext systems (see "Carmody's System," flip side).

In other words, the foot is in the door. Mr. Businessman has the idea that automatic typing and things like that are IBM's special province.

Few firms anywhere have the confidence to advertise generically a product which is made by others as well, as in IBM's "Think of the computer as energy" series.

SHOULD INDIVIDUALS FEAR IBM?

Even if it is true, as Anonymous says (see Bibliography) that IBM intimidates people and keeps its enemies from getting jobs at IBM-oriented establishments, that's not the end of the world. Grosch, Gould, Rodgers and McGurk are alive and working. Extramural harassment like that employed by GM against Nader, for example, has not been reported.

END OF THE DINOSAURS?

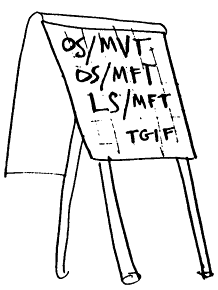
To a very great extent, IBM's computer market is based on big computers run in batch mode, under a very obtrusive operating system.

Many people are beginning to notice, though, that many things are more sensibly done on small computers than on big ones, even in companies that have big computers. That way they can be done right away rather than having to wait in line. Is this the mammal that will eat the dinosaur eggs?

On the other hand, a very unfortunate trend is beginning to appear, an implicit feud within large organizations, which may benefit IBM's big computer approach. Those who advocate minicomputers are being opposed by managers of the big computing installations, who see the minis as threatening their own power and budgets. This may for a long time hold the minis back, perhaps with the help and advice of computer salesmen who feel likewise threatened. But there will be no holding back the minis and their myriad offspring, the microprocessors (see p. 44). And the inroads should begin soon.

(Others are growing to know and love true high-capacity time-sharing as a way of life, like that offered for DEC, GE and Honeywell machines. This, too, may begin to have derogatory effects on IBM's markets.)

Finally, it must be noted that almost all big companies have computers, usually IBM computers, and so an era of marketing may well have ended. It may be possible for IBM to go on selling bigger and bigger computers to the customers who already have them, but obviously this growth can no longer be exponential.



A GROSCH IRONY

Herb Grosch, now editorial director of Computerworld, is perhaps IBM's worst enemy. Once he worked for old man Watson, and was the only IBM employee allowed to have a beard. Now, among other things, he gives speeches and testimony wherever possible about the Menace of IBM, at conferences, at governmental hearings, and in letters to editors.

Yet IBM's main computer sales strategy today is to stress the advantages of big computers with lots of core memory (and persuade you you don't want highly interactive systems or independent minicomputers).

And the fundamental rule stating the advantages of big computers is called Grosch's Law, formulated years ago by none other. See p.

A LITTLE GEM FROM THE IBM SONGBOOK
(Who says IBM doesn't encourage individualism?
To the tune of "Pack Up Your Troubles
in Your Old Kit Bag.")

"TO THOMAS J. WATSON, President, IBM"

Pack up your troubles-- Mr. Watson's here!
And smile, smile, smile.
He is the genius in our IBM
He's the man worth while.
He's inspiring all the time,
And very versatile-- oh!
He is our strong and able President!
His smile's worth while.

"Great organizer and a friend so true,"
Say all we boys.
Ever he thinks of things to say and do
To increase our joys.
He is building every day
In his outstanding style-- so
Pack up your troubles, Mr. Watson's here
And Smile-- Smile-- Smile.

(As a nostalgic public service Advanced Computer Techniques, Inc., of Boston, gave away LPs of IBM songs at the '69 SJCC. They might just have some left...)

NEW CHIPS...

IBM can put pretty much anything on a single chip, to make a functioning machine the size of a postage stamp; but so can a lot of other companies.

The question really becomes whether what goes on that chip is a worthwhile machine that does what people want.

...BUT THE SAME OLD BLOCK?

It is by no means clear that IBM has any general ability to make computer systems easy to use.

This is a psychological problem.

As a corporation they are used to designing systems that people have to use by fiat, and must be trained to use, contributing to the captivity and inertia of the customer base. Thus the notion of making things deeply and conceptually straightforward, without special jargon or training, may not be a concept the company is ready for.

SOME DIVISIONS OF IBM you may hear about

OPD	Office Products Division. Typewriters, copiers.
DPD	Data Processing Division. Computers and accessories.
FSD	Federal Systems Division. Big government contracts: NASA stuff, and who knows what.
ASDD	Advanced Systems Development Division. Very secret.
Components Division.	Makes parts for the other guys, including integrated circuits.
SRA	Science Research Associates, Chicago. Publishes textbooks and learning kits.
Watson Lab	T.J. Watson Research Laboratory, Westchester County, north of New York City. Theoretical and lookahead research.

"THERE IS A WORLD ELSEWHERE."
-- Coriolanus

There is no way to escape IBM entirely. IBM mediates our contacts with government and medicine, with libraries, bookkeeping systems, and bank balances. But these intrusions are still limited, and most of us don't have to live there.

There are many computer people who refuse to have anything to do with IBM systems. Others, not so emphatic, will tell you pointedly that they prefer to stay as far away from IBM computers as possible. If you ask why, they may tell you they don't care to be bothered with restrictive, unwieldy and unnecessary complications (the JCL language is usually mentioned). ¶This is one reason that quite a few people stick with minicomputers, or with firms using large computers of other brands.

It is possible to work productively in the computer field and completely avoid having to work with IBM-style systems. Many people do.

IBM LEGAL MILESTONES

The famous Consent Decree of January 1956. (In a consent decree, an accused party admits no guilt but agrees to behave in certain ways thereafter.) In response to a federal anti-trust suit, IBM agreed to:
- sell as well as lease its computers, and repair those owned by others;
- permit attachments to its leased computers;
- not require certain package deals;
- license various patents;
- not buy up used machines;
- and get out of the business of supplying computer services, i.e., programming and hourly rentals.

Unbundling decision, late sixties. While this was not a government action but a an internal policy decision by the company, it somehow had a public-relations appearance of official compulsion. Beset by pressures from makers of look-alike machines, users of competitive equipment, and the threat of anti-trust action, IBM decided to change its policy and sell programs without computers and computers without programs. Delight amongst the industry turned to chagrin as this became recognized as a price hike.

The Telex Decision, September '73: Telex Corp. of Tulsa was awarded \$352,500,000 in triple damages (since reduced) for losses attributed to IBM's "predatory" pricing and other marketing practices.
Much more important, IBM was required to disclose the detailed electronics required to hook things to their computers and accessories within sixty days of announcing any. This was a great relief for the whole industry. Essentially it meant IBM could no longer dictate what you attach to their machines. Unfortunately, it is not clear whether this will stand.

But what we're waiting to hear about is whether the Nixon Justice Department is, or is not, going to press the big anti-trust suit which has been long brewing, at the persistent request of other firms in the industry.

"THINK OF THE COMPUTER AS ENERGY," says a recent series of IBM ads. But in terms of monopoly, price, and the world's convenience, there would seem only one way to complete the analogy, viz.:

"THINK OF THE COMPUTER AS ENERGY.

"Think of IBM as King Faisal."

FOLDING OF THE IBM UMBRELLA

For a long time, during the sixties, IBM's high prices provided an environment that made it easy for other companies to come into the field and sell computers and peripherals. These high prices were referred to as "the IBM umbrella."

However, this era has ended. IBM now cuts prices in whatever areas it's threatened. A brief flourishing of companies making add-on disk and core memories for IBM computers has become precarious; not only will IBM now cut prices, but they have shown themselves still disposed to invent new restrictive arrangements (the recent "virtual memory" announcement for the 370 claimed that the program will only work on IBM disk and core).

BIBLIOGRAPHY

Harvey D. Shapiro, "I.B.M. and all the dwarfs," New York Times Magazine, July 29, 1973, 10-36.

An objective, factual article, sympathetic to IBM-- although it drew at least one irate letter from an ibmer who didn't think it sympathetic enough.

"IBM: Time to THINK Small?" Newsweek, October 1, 1973, 80-84.

Frank T. Cary, letter to the editor, Newsweek, Oct. 15 73, p. 4. A snappish reply to the above by the IBM Board Chairman, who evidently didn't like the article very much.

Robert Samuelson, "IBM's Methods," New York Times Sunday financial section, June 3, 1973, p. 1.

→This article gives a unique glimpse of some of the interesting things that came to light in the Control Data suit against IBM-- citing trial documents never publicly released.

★ William Rodgers, Think. Stein and Day, 1969. Subtitled A Biography of the Watsons and IBM.

→Concentrates on the days before computers. Fascinating profile of Watson, a business tiger; but the view of the corporation in an evolving nation is general Americana that transcends fiction.

Would you believe Rodgers says Watson was the kingmaker wo put General Ike in the White House?

Unfortunately, the book has relatively less on the computer era, so the inside story of many of their momentous decisions since then remains to be told.

Heywood Gould, Corporation Freak. Tower (paperback.)

Marvelous; hard to get; Gould thinks IBM quietly bought up all the copies.

The musings of a sophisticated, clever and observant cynic who began knowing nothing about IBM, Gould's wide-eyed observation of its corporate style and atmosphere is a jolt to those of us who've gotten used to it. And he thought it was just another big company!

Anonymous, "Anti-Trust: A New Perspective." Datamation, Oct 73, 183-186.

Richard A. McLaughlin, "Monopoly Is Not a Game," Datamation, Sept. 1973, 73-77.

→Questionnaire survey intended to test truth of common accusations against IBM. (Discussed in text above.)

W.David Gardner, "The Government's Four Years and Four Months in Pursuit of IBM." Datamation, June 1973, 114-115.

Almost any issue of Computerworld or Datamation, the two main industry news publications, carries articles mentioning complaints about IBM from various quarters on various issues. Datamation's letters are also sometimes juicy on the topic.

Any issue of On Line, a news sheet of the Computer Industry Association, ten bucks a year. (CIA-- no relation to the intelligence agency -- 16255 Ventura Blvd., Encino, CA 91316.)

T.A. Wise, "I.B.M.'s \$5,000,000,000 Gamble," Fortune, Oct 1966.

Daniel J. Slotnick, "Unconventional Systems." Proc. SJCC 1967, 477-481.
Interesting, among other reasons, for the heaviness of the sarcasm directed at IBM and its larger computers.

★ William Rodgers, "IBM on Trial." Harper's, May 1974, 79-84.
Continues where Think left off; examines some of the dirt that came out in the Telex case, and other things.

The author regrets not being able to list more articles and books favorable to IBM, but these do not seem to turn up so much. However, here are a few.

A Computer Perspective, by the office of Charles and Ray Eames, Harvard U. Press, \$13.

Angeline Pantages, "IBM Abroad." Datamation, December 1972, 54-57.

For an example of the kind of adulation of IBM based on faith, see Henry C. Wallich, "Trust-Busting the U.S.A.," Newsweek 1 Oct 73, p. 90.

The IBM Songbook, any year-- they haven't been issued since the fifties-- is definitely a collectible.

Digital Equipment Corporation, in response to the "Energy Crisis" of 1973, didn't turn out their Christmas tree. Instead they hooked it up to a water wheel they happened to have. Typical.

the Computer Fan's Computer Company DEC The PDPeople

The computer companies are often referred to in the field as "Snow White and the Seven Dwarfs"-- a phrase that stays the same even as the lesser ones (like RCA and General Electric) get out of the business one by one. The phrase suggests that they're all alike. To an extent; but there is one company sufficiently different, and important enough both in its history and its continuing eminence, to require exposition here. This is Digital Equipment Corporation, usually pronounced "Deck," the people who first brought out the minicomputer and continue to make fine stuff for people who know what they are doing.

Other computer companies have mimicked IBM. They have built big computers and tried to sell them to big corporations for their business data processing, or big "scientific" machines and tried to sell them to scientists.

DEC went about it differently, always designing for the people who knew what they were doing, and always going to great lengths to tell you exactly what their equipment did.

First they made circuits for people who wanted to tie digital equipment together. Then, since they had the circuits anyway, they manufactured a computer (the PDP-1). Then more computers, increasing the line slowly, but always telling potential users as much as they could possibly want to know.

The same for its manuals. People who wrote for information from Digital would often get, not a summary sheet referring you to a local sales office, but a complete manual (say, for the PDP-8), including chapters on programming, how to build interfaces to it, and the exact timing and distribution of the main internal pulses. The effect of this was that sophisticated users-- especially in universities and research establishments-- started building their own. Their own interfaces, their own modifications to DEC computers, their own original systems around DEC computers.

This policy has made for slow but steady growth. In effect, Digital built a national customer base among the most sophisticated clients. The kids who as undergraduates and hangers-on built interfaces and kludgey arrangements, now as project heads build big fancy systems around DEC equipment. The places that know computers usually have a variety of DEC equipment around, usually drastically modified.

Because of the great success of its small computers, especially the PDP-8, even many computer people think they only make small computers. In fact their big computer, the PDP-10, is one of the most successful time-sharing computers. An example of its general esteem in the field: it is the host computer of ARPANET, the national computer network among scientific installations funded by the Department of Defense; basically this means ARPANET is a network of PDP-10s.

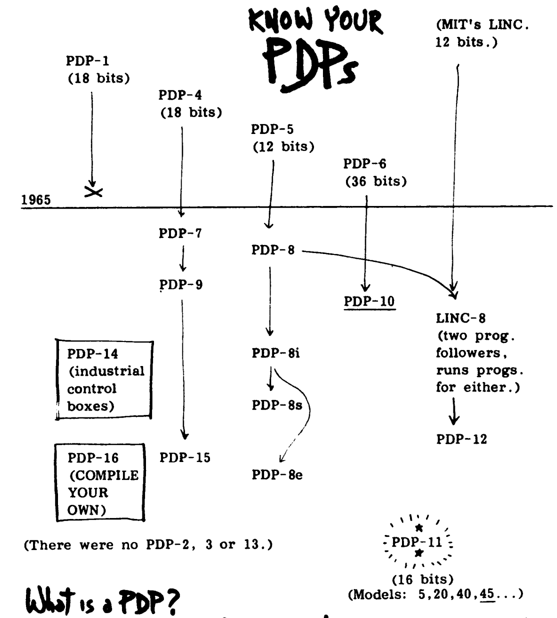
DEC's computers have always been designed by programmers, for programmers. This made for considerable suspense when the PDP-11 did not appear, even though the higher numbers did, and the grapevine had it that the 11 would be a sixteen-bit machine. It proved to be well waiting for (see p. 22), and has since become the standard sophisticated 16-bit machine in the industry.

An area DEC has emphasized from the first has been computer display (discussed at length on the flip side). Thus it is no surprise that their interactive animated computer display, the GT40 (see p. 34) is an outstanding design and success. (And the University of Utah, currently the mother church of computer display, runs its graphic systems from PDP-10s.)

In this plucky, homespun company, where even president Olsen is known by his first name (Ken), it is understandable that marketing pizazz takes a back seat. This apparently was the view of a group of rebels, led by vice president Ed deCastro, who broke off in the late sixties to start a new computer company around a 16-bit computer design called the Nova-- rumored to have been a rejected design for the PDP-11. The company they started, Data General, has not been afraid to use the hard sell, and between their hard sell and sound machine line they've seriously challenged the parent company.

But Digital marches on, the Computer Fan's computer company. If IBM is computerdom's Kodak, whose overpriced but quite reliable goods have various drawbacks, DEC is Nikon, with a mix-and-match assortment of what the hotshots want. That's pluralism for you.

KNOW YOUR PDPs



(There were no PDP-2, 3 or 13.)

What is a PDP? DEC's trade name for a computer.

I'm not getting any favors from DEC, I'm just saying about them what people ought to know.

However, I do have grateful recollections of the warmth and courtesy with which people from Digital Equipment Corporation have taken pains to explain things to me, hour after hour, conference after conference.

In the early sixties they had one man in one small office to service and sell all of New Jersey and New York City. But that one guy, Dave Denniston, spent considerable time responding to my questions and requests over a period of a couple of years, and in the nicest possible way, even though there was no way I could buy anything. You don't forget treatment like that.

PERIPHERALS FOR YOUR MINI

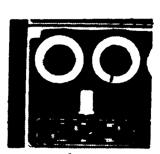
Some kinds of peripheral devices, or computer accessories, are always necessary. Only through peripherals can you look at or hear results of what the computer does, store quantities of information, print stuff out and whatnot.

Trying to print lists of available stuff here is hopeless. There are thousands of peripherals from hundreds of manufacturers. If you buy a mini, figure that your peripherals will cost $1500 (Teletype) on up. But maintenance (see p. 50) is the biggest problem. If you buy peripherals from the manufacturer of the computer, at least you can be sure someone will be willing to maintain the whole thing. (Independent peripheral manufacturers will often repair their own equipment, but nobody wants to be responsible for the interface.)

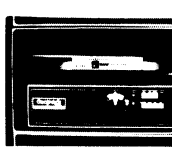
If you want a list see "Table of Mini-peripheral Suppliers," Computer Decisions, Dec 72, 33-5; more thorough poop is offered by Datapro Research Corp., 1 Corporate Center, Route 38, Moorestown NJ 08057.

As to the serious matter of disks, an excellent review article is "Disc Storage for Minicomputer Applications," Computer Design June 1973, 55-66. This reviews both principles of different types of disk drives, and what various manufacturers offer.

Also helpful on disks and tapes: "Making a Go of Ministorage," by Linda Dermer. Computer Decisions, Feb 74, 32-38. Best recent survey.



Scared? It's just a DECtape drive, upside down.



Disk drive for the 11. Most such devices go at 30 spins a second, or 1800 rpm. The heads that read and write information are on moving arms that have to be positioned on the different tracks. (Some disks have a head for every track, which costs more.)

If you have disk drives ($5500 each) you need a controller ($5500). Sigh.

"... a sophisticated electronic computer can store and recall some 100 billion 'bits' of information..."
TIME, 14 Jan 74, 50.

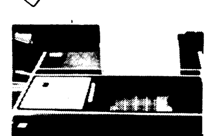
Piffle. That's the overall size of the memory, which is utterly independent of the sophistication or general power of the computer itself.

Trillion-bit memories are available, and you could put one on a machine as small as a PDP-8.

Disk cartridges for this model disk drive.

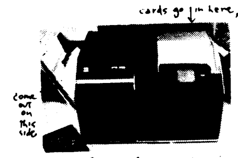
The brown-coated disk itself is hidden in the plastic case. Nevertheless, they sometimes get scratched or break.

A disk costs $75 and holds up to 2,400,000 characters of information (1.2 million PDP-11 words, which are 16 bits each).



A small line printer. Prints some 300 lines a minute (faster if the lines are narrow). Price around $15,000.

(Of course, terminals are peripherals too.)



A card reader. Sends pulses to the computer based on the holes punched in the cards.

TYPICAL PERIPHERALS (for computer shown on p. 36).

MAGNETIC RECORDING MEDIA

Any number of different magnetic devices are used for mass storage of symbolic (digital) information; each has its own medium, or form of storage.

The ones which are removable (called "removable media") are of all sorts.

EFFECTIVELY STANDARDIZED BY IBM

3/4-inch magnetic tape.
  Pre-1965: 6 tracks data, 1 track parity.
  Post-1965: 8 tracks data, 1 track parity.
2741 disk
  Stack of removable platters size of a layer cake.
3330 disk
  Same but bigger cake.
disk cartridge
  Plastic case, size of coolie hat, enclosing disk.
floppy disk
  Flexible, card-thin disk enclosed in square 8" envelope.
data cell (not very common)
  Plastic strips pulled out of wedge-shaped tubes arranged in a rotating cylinder. Strip is pulled out of this carousel, whipped around a drum to make temporary drum memory, returned to case.

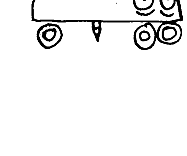
EFFECTIVELY STANDARDIZED BY OTHERS

LINCtape
  3/4-inch tape on a 4-inch reel (fits in pocket), specially coated against friction, developed at Lincoln Labs for LINC computer (see p. 41).
DECtape
  Same size and reel but differently formatted for DEC machines (varies with model). Very reliable. A personal favorite of many programmers.
3M CARTRIDGE
  The Scotch-tape people say the cassette is unreliable, and offer as an alternative a belt-driven quarter-inch baby, costing maybe $1000 without interface.
CRAM (Card Random Access Memory)-- rare
  Big pieces of plastic (about four inches by two feet) pulled by notches out of a cartridge and whipped around a drum. National Cash Register.

HARDLY STANDARDIZED AT ALL

"Cassettes"-- Philips-type audio-type cassette.
  Used by various manufacturers in various ways. Sykes, Sycor, DEC, Data General and others have separate, and usually incompatible, systems.

YOUR TURTLE AND MUSIC BOX

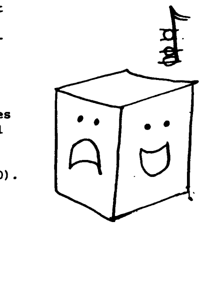


Surely nobody can resist the peripherals offered by General Turtle, Inc., 545 Technology Square, Cambridge, Massachusetts 02139.

The Turtle is a sort of casserole on wheels that takes a pencil down the middle. Attached to your computer, it can be programmed to ramble around drawing pictures, or just do wheelies on the parquetry. $800.

Then the Music Box is $600. It sings in four voices, enough for a lot of Vivaldi, does five octaves and looks to the computer like a Teletype. They will play you samples on the phone (617/661-3773).

For either of these you need a Controller ($1300).



BRAILLE

No joke here. People are still making Braille copies of things by hand. But the way to do it is by computer: the machine can punch out new copies of whatever's stored in it, repeatedly.

A Braille-punching adapter kit is available for the plain 33 Teletype, I believe from Honeywell.

A similar adapter kit for IBM's System 3 is available from IBM.

(It is of interest that an early use of Mooers' TRAC Language was with Braille conversion.)

You never know what you'll see next. In 1969 one firm announced a "high-density read-only memory device" which anyone could see was a plain 45 RPM phonograph-- but with digital electronics. And it made sense. But it doesn't seem to have caught on.

SIMULATION

is an imposing term which means almost anything. Basically, "simulation" means any activity that represents or resembles something. Computer simulation is using the computer to mimic something real, or something that might be, for any purpose: to understand an ongoing process better, or to see how something might come out in the future.

Here again, though, the Science myth steps in to mystify this process, as though the mere use of the computer conferred validity or some kind of truth.

(On TV shows the Space Voyagers stand in front of the "computer" and ask in firm, unnaturally loud voices what will be the results of so-and-so. The computer's oracular reply is infallible. On TV.)

Let there be no mystery about this. Any use of a data structure on a what-if basis is Simulation. You can simulate in detail or crudely; your simulation can embody any theories, sensible or stupid; and your results may or may not correspond to reality.

A "computer prediction" is the outcome of a simulation that someone, evidently, is willing to stand behind. (See "computer election predictions," p. 65.)

These points have to be stressed because if there is one computer activity which is pretentiously presented and stressed, it is simulation. Especially to naive clients. There is nothing wrong with simulation but there is nothing supernatural about it either.

Another term which means more or less the same is modelling.

In the loose sense, simulation or modelling consists of calculations about any describable phoenomena-- for instance, optical equations. In optical modelling (and this is how they design today's great lenses), a data structure is created which represents the curvature, mounting, etc. of the separate glasses in a lens. Then "simulating" the paths of individual rays of light through that lens, the computer program tests that lens design for how well the rays come together, and so on. Then the design is changed and tried again.

Another type of simulation, an important and quite distinct one-- is that which represents the complex interplay of myriad units, finding out the upshots and consequences of intricate premises. In traffic simulations, for instance, it is easy enough to represent thousands of cars in a data structure, and have them "react" like drivers-- creating very convincing traffic jams, again represented somehow within the data structure.

Basically simulation requires two things: a representation, or data structure, that somehow represents the things you're simulating in the aspects that concern you; and then a program does something to these data, that is in some way like the process you're concerned about acting on the things you're modelling. And each event of significance enacted by the program must somehow leave its trace in the data structure.

The line between simulation and other programming is not always clear. Thus the calculation of the future orbits of the planets could be called "simulations."

The most intricate cases, though, don't particularly resemble any other kinds of programs. The intricate enactments of physical movements, especially swarms and myriads with mixed and colliding populations, are especially interesting. (In a recent Scientific American article, simulation helped to understand possible streamers of stars between galaxies as resulting from normal considerations of inertia and gravitation. (Alar and Juri Toomre, "Violent Tides between Galaxies," Sci. Am. Dec 73, 38-48.))

Models of complex and changing rates are another interesting type. Enacting complex things, whose amounts are constantly changing in terms of percentage multipliers of each other, sound easy in principle, but their consequences can be quite surprising. (See "The Club of Rome," p. 68.)

To imagine the kinds of mixed-case myriad models now possible, we could on today's big computers model entire societies, with a separate record describing each idividual out of millions, and specifying his probabilities of action and different preferences according to various theories -- then follow through whole societies' behavior in terms of education, income, marriage, sex, poverty, death, and anything else. Talk about tin soldiers and boats in the bathtub.

Any computer language can be used for some kind of simulation. For simulations involving relatively few entities, but lots of rates or formulas, good old BASIC or FORTRAN is fine. (MAGI's "Synthevision" system, which could be said to "simulate" complex figures in a three-dimensional space, is done in Fortran; see p.DM36.) For simulations involving a lot of separate objects, special cases and discrete events, TRAC Language (see p. 18) is great. If numerous mathematical formulas are involved, and you want to change them around considerably in an experimental sort of way, APL is well suited (see pp. 22).

There are a number of special "simulation" languages, notably SIMSCRIPT and GPSS. These have additional features useful, for instance, in simulating events over time, such as "EVENT" commands which synchronize or draw division-lines in time (the simulated time). Simulation languages generally allow a great variety of data types and operations on them.

The list-processing fanatics, of course, insist that their own languages (such as LISP and SNOBOL) are best. And then there's PLATO (see p.DM26), whose TUTOR language is splendid for both formulas and discrete work-- but allows you only 1500 variables, total (60 bits each).

The thing is, any set of assumptions, no matter how intricate, can be enacted by a computer model. Anything you can express exactly can be carred out, and you can see its consequences in the computer's readout-- a printout, a screen display, or some other view into the resulting data structure.

Obviously these enactments (or sometimes "predictions") are wholly fallible, deriving any validity they may have from the soundness of the initial data or model.

However, they have another important function, one which is going to be very important in education and, I hope, general public understanding, as computers get spread about more widely and become more usable.

The availability of simulation models can make things easier to understand. Well-set-up simulation programs, available easily through terminals, can be used as Staged Explanatory Structures and Theoretical Exploration Tools. The user can build his own wars, his own societies, his own economic conditions, and see what follows from the ways he sets them up. Importantly, different theories can be applied to the same setups, to make more vivid the consequences of one or the other point of view.

(Indeed, similar facilities ought to be available for Congress, to allow them to pour a new tax through the population and see who suffers, who gains...)

I should point out here that for this purpose-- Insightful Simulation-- you don't always need a computer. I have in mind the so-called "simulation games," which if well designed give extraordinary insights to the players. Allen Calhamer's brilliant game of Diplomacy, for instance (Games Research, Boston; available from Brentano's, NYC) teaches more about international politics than you could suppose possible. I am also intrigued by a game called "Simsoc," worked out by a sociologist to demonstrate the development of social structures from a state of random creation, but I haven't played it. (Clark C. Abt, of Abt Associates, Boston, has also done a lot of interesting design here.)

A last point, a very "practical" application. Simulation makes it possible to enact things without trying them out in concrete reality. For instance, in the lens-design systems mentioned earlier, the lenses don't have to be actually built to find out their detailed characteristics. Nor is it necessary to build electronic circuitry, now, to find out whether it will work-- at least that's what the salesmen say. You can simulate any circuit from a terminal, and "measure" what it does at any time or in any part with simulated meters. Similarly, when any computer is designed now, it's simulated before it's built, and programs are run on the simulated computer, as enacted within a real computer, to see if it behaves as intended. (Actually there are some hot-wire types who insist on building things first, but one assumes that the more sensible computer designers do this.)

With automobiles it's harder; but GM, for instance, simulates the handling characteristics of its cars before they're ever built-- so that designers can redistribute weight, change steering characteristics and so on, till the handling characteristics come out the way the Consumers seem to like.

BIBLIOGRAPHY

Simulation magazine is the official journal of Simulation Councils, Inc., the curiously-named society of the Simulators. It costs $18 a year from Simulation Councils, Inc., Box 2228, La Jolla CA 92037.

For all I know you get annual membership free with that. I've always wanted to join but it was always the one thing too many; but their conference programs are sensational. Where else can you hear papers on traffic, biology, military hardware, weather prediction and electronic design without changing your seat?

THAT'S WHAT MAKES HORSE RACING

"Simulation" means almost anything that in any way represents or resembles something. Which is not to say it's a useless or improper term, just a slippery one.

Examples. Here are ways we could "simulate" a horse race:

Show dots moving around an oval track on a completely random basis, and declare the first to complete the circuit The Winner.

Assign odds to individual horses, and then use a randomizer to choose the winner, taking into account those odds. (This is how the PLATO "horserace" game works; see p.DM27.)

Give conditional odds to the different horses, based on possible "weather conditions." Then flip a coin (or the computer equivalent, weighted randomization) to test the "weather conditions," and assign the horse's performance accordingly.

Program an enactment of a horse race, in which the winner is selected on the basis of the interaction of the horoscopes of horse and rider.

Create a data structure representing the three-dimensional hinging of horse's bones, and the interlaced timing of the of the horse's gait. (This has been done at U. of Pennsylvania on a DEC 338.) Then have these stick figures run around a track (or the data structure equivalent).

Using a synthetic-photography system such as MAGI's Synthavision (see p.DM36), create the 3D data structure for the entire surface of a running horse over time; then make several copies of this horse run around a track, and make simulated photographs of it.

And so on.

So don't be snowed by the term "simulation." It means much, little or nothing, depending.

OPERATIONS RESEARCH

is an extension of Simulation in a fairly obvious direction.

If simulation means the Enactment of some event by computer, Operations Research means doing these enactments to try out different strategies, and test the most effective ones.

Operations research really began during World War II with such problems as submarine hunting. Given so-and-so many planes, what pattern should they fly in to make their catching submarines most likely? Building from certain types of known probability, (but in areas where "true" mathematical answers were not easily found), operations researchers could sometimes find the best ("optimal") strategies for many different kinds of activity.

Basically what they do is play the situation out hundreds or thousands of times, enacting it by computer, and using dice-throwing techniques to determine the outcomes of all the unpredictable parts. Then, after all entities have done their thing, the program can report on what strategies turned out to be most effective.

Example. In 1973 the Saturday Review of something-or-other printed a piece on the solution, by OR techniques, of the game of Monopoly. Effectively the game had been played thousands of times, the dice thrown perhaps millions, and the different "players" had employed various different strategies against each other in a varying mix: Always Buy, Buy Light Green, Utilities and Boardwalk, etc.

A complete solution was found, the strategy which tends (over many plays) to work best. I forget what it was.

Using another technique, the game of football was analyzed by Robert E. Machol of Northwestern and Virgil Carter, a football personage. Their idea was to test various maxims of the game, to find out which common rules about beneficial plays were true. What they did was replay fifty-six big-league football games on a play-by-play basis, rate the outcomes, and see which circumstances proved most advantageous on the average. I've mislaid the reprint (Operations Research, a recent year), and being totally ignorant of football can remember none of the findings. Anyhow, that's where to look. (Citation found, below.)

The earlier explanation of Operations Research wasn't quite right. It's any systematic study of what works best. Computers can help.

BIBLIOGRAPHY

Irvin R. Hentzel, "How to Win at Monopoly." Saturday Review of Science, Apr 73, 44-8.

Virgil Carter and Robert E. Machol, "Operations Research on Football." Operations Research, March 1971, 541-544.

GREAT ISSUES

Until now, the obscurity of computers has kept the public from understanding that anything like political issues were involved in their use. But now a lot of things are going to break. For instance--

WHITHER THE FBI?

J. Edgar Hoover's recent death raised a very serious problem. What about all those files he had been keeping? Responsible critics of the FBI, such as Fred J. Cook, have claimed that Hoover's policy basically consisted of chasing lone punks (like Dillinger, Bonnie and Clyde), harassing political dissenters, and keeping vast unnecessary records on innocent citizens-- thus virtually creating the vast network of organized crime in America, which stays off the police blotters. Thus the question of the FBI Succession was an important one.

The question has been answered. In July 1973 Nixon appointed Clarence Kelley, police chief of Kansas City. After the previous goings-on-- for instance, Nixon's seeming to offer the post to Judge Byrne while he was presiding over the Ellsberg trial-- this looked to the press like a staid and uncontroversial resolution. But was it?

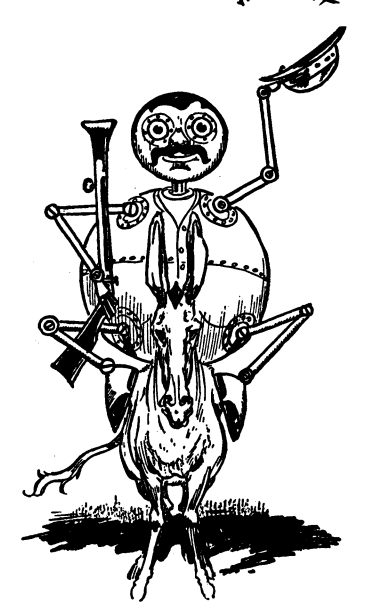
Kelley certainly is aware of technology. It seems to be he that put display screens in Kansas City police cars, created the ALERT system (Automated Law Enforcement Response Team) and COPPS (Computerized Police Planning System), which for your amusement ties into MULES (Missouri Uniform Law Enforcement System). (See Melvin F. Bockelman, "On-Line Computers Keeping Things Straight," which describes the Kansas City computer setup. Communications, June 73, 12-20.) In a more threatening vein, supposedly the Kansas City department kept computer files on "militants, mentals and activists." (Schwartz article, p. 19.)

What Kelley does is thus of interest to us all. The big question is whether, for all his concern with police automation, he is also concerned with the freedoms this country used to be about.

"NECESSITY HAS BEEN THE EXCUSE FOR EVERY INFRINGEMENT OF HUMAN FREEDOM. IT IS THE ARGUMENT OF TYRANTS; IT IS THE CREED OF SLAVES."

EDMUND BURKE

MILITARY USES OF COMPUTERS



A lot of people think computers are in some way cruel and destructive. This comes in part from the image of the computer as "rigid" (see "The Myth of the Computer," p. 9), and partly because the military use so many of them.

But it's not the nature of a computer, any more than the nature of a typewriter is to type poems or death warrants.

The point is that the military people are gung ho on technology, and keen on change, and Congress buys it for them.

No way is there room to cover this subject decently. But we'll mention a few things.

The Pentagon, first of all, with its payroll of millions, with its stupendous inventories of blankets and bombs and toilet paper, was the prime mover behind the development of the Cobol business computing language. So a vast amount is spent just on computers to run the military establishment from a business point of view.

Of course that's not the interesting stuff.

The really interesting stuff in computers all came out of the military. The Department of Defense has a branch called ARPA, or Advanced Research and Development Agency, which finances all kinds of technical developments with vaguely military possibilities.

It is thus a supreme irony that ARPA paid for the development of: COMPUTER DISPLAY (the Sketchpad studies at Lincoln Labs; see p.DM23); TIME-SHARING (e.g. the CTSS system, see p. 45); HALFTONE IMAGE SYNTHESIS (the Utah algorithms: but see all of pp. DM 32-39); and lots more. Some folks might say that proves it's all evil. I say let's look at cases. While they have military applications, that's simply because they have applications in every field, and the military are just where the money is.

Just to enumerate a few more military things--

Command and control-- the problem of keeping track of who's done what to whom, and what's left on both sides, and getting orders through.

It is a solemn irony that the great "465L Command and Control System"-- a grand room with many projectors driven by computer, only something like those in "Dr. Strangelove" and "Fail-Safe"-- may be a prototype for offices and conference rooms of the future.

"Avionics"-- all the electronic gadgets in airplanes, including those for navigation. (A recent magazine piece described how wonderful it felt to fly the F-111-- which has a computer managing the Feel of the Controls for you.)

"Tactical systems"-- computers to manage battlefield problems, aim guns and missiles, scramble your voice among various air frequencies or whatever they do.

"Intelligence"-- computers are used to collate information coming in from various sources. This is no simple problem-- how to find out what is so from a tangle of contradictory information; think about it. Don't think about how we get that information.

"Surveillance"-- it can't all be automatic, but various techniques of pattern recognition (see p.DM12) are no doubt being applied to the immense quantities of satellite pictures that come back. (Did you know our Big Bird satellite either chirps back its pictures by radio, or parachutes them as Droppings?)

Of course, the joker is that all this obsession with gadgets does not seem to have helped us militarily at all. The army seems demoralized, and the navy losing ground to a country that hardly even has computers.

QUIS CUSTODIET, HUH?

Boston welfare recipients have been systematically short-changed for at least 14 years, according to Computerworld (10 Oct 73, p. 2).

A systems analyst recently discovered that the welfare program was not calculating cost-of-living increases on a compound basis, as it should have been, but as a simple increase based each year on an obsolete original figure.

However, it's too late to ask for refunds, and anyway not many welfare recipients take Computerworld.

A PREVIOUSLY UNPUBLISHED STORY

Not all kids who play with computers are quite as law-abiding as the R.E.S.I.S.T.O.R.S. And the temptations are very strong.

One such youngster went on a highschool field-trip to a suburban Philadelphia police station, and saw a demonstration of the police remote information system.

The police who were demonstrating it, not being computer freaks, didn't realize how simple it was to observe the dial-in numbers, passwords and protocol.

When this lad got home, he merrily went to his computer terminal in the basement and proceeded to enter into Philadelphia's list of most-wanted criminals the names of all his teachers.

A few days later a man came to his house from the FBI. He was evidently not a regular operative but a technical type. He asked very nicely if the boy had a terminal. Then the FBI man asked very nicely if he had put in these names. The boy admitted, grinning, that he had. (Everyone in the school knew it had to be he.)

The FBI man asked him very, very nicely not to do it again.

"Of course it didn't do any harm," says the culprit. "I had them down for crimes like 'intellectual murder.' What could happen to them for that?"

Does that make you feel better?

* * * * *

PHILADELPHIANS AND CROOKS PLEASE NOTE:

This happened five or six years ago, and without a doubt the system is by now totally secure and impenetrable. Let's hope.

LOUSED-UP RECORDS: A CASE IN POINT

The question of "privacy" in the abstract isn't really an issue. Who cares if God sees under your clothes? The problem is what happens to you on the basis of people's access to your records.

Margo St. James is a case in point.

Ms. St. James is a celebrated west coast prostitute, once well known for her activities with Paul Krassner as "The Realist Nun;" she is now Chairmadam of an organization called COYOTE, campaigning for the decriminalization of prostitution.

She originally had no intention of becoming a prostitute. Rather, she learned that there was a false record of her arrest for prostitution; and despite her efforts to clear her name, the record followed her wherever she tried to get a job. Finally she said the hell with it and did become a prostitute.

(Membership is $5 a year. COYOTE, Box 26354, San Francisco CA 94126.)

BLACK AND BLUE AND RED ALL OVER

The phone system is bruised and bleeding from the depredations of people who have found out how to cheat the phone company electronically. Such people are called Phone Freaks (or Phreax); articles on them have appeared in such places as Ramparts, The Realist and Oui. For no clear reason, the electronic devices they use have been given various colorful names:

- black box: device which, attached to a local telephone, permits it to receive an incoming call without billing the calling party; it "looks like" the phone is still ringing, as far as the billing mechanism is concerned.

- blue box: device that generates the magical "inside" tones that open up the phone network and stop the billing mechanism. Posession of a blue box can put you in prison.

  As with so many things, the phone system was not designed under the assumption that there would be thousands of electronic wise-guys capable of fooling around with it. Thus the phone system is tragically vulnerable to such messing around. The only thing they can do is get ferocious laws passed and really try to catch people, both of which are apparently happening. Supposedly it is illegal to possess a tone generator, or to inform anyone as to what the magical frequencies are-- even though a slide whistle is such a tone generator, and any engineering library is said to have the information.

- red box: device that simulates the signals made by falling coins.

The fact that the names of these devices are given here is not to be construed as in any sense approving of them, and anybody who messes around with them is a fool, playing with napalm.

Even if people were entitled to steal back excess profits from the phone company-- the so-called "people's discount"-- the trouble is that they mess things up for everyone. We have a beautiful and delicate phone system, one that stands ready to do wonderful things for you, including bring computer service to your home; even if, for the sake of argument, it is run by dirty rats, messing around with it is like poisoning the reservoir for everybody.

"DATA BANKS"

The term "data bank" doesn't have any particular technical meaning. It just refers to any large store of information, especially something attached to a computer.

For instance, at Dartmouth College, where the social scientists have been working hand-in-hand with their big time-sharing project, an awesome amount of data is already available on-line in the social sciences. The last census, for instance, in detailed and undigested form. Suppose you're at Dartmouth and you get into an argument over whether, say, divorced women earn as much on the average as women the same age who have never been married. To solve: you just go to the nearest terminal, bat in a quick program in BASIC, and the system actually re-analyzes the census data to answer your question. If only Congress had this!

The usefulness should be evident.

Because of the way census data is handled, now, it is not possible to ask for the records of a specific individual. But this kind of capability leads to some real dangers.

There is a lot of information stored about most individuals in this country. Credit information, arrest records, medical and psychiatric files, drivers' licenses, military service records, and so on.

Now, it is not hard to find out about an individual. A few phone calls from an official-sounding person can ascertain his credit rating, for instance. But that is very different from putting all these records together in one place.

The potential for mischief lies in danger to individuals. Persons up to no good could carefully investigate someone through the computer and then burglarize or kidnap. Someone unscrupulous could look for rich widows with 30-year-old unmarried daughters. Organized crime could search for patsies and strong-arm victims.

In the face of this sort of possibility, computer people have been worrying for years; noteworthy is the study by Alan Westin that originally sounded the alarm, and his too-reassuring follow-up study of some data-gathering organizations (see bibliography). But the scary data banks, the ones that evidently keep track of political dissenters, aren't talking about what they do (see Schwartz piece).

Basically, the two greatest dangers from data banks are organized crime and the Executive branch of the Federal Government-- assuming there is still a distinction.

Imagine if the Watergate mob had had control over national data banks. Enough said.

It may seem odd, but Nixon has said he is concerned about computers and the privacy problem. Cynics may joke about what his concern actually is; but a more credible stand was taken by vice-president Ford at the 1974 National Computer Conference. Ford expressed personal concern over privacy, particularly considering a proposed system called FEDNET, which would supposedly centralize government records of a broad variety.

Not mentioned by Ford was the matter of NCIC, the National Crime Information Center. This will be a system, run by the FBI, to give police anywhere in the country access to centralized records. THE QUESTION IS WHAT GETS STORED. Arrest records? Anonymous tips? (It would be possible to frame individuals rather nicely if a lot of loose stuff could be slipped into the file.)

Many people seem to be concerned with preserving some "right to privacy," which is certainly a very nice idea, but it isn't in the Constitution; getting such a "right" formalized and agreed upon is going to be no small matter.

But that isn't what bothers me. Considering recent events, and the character of certain elected officials whose devotion to, and conception of, democracy is lately in doubt, things are scarcely as abstract as all that. Considering how helpful our government has been to brutal regimes abroad-- notably the Chile overthrow, which some say was run from here (and which used sports arenas for detention just as John Mitchell did--) we can no longer know what use any information may find in this government. Tomorrow's Data Bank may be next week's Enemies List, next month's Protective Custodial Advisory-- and next year's Termination List. (I don't know if you saw Robert Mardian's eyes on the Watergate hearings, but they chilled my blood.)

Heather M. David, "Computers, Privacy, and Security." Computer Decisions, May 74, 46-48.
Alan F. Westin, Privacy and Freedom, 1967.
Alan F. Westin and Michael A. Baker, Databanks in a Free Society: Computers, Record-Keeping and Privacy. Quadrangle, $12.50.
"Landmark Study of Computer-Privacy Problems Completed." CACM, Dec 72, 1096-7.
Complacent review of Westin & Baker.
Herman Schwartz, review of Westin & Baker book NYTimes Book Review, 8 July 73, 19-20.
Notes that the optimism of Westin and Baker is based on their ignoring various "much-feared information centers" already maintained by the government.
Stanton Wheeler (ed.), On Record: Files and Dossiers in American Life. Russell Sage Foundation (NYC), $10.
"Tax Records: First the Farmers; Then?" Datamation, Dec 73, 105-110.
"How Fair Are Those Fair Credit Guides?" Datamation May 73, 120-124.
Phil Hirsch, "Computer Systems and the Issue of Privacy: How Far Away is 1984?" Datamation, Dec 72, 90-93.

"And the rocket's red glare,
The bombs bursting in air,
Gave proof through the night
That our flag was still there.

"Oh, say, does that star-spangled banner yet wave
O'er the land of the free and the home of the brave?"

-- F.S. Key

THE ABM

THE ABM

Its name has kept changing, possibly to lull the public, possibly to gull the Congress. Anyhow, would you believe a system, totally controlled by computers, designed to shoot down oncoming missiles? If you would, read on.

It's been called Nike-X, Safeguard and goodness knows what. (It's even been called a "thin shield"-- masculine, huh? Perhaps Congress would pay more if they called it the Trojan 4X.) But generally we refer to it as the ABM (Anti-Ballistic Missile). It's the anti-missile missile people have talked about, and in it lie many interesting morals, possible comparisons, etc., for which there is no space here.

Western Electric is the prime contractor. They're the manufacturing arm of the telephone company, remember, the same people who make the Princess™ phone. Of the hundreds of millions of dollars they are taking in on this project, much of it has to go back out-- to Univac, which makes the computers; to Bell Labs, which guides the project, whose Whippany, N.J. facility is totally given over to it; to the rocket-builders and so on.

The system is a turkey.

Note that in telling you this I am drawing only on information that is publicly available, and drawing conclusions from it the way one usually draws conclusions.

Here is how the great ABM is supposed to work.

Immense radars scan over the horizon looking for possible reflections that might be intercontinental missiles.

The radar images are forever constantly analyzed by computers, using every trick of Pattern Recognition (see p. DM 12).

Aha! Something is coming.

Yes, yes, I'm quite sure now, says the computer. We have fifteen minutes.

Great doors swing open, and a long phallic shape arises. It has jagged angular fins, inherited from the smaller anti-aircraft Nike (we say Nikey) rockets that preceded it. This missile is called the Spartan.

It takes off.

The computer system is tracking the oncoming missile. Here it comes-- it's dodging now-- the Spartan is turning, going faster and faster-- they're coming together--

Oncoming missile speed: maybe 15,000 miles an hour. Spartan speed: maybe 10,000, who knows. In these few minutes the Spartan has gone 400 miles.

How's your tennis?

Can you hit a tennis ball fired out of a cannon?

But now comes the good part.

The Spartan goes off. Yay! It too contains an atomic bomb.

If it goes off within five miles of the attacking missile, the hope is that the attacking missile's thermonuclear warhead will get heated on one side and misfire. So it lands in Times Square, just breaks a few buildings and spreads radioactive contamination.

But wait.

What if Spartan missed.

Oops, sorry, Montreal.

Never fear! Have you forgotten Sergeant York? Have you forgotten the Alamo?

There is another missile. It is called Sprint. It is shaped like the point of a pencil. It is almost all propellant. When the great computers realize that the bad guy has gotten through, up goes Sprint! Sprint is eloquently called the "terminal defense system." It only has a couple of minutes.

Brighter than a thousand suns! Sorry, Scarsdale. Can't win 'em all.

If you find this description mind-boggling, that's because it is. Anybody who imagines that this project, on which billions of your dollars have already been spent, can work, is a wishful thinker indeed.

Even if missiles stayed like they were in the good old days of 1962, big helpless clunkers they had to fuel up just before the shoot, the likelihood of the 5-mile ABM detonation they count on was pretty low. (Supposedly ARPA was hoping that Spartan and Sprint could be replaced with ultrapower, fry-in-the-sky laser beams, zapping down all comers with sky-piercing stabs under computer control-- but that is said to have been abandoned.)

But even given, and only for the sake of argument, the feasibility of Spartan-Sprint for fish-in-a-barrel shots, look what's happening now.

MIRVs and FOBs.

MIRV (Multiple Independently Targeted Re-entry Vehicle) basically means Multiple Warheads. One rocket can carry all these little guys, see, that fan out when it gets near the target, and each one goes to its own target city or installation. FOB, or Fractional Orbital Bombardment system, just means that they send the thing into an orbit around the world, and the warheads come in from the opposite side. Any side. Meaning that all those radars pointed at Russia would make good drive-in movie screens.

ABM is sort of a dead duck: the one face-saving installation is in North Dakota, and there won't be any others. But one wonders how such things could ever be funded. But then again I remember once hearing Eric Sevareid, whom some call a liberal, pontificate on this subject. "They describe it as a 'thin shield,'(he said) Why can't we just spend a few billion more and get complete protection?" Otherwise canny people, if fooled by the technologists, will believe anything.

But the ABM is a beautiful example of top-down planning-- like the Vietnamese war. I imagine that the Sprint came about something like this:

> "Garfield, our people in Operations Research have concluded that Spartan won't work."
>
> "Mmm, yes, sir."
>
> "Garfield, I want your team to get on it and find something additional that will make it work."

Now goes Garfield to his cubicle and calls meetings, and it becomes clear: "Lessee now, I can't just say it'll never work, they want something additional, well, I guess it would have to be..." Same as Vietnam. "Gee whiz, they say to search and destroy, I guess that must mean..." Something new, this: the top-down project of the worst sort, where the orders go down, and only news of partial success goes up, rather than the facts of total hopelessness. As in Vietnam.

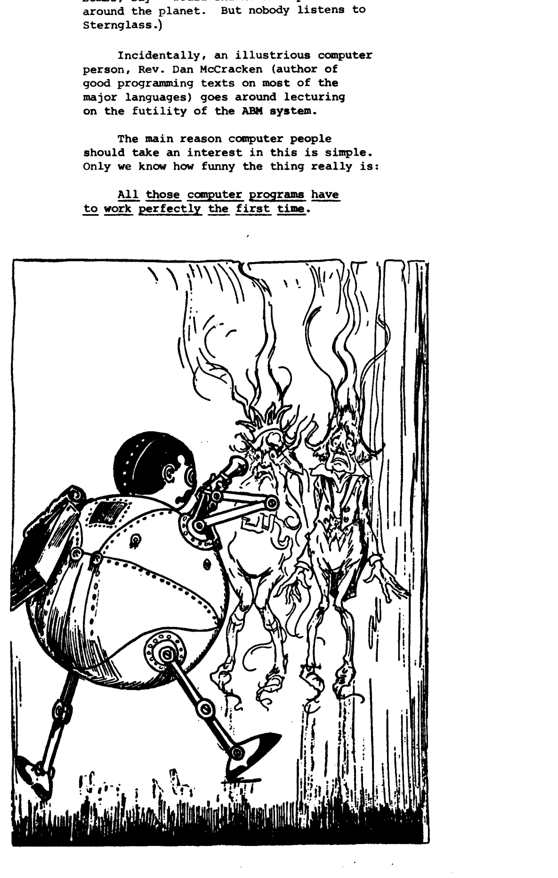
The sophisticated argument is that the ABM effort lets our nation "keep its hand in," "sharpen skills," in case something vaguely like this is ever really needed-- and possible. But this overlooks the overall strategic problem. All this foolishness leads away from the stability of the deterrent; and that may be what keeps everybody alive.

(An interesting point to note: a biologist and population geneticist named Sternglass claims it doesn't matter: that human reproduction is so susceptible to radiation poisoning that just the fallout from the ABM defense itself-- a few dozen bombs, say-- would end human reproduction around the planet. But nobody listens to Sternglass.)

Incidentally, an illustrious computer person, Rev. Dan McCracken (author of good programming texts on most of the major languages) goes around lecturing on the futility of the ABM system.

The main reason computer people should take an interest in this is simple. Only we know how funny the thing really is:

All those computer programs have to work perfectly the first time.



THE MITIEST COMPUTER?

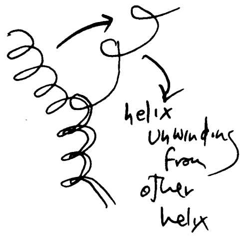
The focus of attention in genetics and organic chemistry has for a decade now been the remarkable systems and structures of the molecules of life, DNA and RNA.

DNA is the basic molecule of life, a long and tiny strand of encoded information. Actually it is a digital memory, a stored representation of codes necessary to sustain, reproduce, and even duplicate the creature around it.

It is literally and exactly a digital memory. Its symbols are not binary but quaternary, as each position contains one of four code molecules; however, as it takes three molecules in a row to make up one individual codon, or functioning symbol, the actual number of possible symbols is 64-- the number of possible combinations of four different symbols in a row of three. (I don't know the adjective for sixtyfourishness, and it's just as well.)

The basic mechanism of the system was worked out by Francis Crick and James Watson, who understandably got the Nobel Prize for it. The problem was this: how could living cells transmit their overall plans to the cells they split into? -- and how could these plans be carried out by a mechanical process?

The mechanism is astonishingly elegant. Basically there is one long molecule, the DNA molecule, which is really a long tape recording of all the information required to perpetuate the organism and reproduce it. This is a long helix (or corkscrew), as Linus Pauling had guessed years before. The chemical processes permit the helix to be duplicated, to become two stitched-together corkscrews, and then for them to come apart, unwinding to go their separate ways to daughter cells.



As a tape recording, the molecule directs the creation of chemicals and other cells by an intricate series of processes, not well understood. Basically, though, the information on the basic DNA tape is transferred to a new tape, an active copy called "messenger RNA," which becomes an actual playback device for the creation of new molecules according to the plan stored on the original.

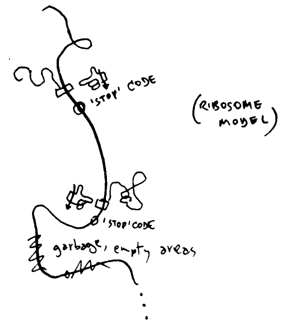
Some things are known about this process and some aren't, and I may have this wrong, but basically the DNA-- and its converted copy, the RNA-- contain plans for making all the basic protein molecules of the body, and anything else that can be made with amino acids. (Those molecules of the body which are not proteins or built of amino acids are later made in chemical processes brought about by these kinds.)

Now well may you ask how this long tape recording makes chemical molecules. The answer, so far as is known, is extremely puzzling.

As already mentioned, the basic code molecules (or nitrogenous bases) are arranged in groups of three. When the RNA is turned on, these triples latch onto the molecules of amino acid that happen to be floating by in the soupy interior of the cell. (There are twenty-seven amino acids, and sixty-four possible combinations of three bases; this is fine, because several different codons of three bases can glom onto the same passing amino acid.)

Now, the tape recording is divided into separate sections or templates; and each template does its own thing. When a template is filled, the string of amino acids in that section separate, and the long chain that results is a particular molecule of significance in some aspect of the critter's life processes-- often a grand long thing that folds up in a certain way, exposing only certain active surfaces to the ongoing chemistry of the cell.

One theory about the mechanics of this is that a sort of zipper slide, called the ribosome, chugs down the tape, attaching the called-for amino acids and peeling off the ever-longer result.



Now, here are some of the funny things that are known about this. One is that there is a particular codon of three bases that is a stop code, just like a period in ordinary punctuation. This signals the end of a template. Another is that the templates on the tape are in no particular order, but distributed higgledy-piggledy. (Geneticists engaged in mapping the genes of a particular species of creature find that the gene for eye color may turn out to be right next to the gene for length of tail-- but where those are really, and what the particular molecules do that determine it, are still mysterious sorts of question.)

Here is some more weird stuff about this.

Large sections of the DNA strand are "dark," it turns out, just meaningless stretches of random combinations of bases that don't mean anything-- or ever get used. This ties in, of course, with the notion that genetic change is random and blind: the general supposition is that genetic mutation takes place a base or two at a time, and then something else activates a chance combination in a dry stretch that turns out to be useful, and this is somehow perfected through successive 1-base changes during the process of successive mutation and evolution.

Amazing use is made of these mechanisms by some viruses. Now, viruses are often thought of as the most basic form of life, but actually they are usually dependent on some other form and hence more streamlined than elemental. Well, some viruses (but not all) have the capacity for inserting themselves in the genetic material: breezing up to the DNA or RNA, unhooking it in a certain place and lying down there, then being duplicated as part of the template, then unhooking themselves and toddling away-- both parent virus and copy. I can't for the life of me think of an analogy to this, but I keep visualizing it as happening somehow in a Bugs Bunny cartoon.

CONTROL MECHANISMS

Now, all cells are not alike. From the first beginning cell of the organism (the zygote), various splits create more and more specialized, differentiated cells. A liver cell is extremely different from a brain cell, but they both date back by successive splitting from that first zygote. Yet they have different structures and manufacture different chemicals.

One simplification may be possible: the "structure" of a cell may really be its chemical composition, since cell walls and other structures are thought to be special knittings of certain tricky molecules. Okay, so that may reduce the question slightly. How then does the cell change from being an Original (undifferentiated, zygotic) cell to the Specialized cells that manufacture particular other complex chemicals?

One hypothesis was that these other cells have different plans in them, different tapes. But this theory was discarded when John Gurdon at Oxford produced a fresh frog zygote from the intestinal cell of a frog (which accordingly, in due time, became a frog de facto). This proved, most think, that the whole tape is in every cell.

Thus there must be something-or-other that blocks the different templates at different times (You there, now you're a full-fledged epithelial cell, never mind what you did before) and selects among all the subprograms on the tape.

Much pressing research in molecular biology, then, is concerned with searching for whatever it is that switches different things on and off at different times in the careers of the ever-splitting cells of our bodies. Not to mention those of all other living creatures, including turnips.

COMPUTERISH CONJECTURES

The guys who specialize in this are usually chemists, and presumably know what they're doing, so the following remarks are not intended as butting into chemistry. However, new perspectives often give fresh insight; and the matters we've covered so far might seem to have a certain relevance.

DNA and RNA, as already remarked, may without distortion be thought of as a tape. Indeed, on this tape is a data structure, and indeed it is a data structure which seems to be involved with the execution of a program-- the program that occurs as the organism's cells differentiate.

There is evidently some sort of program follower which is capable of branching to different selections of (or subprograms) in the overall program, depending on various factors in the cell's environment-- or perhaps its age.

Now, it is one thing to look for the particular chemical mechanisms that handle this. That's fine. On the other hand, we can also consider (from the top down) what sort of a program follower it must be to behave like this. (This is like the difference between tracing out particular circuitry and trying to figure out the structure of a program from how it behaves.)

At any rate, the following interesting conjectures arise:

1. The mechanism of somatic reproduction is a subroutining program follower-- not unlike the second program follower of the subroutining display (see p. That is, it steps very slowly through a master program somewhere, and with each new step directs the blocking or unblocking of particular stretches of the tape.

As the program is in each cell, presumably it is being separately followed in each cell. (This is sometimes called distributed computing.)

2. In each cell, the master program is directing certain tests, whose results may or may not command program branching-- successive steps to new states of the overall program. It may be testing for particular chemical secretions in its environment; it could even be testing a counter.

3. (This is the steep one.) If this were so, we might suppose that this program too was stored on the DNA, in one or more program areas; and it would therefore be necessary to postulate some addressing mechanism by which the program follower can find the templates to open and close. (And perhaps further sections of the program.)

4. Indeed, it makes sense to suppose that such a program has the form of a dispatch table -- a list of addresses in the tape, perhaps associated with specifications of the tests which are to cause the branching.

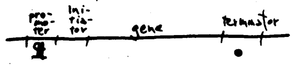
These wild speculations are offered in the spirit of interdisciplinary good fellowship and good clean fun. Whether (1) and (2) have any actual content, or are merely paraphrases of what is already known or disproven, I don't know; somebody may find the rest suggestive.

Two more observations, though. These are not particularly deep, and may indeed be obvious, but they suggest an approach.

5. There is definitely a Program Restart: to wit, whatever it is that turns an old differentiated intestine cell into a fresh zygote.

6. Cancer is a runaway subroutine.

The above remarks seem to be obsolete. The genetic mechanism really seems to be a list processor (see p. 26 col.2), using associative, rather than numerical addressing. The gene is now thought to be divided into four segments, called Promoter, Initiator, gene proper, and Terminator. As I understand it, the promoter and terminator zones contain codes which mean, simply, Start and Stop. The initiator zone, however, is a coded segment which effectively labels the gene. This initiator area contains a chemical code unique for every gene. As suggested in the above article, we may consider both its logical structure-- its mechanisms and effects, considered from a computerman's point of view-- and its chemical structure, or what is really happening. The genes are turned off by grabbing molecules, or repressors, which glom onto the initiator (→) sections of the genes which they have been specifically coded to repress. Research in this area must now find the specific coding of molecules which block and unblock specific genes, and how these fit in the overall graphs of metabolism, immunology, development, and so on. If there is anything to make an old atheist uneasy, it is the extraordinary beauty of this clockwork.



BIBLIOGRAPHY (for note on left)

Har Gobind Khorana, Willard Gibbs lecture, May 1974, "Progress in the Total Synthesis of the Tyrosine tRNA Gene and Its Control Elements."

(→)

From all this, one last speculation creeps forward.

Ivan Sutherland, in considering the structure of subroutining display processors, has noted that as you get more and more sophisticated in the design of a display program follower, you come full circle and make it a full-fledged computer, with branch, test, and arithmetic operations.

If the somatic mechanism should turn out to have a program follower as described, it is not much of a step to suppose that it might have the traits of an actual computer, i.e., the ability to follow programs, branch, and perform manipulations on data bearing on those operations.

In other words, the digital computer may actually have been invented long before von Neumann, and we may have billions of them on our persons already.

It may sound far-fetched, but the mechanisms elucidated at this level are so far-fetched already that this hardly seems ridiculous.

THE COMPUTER FRONTIER

Regardless of what's actually in the cell, it is clear that being able to adapt molecular chemistry, especially DNA and RNA, to computer storage is a beckoning computer frontier.

This would make possible computer memories which are far larger and cheaper than any we now have.

Basically we can separate this into two aspects:

The DNA Readout. This part of the system would create long molecules holding digital information.

The DNA Readin. This would convert it back to electrical form again.

Weird possibilities follow. One is that (if chemical memory is generic, rather than idiosyncratic to an individual's neural pathways) knowledge could be set up somehow in "learned" DNA form, whatever that might turn out to be, and injected or implanted rather than taught. Weird.

As our ability to create clones improves, we could clone new creatures, or genetic "improvements"-- which, considering the racehorse and the Pekinese, means "those sorts of non-viable modifications supported in human society." And of course that ghastly stuff about building humans, or semi-humans; having traits that somebody or some organization, ulp, thinks is desirable...

But the real zinger is this one. It might just be a small accidental printout meant to test the facility, or maybe just a program bug--

-- but the system could output a virus that would destroy mankind.

BIBLIOGRAPHY

James D. Watson, Molecular Biology of the Gene. Beautifully written; meant for highschool science teachers. But potentially formidable; if so, start with his autobiographical The Double Helix, which is a gas.

Mark Ptashne and Walter Gilbert, "Genetic Repressors." Scientific American, June 1970, 36-44.

S.E. Luria, Life: The Unfinished Experiment. Scribner's.

Lewis Thomas, The Lives of a Cell. Viking, $7. Eloquent writing to popularize, among other things, the New Genetic view that your modern animal cells, and mine, actually contain various fungi and other stray ding-a-lings that slid into one of our ancestors and found useful work, joining the basic genetic program.

BRAINS & COMPUTERS

It used to be fashionable to say, "The brain is a computer."

But now people say, "The brain is a hologram."

Fashions change.

THE BRAIN

Almost nothing is known about the brain. Oh, there are lots of picture-books showing cross-sections of brains... Maybe you thought it was just a big cauliflower, but it's full of strings and straps and lumps and hardly anything is known about any of it.

Clinical evidence, of course, tells us that if this or that part is cut out, the patient can't talk, or walk, or smell, or whatever. But that doesn't come close to telling us how the thing works when it does work. The histologists, the perceptual psychologists, the anatomists, are all working at it-- with no convergence. Beautiful example: the split-brain stuff, which I just better not even bring up here (see new Maya Pines book, Harcourt Brace).

We used to dissect brains when I worked down in Dr. Lilly's dolphin lab. Dolphin brains are about 1.2 times the size of ours, and Lilly quite reasonably pointed out that this might mean dolphins were smarter than us.

And, of course, the bigger whales even smarter. We had a killer-whale brain in the deepfreeze that was about 2½ feet across. And whales come much bigger than that; the Killer's maybe a quarter the length of the Blue.

(I should point out here that Lilly's publicity on the intelligence of dolphins was a little too good: it somehow didn't get mentioned that dolphins are just very small whales, the only ones you can feasibly keep in a lab. So think of whales as the possible super-smarties, not just dolphins.)

What's that you say? That "brain size isn't what counts"? That's an interesting point.

People with small heads are by and large just as smart as people with big heads. That's one argument.

However, people have much bigger brains than almost any other animals. That indicates something too.

I believe that the only other animals with very big brains are elephants and whales. (An anatomical explanation: the weight is supported on the man by balancing it, on the elephant by a heavy and comparatively inflexible neck offset by a grappling tool, and in the whale by putting it in the front of a torpedo. But most other anatomies couldn't manage a big brain, so they can't evolve one.)

Anyhow, so the scientific question is whether big-brained species are smart. Well, dogs are smarter than rats...

But about these other guys in our league and beyond. How do we know scientifically that "the size of the brain isn't what counts"? Because obviously they're not as smart as we are, people say. Therefore it isn't brain size that counts. The depth of this logic should be evident. (I've even heard people say, "Of course they're not as smart. They don't have guns.")

Pay close attention to an elephant sometime.

Working elephants in India respond to some 500 different oral commands.

Can you think of a 501st thing to ask an elephant to do? (I rather suppose it could oblige.)

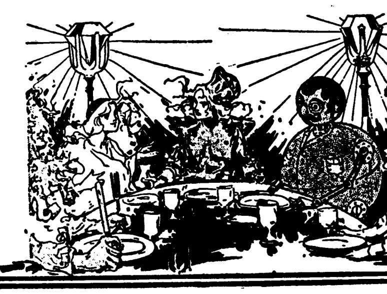
Anyway, the dozen whales I've known personally were smart as hell.

It used to be believed that memory was exclusively a matter of synaptic connections-- the gradual closing of little switches between nerve cells with practice.

It is now known that temporary or short-term memory is synaptic, but something else takes place after that. It's believed that after a certain period, and it has something to do with rest and sleep, memories are transferred to some other form, presumably chemical. But how?

My friend Andrew J. Singer has a beautiful hypothesis that wraps it up. His guess is that memories are moved from synaptic storage to DNA (!) storage during dreaming, or more specifically REM sleep. I like that one.

WHAT NEXT?



By browsing this book you may have more sense of what computers are doing, can do, should do.

What will you do now?

By reading this book in some detail, especially that difficult machine-language stuff (see "Rock Bottom" and "Bucky's Wristwatch," pp. 32-3), or the pieces on specific computer languages (pp. 16-25, 31), you really should be mentally prepared to get into programming, if you dig it.

Maybe you should consider buying your own minicomputer, for a couple of thousand. Or (if you're a parent), chipping in with several families to get one. Or a terminal, and buying (or cadging as cadge can) time on a time-sharing system. Maybe you should start a computer club, which makes it easier to get cast-off equipment; if you're kids, write the R.E.S.I.S.T.O.R.S. (p. 47). If you have a chance, maybe you should take computer courses, but remember the slant these are likely to have. Or perhaps you prefer just to sit and wait, and be prepared to speak up sharply if the computer people arrive ready to push you around. Remember:

COMPUTER POWER TO THE PEOPLE!
DOWN WITH CYBERCRUD!

Computers could do all kinds of things for individuals, if only the programs were available. For instance: help you calculate your tax interactively till it comes out best; help the harried credit-card holder with bill-paying by allowing him to try out different payments to different creditors till he settles on the month's best mix, then typing the checks; WRITING ANGRY LETTERS BACK to those companies that write you nasty letters by computer; helping with letter-writing in general. You'll have to write the programs.

How do you think computers can help the world?
What are you waiting for?

THE COPPER MAN WALKED OUT OF THE ROCKY CAVERN

DAMN THAT COMPUTER!

Everybody blames the computer.

People are encouraged to blame the computer. The employees of a firm, by telling outside people that it's the computer's fault, are encouraging public apathy through private deceit. The pretense is that this thing, the computer, is rigid and inhuman (see "The Myth of the Computer," p. 9) and makes all kinds of stupid mistakes.

Computers rarely make mistakes. If the computing hardware makes a hardware error in a billion operations, it may be noticed and a repairman called. (Of course, once in a billion operations is once in a thousand seconds, or perhaps every ten minutes. That ought to be mentioned.) Anyhow, innocent gadgetry is not what forces you to make stupid multiple choices on bureaucratic forms; mere equipment isn't what loses your subscription records;

IT'S
THE
SYSTEM.

By system we mean the whole setup: the computer, the accessories that have been chosen for it, its plan of operation or program, and the way files are kept and complaints handled.

Don't blame the computer.

Blame the system; blame the programmer; blame the procedures; best of all, blame the company. Let them know you will take your business to wherever they have human beings. Same for governmental agencies: write your congressman. And so on.

A Basic Rejoinder

we should all practice and have ready at the tip of our tongues:

WHY THE HELL NOT? YOU'RE THE ONES WITH THE COMPUTERS, NOT ME!

Let's froth up a little citizen indignation here.

ACCOUNT NUMBERS

In principle we no longer need account numbers.

Now that text processing facilities are available in most (if not all) major computer languages, the only excuse for not using these features is the programmer's notion of his own convenience-- not that of the outside customer or victim.

Example. Someone I know got brand new [redacted] and [redacted] credit cards. He made no note of their numbers. Then he lost them both. Duly he reported the losses. Neither service could look him up, they said, without the numbers. Not having used them, he had no bills to check. Even though he was the only person at that address with anything like that name. And why not, pray tell? Either because they were fibbing, or because they had not seen fit to create a simple straightforward program for the purpose. (See Basic Rejoinder, nearby.)

I have heard of similar cases involving major life insurance companies. Don't lose the numbers. Let's all dance to it:

When anything is issued to you,
Write the number down.

"COMPUTERS" THAT DON'T ANSWER

Few of us can help feeling outrage at the book clubs, or subscription offices, or billing departments, that don't reply to our letters. Or reply inappropriately, with a form printout that doesn't match the problem.

First let's understand how this happens.

These outfits are based on using the computer to handle all correspondence and transactions. The "office" may not have any people in it at all-- that is, people whose job it is to understand and deal sensibly with the problems of customers. Instead, there may just be keypunch operators staffing a Batch System, set up by someone who has long since moved on.

The point of a batch system (see p. 95) is to save money and bother by handling everything in a controlled flow. This does not mean in principle that things have to be rigid and restrictive, but it usually means it in practice. (See "The Punch Card Mentality," p. 29 .) The system is set up with only a fixed number of event types, and so only those events are recognized as occurring. Most important, your problem is assumed to be one that will be straightened out in the course of the system's flow. While there may be provision for exceptions-- one clerk, perhaps-- your problem has not seemed to him worthy of making an exception for.

Here is my solution. It has worked several times, particularly on book clubs that ignored typed letters and kept billing me incorrectly.

Get a roll of white shelf paper, two or three feet wide and twenty or more feet long.

Write a letter on the shelf paper in magic marker. Make it big, perhaps six inches to a word. Legibility is necessary, but don't make it too easy to read.

Explain the problem clearly.

Now take your punch card-- you did get one, didn't you, a bill or something?-- and mutilate it carefully. Tear it in quarters, or cut it into lace, or something. But make sure the serial number is still legible. Staple it lovingly to your nice big letter.

Now fold your letter, and find an envelope big enough for it to fit in, and send it, registered or certified mail, to ANY HUMAN BEING, ACCOUNTING DEPARTMENT, or whatever, and the company's address.

This really works quite well.

I am assuming here, now, that your problem has merit, and you have been denied the attention required to settle it. If we want justice we must ourselves be just.

There is one further step, but, again, to be used only in proportion to the offense. This step is to be used only if a meritorious communication, like that already described, has not been properly responded to in a decent interval.

We assume that this unjust firm has sent you a reply envelope or card on which they must pay postage. Now carefully drafting a follow-up letter, explain once again, in civil language, the original problem, your efforts at attention, and so on. Now put it in a package with a ten or twelve-pound rock, affix the reply envelope to the outside, and send it off.

The problem, you see, has been to get out of the batch stream and be treated as an exception. Flagrantly destroying the punch card serves to remove you from the flow in that fashion. (However, just tearing it a little bit probably won't: a card that is intact but torn can simply be put in a certain slot of the card-punch and duplicated. Destroy it good and plenty.)

In all these cases remember: the problem is not that you are "being treated as a number," whatever that means, but that your case does not correctly fall in the categories that have been set up for it. By forcing attention to your case as an exception, you are making them realize that more categories are needed, or more people to handle exceptions. If more people do this when they have a just complaint, service will improve rapidly.

JUNK MAIL

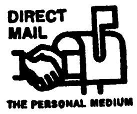
The people who send it out like to call it personalized advertising and the like. But most of us call it Junk Mail. And its vagaries are NOT THE POOR COMPUTER'S FAULT. What gets people angry derives from the system built around the poor computer.

You may wonder why you get more and more seed catalogs, or gift-house catalogs, as time goes on, even though you never order anything from them. Or why a deceased member of the household goes on getting mail year in and year out, regardless of your angry postcards.

How does it keep coming?

Through the magic of something called the Mailing List.

And especially the peculiar way that mailing lists are bought and sold.



Now, a mailing list is a series of names and addresses of possible customers, stored on computer tape or disk.

You can buy the use of a mailing list.

But you cannot buy the mailing list itself.

Suppose you have a brochure advertising pumpkin-seed relish, which you suggest has rejuvenating powers. You want this brochure to go out to rich college graduates.

You go to a mailing-list house.

"I cannot sell you this mailing list outright," says the jolly proprietor, "for it is my business to sell its use again and again, so I do not want anybody else to have a copy of it." So you leave 2500 pumpkin-seed relish brochures with the mailing list company, and pay them a lot of money. And they swear on a stack of bibles that they have mailed the brochures to their special list of rich college graduates.

Well, let's say you get 250 sales from that mailing. (10% is fantastically good.) But out of curiosity you go to another mailing-list house and have another mailing sent out-- this one to people who have low incomes and little education.

This time you get 15% orders.

Now guess what you are acquiring.

A mailing list of your very own. Of people who eat pumpkin-seed relish.

Mailing lists are, you see, generally rented blind, with no chance to see the addressees or check as to whether they've already been mailed to.

And that explains all the duplications.

If an advertiser is going after a certain type of customer, and goes to several mailing-list houses asking for mailings to that particular type of customer, chances are some people will be on several of the lists. And since there's no way to intercompare the lists, these poor guys get several copies of the mailing.

(Another way this can happen is if some cheapskate has his own mailing list and doesn't check it for repeats of the same name. But writing the computer program to check for repeats of the same name is not easy-- there might just be a Robert Jones and a Rob Jones at the same address-- and these things are not usually checked manually. They're big.)

Another possibility exists for eliminating duplications when you rent mailing lists. You can bring in a magnetic tape with your mailing list on it, and they can send out the mailing only to the members of their list who are not already on your list. That way you still can't steal their list, since the tape is on their premises. The trouble is, they can steal your list, by making a copy of the tape. Oh dear.

One possibility, nice and expensive, is to rent a number of mailing lists from a single mailing-list house, with them guaranteeing that they'll compare all the lists you choose and not send to any person more than once.

But as you may be suspecting, this costs money. All this screening and intercomparing requires computer time, and so, even though you are getting a more and more perfect mailing, you are paying more and more and more money for it. So you can see why reasonable businessmen are willing to send out ads even when they know some recipients will get several duplicates.

Another interesting point. There are mailing lists for all kinds of different possible customers. The possibilities are endless. Minority-group doctors. People interested in both stamp collecting and flowers (you'd have to get a company with both lists, and have them go through them for the duplicates... you get the idea).

Note that mailing lists are priced according to their desirability. Weeded mailing lists, featuring only Live Ones, people who've ordered big in recent times, are more expensive. Lists of doctors, who buy a lot, are more expensive than lists of social workers. And so on.

Then there's the matter of the pitch.

The ad's phrasing may be built around the mailing plan. Some circulars come right out and tell the recipient he's going to get several copies because he's such a wonderful person.

THEN there are those advertisements that are actually printed by the computer, or at least certain lines are filled in with the recipient's name and possibly some snazzy phrases to make him think it's a personal letter. Who responds to such things I don't know. My favorite was the one-- I wish I could find it to include here -- that went something like

> You'll really look swell, Mr.   Nelson
> walking down Main Street of   New York
> in your sharp-looking new slacks...

I don't know whether I enjoyed the spaces or the Main Street more.

But you see how this works. There's this batch-processing program, see, and the names and addresses are on one long tape, and the tape goes through, and the program takes one record (a name and address), and decides whether to call the addressee "Mr.," "Ms." or whatever, and then plugs his name into the printout lines that give it That Personal Touch; and then the mailing envelope or sticker is printed; and the tape moves on to the next record.

We may look forward to increasing encroachments on our time and trust by the direct mail industry: especially in better and better quack letters that look as though they've really been personally typed to you by a real human being. (It is apparently legal for letters to be signed by a fictitious person within a company.) In the future we may expect such letters to be sent on fine paper, typed individually on good typewriters, and convincingly phrased to make us think a real personal pitch is being tendered.

There is, however, a final solution.

YOU CAN GET OFF ALL MAILING LISTS -- that is, the ones "participating" in the Association-- by writing to

> **Direct Mail Advertising Association**
> **Public Relations Department**
> **230 Park Avenue**
> **New York, NY 10017**

They will send a blank. If you fill it in they'll process it and delete your name from mailing lists of all participating companies.

Presumably this won't help with X-rated or stamp-collecting lists, but it ought to keep you from getting semiannual gift catalogs from places like The House of Go-Go Creative, Inc. and those million solicitations from Consumer Reports and that File Box company.

[redacted] BANK

Metropolitan Division
P.O. Box 5144 Church Street Station, New York, N.Y. 10049

[redacted]
Senior Vice President

Theodor H Nelson
458 W 20Th St
New York, Ny 10011

Branch 014

FILL-INS

Great news for the Nelson family!

Wouldn't you like your money to work for you full time... even when you're asleep?

Now the Nelson family can save...right at their own bank. [redacted]

... 5% Passbook Savings Plan which
... quarterly or even

NO-RISK ...
If not delighted with ... package, I may return it with... 10 days and my membership will be canceled; I will owe nothing.
DO NOT RETURN THIS FORM IF YOU ARE ALREADY A MEMBER

(whoever you are)

SEPARATE ON THIS LINE AND MAIL ORDER FORM IN POSTPAID ENVELOPE TODAY!

21

Yes, Mr Ainsworth, as a former member of [redacted]'s Literary Guild you qualify for this exciting offer 5 books for 10¢ sent to you as soon as you enroll in the BARGAIN BOOK CLUB. And instead of taking a book a month for 6 months as regular members do, you'll have 12 months -- TWICE THE TIME -- to find the 6 books you want.

FILL-INS

We can make this offer to you because as a member of Doubleday's book club family for 20 months, you proved an interest in good reading at bargain prices. Since you left us in the autumn of 1972, Bargain Book Club members in Chicago have enjoyed such big books as Irving ...llace's THE WORD, Dorothy Eden's SPEAK TO ME OF LOVE, ...erfield's TO SERVE THEM ALL MY DAYS and Frank ...ENTION, M.D. -- and enjoyed them all at

Dear Reader:

If the list upon which I found your name is any indication, this is not the first ~~-- nor will it~~ be the last ~~subscription letter you~~ receive. Quite frankly, your education and income set you apart from the general population and make you a highly-rated prospect for everything from magazines to mutual funds.

You've undoubtedly "heard everything" by now in the way of promises and premiums. I won't try to top any of them.

If you subscribe to [redacted], you won't get rich quick. You won't bowl over friends and business ... clever remarks ...

[redacted]

1255 PORTLAND PLACE, BOULDER, COLORADO 80302

Dear Mr. Nelson:

LOUSY MAILING LIST / I HADN'T LIVED IN POUGHKEEPSIE FOR SIX YEARS.

Let's get straight to the point. This is not an ordinary letter. It's a subscription offer, from a magazine. It can save you money off the regular price. ...haps won't appeal to everyone in Poughkeepsie. And if it per... will appeal to you, Mr. Nelson, we think it

Because it
... gives you

You call up the bank and ask your balance and they say, "I'm afraid I can't get that information. You see, it's on a computer."

(See Basic Rejoinder, nearby.)

Well, the reason it's this way is that they're handling things in Batch (see p. 45) and they aren't storing your account on disk, or if they are they don't have a terminal they can query it with.

But to say that they can't get the information because it's on a computer is a typical use of the computer as an excuse (see Cybercrud, p. 8); and second, if the person believes this to be an explanation, it's a sign of the intimidation and obfuscation that have been sown among the clerks who don't understand computers.

Write them a letter. Change banks. Let's get the banks to put on more and more citizen services. Rah!

THINGS YOU MAY RUN INTO

Everywhere you go computers lurk. Yet they wear so many faces it's impossible to figure what's going on.

Guidelines are hard to lay down here, but if you look for examples of things you've already run into in this book, it may help some.

Terminals you can presumably recognize.

Microprocessors are harder, because you don't see them. Good rule-of-thumb: any device which acts with complexity or apparent discretion presumably incorporates a terminal, minicomputer or microprocessor.

Two other things to watch for: transaction systems and data base systems.

A transaction system is any system that takes note of, and perhaps requires verification of, transactions. Example: the new point-of-sale systems (POS). This is what's about to replace the cash register.

In the supermarket of the future, every package will have a bar code on a sticker, or printed on the wrapper. Instead of the checkout clerk looking at the label and punching the amount of the sale into the cash register-- an error-prone and cheat-prone technique which requires considerable training-- your New Improved Checkout Clerk will wave a wand over the bar code. The bar code will be sensed by the wand, and transmitted to a control computer, which will ring it up by amount and category (for tax purposes), and even keep track of inventory, noting each object as it is removed from stock.

Here is what your bar code will look like. (A circular code, which was already turning up on some TV dinners, has been eliminated by the bar code. This is unfortunate, since the scanner necessary to read the bar code is electronically more complicated, but there we are.)

(Different versions.)

(Incidentally, while this does arrest the classic cashier's cheat-- ringing up excessive purchases on the customers, then having a confederate walk through equivalent amounts-- the consumer is still entirely prone to cheating by the store in the computer program. Remember, it's 1974. So you still may have to check your tapes, folks.)

Data base systems are any systems which keep track of a whole lot of stuff, often with complex pointer techniques (see "Data Structures," p. 26). A cute example is the message service now offered by Stuckey's snack/souvenir stands all over the country. You may leave messages for your friends or loved ones on the road; they can stop at any Stuckey's and ask for their messages, just as if it was a telephone answering service. (You're listed by your phone number-- is this to avoid pranks? And what about people with no phones?) It's free and a neat idea. (Obviously, the messages are stored on the disk of a big central computer, and queried from terminals at the individual stands.)

Now, most of the big systems you run into tend to be a combination of transaction and data-base system. For instance, suppose you make an airline reservation. The airline has a large data base to keep track of: the inventory of all those armchairs it's flying around the country, and the list of who so far have announced plans to sit in them, and in some cases what they intend to eat. When you buy your ticket, that transaction then gets you put in the listing. Same for car rentals and so on.

The potential dangers of transaction systems are fairly obvious from the supermarket example, but they fan out in greater complexity as the systems get more complex. Credit cards, for instance, were only made possible by computers and computerized credit verification; but it is only now, fifteen or so years into the credit-card era, that laws protect the cardholder against unlimited liability if he loses it.

Yet we plunge ahead, and it is obvious why. Transaction systems managed in, and by, computers allow more flexible and (in principle) reliable operations. For instance, in the securities business, thousands of stock certificates are lost and mislaid, and the transaction paper must be typed, shuffled, put in envelopes, sent, opened, shuffled again, compared... all by hand. Little wonder they're working on an Automated Stock Exchange System. But if it's taken fifteen years to get the implicit bugs out of credit cards ... not to mention the frequent allegations that much Wall Street "inefficiency" is actually the disguised marauding of Organized Crime... uh-oh. (If they can buy the best lawyers, they can probably buy the best programmers.)

Then there is the Checkless Society. This is a catchphrase for an oft-advocated system that allows you to transfer money instantly by computer; supposedly some such thing is working already in France. Again, they better get it pretty safe before a sane man will go up in it.

The safety of such systems is of course a matter of immense general concern. IBM portentiously (sic) announced its intent to spend millions of dollars on "computer security" a few years ago. However, a few million dollars is not going to plug the security holes in the IBM 360, and evidently the 370 is just about as vulnerable.

(In this light, even the greatest IBM-haters will have to admit that there may be a proper motive behind IBM's current refusal to let others use its new operating system language: that way they may be able to prevent special holes in the system from becoming known to programmers.)

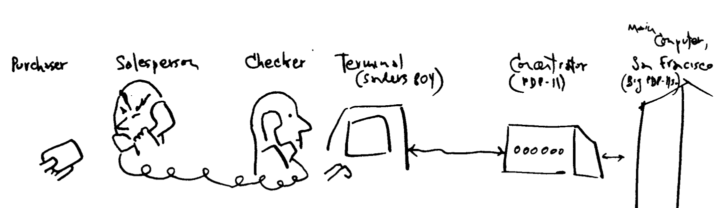
It is interesting that one profession seems to be stepping forward to try to improve this situation: the auditing profession, devoted to verification of financial situations of companies, seems to be branching into the verification of computer programs and the performance of complex systems. This will be great, if it works. Cynics, however, may note that auditors have permitted some remarkable practices in the "creative" accounting of recent years. (Obviously the way to check out the safety of big systems is to offer bounty to those who can break its security. But who is willing to subject a system to a test like that?)

Hereabouts are a few other computerish things you may run into which more or less defy categorization.

THE COMPUTER GRAVEYARD

In the mid-sixties there was a junkyard in Kingston, N.Y. that was like an automobile graveyard-- except piled high with dead computers.

They were from various manufacturers. The guys would smash them with sledgehammers, or other awful things, to make sure they could never work again. Then you could buy the circuit cards. I saw 1401s five high, Univac File Computers, tape drives... it was an electronic nut's paradise. You could decorate your den with huge old control panels, mag disks and whatnot. It seems to be gone now. They forbade pictures.



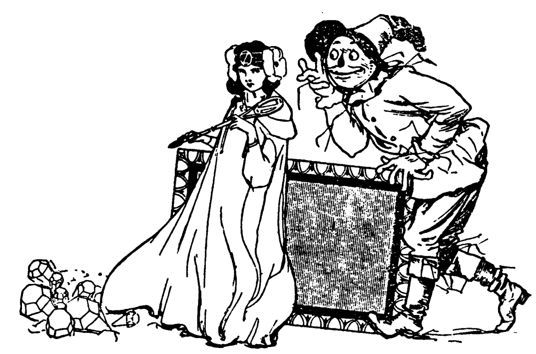
HOW BANKAMERICARD CHECKS YOUR CREDIT
(source: Datamation.)

"COMPUTER DATING"

should of course be called MATCHUP DATING, since there is nothing particularly computerish about either the process or its intended result. But there we go again: word-magic, the implicit authority of invoking the word Computer. (See "Cybercrud," p. 8.)

In the early sixties, a perky young fella at the Harvard B-School, I believe, one Jeff Tarr, came up with the notion of a computerized dating service. The result was Operation Match, an immense financial success, which sort of came and went. No followup studies were ever done or success statistics gathered, unfortunately, but they certainly had their fun.

The basic principle of "computer dating" is perfectly straightforward. Applicants send in descriptions of themselves and the prospective dates they would like to meet. The computer program simply does automatically the sorts of thing you would do if you did this by hand: it attempts to find the "best" match betweeen what everybody wants and what's on hand.



Obviously this could be a matter for serious operations research: attempting to discover the best matchup techniques among things that never really fit together, detail for detail; trying to find out, by followup questionnaires, what trait-matchings seemed to produce the best result, etc. But such serious matchup-function research remains, so far as I know, to be even begun.

Obviously there are several problems. Demographically it is almost never true that "for every man there's a woman"-- in every age-bracket there's almost always an imbalance of the opposite sex in the corresponding eligible age-bracket, either too many or too few. But more than that, there is little likelihood that the traits women want are adequately represented among the available males, or vice versa. For introduction services it's obviously worse: there is no balance likely between what comes in one door and what comes in the other. The service can only do its best with the available pool of people-- and make believe it's somehow made ideal by the use of the computer. It's like an employment office: applicants don't match openings.

Numerous other dating services have appeared, some of which don't even pretend to use the computer (and others which claim to be a registry for nonstandard sexual appetites), but none that's gotten the attention of the original Project Match.

But there's no question who got the best dates out of that one. Jeff Tarr.

DO YOU GOT RHYTHM?

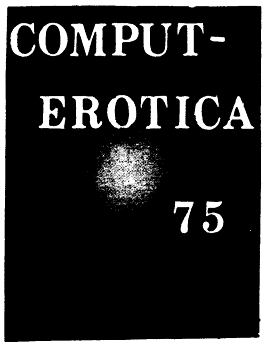
A device called the BIO-COMPUTER (trade mark) purportedly helps you predict your "body beats," telling you what days are the right sort of time to do particular things in terms of your own biological energies. The object costs $15 postpaid from BIO-COMPUTER, Dept. CLB/DM (why not?), 964 Third Ave., NY NY 10022.

The question with all such special purpose devices-- "fishing computers," horse-racing computers, etc., is always whether the theory and formulas which are built into them are correct. There is no ready way to tell.

ASTROFLASH, etc.

There are various computerized astrology services. Given your date of birth, and hour if known, they'll type out your signs, explanations, etc. Presumably there is a text network which the system selects among according to "reinforcing tendencies," etc., among the entities thought to be influential.

Conceivably this could do nine-tenths of what a talented human astrologer does, and with the same validity, whatever that may be. In any case it's probably a lot cheaper.



Is it too soon for a computer pornography contest?

(Is it too late?)

See p. DM35.

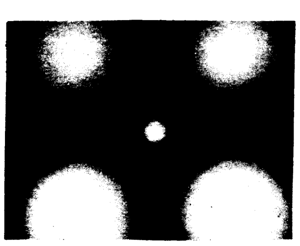
SUPER-CUSTOMIZATION

People think computers are rigid and invariant. This (as stated elsewhere in this book) is due to the systems which people have imposed, and then blamed, on the computer.

The fact is that computers are now being set up to give new flexibility to manufacturing processes. Computers, directly connected to milling machines, grind metal into any conceivable shape much faster than a human craftsman. To change the result, change the program-- in a fraction of a second. Fabric design has been done on computer screens; the obvious next step is to have the computer control the loom or knitting machine and immediately produce whatever's been designed.

Custom clothing: soon we may look forward to tailoring services that store your measurements and can custom-tailor a suit for you to any new fashion, in minutes. (But will the price beat Hong Kong?) Customized printed matter is already here (see "Me-Books," p.67). Wherever people want individual variations of a basic manufacturing process, computers can do it.

The Telephone Company (at least in Illinois and Indiana) offers a speaker on "The Shadowy World of Electronic Snooping" to interested groups.



Modern menage, she 29, interested in recursive relations and reverse Polish culture. Phone a must. Contact box RS-232 (& see p. DM35).

BETCHA DIDN'T KNOW...

that the IRS hasn't been able to do instant matching of W-2 forms to tax returns. That'll be fixed in fiscal '74, and interest and dividend payments in '75. (TIME, 31 Dec 73, 17.)

"COMPUTER ELECTION PREDICTIONS"

This is an outrageous misnomer. The computer is only carrying out, most speedily, what hardened politocoes have always done: FACTIONAL ANALYSIS, now possible with new-found precision on the basis of certain election returns.

This is based on the cynical, and fairly reliable, view that people vote according to what faction of the greater populace they belong to-- middle-class white liberals, blue-collar non-union members, and so on. The factions change slowly over time, and people move among them, but the fact of factionalism remains unchanged.

Well. By the close of a major election campaign, most factions can be pretty well predicted, especially as to presidential choice, or what proportion of that faction will go for a given candidate.

But some factions' reactions are not certain up to the day of the ballot.

So. "Computer predictions" of elections basically break the country into its factional divisions, state by state and district by district, and then tabulate who can be predicted to vote for whom on a factional basis.

Then what's the suspense?

The suspense comes from the uncertain factions-- groups whose final reactions aren't known as the election starts.

Certain election districts are known to be chock full of the types of people whose reaction isn't known.

The final "computer prediction" simply consists of checking out how those districts voted, concluding how those factions are going in the present election, and extending this proportion through the rest of the country.

It's often painfully accurate-- but, thank god, not always. When it isn't don't blame "the computer." Thank human cantankerosity.

THE VW CHECKOUT COUPLER

may or may not be a real computer-- friends have told me it isn't-- but it's certainly a good idea.

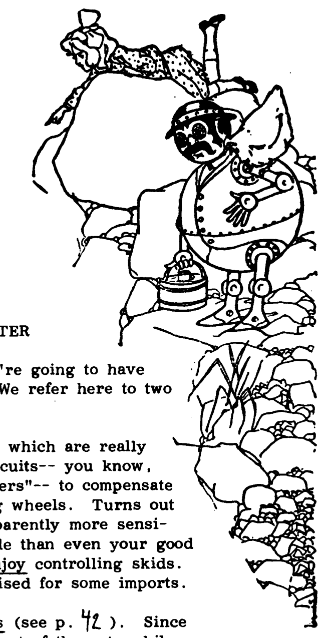
When you pull your late-model Volkswagen into a dealer's service area, the guys can just roll out a cable and plug it into the corresponding socket in your vehicle. At the other end of the cable is some sort of device which tests a series of special circuits throughout the car for Good Condition. These circuits indicate that things are working properly-- lights, plugs, points, brakes and so on.

This is the same technique used by NASA up to the final moment of COMMIT LAUNCH-- a system of circuits monitors the conditions of whatever can be monitored, to make sure all's functioning well. It's more expensive to wire it up that way, but it makes checking out the rocket-- or the car-- that much easier.

SIC TRANSIT

Some of the zappier new Urban Transit Systems give you a ticket with a magnetic stripe on the back. Each time you ride you must push the card into an Entrance Machine, which presumably does something to the stripe, till finally the ticket runs out and you have to pay more money.

Secrecy of the recording code is an important aspect of the thing. Indeed, waggish gossip claims that some such systems start with a blank magnetic stripe and just add stuff to it, meaning the card can be washed clean with a magnet by larcenous commuters. But this seems unlikely.



YOUR AUTOMOBILE COMPUTER

Didja know, huh, we're going to have computers in our cars? We refer here to two things--

anti-skid controllers, which are really just special circuits-- you know, "analog computers"-- to compensate among skidding wheels. Turns out that this is apparently more sensitive and reliable than even your good drivers who enjoy controlling skids. Already advertised for some imports.

grand bus electronics (see p. 42). Since the electrical part of the automobile is getting so blamed complicated, the Detroit Ironmongers have decided to switch to a grand bus structure instead of having all those switches and things separate anymore. Should make the whole thing far easier to service and customize.

Presumably this will all be under the control of a microprocessor. (See p.44.) This means that the car can have things like a Cold-Weather Startup Sequence-- a program that starts the car, turns on the heater, monitors the engine and cabin temperature, and bleats the horn, twice, politely when it's all ready-- all at a time preset by the dashboard clock.

Presumably Detroit is not yet planning to go this far. But because of the auto industry's anomalously huge influence in America, some have expressed the fear that this move -- toward the integrated-circuit, digitally-controlled grand bus-- would effectively put Detroit in control of the entire electronics industry.

The ever-clever Japanese are computerizing faster, better and more deeply than we are.

They now have a prototype taxi operating under computer control. They're calling it, at least for export, Computer-controlled Vehicle System (CVS).

Basically it's like an Elevated Railway-- you climb up and wait-- but when you get in, you punch a button for your destination. According to Hideyuki Hayashi of the Ministry of Industry and International Trade, the system will be operational in Tokyo within the decade, and is the "cleanest, safest, quickest transport system ever devised by man." Think fast, Detroit.

(A nice point: one of the most important features of such a system is that the vehicles don't react to each other, as do vehicles in the existing Human-controlled Vehicle System (HVS). A whole line of the cars can be accelerated or slowed simultaneously, a crucial aspect of their flexibility and safety. Nothing can possibry go long.)

(Leo Clancy, "Now-- Computer-Controlled, Driverless Cars," National Enquirer 3 Mar 74, 24-5.)

THOSE THINGS ON THE RAILROAD CARS

As we lean on the fence a-chawin' an' a-watchin' the trains go by, we note strange insignia on their sides, in highly reflective Scotch-Lite all begrimed by travel.

Basically it's a stack of horizontal stripes in red, blue and other colors. This is ACI, for Automatic Car Identification. It may yet straighten out the railroads.

In this neolithic industry, it is not known at any given time where a railroad company's cars are, and some peculiar etiquette governs their unrequested use by other firms in the industry. Yet the obvious solution may come about: a running inventory of where all the cars are, where each one is going, what's in it, and who that belongs to. But, of course, that's still in the works. Revolutionary ideas take time.

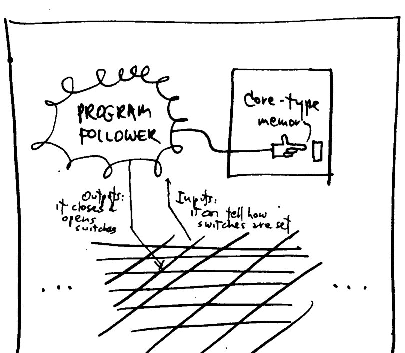
THE ESS

The national phone company (usually called affectionately, "Ma Bell") has drastically changed its switching methods in the last few years. They are replacing the old electromechanical switches, or "crossbars," with a new device called the ESS, or Electronic Switching System. If there's one in your area you may hear about it in their jolly news sheet that you get with the bill.

In the old crossbar days, a phone connection was a phone connection and that was that. Now, with the ESS, all sorts of new combinations are possible: the ESS has stored programs that determine its operation. If you dialled a non-working number, it jumps to a program to take care of that. It does all sorts of things by special program, and new programs can be created for special purposes. Now the phone company is trying to find the services that people will pay for. Having calls rerouted temporarily to other numbers? Linking up several people in a conference call? Storing your most-called numbers, so you can reach them with a single or double digit?

These particular services are now being offered experimentally.

The way it works is this: there are a number of programs stored in a core memory; the only "output device" of the system consists of its field of reed switches, arranged to close circuits of the telephone network.



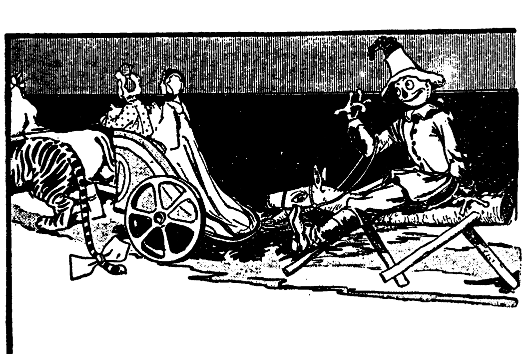
Depending on the numbers that have been dialled, and whatnot, the ESS jumps to a specific program, and that tells it to connect an incoming call to particular other circuits, or to ring other lines, or whatever.

It's really neat.

There are only a couple of things to worry about.

One is that it makes wiretapping, not a complex bother involving clipped wires and men hunched over in cramped spaces, but a simple program.

Another is that some people think that blue-boxers (see nearby) may be able to program it, from the comfort of their own homes. Meaning that not just court-authorized wiretaps, but Joe Schmoe wiretaps, would be possible. Let's hope not.



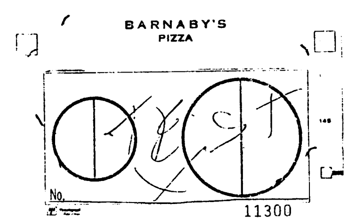
TELAUTOGRAPH

This has been around for decades, and has nothing to do with computers, but isn't it nice?

You write with a pen attached by rods to a transmitter; somewhere else, a pen attached by rods to a receiver duplicates what you have written.

What is being transmitted consists of the measured sideways motion ("change in x"), the measured up-and-down motion ("change in y"), and the condition of the pen ("up" or "down"). What would these days be called "three analog channels, multiplexed on a single line."

These only cost a couple of hundred dollars. Why has nobody been using them for computer input?



Sugar Creek, Texas will have 3000 homes with a minicomputer-based alarm system. Evidently various automatic sensors around each house sniff for fires and burglars, as well as providing panic buttons for medical emergencies.

The system uses dual Novas (one a backup), and prints out the news to fire and police dispatchers on a good old 33ASR Teletype. (Digital Design, May 73, 16.)

ONE OF THOSE MYTHS

"Overpay your phone bill by one cent. It drives the computer crazy."

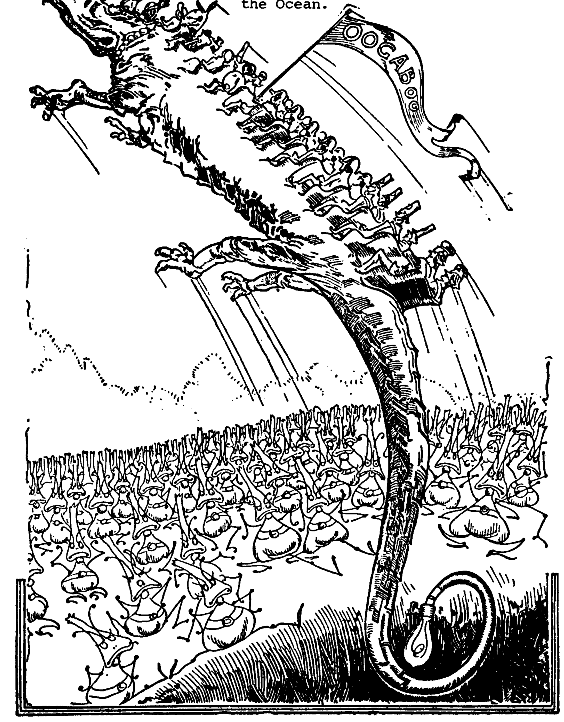
Nope. The amount of payment gets punched in and goes through the gears quite normally.

If you want to put together your own computer-on-a-chip, or any other complex integrated circuit, a complete simulation-verification-layout-and-fabrication service is available from Motorola, Semiconductor Products Div., P.O. Box 20924, Phoenix, Arizona. Presumably it costs a mint, but after that you can roll out your circuits like cookies.

Your circuit is overlaid on their beehive-chip of logical subcircuits, called a Polycell. You use their MAGIC language (Motorola Automatically Generated Integrated Circuits), which then feeds a resulting circuit data structure to a program called SIMUL8 (yuk yuk) to try out the circuit without building it. That way you can supposedly be sure before they make the final masks.

I always figured that the day of Computer Hobbyism would arrive when the folks at Heathkit offered a build-it-yourself computer. But you know what they came out with instead last year? A general interface for hooking things to the PDP-8.

Minicomputers handle various control functions in our mighty new Aeroplanes and Ships of the Ocean.



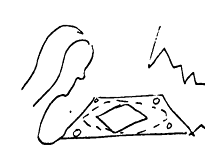
It was a truly stellar group that reported to Judge Sirica on 15 Jan 1974 that the 18-minute Watergate tape buzz had at least five starts and stops.

The six panelists included:

Richard Bolt, a founder of Bolt, Beranek and Newman, Inc.
Franklin Cooper, head of Haskins Laboratories, ~~(see p.~~)
Thomas Stockham, audio resynthesizer extraordinary (see p. DM 11)

The news, however, generally referred to them as "technicians."

QUADRAPONG,



a swell video game now in bars, probably controls the four-player pingpong on the screen with a minicomputer or microprocessor.

Especially exciting is the social possibility of horizontal screens for other fun interpersonal stuff. As well as collaborative work. (But boy, let's hope the radiation shielding is good.)

The Computer Diet by Vincent Antonetti (Evans Pub.) shows the author sitting on the deskplate of a 360 console.

The inside consists principally of charts he recommends for weight loss. "The power of a modern digital computer" interpolated the tables. A slide rule might have have been simpler.

The thing is, he presents a paper on the thermodynamics of weight loss which may be important; in this he states the difference equations which are the heart of his diet. And these may indeed be perfectly valid. So why not call it what it is, The Thermodynamic Diet?

Kirk Brainerd, of L.A., is using computers for a registry of people with something to teach. He hopes that if people are mutually available to each other at a deep enough level, people can begin to act out of altruism in general.

ME-BOOKS(TM)

Would you believe that the greatest available computer service is for the kiddies?

For four bucks and a half, an outfit called Me-Books will send, to a child you designate, a story of which he is the hero, in which his friends and siblings appear, and whose action involves his address and birthday.

Kids adore it. Children who don't like reading treasure the volumes; children who do like reading love them just as much.

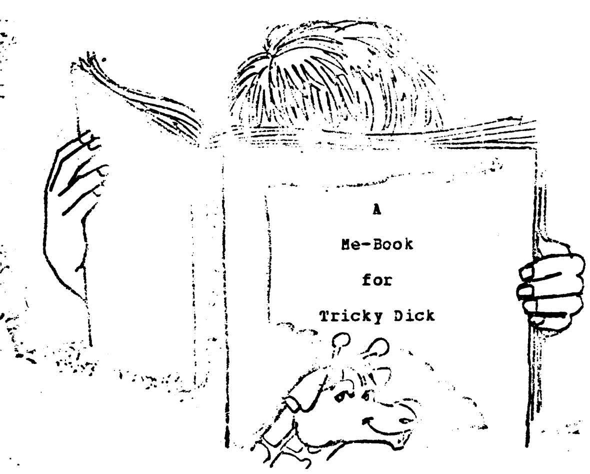
I can personally report, at least on the basis of the one I ordered (My Friendly Giraffe) that the story is beautifully thought out, warm, loving, and cleverly plotted. In other words, far from being a fast-buck scheme, this thing has been done right. It's a splendid children's story. (I won't reveal the plot, but the Giraffe's birthday, name and home address are related to those of the protagonist.)

Moreover, it has three-color illustrations, is on extra-heavy paper and is bound in hard covers.

(In case you're interested, any of the three programming languages expounded earlier in the book would be suitable for creating a Me-Book: depending on the language chosen, the holes left for the child's own name would be alphabetic variables, segment gaps or null arrays -- anyhow, you could do it.)

Astute readers of the Me-Book will note that while it's not readily obvious, only the lines on which personalized information appear have been printed in the computer's lineprinter. The others have all been pre-printed on a press. Indeed, the personalizations appear on only one side of each page, the whole book being one long web of paper that's run through the lineprinter just once before being cut and bound. But it's so cleverly written and laid out that the story moves on beautifully even on the pages that don't mention the child's name.

As an experiment, the author tried sending for a copy of My Friendly Giraffe as told about a little boy named Tricky Dick Nixon, residing at 1600 Pennsylvania Avenue in Washington, D.C. The result was extremely gratifying, and well worth the $4.50. Herewith some excerpts.



COMPUTER QUALITY CONTROL SHEET

*ACCOUNT NUMBER: 1344563005

DATA SUMMARY

1

CHILD'S NAME (B): Tricky Dick Nixon
ADDRESS: 1600 Pennsylvania Ave
CITY, ST ZIP: Washington DC 20050
BIRTHDATE: July 4, 1976 I said 1776.
ADDITIONAL NAMES (B): Spiro
Mitchell
Vesco
DOG'S NAME: Checkers
CAT'S NAME:
BOOK PLATE:
GROWN-UP'S NAME: The Founding Fathers
ADDRESS: T Nelson

BOOK SHIPPED TO GROWN-UP

Once upon a time, in a place called Washington, there lived a little boy named Tricky Dick Nixon.

Now, Tricky Dick wasn't just an ordinary little boy. He had adventures that other little boys and girls just dream of.

This is the story of one of his adventures.

It's the story of the day that Tricky Dick met a giraffe.

> As the giraffe came closer and closer, Tricky Dick started to wonder how in the world he was going to look him in the eye.

Tricky Dick knew there were no jungles in Washington. Especially on Pennsylvania Ave.

But Tricky Dick wasn't even a little bit worried.

First, because he was a very brave little boy.

And second, because he knew that his friend, the giraffe, would never take him anyplace bad.

Tricky Dick Nixon was home.

Back in Washington.

Back on Pennsylvania Ave.

And with a story to tell his friends, that they wouldn't have believed if they hadn't seen Tricky Dick riding off on the giraffe's back.

Tricky Dick would long be a hero to those who had seen him that day.

There would be many other exciting adventures for Tricky Dick and his friends.

And maybe, just maybe, if you're a very good boy, someday we'll tell you about those, too.

PERSONALIZED ME-BOOKS™ NOW AVAILABLE:

My **Friendly Giraffe**
Your child and the child's friends and pets take a jungle trip with a friendly giraffe. Personalized in over 70 places.

My **Jungle Holiday**
The child of your choice and the giraffe visit the animals in an amusement park. Personalized throughout.

My **Birthday Land Adventure**
People in the land of candy and cake tell all about your child's exact birthday, from birthstone to famous birthdays.

My **Special Christmas**
As Santa's helper, your child visits the Santas of the different countries and learns the true meaning of Christmas.

For additional Me-Books™ written around a child of your choice, complete an order form at your favorite bookstore or write: Me-Books Publishing Co., Dept MB2, 11633 Victory Blvd., North Hollywood, Calif. 91609. Enclose $3.95 plus 50¢ for postage and handling. (Calif. residents add 20¢ for sales tax.) Be sure to state which Me-Book™ you desire and include the following information:

PERSONALIZED STORY DATA.
If certain information below is not available or not applicable **LEAVE BLANK**. This charming story will be written without it. **PRINT CLEARLY**, one character per space and one space between words. (Example ED JONES)
If not enough space, abbreviate.

Child's first name or "nickname" ____ Last name ____ ☐ Boy ☐ Girl
Child's address ____ Apt ____
City ____ State ____ Zip code ____
Child's birth date ___/___/___ (Month Day Year) Below, list up to 3 friends, brothers or sisters
____ ☐ Boy ☐ Girl ____ ☐ Boy ☐ Girl ____ ☐ Boy ☐ Girl
Dog's name ____ Cat's name ____
Grown-up's name to appear on personalized book plate:
("Aunt Jane, Grandma, Mom & Dad" etc.) ____
Grown-up's name (Person buying book) ☐ Mr ☐ Mrs ☐ Miss First initial ____ Last name ____
Grown-up's address ____ Apt ____
City ____ State ____ Zip code ____

I bought my last Me-Book™ at ____ (Name of Retailer)

About those funny numbers on your checks.

You will note that all bank checks now have funny-looking numbers along their bottoms. They go like this:

0123456789 ⑆ ⑇ ⑈ ⑉

The numbers are odd but recognizable. The last four thingies are punctuation marks, which presumably can mean anything the programmer wants them to. (In other words, frankly, I don't know their names or standard functions.)

The name of these numbers is **MICR**, which stands for Magnetic Ink Character Recording. They are printed in magnetic ink-- not magnetic so's you could record on it, like magnetic tape, but chock full of iron and vitamins so that as its blobs whiz past a special read head, they cause a specific sequence of pulses in the parallel circuits of the read head that can be decoded as the specific number or mark.

The MICR system was designed in the late fifties, with the technology convenient at that time, and would certainly not be designed that way now. Nevertheless, these weird-looking symbols have inspired various

RIDICULOUS TYPE-FACES,

which apparently look to the public like the latest hotcha whizbang zippity up-to-date futuristic stuff, even though to the knowledgeable person they bring back the late fifties. (In fact there are no letters in the MICR character-set.)

What, then (you may ask) would symbols designed for computers look like if they had been designed more recently?

We were just getting to that. In fact, there are two such alphabets, called OCR (for Optical Character Recognition). They have been standardized so everybody can design equipment and/or programs to work with them. They are called the A and B optical fonts, or, for completeness, OCR(A) and OCR(B).

They are very disappointing.

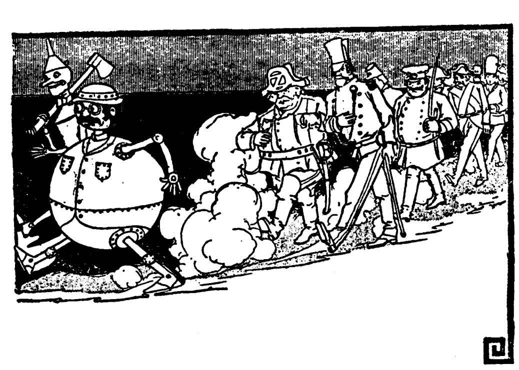
OCR(A) is a little sexier. At least it looks like something. (Evidently it's slightly easier to deal with and design for.) But the other one, OCR(B), just looks like the alphabet next door. Here they are.

ABCDEFGHIJKLM
NOPQRSTUVWXYZ
0123456789
. , : ; = + / $ * " &
' - { } % ? ⑁ ⑂ ⑃
Ü Ñ Ä Ø Ö Æ Å £ ¥

OCR(A)

1234567890
ABCDEFGHIJKLM
NOPQRSTUVWXYZ
abcdefghijklm
nopqrstuvwxyz
*+-=/.,:;"'_
?!()<>[]%#&@^
¤£$|\█ —— ¥

OCR(B)



THE CLUB OF ROME

One of the world's most exclusive clubs is also one of its most dismal. It is The Club of Rome, founded by Italian businessman Aurelio Peccei, having (as of 1972) some seventy members from twenty-five countries.

Their concern they call The Predicament of Mankind, or the "problematique." It is the problem of growth, pollution, population, and What's Happening in general.

On funds from Volkswagen, they have sponsored studies which thinking men can only regard as the most dismal in portent of anything we've seen in years. Or ever.

Basically the prediction is that mankind has perhaps forty or fifty years left.

Not because of war, or bombs, or dirty movies, or Divine retribution, but for simple economic reasons. However, the studies are often called "computer studies," because computers are the viewing mechanism by which we have come to see these coming events.

MALTHUS AGAIN

In the nineteenth century, a pessimistic economist named Thomas Malthus predicted that there would always be starving people, because people increased geometrically-- expanding at compound interest, with a fixed rate of increase creating an ever-steeper growth-- while agricultural production, which must feed us all, expands arithmetically, not as a rate but a few acres or improvements at a time.

This meant, Malthus thought, that there would always be the starving poor. For various reasons this did not happen in Europe. But the regrettable soundness of the general principle persists: when rates of food production can't nearly keep up with rates of population growth, people are going to starve.

This is basically the prediction.

DYNAMIC MODELLING

Basically what has happened is this. One Jay Forrester, of MIT, has for some years been studying "dynamic models" of things, a new breed of simulation which couldn't have been done without computers. And now dynamic models of the world's entire economic system can be created and tried out.

Basically dynamic models are mathematical complexes where things change at rates that change themselves over time. For instance, the more you eat, the fatter you get, and the fatter you get, the hungrier you are going to be. Now, just because this is simple to say in words, and sounds as though mathematicians would have had solved the whole class of problems centuries ago, that's not how it is. The intricacy of such models, even for just a few variables, made it impossible to foresee what happens in such complexes exact by techniques of computer enactment. Forrester, who has studied such systems since the fifties, has become alert to their problems and surprises. The culmination of his work has been a model of the entire world's economic growth, agriculture, population, industrialization and pollution; this is described in his book World Dynamics (Wright-Allen, 1971).

The insidious portents of Forrester's work did not go unnoticed. The dangers of population increasing at compound interest on a planet of unchanging size, and further derivatives of these changes, suggested that things might be getting worse than anybody thought. An alert Italian businessman brought together a group of scholars from all over the world to study these problems, and called the group The Club of Rome. Their first work is out now, and it is very scary and all too real. The book is called The Limits to Growth.

Basically what they have done is a very elaborate computer simulation, modelling the entire economy of the planet in the years to come as a structure of rates. They have taken into account population, food-growing capacity, industrial growth, pollution, and a lot of other things. The model is precise and elaborate.

Unfortunately the findings are precise and simple.

They tried all kinds of alternative futures using the model-- what would happen if the birth rates were different? What if there were no pollution? What if resources were infinite?

The results of the simulations are always the same.

According to all the simulations, the human race will be wiped out-- mostly or completely-- by the year 2100.

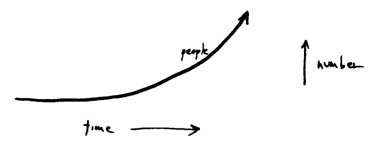
Let's go briefly through the model. Note that it can't be exact, and we can't know what years things are going to happen. The curves themselves-- the shape of things to come-- tell the story all too clearly. (For those who would like a little more drama with their numbers, finding these matters too abstract, I strongly recommend the very beautiful Indian film "Distant Thunder," a sort of "Mr. Smith Starves to Death." Or just stick around awhile.)

HUH?

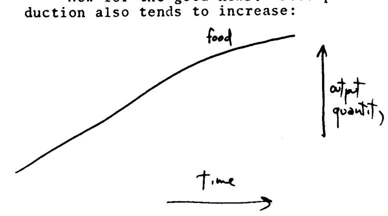
The model assumes that birth rates stay relatively constant in particular parts of the world, and that new land and agricultural techniques increase food production in relatively well-understood ways.

Of course, population continues to go up, on the familiar but deadly curve.

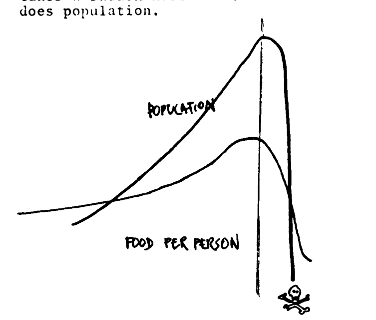
> **Civilization, and the bulk of mankind, have about forty years to live, according to certain studies (see p.68). The studies are depressingly good, although unfinished.**
>
> **There are four possible things to do.**
>
> **1. Ignore it.**
>
> **2. Deny it.**
>
> **3. Seek individual salvation somehow. Hide in a remote corner. Lay in stores.**
>
> **4. The glorious flameout. Eat, drink and be merry, for tomorrow we die. Or apocalyptic occultism, or whatever.**
>
> **5. Work starting now. In whatever directions might, just might, point or contribute to a way out.**



Now for the good news. Food production also tends to increase:



Now for the bad news. The running ratio of food to people, Food per Capita, takes a sudden nose-dive. And then so does population.



It is not any individual prediction that is frightening, since the numbers plugged into the separate runs are merely hypotheses, to show the shape of the consequences. It is the overall set of runs that is so ghastly, because they always come out the same.

PAY CLOSE ATTENTION

Now, it is important to clarify what is happening here and what is not. What is not happening: an oracular pronouncement by "the computer," showing some transcendental prediction by a superhuman intelligence. What is happening: people are trying out separate possible assumptions to see what their consequences are, enacted by the computer according to the economic rules they set up. Result: always the same. Any set of rules, played out in the unstable exploding-population world beyond the seventies, appears to have similarly dire results.

WHAT HOPE IS THERE?

The original model is only an approximation, and the basic results, as published in The Limits to Growth (see box) reflect those approximations. One of the things that can be done is to fill in and expand the model more, to see whether any hopes can be found in the details and fine cracks which don't appear from the gross results. And, of course, to study and re-study the basic findings. (For instance, a small error was recently found: a decimal point was misplaced in the "pollution" calculation, leading to an overstatement of the pollution in some of the runs. (But pollution, remember, is only part of the problem.)

So there you are. This is a study of the greatest importance. We may, just may, be getting wind of things in time to change the outcome. (If only we knew how. But again, this study is where serious discussion must begin.)

IBM IS BULLISH ON THE FUTURE

Lewis M. Branscomb, who has the awesome title of Chief Scientist of IBM, has been giving numerous talks recently that seem to be directed against pessimism resulting from the Club of Rome studies.

> "'On the shoulders of the information processing community rests the responsibility for convincing the public that we have the tools, if it has the will, to address the complex systems management problems of the future,' Branscomb said.
>
> "'More than any other profession our community can restore the public's confidence that from the limited resources of the world can be fashioned a life of well-managed abundance for all,' he concluded."
>
> (Keynote speech, ACM 73, quoted in Computerworld, 5 Sep 73, p. 4.)

ENDGAME.

Now begins the winter of the world.

We are poisoning everything.

With so little time left, we are of course expanding and accelerating every form of pollution and destruction.

We are killing the last of our beautiful brothers, the whales, just to provide marginal amortization of the whale-ships that are going to be scrapped anyway.

Item: supposedly the Sahara Desert was man-made. It is growing fast.

Set down upon this beautiful planet, a garden spot of the universe, we are turning it into a poisoned pigsty.

You and I may starve to death, dear reader. In some year fairly soon now, around the turn of the century, there will no longer be nearly enough food for the teeming billions.

That, anyway, is what the predictions say. The predictions are compelling, not because a computer made them -- anybody can make a computer predict anything-- but because the premises from which the predictions grow were very well thought out.

It is now up to us to make the predictions come out wrong.

Not by killing the bearer of bad tidings, or by pretending they were not clearly stated-- but by seeing what possible alternatives remain in the few moments of real choice we may yet have-- scant years at best.

To haggle now about ideology is like arguing about who is driving while we are headed toward a brick wall with the gas pedal jammed to the floor.

The public thinks, "science will save us," a view at which many scientists snicker bitterly. Perhaps we will be shrunk to an inch's height, or fed on rocks, or given gills and super-kidneys to live in the ever-more-poisoned sea. Or perhaps we will do what science says others have done: die out.

This science-will-save-us ostrich-position is nicely exemplified by Albert Rosenfeld, Science Editor of Saturday Review/World.

Since "science" has given us the Boeing 747 and the neutrino, neither of which could once, he thinks, have been imagined possible, obviously (to him) science can do anything else we think is impossible! He fully imagines that science will come up with something to take care of geometrically increasing numbers of people. In perpetuity?

"Take a lesson from the neutrino," he says. "We can solve our problems." ("Look to the Neutrino, Thou Doomsayer," Saturday Review/World, Feb 23 1974, 47.)

OTHER FUN

The growing diffusion of weapons and grudges, and the great vulnerability of almost everything, assure that terrorism and political extortion will will increase dramatically for the foreseeable future. On the other hand, whole economic blocs and industries have lately mastered and demonstrated by example how to hold the country at bay in order to get their wishes; as everybody can see what's happening, and learn from it, the number of wrenching unpleasantnesses created by terrorists and industries and interest blocs will increase.

All these were essentially foreseen by Thomas C. Schelling in his masterly 1960 work, The Strategy of Conflict. Schelling formalizes a theory of intimidation as part of his study of communicating adversaries. (His is a structural rather than a psychological study, examining the properties of situations whether or not they are psychologically perceived. Regrettably, perception of situations is improving all the time.)

Cousteau says the oceans are dying a lot faster than he anticipated -- and gives mankind fifty years after life ends there.

But even if everything else were all right, the Breeder Reactors are sure to get us. I refer to those wonder machines that the electric companies are calling Clean Energy for the Future. What is not explained with such slogans is that breeder reactors not only create energy, they create atomic waste, breeding new fissionable material-- including plutonium. Plutonium is well named for the god of hell. Chemically a poison and radioactively a horror, it does not go away; wherever we put it, it will get back to us.

The mere radiation from the atomic crap is hardly the problem. The radioactive poisons are getting into the oceans. They are getting into the clean waters of the land. (A December 1973 news report, for instance, revealed that a 1968 leak of radioactive chemicals was into the water supply of Bloomfield, Colorado.) Now, atomic enthusiasts call it a Disposal Problem, like the question of where to bury the garbage. But it's a very different problem. Wherever we put it, it will come back. The sea? No, that'll be poisoned after the containers go. Deep wells? The mountains? But there is no place that cannot be guaranteed against earthquake and recycling. It will come back. Though dozens of generations might survive it, it will be waiting.

But the breeder reactors multiply this output. Perhaps we could survive the the waste for a few hundred years, till it comes back out. But the other part of it is the fissionable material which can be made into backyard bombs.

That's the kicker. With more and more fissionable crud being disgorged, its availability for terrorists who want to build their own increases. Ralph Lapp pointed out last year that the stuff was shipped in unguarded trucks, and one or two good hijackings would enable any bright kid to build his own dirty A-bomb. By the year 2000 it is not inconceivable that bootleg atomic weapons will be as widespread as handguns in Detroit-- and as much used.

But now, with the breeder reactors-- in lots of countries-- pouring the stuff out, the era of atomic plenty is here. The smaller countries who want them are getting their atomic weapons -- though holding back assembly of the parts, for various reasons. It is generally believed among bomb-watchers, for instance, that India and Israel have theirs anytime they want.

Add this to the great avalanche of missiles, tall and horny in their silos, ready to go on two, later three or four, sides. (The U.S. official arsenal now stands at the explosive equivalent of 5 billion tons of TNT, a ton of TNT for every human being. And that's just the explosive part, not the fallout; a fraction of these bombs could destroy all life on earth by its seething residue.) And now, because of the SALT talks, we may expect a new and drastic increase of this Readiness Posture. Hoo boy. What is there to say.

So there it is, folks, merry times ahead. Humanity may end with a bang (thermonuclear exchanges, or just desultory firings until we're all poisoned or sterile), or a whimper (universal starvation), or, I would anticipate, some spastic combination of the two, and all within the (possible) lifetime of the average reader. This is, at any rate, what I think most likely.

Except of course we won't see it happen that way. We'll watch the starvations on TV (as we did Biafra, Bangladesh, now West Africa, what next... India?), and tsk about the poor foreigners who can't take care of themselves. And as the problems increase and move toward our heartland, it'll be blamed on environmentalists and on the news media, till bang.

Or maybe not. Just maybe.

But we've all got to get access to the Club of Rome models, and look for holes or strategies. If computer modelling systems doing this kind of work are made widely enough available, perhaps some precocious grade-schooler or owlish hobbyist will find some way out that the others haven't hit on...

We've got to think hard about everything.

BIBLIOGRAPHY

Frederick Pohl and C.M. Kornbluth, The Space Merchants. Ballantine, paper.

Thomas C. Schelling, The Strategy of Conflict. Paper.

The Great American Bomb Machine (citation not handy). Paperback.

A book called Cold Dawn (citation not handy; originally published in the New Yorker) presents a most discouraging view of this country's actions in the SALT talks.

One Access Catalog, not to be named here, gives a recipe for an atomic bomb. Very funny, ha ha. "The U-235 we are using, (although Plutonium will work just as well) is a radioactive substance and deserves some care in handling. It is NOT radioactive enough to kill with limited exposure, but don't sleep with it or anything." And so on. Thanks a lot, fellas.

Ralph Lapp had a piece in the New York Times Magazine last year, pointing out that plutonium is shipped in unguarded trucks and it's only a matter of time before punks get their hands on it...

A piece in a recent Esquire, "Did There Ever Come a Point in Time When There Were Forty-Three Different Theories about Watergate? Yes, to the Best of Our Recollection," is a very helpful general source, especially for those who suspect a connection between "Watergate" and the assassinations of the Kennedys, Malcolm X, Martin Luther King, etc. But for a real chill see "Mae Brussell's Conspiracy Newsletter" in the March (?) 1974 Realist, as well as "Who Is Organized Crime and Why Are They Saying Such Awful Things About It," same issue.

Glen A. Love and Rhoda M. Love, Ecological Crisis: Readings for Survival. Harcourt, $4 (paper). A quick way to catch up on some bad stuff. Four bucks well spent.

William Leiss, The Domination of Nature. Braziller, $7.

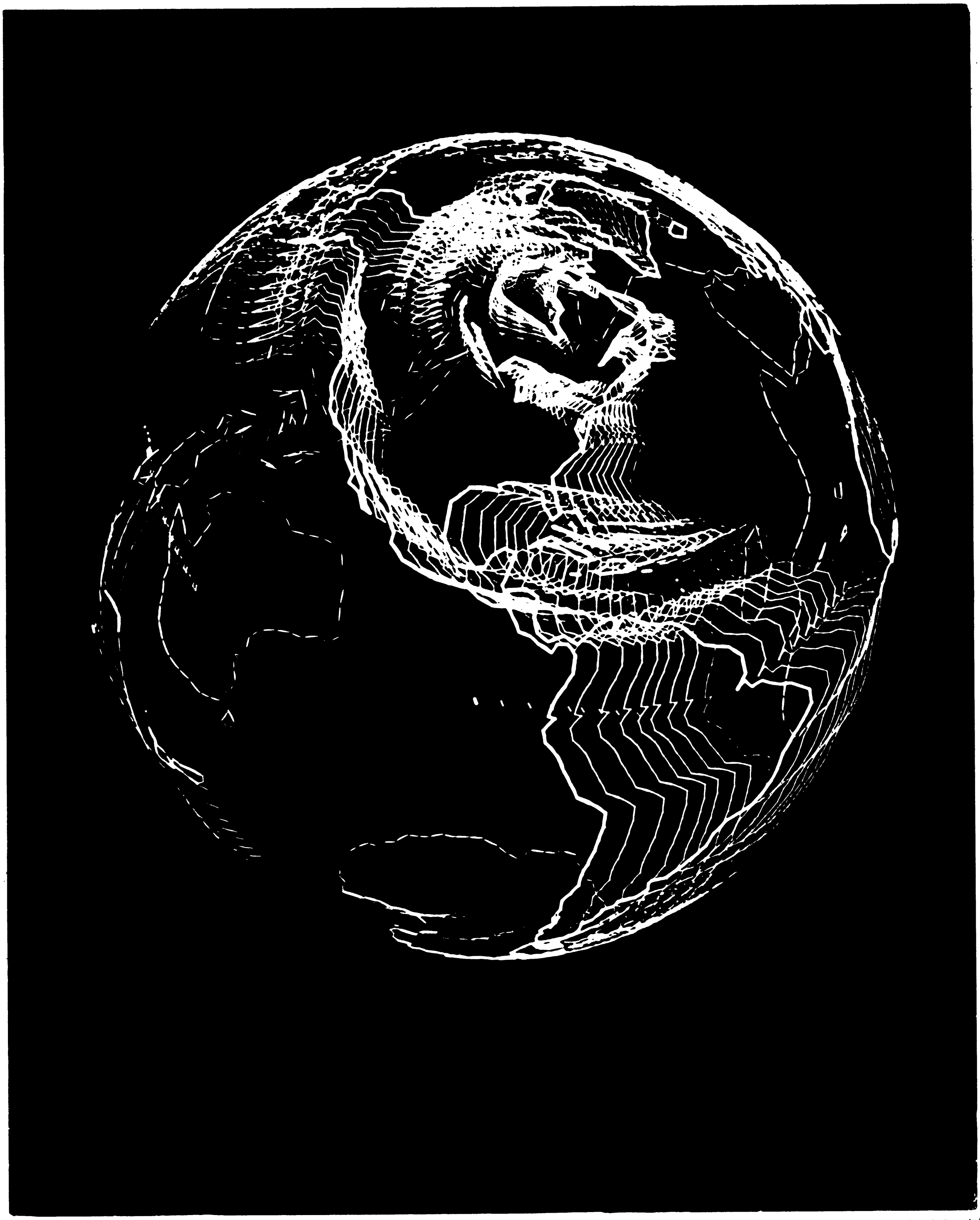
For a dazzling, romantic and optimistic view of the future, see Dimensions of Change by Don Fabun (Glencoe Press, $5 in paper).

The Futurist magazine goes out to members of the World Future Society, An Association for The Study of Alternative Futures, Post Office Box 30369, Bethesda Branch, Washington, DC 20014. The magazine used to be pretty sappy and optimistic, but seems to be acquiring sophistication.

Ronald Kotulak, "The Lifeboat Ethic." Chicago Tribune Magazine, 28 Apr 74, 19-22.

Perhaps the Club of Rome study should be called--

THE HOLE EARTH CATALOG



Tom DeFanti (see p. DM 31)

"I have a dream..."

P. My feeling frankly is this. That you know I was just thinking tonight as I was making up my notes for this little talk, you know, what the hell, it is a little melodramatic, but it is totally true that what happens in this office in the next four years will probably determine whether there is a chance, and it's never been done, that you could have some sort of an uneasy peace for the next 25 years.

E. Uh huh.

(Nixon to Ehrlichmann. Apr 73.)

Thank you, Mr. President.

READ IT AND WEEP

Donnella H. Meadows, Dennis L. Meadows, Jørgen Randers and William W. Behrens III, The Limits to Growth: A Report for THE CLUB OF ROME'S Project on the Predicament of Mankind. Universe Books, paper, $2.75.

"Things are going to get worse and worse and never get any better again."

-- attributed to Kurt Vonnegut, Jr.

" FOLKS DON'T NEED THESE LI'L SHMOOS!!-- THEY ALREADY GOT ONE-- TH' BIGGEST SHMOO OF ALL-- TH' EARTH, ITSELF! JEST LIKE THESE LI'L SHMOOS, IT'S READY T'GIVE EV'RYBODY EV'RYTHING THEY NEED!! EF ONLY FOLKS STOPPED A-FIGHTIN', AN' A-GRABBIN'-- THEY'D REE-LIZE THET THIS SHMOO-- TH' EARTH-- GOT PLENTY O' EVERYTHING-- FO' EV'RYBODY!!"

-- Li'l Abner

(Al Capp, The Life and Times of The Shmoo, Pocket Books, 1949, pp. 121-122.)

CHEAP COMPUTERS

You already saw about MITS and the Altair on p. X.

MITS' new computer will be based on the Motorola 6800, and sell in kit form for around $300. But their main commitment is to the Altair, a line based on the Intel 8080, and the customers already into that machine will not be in any way let down, they say.

A computer kit based on the Motorola 6800, with 21K bytes of core, cassette recorder and TV display (32 chars. by 16 lines) is offered for $1745 by THE SPHERE, 96 East 500 South, Bountiful, Utah 84010.

Two computer kits, one built around the PACE and another a Nova lookalike, have eeen announced by Bill Godbout Electronics, Box 2355 Oakland Airport, Oakland CA 94614. He also plans an 11 lookalike.

Or you might get an LSI-11. An LSI-11 buying pool is being formed by Hal Lashley, Southern Cal Computer Society, P.O. Box 987, S. Pasadena CA 91030.

CHEAP TERMINALS

MITS has a 'very low-cost terminal' (the VLCT, yuk yuk) for $170 ($129 for kit).

Processor Technology, 2465 4th St., Berkeley 94710, makes a text display kit for the Altair for $160 (you supply the TV monitor and evidently the keyboard). 64 character per line, 16 lines.

A similar kit at a similar price is made by Southwest Technical Products.

Bootstrap Enterprises, Ann Arbor, are also working on a similar unit, called "The Dumb Terminal," with a color option.

MITS is committed now to building a video terminal, the CT-8096, that will provide both text and graphics. Following specs are not final.

PRICE IS TO BE ABOUT $1000.

It will have a keyboard and video monitor, plug straight into the Altair, and refresh from Altair memory modules-- which may double as regular memory, if you don't mind garbage on the screen.

It will have 24 lines of upper-case characters, 5x8 dots to the character, 80 characters to the line on a built-in monitor. In addition it will offer graphics from bit maps (see p. Z), either 120x120 or 240x240. (The resolution will be switch-selectable, if you have enough buffer memory; a screen of text takes 2K, so does a 120x120 picture, and 240x240 takes a whole 4K.) Buffer memory will also be dividable into separate "pages" of text or graphics; and two pages will be superimposable, interlacing alternate video fields (see pp. DM6-7). Note that refreshment is from random-access, rather than serial, memory, so that multiple fields cannot be overlaid.

OTHER ACCESSORIES

While none has been announced as yet, a music synthesizer that plugs into the Altair will almost certainly be available in 1976.

(Note that this could provide an entirely new form of interactive terminal if used with the Wachspress equipment; see nearby.)

A Selectric interface to the Altair is in the works at MITS.

Altair interfaces to the PDP-8, PDP-11 and Nova have not yet appeared. Why not?

DEC's own floppy disk, for the 8 and 11, finally came out. Price for 11: $3000 for one drive, $4000 for double.

LINCtape, which is virtually the same as DECtape but unpatented, has just come out at $2000 for one drive, including controller and interface to 11 or Nova (interrupt-driven). Note that the unit is compact and rugged, and may be more suitable than disk or cassette for those of us concerned about portable rigs and van-mounting. Computer Operations, Inc., 10774 Tucker St., Beltsville MD 20705. (The bad news: software costs $300 for the driver, plus $750 to DEC if you want operating system RT-11.)

Cambridge Memories, Inc., cleverly sells main memory banks for the 11 which can attach to to two PDP-11s at once-- thus connecting the two machines without using DEC's expensive Unibus coupler.

Also for 11s: Formation, Inc., sells a curious programmer's console that traps and displays the last sixteen Unibus addresses referenced; and Fabri-Tek offers a cache memory for the PDP-11/45.

ROLL THE DRUMS...

IBM has announced a laser printer, the model 3800, which prints at 13,360 lines per minute. Like most of their printing machines, it's good. It is basically an electrostatic drum copier, like the original Xerox 914, on which the patents have run out. (Even Toshibafax now makes one.)

Anyway, this spectacular beast writes with a scanning laser on the electrostatic drum. But IBM has cleverly found the way around the problem of spoilage of the drum surface: instead of just a polished metal surface, it's a renewable surface, which is itself changed automatically when required for new images.

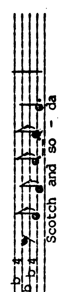
Moreover, you can have up to 18 type fonts, defined in 18x24 dot matrices. (THESE ARE KEPT ON A FLOPPY DISK, AND UP TO FOUR MAY BE CURRENT AT ANY ONE TIME.) Fonts are user-designable.

A flash-projector can put business forms on the drum.

Now for the bad news: base price is $310,000, plus extensive extra charges. Also it doesn't make carbon copies, and needs a new box of paper spliced on the end every 20 minutes.

(Canon, of Japan, has out a more modest laser printer that goes at only 4000 lines per minute. And it also plots. Burroughs is said to be trying to get the bugs out of a similar device.)

NOTE WELL: this musical staff was done in a few seconds on the Diablo, using its ordinary typewheel.



THE HIGHER-PRICED SPREAD: the Diablo can position the characters exactly, and repetitively, anywhere, by fine degrees. This means, for instance, that justified typesetting by computer becomes easy and cheap.

My spacing is now being incremented both horizontally and vertically.
My spacing is now being incremented both horizontally and vertically.
My spacing is now being incremented both horizontally and vertically.
My spacing is now being incremented both horizontally and vertically.
My spacing is now being incremented both horizontally and vertically.
My spacing is now being incremented both horizontally and vertically.

TRULY AMAZING NEWS

One of the buys of computer history is waiting up at American Used Computer, in Boston, 617/261-1100.

Memorex, for some unfathomable reason, built in the early 70s a computer intended to be upward-compatible from the 360/20. But it was not a 360.

Why did they do this? The kind of people who shop around would not buy 360/20s, and the kind of people who buy 360/20s would scarcely leave IBM's skirts at upgrade time.

Thus the Memorex 40 has, quite understandably, been discontinued. And all the ones they had left are waiting for you brand new up at American Used Computer for the heart-stopping price of

$3500.

That price includes 48k bytes of core.

Now for the bad news.

It comes bare-bones, with no software, and no hardware support. You get the wiring diagram with it, and a list of other owners, and you're on your own. AUC does have spare parts, however. And peripherals, mostly more expensive.

Mr. Monoson of AUC told me on the phone that it had 158 instructions, including 64-bit floating point, 32-bit binary. On studying the literature, however, it appears to me that the instruction-set he described is microprogrammed, with the microcode intended to be read in at startup time. (There are 65 microinstructions.) Maybe you can get the microcode for those 158 instructions and maybe you can't. Maybe you don't care, if you're well enough fixed to handle one of these.

It comes in basic black, 2x5x4 feet, fits in a van, and supposedly does not need airconditioning. Supposedly plug-compatible with 370 peripherals! It's really a sixteen-bit machine, and it has eight sets of eight registers, having been designed to perform up to eight functions simultaneously.

So. 64 main registers, 4K dynamic microstore, 48K of memory, for about the price of a used PDP-11/10 with 4K. Smelling salts, anyone?

THE DIABLO & others

"Diabolo" was a game of the twenties that involved poking a spinning object. Oddly, that's what today's Diablo involves.

Redoubtable Max Palevsky, who brought you Scientific Data Systems (which Xerox bought and recently shut down), Rolling Stone and the movie "Marjoe"-- has another winner, which he's also sold to Xerox.

This is the Diablo company, which first made disks and now makes a sensational printing machine. It has a whirling plastic "daisy wheel" of type, interchangeable, and can type 30 characters per second in either direction, as well as draw pictures-- of a sort.

The basic difference between these printers and conventional typewriters, like the Selectric, is their use of servos rather than ratchets. This means their characters can be positioned in many intermediate positions, unlike the fixed positions available on an ordinary typewriter. For instance, the Diablo can position the type to 1/60 of an inch horizontally and 1/48 of an inch vertically. (Nice for justified typesetting.)

There are now a number of machines of this kind. First came the Diablo printer, officially the HyType I; then the engineers who built that went off and created a competitive printer called the QUME (pron.'kyoom'); now there's an improved Diablo HyType II; Interdata makes a competitive unit, the Carousel printer, with a little print cup; and to make things totally confused, there's a special model Diablo called the 800, which can't be connected to computers but is sold for office use as a "word processor."

A number of companies make terminals in the $5000 ballpark embracing one or the other of these printers. Gen-Com Systems makes one around the Diablo; Anderson-Jacobson makes one around the QUME. Xerox makes its own computer terminal, the 3010, around the Diablo I-- which, it should be noted, can be rented for as little as three months, at $190/month.

The one everybody wants for their computers is called the Xerox 800, but so far that is not available as a computer terminal. It goes faster than the other Diablos and offers typefaces that look beautiful for typesetting; much nicer, it seems, than the types currently available for the other Diablos.

For those interested in just hooking up the printer mechanism, for substantially less money than a whole terminal, interfaces for hooking the Diablo or QUME printers to PDP-8 or PDP-11 are available from Data Systems Design, Inc., 1122 University Avenue, Berkeley CA 94702.

SUGGESTIONS TO XEROX CONCERNING DIABLO PRINTERS.
No charge.

1. *Sell the 800 as a terminal, for goodness sake.*
2. *Failing that, make those pretty typefaces available for the others.*
3. *Already you offer black and red ribbons; a blue and yellow ribbon would permit printing PICTURES IN FULL COLOR, a development of great interest to the many computer graphics freaks.*
4. *However, that would require somewhat finer positioning of the platen; say, 1/120 in both directions.*
5. *... failing which, you could put out a "graphic daisy wheel" with intermediate dot positions equivalent to dot positions between those now available.*
6. *Could the Diablo somehow be made to sound less like a dentist's drill?*
7. *How about a portable?*

LOGOMANCY

Dan Hillis and Radia Perlman, of the LOGO group at MIT, are working on a special "preliterate" terminal to allow non-readers (possibly including chimps and gorillas) to program in LOGO, especially on the General Turtle 2500 (see "Minsky's Computer," nearby). Plastic credit cards will have symbols for the various picture and music-box functions. To write a program, or create a movie on the scope, the user will insert function cards in slots. Color coding will be used for program transfer: a red card means "jump to the red subroutine." Since this is MIT, the full recursive power of the system will of course be available. (My hope is that chimpanzees and other little slotniks can be taught recursive program definition. Then will the public wake up to computers being easy?)

MINSKY'S COMPUTER also affectionately known as MARVIN'S COMPUTER, THE FLYING TURTLE

The great Marvin Minsky is renowned on five continents. Dean of the amorphous field of "artificial intelligence," and referred to without ambiguity as "Marvin" throughout computerland, he is a theoretician's theoretician.

But at the heart of every theoretician, I think, burns the dream that he will someday prove the outright, worldly importance of his thoughts. Like Destry, at last he will go to his suitcase and get out his guns, and the audience will cheer.

The great Marvin Minsky has come out blasting.

General Turtle, Incorporated, is a toy company that the team of Minsky and Papert put together to market their educational computer accessories. (See p. 57.)

They've sold a few, but the impact has been modest. And, as a member of the project puts it, "We wanted to get our ideas for education out to the world."

So they decided to build a terminal. But it grew, as terminal designs will. It is now the GTI 2500.

Remember the tortoise and the hare? This is the hairiest tortoise on four wheels. First deliveries this fall.

And here's what you get for your five thousand dollars--

THE KILLER CHELONIAN—

a 16-bit computer like none you ever saw.

- 8 working registers, in addition to PC.
- 32 scratchpad registers (70 nanosecond).
- 250-nanosecond I/O.
- 4K of main memory, 250 nanosecond. (Expandable, of course.)
- 1K OF MICROPROGRAM MEMORY, 40 NANOSECOND, DYNAMICALLY ALTERABLE. (Expandable to 4K.) Likewise 16 bits.

(YOU PROGRAM YOUR OWN INSTRUCTION-SET.)

- Cassette memory, 1 drive.
- Alphabetical display, standard video, with 8x16-dot character generator, 64 characters, DYNAMICALLY ALTERABLE. Also expandable.
- Vectoring graphic display with 2D rotation ("turtle geometry"-- lines are specified not by endpoints but by angles and length). 512x512 resolution, 1 million endpoints/sec refresh.
- Keyboard.

I asked Dan Hillis, a member of the group, about the possibility of installing the 2500 in a van. "Think of it as a recreational vehicle with the van optional," he said.

IN THE CHIPS

What makes possible the computer counterculture and everything else is, of course, the spectacular development of electronic chip technology, the techniques of shrinking great electronic circuits to almost no size. Electronic rigs that were shoebox-size ten years ago are typically now etched on chips the size of your thumbnail and sold for a few dollars, no matter what they contain.

A few years ago, the chips only contained building blocks, such as registers-- units for holding information temporarily. But now in the mid-seventies they have come to contain whole computers, or large sections of them. (The distinction between microprocessors and computers is taken up on p. 44.)

The first biggies were from Intel: the 8008 and then the 8080, a chip that has become the heart of the Altair (see p. X), as well as rival computers.

New computer chips keep coming out; people keep telling me to mention specific ones, but I can't keep track of them. The Motorola 6800 seems popular; it will soon be the heart of new computers from MITS and SPHERE (see p.W and Y). (An augmented and faster copy of the 6800 is reputedly being sold by MOS Technology for $20.) Another interesting computer chip is the PACE microprocessor from National Semiconductor, with four working registers and a ten-word stack; with 16K memory it costs $500. (The PACE is hidden in an automatic drink mixer and booze inventory controller from Electro Units Corp., San Jose, Calif. Adjusts prices to hours and can even water the drinks precisely. Claimed to make absentee ownership of bars practical.)

Because of chips, the price of computer main memory is collapsing apace. Something like a dollar a word in the sixties, it is something like like a dime a word now. But Intel now offers a storage chip holding 16K bits for $55, which is 3¢ a bit, and a friend of mine estimates that memory chips will cost 1/10 of a cent per word in 1976.

These cost collapses cause many to predict the end of disk and tape. But that's premature. While these zappier chips hold a lot for a little, their contents disappear when the lights go out. Until laser-punched tape comes along, disk and magnetic tape will be very much with us as long-term and backup storage devices.

Because of the action in chip technology, a potentially important movement in computer design may have been passed over: the "macromodules" developed at Washington University in St. Louis by Wes Clark (father of the original DEC modules), and associates.

The basic idea of the macromodule approach was to have computer subsections that were completely interpluggable. With them you can build any computer, to your own design, in a couple of days. The system exists now and it works just fine: counters, registers, memories can be attached quickly by cable.

Unfortunately, the cost is high and they haven't found a manufacturer. With chip prices falling, and chip know-how widespread, it's hard to justify charging ten or so times as much for components just because they can be plugged together faster. (Just as unfortunately, everything in the macromodule system is built on sections of twelve bits.) For this reason the St. Louis folk are having trouble getting commercial sponsorship. However, perhaps some bright hungry chip company, reading this, would like to get into the macromodule game. And presumably whittle the module down to the now-universal 8 bits.

EQUIPMENT PAGE

WEIRD AND SEXY COMPUTERS

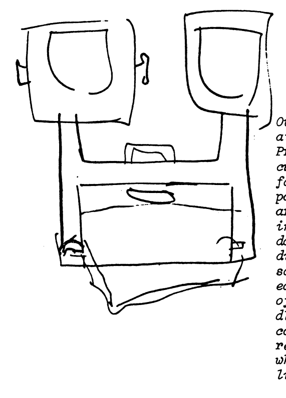
The most glamorous computers being built today all seem to be openly called by their developers' names: the Greenblatt computer, Minsky's computer, the Amdahl machine, the Cray computer.

THE AMDAHL COMPUTER

The Amdahl computer or System 470, a supercomputer of the 360 series by one of the guys who designed them originally-- see p. 41-- is now available from Amdahl Corporation, 1250 East Arques Avenue, Sunnyvale CA 94086. (They are now advertising for systems people who know the insides of OS/MVT, VS, etc.). The first 470 is up and running at NASA's Institute for Space Studies, Columbia U. But IBM is said to be readying one of their famous "knockout" machines to do it in. (Datamation, July 75, 96.)

OCTOPUTSCH

Of course you've thought that hardwired setups were for sloppy analog types of thing. But here now we have THE CHESS MACHINE, under straightfaced construction at the MIT AI Lab, which will provide HARDWIRED THREAT ANALYSIS. Yes, its advanced perceptron architecture will supposedly be capable of analyzing threats to any given position in a GRAND PARALLEL FLASH. The impact of this astonishing development on the world of Electronic Chess, or anything else, for that matter, is totally impossible to predict.



Over a very nice lunch at Roditys in Chicago, Prof. Minsky and I discussed possible styling for his computer. He particularly liked the arrangement suggested in this sketch: a fold-down keyboard and the displays sort of on poles so they could be seen easily through a crowd of bystanders. The handle would only work, of course, with the scopes removed. We'll see later what it finally looks like.

THE GREENBLATT MACHINE

Unsatisfied with the structure of normal computers, they are building at MIT's AI Lab a computer whose native language is LISP. It will have 32 bits with virtual memory, and execute LISP like a bat out of hell.

In a refreshing reversal of trends, it will be for one user at a time. "Time-sharing is an idea whose time has gone," chuckles one participant. (Project MAC, where time-sharing grew up, was there.)

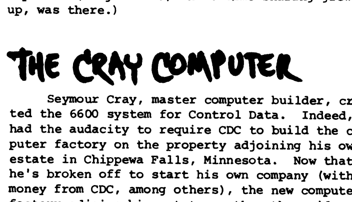
THE CRAY COMPUTER

Seymour Cray, master computer builder, created the 6600 system for Control Data. Indeed, he had the audacity to require CDC to build the computer factory on the property adjoining his own estate in Chippewa Falls, Minnesota. Now that he's broken off to start his own company (with money from CDC, among others), the new computer factory adjoins his estate on the other side. The Cray-1, another supercomputer, is nearing completion there.

AUDIO TRANSDOUCHER

Patent #3,875,932 has now been issued for How Wachspress' electronic sex machine or whatever it is (you saw it first on p. DM9). In the illustration we see it tickling a shmoo.

After you send Wachspress his fifty-buck royalty, you can either buy the kit or a prebuilt model. Concave or convex, as the poet says. (Etchings are antediluvian and waterbeds are commonplace; as an invitation, what more incisive comeuppance could be proffered?)



Speaking of Wachspress, it seems that the unusual I/O equipment offered by the Federal Screw Works (Troy, Mich.) is only a voice output device.

Surprisingly, a voice input device is now commercially available from Threshold Technology, Inc., Cinnaminson, NJ. For $10,500 you get a device that will recognize 32 spoken words, and microphones. (Each user has to train it on his 32 words, but separate vocabularies may be stored on the computer for different users or purposes. This is still some way from the fabled "talking computer"-- see pp. DM 13-14 for problems and objections-- but it's undeniably a useful step.)

CORPORATE PAGE

• • • • •

BIG BROTHER AND AUNTIE TRUST

(All in the Family?)

Well, the anti-trust trial of IBM is underway. In an awkward start, opposing lead attorneys accused each other of professional misconduct, placing both men's careers under a cloud as the fight began.

The way it comes through in the trade press, the Government seems to be pulling punches and missing the point of what its own witnesses say. A large-scale botch may be in progress. (The Computer Industry Association, or IBM-hater's club, offers transcripts of the IBM trial, as well as daily summaries. The group's headquarters are now at 1911 N. Fort Meyer Drive, Rosslyn, Virginia.)

What is the point of it all? The Justice Department is seeking to break up IBM. (According to one theory, it needs points after the ITT-Hartford business.)

There is a lot of superstition about IBM in the land. The stock market took a huge dive when the Justice Department announced it would prosecute. But why? Hesh Wiener, editor of Computer Decisions, thinks IBM will be broken up: "The Justice Department wants it, and IBM wants it, and the stockholders will make more money. They've already drawn the dotted lines."

A key question is what difference it would make. (Remember what happened after they broke up Standard Oil? Not much.) A phony breakup would simply make the different divisions into different companies, leaving the product line and the cooperation intact; a more effective split would in some way foster competition among the daughter corporations. But what way?

One of IBM's more recent tricks is to overwhelm litigators by the quantity of documents supplied, many of which are stored on computers in full-text form. To give you an idea of the humungous magnitudes involved, some figures just came up in recent litigation with Sanders Associates. Sanders is suing IBM, and recently asked IBM how many documents IBM had that were 'pertinent' to the case. The reply: "Active files, approximately 906,054,000 pages; inactive files, approximately 421,660,000 pages." (Datamation, July 75, p. 129.) An unfortunate aftermath of a suit by Control Data, in which IBM settled, was the destruction of the great indexes which had been constructed to the vast file of IBM's records; the index is gone and unavailable for this case.

To a lot of people this just seems to have to do with the size of one corporation. In the author's opinion, however, the issue is the one big usual question, the issue of freedom in our time; and that is not a matter of bigness, but the style of IBM's control. Computers should make things easier in both our work and our private lives, and should help lighten our loads and enlighten our minds, clarifying the complexities of everything. Unfortunately, IBM's method of making money has a little too much to do with creating rigid and oppressive and pointlessly complex systems, fobbing them off as "scientific," and ensnaring its customers in complications by the techniques discussed on pp. 52-56.

People should be free to use computers as they ought to be used, each in his personal style regardless of his job title, amidst rushing menus of options and clarifying screen graphics, rather than each person and office worker being locked into his own "sternly allotted sandpile," as cummings put it. And that is the problem.

RECENT IBMOGRAPHY

Nancy Foy, The Sun Never Sets on IBM. Not reviewed at press time.

William Rodgers' Think is out in paperback, with an added chapter, from the New American Library.

Datamation devoted large sections of its February and March '75 issues to material on the IBM Problem.

THE BIG LIE

(Cybercrud '75, Antitrust Division)

The backbone of IBM's defense in antitrust is this whopper:

MAJOR PREMISE. "Computers are so complicated only a company as large as IBM can put together the technical teams necessary to make them work."

THE COROLLARY. "Computers are so complicated that there's just no way to make it possible for competitors to hook up their equipment to ours."

The truth: almost anybody can make sensible computers that work and tie together sensibly. Only IBM can do it wrong and make it stick.

THIBMK

Some useful words for discussing the IBM problem. (Thanks to Computer Decisions magazine, in which some of these were first published. ©1973, 1975 Theodor H. Nelson.)

ibmology
  the study of IBM.
ibmosophy
  the wisdom of IBM; ibmosoph, one wise to IBM.
ibmperceptible
  officially noticed by IBM.
enibmatic
  puzzlingly ibmish.
ibmbroglio
  IBM software.
ibmologism
  clumsy or inappropriate term, esp. one which misspeaks itself, such as "random access" for cyclical access, "direct access device"for indirectly accessible device, and "virtual system" for real system involving virtual huge memory.
hexadecimal
  trying to put a curse on the PDP-10.
EBCDORK
  rumored forthcoming code for the Future System (Extended Binarily Coded, Decimally Organized, Rbitrary Kludge).
ibmophily
  love of IBM.
ibmolatry
  worship of IBM.
ibmecile
  someone immersed in, or ibmued with, the ibmage.
ibmopoly
  65% of the market, a great ibmposition.
ibmunity
  the safety and togetherness of IBM.
ibmpunity
  judgments against IBM, if any.
ibmoclasm
  the breaking up of IBM by the Justice Department.

On page 53 of this book I say: "I hope to be able to report in future editions of this book that IBM has moved firmly and credibly toward making its systems clear and simple to use, without requiring laborious attention to needless complications and oppressive rituals."

This has in fact occurred, and I so report.

In an earth-shaking announcement in January, IBM totally reversed the policy of its computer division for the last ten years. Yet so jaded is the press that this event was not, I think, properly recognized.

Astounding as it may be from the company that gave the world JCL and the MT/ST, in January IBM stepped into the world of easy computers, bringing out the System/32, a minicomputer for business. You can only rent it as an interactive terminal, with a program created by IBM which cannot be modified (called an Industry Application Package or IAP). But these little programs prompt users step-by-step through what they are supposed to be doing, and apprently are very clear and helpful for the naive.

This about-face is in many ways gratifying for those of us who have been advocating easy, screen-based systems for years and years. At long last it gives IBM's "legitimacy" to minicomputers for business, and it helps companies that already provide such services, such as Basic/Four.

It will be interesting to see if IBM knows how to make things simple, considering the experience they have lavished on the opposite policy. Anyway, with this move I would say that IBM has purged itself of at least 20% of its discernible evil, if this begins a real change.

A delicate problem will restrict the impact of the 32 itself, however. That is that IBM wants it used only as a gateway to its big computers; presumably, if users were allowed to program it, they'd find ways out of having to use the biggie.

WHAT WILL IBM DO NEXT?

As it happens, we know what IBM's biggest next move will be. It is something to be called the Future System (FS). FS will be a complete line of computers and communications techniques for them, but that's all we know; security is very tight. Supposedly FS exists and is running; but what is it? All we know is that its scheduled introduction has been pushed back from 1978 to sometime after 1980.

Anyway, I have asked a lot of savvy people what they thought FS was going to be, and here are some of the answers:

A completely modular line of computers and terminals with a Unibus-type architecture. (RUMOR: this would eliminate SEs, CEs and Systems Analysts. Thus the postponement.)
A micrprogrammed line of equipment, whose underware uses APL.
A totally PL/I system.
A line of equipment with ever-changing microprogrammed "fan-dance" interfaces, such that no competing manufacturer can ever find out what they are. (A charge by Herb Grosch and numerous peripheral manufacturers.)
A complete and impregnable total system for all symbolic information, which can only be keyed into through IBM terminals, processed on IBM computers, transmitted through IBM satellites, and read out through IBM terminals. (FACT: IBM has applied for a satellite.)
Totally compatible with existing 370 hardware.
Totally incompatible with existing 370 hardware, but software-convertible. (IBM makes a lot of money on adapters and conversions.)
A line of pocket-sized and portable equipment built around Magnetic Bubble Technology.
A line of easy-to-use equipment with easy-to-use interactive software. (This would suddenly eliminate hundreds of thousands of programmers, but IBM doesn't owe them anything.)
"Man, whatever it is, it'll be sick."

CYBERCRUD '75

County Fair Division

At the Dutchess County Fair this year, there was a "computer handwriting analysis" booth. You wrote your name on a card (Hollerith, natch) and this was put through a slot. A typewriter (marked "IBM") printed out the "analysis."

I wasn't there, but it was almost certainly a brazen fake. Presumably the typewriter was an ordinary Mag Card Selectric, Memory Typewriter or the like. The flathouse operator could simply choose what he wanted the printout to say by the insertion of a card (on the former) or the twist of a dial (on the latter).

Incidentally, while IBM is probably the principal employer of Dutchess County, we should not assume direct complicity.

"THE OFFICE OF THE FUTURE"

A remarkable issue of Business Week (June 30, 1975) carried a 36-page section called an "executive briefing," whatever that is, on the Office of the Future, whatever that is.

The article was actually two things spliced together: "futuristic" gab around the title, and a report on the so-called "word processing industry." Word processing, a silly IBM term, means handling text by tricky office equipment (see "Type Righter," p. 14).

IBM controls the word processing market, with such machines as the Mag Card Selectric and the abominable (in my opinion) MT/ST. As reported by Business Week, IBM's basic strategy is to tell businessmen that they have to have a centralized typing pool of specially trained typists to use these things, so the office has to be reorganized. The secretaries hate the new organization because it makes them into keypunch operators-- the peon/executive dichotomy is a traditional aspect of IBM products, it would seem-- but the whole thing is put over as Modern.

Now Xerox has come up with a competitive machine, the 800 (see Diablo, p. Y), and Business Week intones that only these two firms have the savvy and capital to succeed in competing to create the Office of the Future. Well, this is hogwash.

The big mistake IBM's competitors always seem to make is to let IBM define the problem, and then go in to try to compete on the battleground, and in the terms, that IBM has laid out. But it is not sensible to play follow the leader on slippery logs through a boobytrapped swamp. Now Xerox has stepped onto the slippery log. But the right thing would be to unmask the absurdities of the IBM game with new initiatives which they cannot possibly emulate.

The office of the future, in the opinion of the author, will have nothing to do with the silly complexities of automatic typing. It will have screens, and keyboards, and possibly a printer for outgoing letters, but possibly not. All your business information will be callable to the screen instantly. An all-embracing data structure will hold every form of information -- numerical and textual-- in a cats'-cradle of linkages; and you, the user, whatever your job title, may quickly rove your screen through the entire information-space you are entitled to see. You will have to do no programming, and indeed "programs" will never be explicitly invoked at all; they will simply take effect as you get near, in the display space, something which needs update. A display-driven information complex.

X MARKS THE SPOT

When Xerox Corporation entered the computer business a few years ago, it announced that it was going to challenge IBM head-to-head for domination of the whole broad field of Information, whatever that is. Xerox made copiers, but saw the handwriting on the drum: eventually the handling of written materials would cross over into the computing realm; few are sure in what way. (For three future directions that have been proposed, see Engelbart, pp. DM46-7, PLATO, pp. DM26-7, and Xanadu™, pp. DM56-7.)

The last July issue of Computerworld in '75, however, tolled of Xerox's abandonment of the computer field. Specifically, Xerox will stop making computers themselves, though they still will make accessories such as the hot Diablo printer (see p. Y). The news was presented in the framework of grand tragedy, the Promethean collapse of overextended ambitions. Evidently Xerox management pushed too hard in two incompatible directions-- building slowly for the eventual challenge of IBM, vs. showing profits quickly. The firm fell between the boat and the dock, joining RCA and General Electric and the other big companies that found they couldn't make it selling computers.

But Xerox is not as far out of the field as some might think.

In a secret mountain hideaway-- well, not too secret-- Xerox still has perhaps the sharpest bunch of computer rascals in the world. And they are planning way way ahead, to the time computers are practically free. If Xerox gives them their head, and doesn't cut back, the corporation will have little trouble in triumphantly returning to the field five or ten years from now, conceivably knocking IBM off its feet in the new markets of that day with a karate-like sweep.

This Place of Power is called Xerox Palo Alto Research Center, or Xerox PARC, and its atmosphere of California Mellow can mislead the unwary.

I spoke there a few years ago and found it an astonishing experience. First, there was a busy volleyball game outside when I arrived, and when I asked for the person I was going to see, the receptionperson said to pull up a beanbag and wait till he had finished playing volleyball. Later, when I addressed a group, it was in a room furnished only with a mountain of those beanbag sacks. As people came in, they would pull beanbags off the mountain and sit down on them.

So far so good: California Mellow. So I went into my rap, and everybody sat listening. I had no idea if I was getting through. Since what I try to tell people begins where technology stops-- moral precepts, as it were, for organizing ideas and systems in the world of the future (see this whole DM side)-- I'm used to people looking confused, or worried, or angry, or even walking out. There was none of that. Was I getting through? Or were they all just stoned?

I think I just sort of stopped and said, "Is everybody following this?"

There were smiles and I think someone said, "We're with you, Ted."

And they were. It was the only place I've ever spoken where the audience was on the same wavelength, going straight on into Systems Design for Future Man. Very moving.

This is obviously the place to tell you about Alan Kay and the Dynabook.

The hottest project at Xerox PARC is Alan Kay's Dynabook, formerly the Kiddy Computer. As lots of people will tell you, it's going to cost five hundred dollars, be small enough to carry around on a shoulder strap, have a built-in screen, run on batteries, and have all the books a kid wants to read from the screen stored on a cassette.

And the demos! They'll knock you out. On a color TV screen, they'll show you a wildly changing pageant of toy soldiers, photographs, beautiful patterns, all generated by the computer in real time (see "Bit Maps," p. Z). And if you're into computers, they'll show you how all this is run by the beautiful SMALLTALK language (it was previously called the Kiddy Computer, remember), which any bright child can learn and which has some awfully powerful features.

Now let's sort this all out.

There have been a lot of cons in the computer field, but this is not one of them. It's marvelously real.

So how come Xerox is leaving the computer field?

Answer: they're not exactly leaving; they're taking a break until they can sell this beauty for five hundred dollars.

What's the delay?

The Dynabook, or Kiddy Computer, is actually a PDP-10.

You're supposed to laugh. A PDP-10 is a big computer, the best. (See page 41.) A PDP-10 system costs hundreds of thousands of dollars.

But the last laugh will be Xerox's. The way computer prices are coming down, through integrated circuits ever more powerful and cheap, that PDP-10 can be sold for $500 in... (check your choice) __1978 __1979 __1980 __1981 __1982.

(Interesting anecdote: the guys at Xerox PARC asked to buy a PDP-10, but management bridled, seeing as how Xerox was in the computer business and made competitive machines. So the fellas, nothing daunted, built their own. They modestly say the parts only cost a few thousand.)

(Note: the above predictions are based, of course, on the assumption of Xerox management knowing what it's doing. Assumptions of this type in the computer field all too often turn out to be without basis. But we can hope.)

The TRAC® Language is now running time-shared, for general customers, on Computility (as mentioned on p. 21), and in a fancier version offered by Interactive Sciences Corp., 60 Brooks Drive, Braintree, Mass., 617/848-2600. Mooers has licensed the latter firm to run both his basic processor and "Advanced Developments" (rather secret) in file systems and computer control. Apparently he has some spectacular data-base stuff in there, but you won't be able to find out about it directly. Special packages are the specialty of Interactive Sciences, and with TRAC they can offer packages with both the data base stuff and other unusual capabilities. For instance, this time-sharing TRAC can itself call up other computers and sign into them, responding to messages as if it were a user.

The Computility version seems to run for about $12 an hour, the Interactive Sciences version for somewhat more-- but the latter firm is interested in selling whole packages, not user-diddling.

• • •

Mooers has recently received registration for his trademark

"THE HAPPY ROBOT."

"DEC IS GETTING LIKE IBM"

is a complaint you hear everywhere. The resemblance is certainly not in salesmanship-- ha ha-- but in the way that the standard answer to questions has now become, "I don't know, that's not my department." People feel this with a certain bitterness because so many of DEC's fans loved it for not being like IBM. It's like when Jackie Kennedy married Onassis...

News of DEC

Despite its steadfastly insipid marketing, DEC has consolidated its position at the center of the small-computer maelstrom, and the PDP-11 has been consolidated as the small and medium-sized computer of choice among sophisticates. (The PDP-11 is also attracting considerable interest as a network computer. In one curious instance, First National City Bank of New York is creating a network of 11/45s.)

NEWS OF THE 11

The PDP-11 has now become the first computer to range in size, genuinely, from the tiny to the grand. During the last six months, DEC has brought out the smallest of the line, the LSI-11, all on a board the size of a sheet of typewriter paper, for SIX HUNDRED AND FIFTY DOLLARS. That includes the full computer and 4K of volatile fast memory, as well as built-in debugger.

However, as with many announcements, this is not quite the full story. This LSI-11 is without power supply and without Unibus. Indeed, it seems that the LSI-11 happens to be the very same thing as the 11/04, demurely announced last fall, which costs $2500 with power supply and Unibus, no front panel. The announcement of the LSI-11 then takes on the appearance of a reply to the grand MITS announcement of January (see p. W). Especially when it turns out that if you want one LSI-11, it costs a thousand. ("Buying clubs" are being formed with the idea of pooling resources for the quantity price; see "Cheap Computers," p. Y.)

(Sophisticates interested in putting the LSI-11 in other equipment have been quick to notice an unusual feature: it has an empty socket in which you may insert the ROM that gives you floating point (a very cheap option). For those of us who daydream about unusual functions, such as list processing or graphics or the like, this opening is very suggestive: with access to the microprogram instructions, a different ROM could be put in for fast implementation of whatever it was you wanted-- and your program would use for your nefarious purposes the binary commands ordinarily reserved for floating point.)

(While he may not be able to deal with that, a very savvy source person for the LSI-11 is Daniel L. Lewis at DEC in Rolling Meadows, Ill.)

At the high end of the line, a big PDP-11-- the model 70-- has been unveiled, revealing a full 32-bit machine, in the hundred-thousand-dollar class, with cache memory and time-sharing. (But what of the even bigger PDP-11 model 85, rumored to be whirring its thirty-six bits unseen in the Marlboro plant under yet another operating system? Will it mean that all the other 11s have had two more bits all this time? Ah, pity that nothing can be said about that here.)

Multiple operating systems are, indeed, the bane of the PDP-11 line. Not only are there DEC's own, like RSTS, RT-11, DOS and RSX, which suffer from a lack of file compatibility and sometimes won't even run the same object code; but now there has arisen a far grander operating system, UNIX.

UNIX-- the name's suggestiveness of harem guards is deceptive-- is really the son of MULTICS (see p. 45). But it was finished in much less time. Like Multics, it's a beauty. Like Multics, it was programmed in a higher language; the language it's programmed in, however, is called simply "C". The language was created by Brian Kernighan, author of a widely-praised book which he audaciously compiled out of incorrect programming examples from other people's books on programming. Unix itself was programmed in "C" by Ken Thompson and Dennis Ritchie.

Unix is a demon. Aside from all the usual features, it allows programs the magic property of splitting. Each program can throw off copies of itself, which run independently and themselves initiate further events. This sorcerer's-apprentice structure comes mainly from a Norwegian language called SIMULA, and also appears in Alan Kay's SMALLTALK language at Xerox PARC: regrettably, there is no room to discuss these here. (For Simula, see Ole-Johan Dahl and C.A.R. Hoare, "Hierarchical Program Structures," in Dahl, Dijkstra and Hoare, Structured Programming, Academic Press.) These features effectively change the character of programming completely. For instance, to simulate a number of objects interacting, the program can spin off a copy of itself for every object, and each copy (mimicking the real-world object), can then respond to its continually-changing environment as required. In other words, this type of language means that programs behave much more like the things being simulated than they ever did before.

SIMULA costs $20,000, and, as it happens, UNIX costs $20,000 (free to non-profit organizations). Unfortunately this raises certain grave questions, since the telephone company (of which Bell Labs is a branch) is not supposed to be in the computer programming business; and those who are in the business are dismayed by the idea of such a competitor.

DEC'S OTHER COMPUTERS

Rather than throw its corporate weight entirely behind the PDP-11, DEC has carved out certain areas in which it is trying to market its 12-bit and 18-bit machines, the PDP-8 and PDP-15. The PDP-8 is being pushed for business applications, with DEC's COBOL-like language; also a very nice version of the 8 has appeared, an excellent home computer, with 8K of core, two floppy disks, keyscope, and wet-printer option; this is the "Classic," at $12,000.

The 18-bit PDP-15 line is still being marketed. Perhaps in order to save it, it is being marketed as a "medium-sized" machine, with MUMPS (DEC's data-base system), with virtual huge memory, and with hot displays.

COMPETITIVE LOOKALIKES

Imitation of DEC computers is continuing.

One firm, Intersil, has put the PDP-8 on a chip for some $300. (However, as it usually turns out, by the time you get all the parts together it costs $3000 after all. But in quantity it's another story, and the individual price will drop soon enough.)

Intersil has also intimated that they are working on a chip to simulate the PDP-11. If so, this will of course bring them smack up against the patent that seems to have knocked out the Digital Computer Controls lookalike, the Lockheed Sue (at least its direct marketing), and engendered a lawsuit against Cal Data. But that remains to be seen. (Same for Godbout's 11 lookalike, mentioned on p. Y.)

FINANCING YOUR PDP

As you may know, you can't in general just rent a computer (except from IBM), but must commit for its full purchase price, since the falling prices of computers mean it will probably have no market value in a few years. (IBM's great power stems in large part from being the only computer company big enough to rent.)

Well, good old Digital Equipment Corporation has finally gotten into the leasing business! They have started a computer leasing company, Digital Leasing, in collaboration with U.S. Leasing. They will lease DEC equipment to individuals of good credit on terms up to seven years. Current rate on a 7-year lease is 2.3 percent a month.

DEC, the Halls

A wickedly funny description of DEC's home factory, fairly accurate, can be found in a neatsy belletristic book called Travels in Computerland by Ben Ross Schneider, Jr. (Addison-Wesley, paper, $6), esp. pp. 73-5.

THE ALTAIR STORY (cont.)

But MITS took it seriously, and offered with the Altair a small but complete line of terminals, disks, printers, interfaces, and, most important, service facilities.

The firm had innovated before, notably when they brought out the first hand-held calculator several years before. Just as they correctly anticipated that demand, they foresaw this one.

They also chose unerringly the right market to begin on: electronic hobbyists and kit-builders. The kit-maker enjoys the challenge of building a machine from only a diagram and a box of parts; and to be far from a repairman holds no terrors for him, for he is the repairman.

The price drop was not as dramatic as it might seem to the general public; nor is the computer quite as cheap as it seems at first glance. Contrary to a public impression, created by IBM and a muddled press over the years, that computers are huge and cost millions of dollars, very good computers have been available lately for a couple of thousand, not counting accessories.

But the accessories present a problem. On that score, the apparent rock-bottom price of the Altair may have been misleading, especially to kit-builders. A computer itself is a limp dishrag without memory, terminals and programs-- all of which pad the cost of the package. By the time you've added 8K memory, a terminal and BASIC software to your kit-built Altair, a thousand dollars has flown ($1400 if you buy it already assembled). Then if you want the disk (and who doesn't), that's at least fifteen hundred more.

Now kit-builders just starting may not see the point of all these fripperies; they aren't used to powers like that of a full computer, so coming to realize the immensity of it all may be a gradual awakening, with many happy soldering experiences on the way. Others may be brought up short as they sense what they're getting into.

This is partly a problem of MITS' trying to reach two consumer groups at once: the kit-builder, who may have thought a computer was a fancy switchbox, and now must enter a world he doesn't know, and the computer sophisticate, who looks at the bottom line for the cost of a complete package.

Indeed, MITS' low prices aren't that low. When it comes to price, they are about 50% ahead of the conventional competition. For instance, their $5000 setup (with terminal and disk) might be taken as roughly equivalent to the DEC Classic at around $10,000 (see p. Y).

But what you usually pay for in this field is service and fringe benefits. The fundamental test, it seems to me, is whether you can come back to the company with your problems. (They even answer correspondence about their customers' computer troubles.) MITS' principal contribution is really in the thought they have given to their market, and the depth with which they are serving it. They no doubt anticipated competitors who would supply accessories and undersell them (see p. Y). But they see the advantage in this: they even give out their mailing list to competitors who sell Altair memory boards cheaper! They were not out just for a quick buck; they appear to be thoroughly committed to full-spectrum computer service.

In eight months, MITS has gone from twenty-five to a hundred employees and sold OVER FOUR THOUSAND COMPUTERS, which is something like two or three percent of the computers in America. Today, the electronic nuts; tomorrow, the world.

Bob Albrecht, caliph of counterculture computerdom, highly endorses Altair Extended BASIC. Says it's terrific.

The main service center for Altairs has been the Albuquerque factory, but the first of their regional service centers has now opened in Nashville.

An Altair assembler is running on the PLATO system (see pp. DM26-7).

MITS prices are quite reasonable. If you buy a kit for anything in the Altair line, it's generally about 25% less than the assembled and fully-checked-out version.

The basic computer kit costs $439 ($621 assembled), but ignore that; it's like a car without an engine, seats or wheel. A complete package (their "Basic 1" set), with the computer, 8K of memory, terminal and 8K BASIC language is $1391. A more high-powered system with 12K of fast memory and double floppy disk is $6650, complete with their Extended Basic. There are many separate items, plans and options; it is possible, of course, to buy a packaged system from them for as much as you want to spend.

THE ALTAIR FACTORY OUTLET

Naturally it had to be in Los Angeles. The first "computer store," it seems, is at 11656 Pico (at Barrington), West L.A., ¼ mile west of the San Diego Freeway; 213/478-3168. Hours are 2 to 8 Wednesday to Friday, 10 to 6 on Saturday and Sunday. It's called the Arrowhead Computer Company, and they stock a line of Altairs.

The BIZ MINI BIZ

Andrew J. Singer (who says he is now "consultant to a small firm of astrologers"), announces to me that

"SOMEONE HAS DONE BUSINESS PROGRAMMING RIGHT."

He has only praise for the BASIC-oriented business systems offered by BASIC-FOUR Corp., to be found in major cities. Not only does the contract spell out what you get in sparkling detail, but the manual is written in English. And Andrew himself couldn't crash the system.

(The Basic-Four setup uses a mini from Microdata Corporation. Microdata itself sells a time-sharing business-type setup called REALITY, which is highly praised by John R. Levine, another young heavy.)

Very much in the BASIC game is Wang Labs; they offer a system with 4K, a BASIC interpreter (in firmware), display and cassette for under $6000. Wang has cleverly farmed the local programming problem out to a network of software houses, each responsible to its customers for their programs.

CUSTOM AUDIO WORK

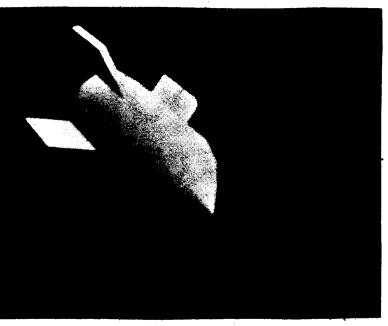
Two bright guys in New York, Norman Schwartzman and Jerry Fischer, do good custom audio work. They are also an authorized TEAC repair station. NJ Electronics, 212/265-0116, 359 W. 45 (next to the Flying Saucer News).

DREAM MACHINES

New Freedoms Through Computer Screens
— a Minority Report

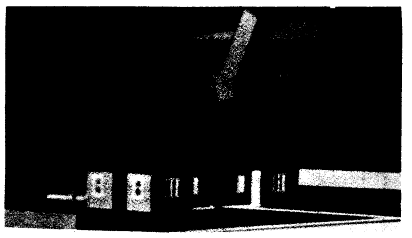
This is the flip side of Computer Lib.

GRAPHICS PAGE



The halftone system of HUMRRO, rumored on p. DM38, is real. Clever indeed: it divides the half-tone problem into two parts, one the original picturing of the scene, the other its presentation in the terminal. That means that their system permits one central image generator to send out pictures to as many terminals as desired. Unlike the Watkins Box (see p. DM37), whose half-million-dollar opulence can be poured only on a single user at once, in this system the central resource can be distributed among various users, with each one's picture changed intermittently, or poured on a single user for full animation. Currently it runs in Fortran, transmitting encoded pictures to the unusual terminals required (built around Trinitrons). But a special central processor is foreseen.

The system is called CHARGE, and Ron Swallow, its developer, is indeed a hard charger. (Software: Bill Underhill and Roger Gunwaldsen.) Swallow's game isn't movies or engineering graphics; he wants CHARGE to compete head-to-head with PLATO (see pp. DM26-7). And at the prices he's talking about-- $5000 per terminal and $150,000 for the central processor-- who knows?



UNREAL ESTATE: for relaxation, Ron works on the "dream house" he keeps inside the system.

Millions of people saw computer graphics for the first time on the PBS "Ascent of Man" series, where a screen drawing of Early Man's skull was seen to rotate and gradually change in its features. This was startling even if you know about computer graphics, since it seemed to be proceeding from complex data concerning the entire skulls and their changes.

Not so. Actually what you saw was a series of skull drawings by Peter Foldes, a Parisian artist, with the computer generating transitional drawings between them. (Indeed, though you saw Prof. Bronowski next to the screen, you did not see him next to the screen at the same time the drawings were changing-- because that had to be filmed very slowly.)

The system was created by Nestor Burtnyk and Marcelli Wein, of the National Research Council of Canada. It currently runs only on an SEL 840A. (It was also used by the National Film Board of Canada for creating Foldes' splendid film "Hunger.") They can preview by rolling through bit-map video on a moving-head disk. (See Burtnyk and Wein, "Computer Generated Key-Frame Animation," J. SMPTE, March 71, 149-53.)

What about the animated figure that talks to Joe Gariagiola before baseball games? Haha. That's a rubber puppet matted in from a black box; the guy who does the voice works the mouth.

Many unlikely individuals have stormed that heartbreak town of Hollywood, leaving sadder but wiser-- but Ivan Sutherland, dean of computer graphics? Well, having found that the movie-makers are not ready for image synthesis-- the dreamsmiths unprepared, as it were, for the Total Forge-- he is sojourning at the Rand Corporation.

A fella named Charles McCarthy, of suburban Chicago, bought the "Computer Eye" from Spatial Data Systems, and will do mail-order picture conversions. He'll convert your favorite snapshot to a printout of the same subject made of light and dark letters. If you're interested in having the actual grey-scale data for processing in your own computer, inquire.

The Möbius Group, Inc., P.O. Box 306, Winfield IL 60190.

Want a computer-controlled videocassette recorder? The model to ask for is the Sony 2850, costing (gasp) some six thousand bucks. An interface to the PDP-11 is made by CMX Systems, 635 Vaqueros, Ave., Sunnyvale CA 94086.

Incidentally, scaled-down CMX editing setups are beginning to get around. For instance, they have a small setup in the pleasant offices of DJM Film & Tape, 4 East 46, NYC: three of the above Sonys and the CMX Model 50 control setup, using a PDP-11 and keyscope. Though prices are by the job, the basic charge is $75/hour. (Note that the big CMX setup, with a disk, is the model 300.)

Video Disks, Supposedly TURN, TURN, TURN

Since the forties, there have been continual announcements that video disks-- movies you play on your TV off a record-- were right around the corner. Earlier this year they were supposedly going to be available before Christmas. Now they might be on sale, "on a limited basis," in 1976. (TV Guide, 16 Aug 75, p. 7.) Because of the grave difficulties of engineering-- inaccuracies in punching the center hole mean the track can't help being off center, for instance-- some of us are skeptical.

Two systems have been confidently announced. Philips, the firm that gave us the audio cassette, has a system that will follow the spiral track on the disk from underneath with a laser. The disk turns at 30 revolutions per second, or one turn per TV frame, so it can supposedly freeze on one frame when desired.

The other system is from RCA, which has a long history of me-too announcements, but at least two of them made it big (the 45 record and the color TV system now used in the USA), so RCA should not be dismissed out of hand. Their disk system will supposedly go at 450 rpm (7.5 revolutions/second), but they still mean to track it with a needle. The man from TV Guide says he's seen it and it works perfectly, but I would personally look for hidden wires.

(MCA, an entertainment conglomerate, has hitched up with Philips and printed a catalog of all the movies they will supposedly make available on disk for the "MCA-Philips" system-- such as Destry Rides Again for around ten bucks. This is probably just a bluff; with the price of audio records what they are, no way is a movie going to cost ten bucks. But it makes RCA look weaker, which is probably the purpose.)

MOVIES FROM YOUR COMPUTER

The prospect surprised them, but MAGI (see p. DM36) allows as how they might let you make movies on their over-the-phone movie-making setup (sketched on p. DM36). Price to capable outders, if the software meshed, would be about $50 an hour. (Six hours makes one minute of film, not counting the phone bill. Cheap if you know movie economics.)

Meanwhile, John Lowry, at Digital Video Laboratories in Toronto, has been developing high-quality video suitable for transfer to theatrical film. He and they have developed a 655-line color system-- with heavy digital enhancement (see "Picture Processing," p. DM10). I scarcely believe my notes, but I saw it, and wrote down that it was comparable to 35mm studio color. The day of "electronic cameras"-- that is, film-quality video-- may be upon us soon.

About 1972, there was announced an electronically-controlled color filter that could change to any hue in nanoseconds. That would be just what we all need for color movies from COMs-- but what happened to it?

VECTOR DISPLAYS

At the high end of things, a firm called Three Rivers Company has come in with a 3D vectoring system (competitors discussed p. DM30). Supposedly they can pack a lot more lines on the screen.

The price of the GT40 display (see p. DM21), which all in all is one of the best displays on the market, has just dropped to $6500. To disguise this price drop, DEC gives you the smaller tube and no keyboard.

And at the low end, a firm called Megatek in San Diego offers line-drawing CRT controllers for $1000 to $3000. All permit animation. You have to supply the oscilloscope. Their equipment plugs into the PDP-11 or the Nova, or in one case connects in tandem to an ASCII time-sharing terminal (!).

The 11 and Nova models work directly from BASIC; your program in Basic puts line lists in the device's buffer memory. The time-sharing model converts incoming line lists from ASCII to binary and stores them internally. 256 lines with 8-bit resolution cost $1900, $1100 and $1600 for 11, Nova and t-s respectively; 1024 lines with 10-bit resolution cost $2800, $2000 and $2500 respectively. (Nova and 11 models can be completely updated in two refresh cycles, yielding as much animation as anyone can decently expect for the price. Software is supplied to provide display output from Nova, PDP-11 or time-sharing BASIC; also t-s Fortran.)

Meanwhile, for the hands-on electronics guy, Optical Electronics, Inc. makes all kinds of rotation modules. You can build your own 3D rotation setup out of their modules for a couple of thousand; but, of course, the fancy digital I/O for high-speed refreshment is not available. An interesting capability of the OEI equipment, though, is that you can build 4D- or even 5D-rotation systems out of their modules. Hmmm.

PLATO news

Excellent manuals on the PLATO system and TUTOR language are now available from CERL, University of Illinois, Urbana.

The next generation of PLATO terminals is coming down the line. The microfiche projector is withering away, as was easily foreseeable; meantime, steps are being taken toward a more high-performance terminal, by putting a computer in it. This is being done both by Jack Stifle, who has done it with the Intel chip, and Roger Johnson, who has the panel interfaced to an 11. (11 fans please note the implication: it is possible that the interface may be marketed.)

Meanwhile, PLATO-like terminals (the model AG-60) are about $5000 from Applications Group, Inc., P.O. Box 444B, Maumee, Ohio 43537. Note that these have standard non-PLATO interfaces and standard keyboards, but the Owens-Illinois plasma panel (erroneously called Corning elsewhere in the book) blazes in all its glory.

BIT MAPS

The main development in computer graphics in the last year has been the sudden upsurge of the bit-map approach to computer display. While the approach, and equipment for it-- like the Data Disk system-- have been around for some time, the falling price of electronics, especially in the memory area, have made it abruptly the cheapest and thus the most popular type of computer display for graphics.

A "bit map" is a series of dot positions, or bits, recorded in some form of fast memory and read out in sync to a conventional scanned video system (see pp. DM6-7). The one bits stand for dots or little squares, the zeroes for nothing, and the video system brightens the corresponding zones on the screen. This method has certain disadvantages-- parts of pictures cannot be automatically distinguished or separately animated, as with subroutining display (see "The Mind's Eye," esp. p. DM23)-- but for the money it's great. Sizes given refer to the number of squares in the rectangle of the picture.

BLACK-AND-WHITE

An off-the-shelf bit-map system for the PDP-11 or the Nova is available from Intermedia Systems, 20430 Town Center Lane, Cupertino CA 95014 ($2750 or $2500 respectively). May be ganged for grey-scale or color. It's 256x256.

For the Altair, the forthcoming 8096 display (see p. Y) will have 120x120 or 240x240 bit-map graphics, for prices starting around $1000.

COLOR

Extra bit maps, plus electronics, can get you color; if you double the number of bits you can double the number of available colors on your display, ad infinitum.

On the small side, 64x64 color wll shortly be available for the Altair from the Digital Group, Denver. A 128x128 color bit-map system for the 11 has just been announced by DEC (for "nuclear medicine" of all things-- but they will part with it to anybody for 8 or 10 thousand (not yet fixed)). They stress that this will be the first of a modular series of bit-map displays, with plugins for different degrees of resolution and different character generators.

Ramtek and Comtal both make 256x256 bit-map systems, priced in the $16,000 area.

Above this resolution special TV systems tend to be necessary. Both Ramtek and Comtal make very expensive systems for the purpose, using solid-state and disk respectively.

You may or may not have heard of the Advent TV projector, the most glorious TV thing there is. It costs $3500 and projects a four-foot picture in the best TV color you can find. A lot of guys are bit-mapping to it.

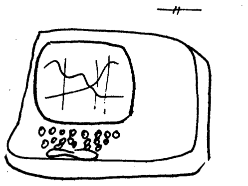
At MIT they've got bit-map color on the Advent at better than 400x500 resolution. (An option planned for the Flying Turtle (see p. Y) will allow its core memory to be used with the Advent as a bit-map display refresher.) At Comtal they're going for 1000x1000 on the Advent, rejiggering the electronics from scratch.

The most spectacular demonstration of bit-map color so far has no doubt been the film done by Dick Shoup et al. at Xerox PARC (see p. X), showing the super animation that's possible when big-computer resources are given over to bit-map animation. Their system is 600x800.

YOUR BIG DISPLAY PANELS

All those scoreboards and wisecracking light-grids, now that they are computer-controlled, raise all kinds of possibilities for non-frame animation. The big ones cost in the millions; a small one for shopping centers costs a hundred grand (Millenium Info Systems, Santa Clara CA).

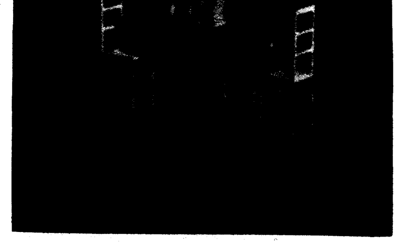
Within a year or so, though, you ought to be able to get a nice animated display-panel of some sort for the side of your van, assuming you've got the computer inside.



2D FROODY

A surprise something-or-other from DEC, the VT55, represents a breakthrough of some sort. But what were they thinking of?

"Graphic capability" has been added to an ordinary upper-case keyscope. Specifically, the ability to make two graphs, i.e., two wiggly lines (no more) somewhere between the left and right sides of the screen. You can also shade in under them, and add coordinate grids. It's $2500, and obviously great if you're bonkers for 2D graphs.



GUESS WHO'S COMING TO DINNER

IBM, which did not take part in its development, is sponsoring a $100,000 CHARGE installation at the University of Waterloo, in Canada.

NOTE FROM THE AUTHOR & PUBLISHER

Labor Day, 1975

MORE THANKS

In banging together this volume originally, I omitted thanking Hesh Wiener, brazen & brash young old-fashioned new editor of Computer Decisions, who has changed that publication from stolid to peppery.

Thanks also to my good friend Robert W. Fiddler, Esq., patent attorney and still an ex-philosophy professor at heart, for many delightful and witty conversations on problems of patent, copyright and the vagaries of intellectual property. Any harebrained ideas on these topics expressed here, however, are almost assuredly my own.

For much of the information in this supplement I am grateful to Bob Albrecht of PCC, mentioned here and there.

Finally, special thanks to Commander Hugo McCauley, better known to you as Hugo's Book Service, for his yeoman performance in shipping out the books-- not to mention carrying them up and down stairs, typing the mailing labels, checking for bad ones, and sending out all those notes of apology when we were out of books again and again and again. And to long-suffering Lois and Megan McCauley, my especial gratitude.

WHATEVER

The sea-to-shining-sea Nelson Empire now consists of a lot of unsold books, a 1K Altair and a second pair of shoes. My scheme for taking on Apprentice Generalists may have to wait awhile. So may Computer Lib, the film. But just wait.

Speaking of which, what about this book, hey, now?

Eventually there will be a new edition. Yes, the type is horrendously small, and that will have to be fixed. But that involves new negatives for every page, an expenditure of thousands of dollars, and some reconsideration of how this should all be set up.

There have been several interesting plans. One was to split the contents of this book into three books, add material, enlarge the type and have them each this size and price. Tentative titles were Computer Lib /Dream Machines, Computers Arise!/Computers Arouse!, and Guerrilla Computing/Electronic Monkeyshines. Sample cover, for Guerrilla Computing: King Kong climbing the front panel of a 370 holding Patty Hearst. (I also daydreamed about putting out a 10-volume encyclopedia in the same format, embracing psychology/sociology, biology/evolutionary strategy, history (as strategy)/more history (as mood and feeling), revolution versus continuity (a two-sided position paper)... the Gem-Maniacal Encyclopedia^{tm}. But reason has prevailed, and such forays have been postponed indefinitely.

The present plan is for Computer Lib to be rewritten and reset in bigger type, at least 256 pages, with at least 8 color pages and color cover. (We're talking about fall '76 or later.) Price will have to be $15. If you think that's a ripoff you can still get this one. (A number of people have complained to me about the $7 price tag of this volume. Have they ever bought other books?) Later I would like to put out an anthology of my favorite articles in the field, using the Computers Arise!/Computers Arouse! title and format, and with some good 3D if possible. In any case, I want to stay in the publishing game; I haven't had so much fun in years. Other projected volumes include The Inner Beyond, by Sheila McKenzie; Dirty Driving and the Strategy of Traffic by "Driver Ed;" and The Nelson Computer Glossary. Soon I hope to be able to typeset from my own computer, and possibly to share this facility.

This has been a most interesting year. I have been pleased to meet, and otherwise enjoy, the variety of clever, charming and/or lubricous persons who have sought me out since the book first appeared; as well as all the speaking engagements, soirees and whatnot.

I am delighted to receive relevant material and communications of any kind, although problems of time, disorganization and mood often preclude a Personal Type Reply.

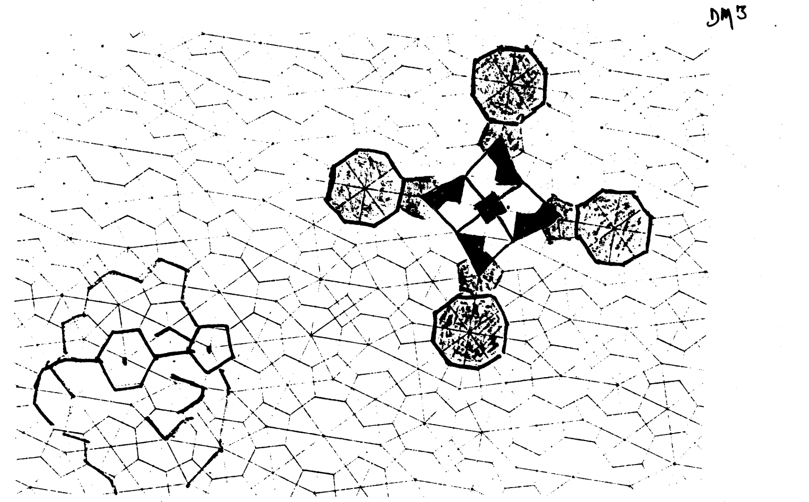
It has been a real lift for my morale to share some of these ideas and enthusiasms with a wider public at last. It is you, finally, who have to care; and I am very glad you do.

As to the most important matters, there is a news blackout for the indefinite future. Please stand by.

Next year in Xanadu.

TN

This book (both sides) is based in part on my talks at or before the American Chemical Society, the American Documentation Institute, the American Management Association, the Associated Press, the Association for Computing Machinery, the Central Intelligence Agency, the Institute of Electrical and Electronics Engineers, the Printing and Publishing Association, the Rand Corporation, the Society for Information Display, the Society of Motion Picture and Television Engineers, TIME Incorporated, Union Theological Seminary (the Auburn lectures), Xerox Palo Alto Research Center, and various art schools, colleges, universities and Joint Computer Conferences.



ACKNOWLEDGMENTS

Everybody at Chicago Circle Campus has been very sporting about this project. I am grateful not only for the encouragement and assistance of various individuals (especially Joseph I. Lipson, David C. Miller and Samuel Schrage), but for the atmosphere of support which has made this possible. My thanks to the Department of Art and the Office of Instructional Resources Development for freeing me from teaching duties, to the Computer Center and the Department of Chemistry for letting me use pictures of their equipment, and everybody for their encouragement.

I would like to thank the Walt Disney organization for their permission to depict their wonderful characters, and everyone else who furnished materials and permissions for the things herein.

Thanks also to those who looked over some of the material, especially Herbert Grosch of Computerworld, Dan McGurk of the Computer Industry Association, and William Rodgers.

I am particularly grateful to the many who have explained computers to me over the years, especially Dave Denniston, Robert Fenichel, Andrew J. Singer, John R. Levine.

My thanks to Tom Barnard for some of the early typing, and for the Porta-Xan.

I am grateful to Computer Decisions magazine for their good will, and help in researching computer image synthesis.

My roommate Tom DeFanti, mentioned elsewhere in this book, has been considerate beyond the call of duty in giving over all the first-floor rooms of our house to this project for six months.

My thanks finally to the many others whose good will has kept me going, in particular my former wife and eternal friend, Deborah Stone Nelson.

Special greetings to my friend and neighbor, Mrs. John R. Neill: I hope you enjoy the uses which your husband's illustrations of Tik-Tok the Machine Man have found here.

Lastly, for her contributions to morale (and for not footprinting the pasteups), let's have a warm hand for Pooky the Wonder Dog.

The occasional Oz illustrations are all by John R. Neill, from various out-of-copyright Oz books by L. Frank Baum, especially Ozma of Oz and Tik-Tok of Oz. Tik-Tok, the Machine Man, is the figure to whom occasional allegorical significance is attached here by juxtaposition.

The Oz picture in this spread is from The Patchwork Girl of Oz.

Thought you might wonder.

ARE WE MATERIALISTIC?

Persons of sagacity have been saying for some time that we are materialistic.

In an important sense this is not so.

The machines, and toys, and involvements we buy into, are in but a small proportion of cases owned simply as scores, for their cost as consumption symbols.

Rather, we buy things that REPRESENT IDEALS, hoping ourselves to partake of some abstraction or image-- the Playboy man, the Smart Businessman, the Clever Homemaker.

Each product tries to tell us it is the keystone of a way of life, and then, at least at that moment of purchase, we step into, we embrace that way of life, covering ourselves with the feeling, the aura, the magic we saw in the commercial.

This is not materialism. It is wishful grasping at miasma. (Following sentence optional.) It is communion, with the object seized simply the Objective Correlative of a hoped-for transsubstantiation. (Sorry.) It's a seeking, not to possess, but to belong.

GREAT AMERICAN MACHINE-DREAMERS

D.W. GRIFFITH-- took the movie-box and created the photoplay, no longer a twisted stage production.

WALT DISNEY-- created a hypnotic pantheon of kindly and innocent semi-animals, sentimentally universal, generally acceptable.

JOHN W. CAMPBELL-- as author and then editor of Astounding, turned American science-fiction from the Buck Rogers space opera to the human story, built around thought-out premises and structures.

IVAN SUTHERLAND-- programmed and systematized a computer setup for helping people think and work with deeply-structured pictorial information. (See p.DM 23.)

DOUG ENGELBART-- foresaw the use of computer screens as a way of expanding the mind, and over the last decade and a half has brought about just that.

And more, and on.

ANOTHER QUICKIE

Compare Alice, when she gets to Wonderland ("Deary me! Curioser and curioser!") with Dorothy Gale, transported to Oz ("How do I get back to Kansas?!!!") Fantasy ties in with everything, including American git-out-n-do-it.

THE GREAT AMERICAN DREAMER

I already said on the other side that the computer is a Rorschach, and you make of it some wild reflection of what you are yourself. There is more to it than that.

America is the land where the machine is an intimate part of our fantasy life.

Germans are too literal, they can get off on well-oiled cogs. The French are too vague. (I've noticed that German science-fiction magazines had covers of machines and planets; French science-fiction magazines, of dragons and people with wings. Our science-fiction covers show people with machines. Intimately, emotionally.) German fantasy is icy and impersonal, French fantasy too personal, and American fantasy is splat in the middle, uniting both: man and machine, means and ends, emotion and details.

Men always longed to fly, but it was here that they first did. This is the land of the MOVIE, a fantasy fabricated with endless difficulty using various kinds of equipment.

The mad tinkerer is a fabled character in our fiction.

This is the land of the kandy kolor hot rod, the Hell's Angel chopper, the drive-in movie. And the wild hot-rod, in fact, is just the flip side of the deep-carpeted Cadillac: each is a fantasy, an extension of its owner's image of himself in the world.

Thus it was not an historical accident, but utterly predetermined, that in the hands of Americans the computer would become a way of realizing every conceivable wild fantasy that was dear to them.

This is perfectly all right. This is as it should be. This is the best part of our culture. Not "Let a hundred flowers bloom," but "Let a hundred gizmos clank." This has sped immeasurably the imaginative development of many different things we might want. I try here fairly to explain a few differences among them.

There is just one problem with all this. Now that all these things exist, or come nearer to existing, which ones will other people want? What will it be possible for everyone to have? And how can we tie all these things together?

(Note: this thesis is being advanced only half-seriously. There have been a number of exactly-dreamful Frenchmen, and for this three-nationality split to be really true, they would all have to have come from Alsace, next to Germany: Jules Verne, Daguerre, the brothers Montgolfier, the brothers Lumière, to name a few.)

CONTENTS of Dream Machines.

2 DREAM MACHINES
4 APPARATUSES OF APPARITION
6 VIDEO
6 LIGHTNING IN A BOTTLE: THE CATHODE-RAY TUBE
8 HOLOGRAPHY
8 Sandin's Image Processor
9 BODY ELECTRONICS
10 PICTURE PROCESSING
11 AUDIO & COMPUTERS
12 THREE COMPUTER DREAMS:
12 AI (artificial intelligence)
15 IR (information retrieval)
15 CAI (computer-assisted instruction)
16 "No More Teachers' Dirty Looks."
20 THE MIND'S EYE (computer display)
24 COMPUTER MOVIES
26 PLATO
28 "Laws of the Universe Hyper-Comics"
30 THE MIND'S EYE MORE: 3D LINE SYSTEMS
31 DeFanti's Coup de GRASS
32 HALFTONE IMAGE SYNTHESIS
1. Polygon Systems
34 2. Shades of Reality (nicer greys)
37 3. Hardening of the Artistries (special hardware)
39 4. Computer Image Corp.
40 THE MIND'S EYE MORE: n Dimensions
41 The Circle Graphics Habitat
42 The Tissue of Thought
42 How to Learn Anything
43 On Writing
43 The Heritage
44 HYPERMEDIA, HYPERTEXTS
46 Engelbart
48 FANTICS
52 THINKERTOYS
56 XANADU
58 WHAT NELSON IS REALLY SAYING
59 FLIP OUT

perhaps it is time

THE LEGEND OF HYPER-MAN

In the fantasies of their subjects, which they feel are the precursors of new artistic images that will in turn actualize themselves as another form of being, Masters and Houston see a new hero figure constantly recurring. This new hero is not the old "hero of a thousand faces," the individualist who suffers, dies, and is reborn, slaughtering and conquering along the way. Instead, he is Protean, capable of infinite changes in appearance and style, a magician, a Balthazar bringing gifts. He ruptures categories and confuses the senses, and in doing so he holds out the promise of fusion on a higher level.

If such a hero were to become the model for the approaching age, he would probably not be the founder of a mass movement or the god of a new religion. He would be more elusive, more changeful than his predecessors. He would be a sorcerer who treats the external world and the internal world on equal terms, giving spirit to the former and flesh to the latter. He would be a master of paradox and a player of games, speaking a new language. His one prayer might be the lines of Blake:

May God us keep
From single vision
And Newton's sleep.

-- *Kenneth Cavander, "Voyage of the Psychenauts." Harper's, Jan 74, p. 74.*

OUT THE DOOR IN '74

I have wanted to write an introduction to computers, and a separate book on Fantics, for years. But the idea of binding them back-to-back in a Whole Earth format, with lots of mischievous Enrichment material, didn't hit me till Jan 73. I have tried to add all the stimulating and exhilarating stuff I could find, especially personalizations, as on the other side; computers are deeply personal machines, contrary to legend, and so are showing-systems. I regret having to throw so many of my concerns into comic relief, but I hope that some readers will sense the seriousness below.

The final inspiration for this book came from something called the Domebook, that tells you straightforwardly how to make Geodesic Domes. And of course I'm blatantly imitating, in a way, the wonderful Whole Earth Catalog of Stewart Brand. As I think back, though, the tone also comes in part from Pete Seeger's wonderful banjo book, and Tom McCahill's automobile reviews in Mechanix Illustrated. As to the last aspect, that of taking my case to the public because the experts won't listen, the only precedent I can think of is Maj. Alexander de Seversky's Victory Through Air Power, telling the country how he thought we should win World War II.

This project, simple in principle, has been infinitely bothersome. Self-publication was necessary because no publisher could have comprehended the concept of this book; I heartily recommend Bill Henderson (ed.)'s The Publish-It-Yourself Handbook, $4 from The Pushcart Book Press, Box 845, Yonkers NY 10701.

The present product is not the book I had meant to write. Most is first-draft; how the sentences do run on. (Believe it or not, I do not like underlining things-- a first-draft expedient.) Fact-checking and bibliographies had to be largely abandoned. Better planning could have increased type size; and so on. Half the manuscript, and the glossary, had to be kicked aside; including sections on movies, "multi-media," microfilm, training simulators, augmented stage productions of the future, and goodness knows what. Sorry for all that.

... WITH A LITTLE HELP FROM MY FRIENDS

This project could never have been completed without the dedicated and extraordinary efforts of my wise and warm friends Sheila McKenzie and Wade Freeman, both faculty members at Circle, who have my deepest gratitude. They gave months and weeks of their good time to the tedious aspects of this project (which I continuously underestimated.) I hope it has been worth their work as well as my own. Ms. McKenzie, whose concern for intelligent change in education drove her to boundless efforts on this project, has also my deepest admiration.

The sad thing about it all is that 90% of these efforts are unnecessary. A decent computer text system (of which only a couple exist as yet) would have obviated all the finding-and-retyping problems. I feel deeply for everyone who has trouble writing by conventional means, and who wouldn't if only decent systems were available.

Affectionately Dedicated to my beloved grandfather, Theodor Holm.

NT

APPARATUSES OF APPARITION

It seems different companies are all the time introducing wonderful new devices that will revolutionize, uh, whatever it is we do with, uh, information and stuff. Things you'll attach to your TV to get highbrow programs or dirty movies. Microfilm devices that will shrink the contents of the Vatican Library to a dot on your glasses. Goggles that show you holographic color mòvies. A pince-nez that lets you see the future. And so on.

Reading Popular Mechanics or the Saturday review of patents in the New York Times, you get the idea of Something Big, New and Wonderful About to Happen, so we'll all have access to anything, anytime, anywhere.

But it's been that way for decades, and with certain exceptions hasn't happened yet.

Here are some things that have caught on, and are mostly familiar to us all.

Book. Newspaper. Magazine. Radio (AM). Phonograph record (78). Tape recorder, ¼". Black-and-white television. Radio (FM). Phonograph record (33). Phonograph record (45). Color television. Tape cartridge (¼"). Tape cassette (Philips, ca. 1/8"). Stereo records and tapes. Oh yeah, and movies: 35mm, 16mm, 8mm, Super 8mm. Carousel projectors. View-master stereo viewers.

Here are some things in the process of catching on (and not assured of success): Quadrophonic sound. Dolby. Chromium dioxide tape emulsion. Super 16 movie format.

But for everything that did catch on, dozens didn't. Some examples: 12-inch 45 rpm records. 11.5 millimeter movies. RCA's ¼-inch tape cartridge, which became a model for the much smaller Philips. Wire recorders.

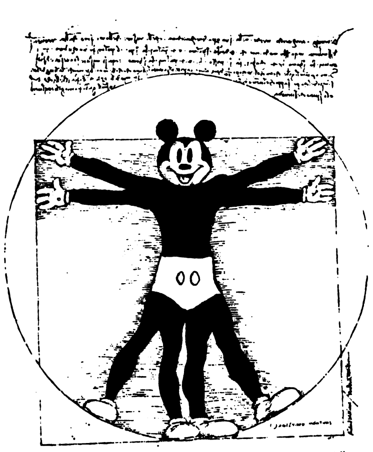
Then there are the things that caught on for awhile and went away. Stereopticons (and their beautiful descendant, the Tru-Vue, which I loved as a kid). Cylindrical recordings. Piano rolls. And so on.

Then there are the video recording systems. CBS' EVR died before it got anywhere. RCA's SelectaVision isn't out yet. 2-inch quad is standard in the studios, ½-inch Porta-Pak is standard among the Video Freaks, and it looks like Sony's 3/4" cartridge will win as the main sales and storage medium. (The Philips system here looks as though it won't make it, and 1-inch is dubious.) But what's this we hear about video disks (twenty-five years after they announced Phonevision. Ah, well.)?

The thing is, so many of these things seem to sound alike. They all mention "information retrieval," education, technology, possibly "the information explosion" and "the knowledge industry." Press releases or effusive newspaper articles may use phrases like "space-age," "futuristic," "McLuhanesque" or even "Orwellian" (though few people who use that word seem to know what Orwell stood for; see p.DM59).

And the intimidating company names! Outfits with names like General Learning, Inc., or Synergistic Cybernetics, Inc., or even Communications | Research | Machines, Inc. Surely such people must know what they are doing, to use such scientific-sounding phrases as these!

Then there are the business magazines. In the late sixties they were talking about "The Knowledge Industry" (a fiction, it turned out, of an economist's lumping a lot of things together oddly). Now they talk about the Cable TV outfits and the Video Cartridge outfits as though they're the cat's pajamas.



Emblem of 2d International Animation Film Festival in New York, Jan 74. © Walt Disney Productions.

THERE'S SNOW BUSINESS LIKE SHOW BUSINESS—

You Can't Tell the Experts Without They Program You
(Cf. "Calling a Spade a Spade, p. 12)

BABEL'S IN TOYLAND

Guy's Background	Tell-Tale Phrases & Jargumentation
Television:	"Media" (meaning television);
1. Video freaks	"Software" (meaning videotapes).
2. Network People	"Programming" (meaning competitive scheduling); "Software" (meaning fixed-length TV shows).
3. Cable Operators	Head end, upstream & downstream, back-channel, "interactive TV" (meaning any form of interactive computer system they can get in on).
Math/Engineering	Information theory, channel capacity, bandwidth, feedback, anything complex and irrelevant.
Display Engineering	Full duplex, echoplex, aspect ratio, scroll, cursor; "information transfer" (meaning telling or teaching); "data delivery" (act thereof).
Programmed Instruction, Computer-Assisted Instruction	"Software" (meaning sequential or branching tell-&-test materials); "Programming" (creating these); reinforcement schedules (meaning presentational order); "inputs" (meaning ideas and information); "feedback" (meaning replies); "simulations" (meaning pictures or events a user can influence).
Publishing	"Software" (meaning books).
Advertising, Public Relations, Marketing	"Demographics" (meaning factions); campaign strategy (meaning how you hit a market); "penetration" (meaning extent to which your stuff catches on); "Programming" (meaning anything whatever).
Artificial Intelligence	Anything mathematical; theorems, discriminators, neural nets; "programming" (meaning setting up anything very complicated and incomprehensible).
McLuhanatic	Global Village, mosaic, surround; "Programming" (meaning psychological indoctrination); anybody else's terms, dynamically infused with new senses.
Nelsonian	Medium (meaning stabilized presentational context); Writing and Creation (meaning thoughtful production of something presentable, whether sequential or not, in a medium); "Programming" (meaning giving exact instructions to a computer); media integrity, inventions & conventions; hypertext, thinkertoy, fantics.

Having spent some considerable time around and among these areas, I have developed considerable cynicism and a bad case of the giggles. Originally it all seemed to fit together and to be leading somewhere, but talking to people at all levels, and either giving advice or trying to interpret the advice of others, I am convinced that what we have here in this whole audio-visual-presentational whizbang field is nothing less than a very high order of collective insanity. The strange way companies adopt and drop various product lines, and verbalize what they think they are doing, seem to me a combination of lemmingism and a willingness to follow any Authority in an expensive suit. I have talked to enough vice-presidents and presidents of computer companies, publishing companies, networks, media outfits and so on, to be totally certain that they have no special knowledge or unusual basis of information; yet these people's remarks, as amplified through the business reporters, send the whole nation a-dithering. There are times I think everybody in Media is either deluded, misguided, lying or crazy.

THREE CRUCIAL POINTS.

1. SYSTEMS "IN THE HOME."

The emphasis has changed from trying to sell snazzy systems to the schools (which don't have the money) to the home. This in turn has convinced most people that the new systems have to be very limited, like jimmied-up TV sets. (We easily lose track of the fact that you can have anything "in the home" if you want to pay for it; and an economy in which Marantzes and snowmobiles have caught on big indicates that some people are going to be willing to pay for really hot stuff.)

2. CATCHING ON.

The key question is not how good a system is in the abstract, but whether it will catch on. (Obviously if we're public-spirited we want the best systems to catch on, of course.)

This matter of Catching On is a fickle and crucial business.

According to one anecdote, Mr. Bell couldn't interest anyone in his invention, which he was showing at some trade fair. Then who should come by but the Emperor of Brazil (!), who was about to leave with his retinue of advisers. "What is that?" asked the Emperor of Brazil. "Nothing to bother with," they said, and tried to rush him by, but he stopped and loved it, and ordered the first pair of telephones sold. This made the headlines, and the sale of telephones began.

Another anecdote. It is legendary that inventors overvalue their own work. Yet after Thomas Edison had invented the kinematograph, or "moving picture," a device you looked into turning a crank, he declined to build a projector for it, saying that the novelty would wear off. Obviously he did't quite see what "catching on" would mean here.

Wonderful Systems That Were Gonna Be

WHERE ARE THE SHOWS OF YESTERYEAR?

I once read a mind-blowing review article in Films in Review, early sixties I think, on schemes to make three-dimensional movies before 1930. There were dozens.

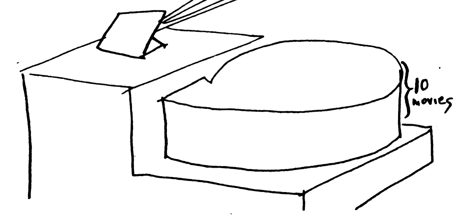
Then there was that multiscreen film Napoleon -- a legend-- done in the nineteen-twenties. (That one really existed.)

Phonevision, about 1947 or so, was going to store a half-hour movie on a 12-inch disk. Did they get the idea from the LP? Did they really think they could do it?

The German photo-gizmo, around 1950: a special camera that supposedly created a sculpture of what it was pointed at. (But how did it know what was behind things?)

A weird lens around 1950-- I think it was depicted as having a blue center and a red periphery, like a fifties hoodlum tail-light-- that was somehow going to find "residual traces" of color in black-and-white pictures, and make 'em into color, zowie, just by copying them.

Then there was the Panacolor Cartridge. During the Days of Madness-- 1968, I think it was -- a rather good little movie gadget was being pushed by a firm called Panacolor. It had ten parallel movie and audio tracks, I believe, on a 70mm strip. The prototypes were built by Zeiss.



Their idea was that this was a compact movie projector. I kept trying to persuade the company's president that they had inadvertently designed a splendid device for branching movies (see "Hyperfilms," p.DM44).

Exercise for the reader: map out properties of the branching and expository structures implicit in such a device. (It's one-directional. Gotta rewind when you get to the end. But you can jump between tracks when it seems appropriate.)

Anyway, it's gone now.

The Great Robert Crumb. (From Zap Comix #0.)

"In the news*
there is no truth**;
and in the truth**,
there is no news*."

— Modern Russian proverb.

* Izvestia.
** Pravda.

HARDWARE, SOFTWARE AND WHATNOT (reprise)

Among the many odd things that have resulted from the collision of computer people with educators, publishers and others has been the respectful imitation of computer ways by those who didn't quite understand them. Again, the cargo cult.*

The most dismal of these practices has been the adoption of the term "software" for any intellectual or artistic property.** This wholly loses the distinction, made on the other side of the book, between:

hardware (programmable equipment)

software (programs, detailed plans of operation that the hardware carries out)

contents or data (material which is worked on by, moved in or presented by the hardware under control of the software)

In other words, hardware and software together make an environment; data or contents move and appear in that environment.

The publishing-and-picturefolk have missed this distinction entirely. Not realizing that their productions are the contents (material, matter, data, stuff, message...) that come and go in the prefabricated hardware-software environments, they have mushed this together into a state of self-feeding confusion.

(The matter has not been helped by the computer-assisted instruction people-- see p. DM 15 -- whose branching productions seemed to them enough like computer programs to be called "software.")

* Primitives exposed to "civilized" man imitate his ways ridiculously in religious rituals, hoping for the shipments of canned goods, etc. that his behavior seems to bring down from parts unknown.

ON EX-SPURT-TEASE

** "Mere corroborative detail, to enhance an otherwise uninteresting narrative..."

Pooh-Bah,
Lord High
Everything
Else

3. STANDARDIZATION

In order for something to Catch On, it has to be standardized. Unfortunately, there is motivation for different companies to make their own little changes in order to restrict users to its own products. The best example of how to avoid this: Philips patented its audio cartridge to the teeth, but then granted everybody free use of the patent provided they adhered to the exact standardization. The result has been the system's spectacular success, and Philips, rather than dominating a small market, has a share of a far larger market, and hence makes more money. That's a virtue-rewarded kind of story.

The other problem with standardization, though, is that we tend to standardize too soon. We standardized on AM radio, even though FM would probably have been better. (One Major Armstrong, a great figure in the development of radio, committed suicide when nobody would accept FM. If he could only have heard our FM of today, he might have said "Oh, nuts," and lived.)

Another example. When they designed the Touch-Tone phone pad, the Bell people evidently saw no reason to have it match the adding machine panel, so they put "1" in the upper left rather than the lower left. Now there are lots of people who use both arrangements, every day, and at least one of them curses the designers' lack of consideration.

Another interesting example of Catching On: during the early sixties, it was fun being at places where they were just getting Xerox copiers for the first time. Everyone would argue that nobody needed a copier. Then, grudgingly, one would be ordered. The first month's use invariably would exceed the estimate for the first year, and go up and up from there.

The worst aspect of the confusion among the corporations is that certain deficiencies and crudities of vision slip into the mix. Unless our new media and their exact ramifications and concomitants are planned with the greatest care, everybody stands to lose. We must understand the detailed properties of media. (The first question to ask, when somebody is showing you the Latest and Greatest, is: "What are the properties and qualities of the medium?" The followup questions come easily with experience: How often do you have to change it, what are the branching options, what part could somebody accidentally put in backwards, are there distracting complications? etc.)

I am unpersuaded by McLuhan. His insights are remarkable, yet suspicious: he supposes that electronic media are all the same. How can this be? Here we may now decide what electronic media we want in the future-- and this decision, I would say, is one of the most important we have to face.

The engineers seem to be quite the opposite of McLuhan: somehow to them it's always a multiple-choice, multi-engineering problem, different every time; "this technique is good for A, that technique is good for B." But the net effect is the same: "electronic media are generally the same." I would claim that the're all different, all ten million of them (TV being only one electronic medium out of the lot), and the differences matter very very much, and only a few can catch on. So it matters very much which. Some are great, some are lousy, some are subtly bad, having a locked-in information structure, built deep-down into the system. (Example: the fixed "query modes" built into some systems.)

One last point. Everybody only has a 24-hour day. Most people, if they increase consumption of one medium (like magazines or books) will cut down on another (like TV). This drastically reduces the sorts of growth some people have been expecting. Except, now, if we can begin to replace some of the inane paper-shuffling and paper-losing of the business world, and replace the creepy activities of the school (as now generally constituted) with a more golden use of time and mind. Read on.

THANATOPSYS

A self-employed repairman of mobile homes named Donald Wells has invented a solar-powered tombstone that can show movies and still pictures of the departed, along with appropriate organ music and any last words or eulogies selected by the deceased.

The device is activated by a remote control device carried by a visitor to the gravesite. The movies would be shown on a twelve-inch screen mounted next to the epitaph.

"You could also have pictures of Christ ascending to heaven or Christ on the cross, whatever you want," says Wells. "It adds a whole new dimension to going to the cemetery...."

Cleveland Plain Dealer
(Quoted in National Lampoon True Facts, May 74, 10.)

"The Emperor has no clothes on!"

Small Boy
(name withheld)

Last year I actually heard a phone company lecturer say that in the future we will have "Instant Access to Anything, Anytime, Anywhere."

What they're pushing is Picturephone, which it seems to me is unnecessary, wasteful and generally unfeasible.

(See: Robert J. Robinson, "Picturephone-- Who Needs It?", Datamation 15 Nov 71, 152.)

ON USING MEDIA

In any medium-- written, visual, filmic or whatever-- you generate instantaneously an atmosphere, a patina, a miasma of style, involvement, personality (perhaps implicit), outlook, portent. Consider--

The complacency of the Sulzbergers' New York Times--
The cynicism and mischief of Krassner's Realist--
The perkiness and sense of freedom of "Sesame Street"--
The personalized, focussed foreboding of Orson Welles films; as distinct from the impersonalized, focussed foreboding of Hitchcock--

Next to this matter of mood, all else pales: the actual constraints and structures of media, the expositions and complications of particular cognitive works and presentations within media, are as nothing.

"MEDIA" IN THE CLASSROOM

Time after time, the educational establishment has thought some great revolution would come through getting new kinds of equipment into the classroom.

First it was movies. More recently it's been "audio-visual" stuff, teaching machines, film loops and computer-assisted instruction.

In no cases have the enthusiasts for these systems seen how the equipment would fit into conventional education-- or, more likely, screw the teacher up. Teachers are embarrassed and flustered when they have to monkey with equipment in addition to everything else, and fitting the available canned materials into their lesson plans doesn't work out well, either.

The only real possibilities for change lie in systems that will change the instructor's position from a manager to a helper. Many teachers will like this, many will not.

PAY CAREFUL ATTENTION

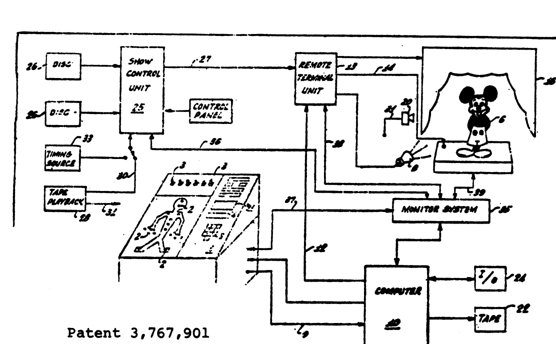
when somebody shows you an electronic or other presentational system, device or whatever.

A certain kind of slight-of-hand goes on. It's very easy to get fooled. They may show you one thing and persuade you you've seen another.

And if you're canny enough to ask about a feature you haven't seen they'll always say,

"WE'RE WORKING ON IT."

It's only dishonest if they say, "It'll be ready next month."



Patent 3,767,901 is for the Disney Audio-Animatronics system, which now basically consists of the manipulation of rubber puppets by minicomputer, through cables and puffs of air.
© Walt Disney Productions.

VIDEO. The happy medium? some mutterings

Would you believe there was television broadcasting over the airwaves in the nineteen-twenties? The thing is, it used bizarre spinning equipment because there were no CRTs (see "Lightning in a Bottle," nearby.) Only with the development of radar in World War II did there also come a practicable Cathode Ray Tube, making home television feasible.

But the big companies were at first very conservative in their marketing, figuring television would be a luxury item only. It took a man named Madman Muntz, who caricatured himself in a Napoleon hat, to see that millions would buy television if the price was right. So he came out with Muntz TV in the late forties. As I recall, the Muntz TV cost $100 and had one tuning knob. (This was less intimidating than the row of knobs on more expensive sets.) I don't know how Muntz came out on it all, but his opening of the mass market made the bigger corporations realize it was there. (This same thing may yet happen again in newer media.)

Originally all there was was Krazy Kat and Farmer Brown cartoons. But behold, sooner than you could say "vertical hold," there were Sid Caesar and Imogene Coca on the Admiral Show, and we were off.

A quarter of a century later, the best of television is no better and the bulk of television is about as bad as it ever was.

We "understand" television. That is, we know what a TV show is, how it fits together and so on.

ICECUBES

But what people don't realize about TV is that the governing feature is the time-slot. In any medium with time-slots, whether TV, radio or classroom education, the time-slot rules behavior. Whatever can happen is as constrained as icecubes in a tray.

This is the limiting factor when optimists try to use TV for teaching. If it's coming over a cable, everything has to be scheduled around it, and the contents are clipped and constrained to fit the time-slot. It may be better with videotape.

CABLES

In the last dozen years, Cable TV, or CATV, has become big business. A Video Cable is a high-capacity electrical carrier that runs through a given neighborhood or region. Business and individuals may "subscribe" and get their own sets hooked onto the cable.

What this does first of all is improve reception. The fouled-up video picture caused by such extraneous objects as the World Trade Center in New York can be corrected by hooking into the video cable: you get a nice, sharp picture.

In addition, though, the cable offers extra channels.

Now, the businessmen who have been throwing together these video cable outfits are aiming for something. They have been thinking that these extra channels would net them a lot of money: by showing things on them that can't be offered on the air-- highbrow drama, or perhaps X-rated stuff-- they could get extra revenue. (You'd pay extra to watch it by buying an unscrambler, or whatever.)

This is turning into somewhat of a disappointment.

The cable people had foreseen, evidently, that people would stay home in droves to see the new offerings on the cable. In Show Business it's easy to forget, though, that everybody has only twentyfour hours in a day, and far less than 24 hours to dispose of freely; so every leisure occupation is competing with every other leisure occupation. Moreover, the residual leisure occupation, when there's nothing else to do, is TV. It would seem that few people would watch more television if it were better, but many would watch less if they could afford to go out.

EXTRA CHANNELS

In recent years, a number of extra channels have been made available by law. These are the UHF, or Ultra High Frequency channels. These, like cables, represent a consumer breakthrough but will have only negligible impact.

THE PROBLEM OF ORGANIZATION

Whatever else you may say about them, the networks and TV stations are at least organized as going concerns within the institutional structures of the country. Ideas of "community television" and other such schemes which call for some new form of social organization to spring forth are about as plausible as "community control" of schools and police-- or at best likely to be as influential as "community social centers."

INTERACTIVE TV?

Some people, I won't say who, have gotten a lot of money for something they call "interactive television." What this turns out to mean is any form of computer time-sharing that will use home TV terminals and video cables. The questions are why use home TV terminals and video cables, insofar as they would seem to promise only comparatively low-grade performance; and whether these people have thought out anything about the potential characteristics of the various media they propose with such abandon. Nothing I have seen or heard about this is reassuring.

"ALTERNATE" TELEVISION, or VIDEO FREAKS

In recent years, many young folks have taken to video as a way of life. In the most extreme cases they say things like "the written word is dead," prompted perhaps by McLuhan. I have found it rather difficult to talk to video freaks. (It may be that some of them are against spoken words as well.) I really just don't know what they're about.

The work of these people is as exuberant as it is strange. I haven't seen much of it or understood much of what I have seen.

In some cases, "alternative television" simply means documentaries outside the normal framework of ownership and reporting. In one example cited by Shamberg (see bibliography), video freaks did excellent coverage of the 1968 Republican convention. People were allowed to speak for themselves, unlike "normal" TV journalism where "commentators" tell you what they see.

Now, this is hardly revolutionary; it is just good documentary-making that shucks dumb traditions artistically, much like the Pennebaker films. However, video enthusiasts claim it is somehow different, and indeed claim that video is different in principle from films. I have been unable to get a satisfactory clarification of this idea.

Video is being used in other ways, harder to understand, by artists (best defined as persons called "artists" within the art world today). Very odd "video pieces" have been shown at art shows, where the object seems to be to confuse the viewer-- or knock him into a condition of Enlarged Perspective, shall we say. And a variety of non-objective videotapes are now being created. (A gallery show in 1969 was called "Video as a Creative Medium" -- implying sarcastically that it had not been before, on the airwaves.)

Some video freaks think of video as intrinsically radical or Revolutionary. In this respect they differ interestingly from, say, the editors of the National Lampoon. The editors of the National Lampoon appear to be political radicals, but do not suggest that the very media of cartoon and joke-piece are themselves revolutionary. Some video freaks appear to be persuaded that the medium of television itself is inherently a vehicle for change.

I can understand one interesting sense in which this may be true: Shamberg talks about video as a method of self-discovery. Seeing yourself on TV does, of course, confer certain insights. But Shamberg suggests it may expand people's consciousness in larger ways-- allowing people to see the bleakness of certain pursuits (he uses the example of Shopping), for instance. But if this does hit home to people, it doesn't seem to me to be the medium that's doing it but the selected content-- as in all previous media. Maybe I've missed the point in some way.

These developments are all very interesting. It can be hoped that those trying to develop new forms of communication will make an effort to communicate better with those who, like the author, often cannot comprehend what they are doing.

> " But decentralized transmission of information should be dominant, not fugitive. Each citizen of Media-America should guaranteed as a birthright access to the means of distribution of information."
>
> (Shamberg, p. 67)

> " Well, we went down there with our Porta-Pak and tried to take it inside. A guard came over and said we couldn't and even threw one of us out of the booth while the other was inside. A guard telling you what to do in a cybernetic environment?"
>
> (Shamberg, p. 53)
>
> ("Cybernetic" is evidently a code word here for what they think is good, true, beautiful and inevitable. Cf. p. DM 13.)

> " About the only generalization to be made is that community video will be subversive to any group, bureaucracy, or individual which feels threatened by a coalescing of grassroots consciousness. Because not only does decentralized TV serve as an early warning system, it puts people in touch with one another about common grievances."
>
> (Shamberg, p. 57)

BIBLIOGRAPHY

Michael Shamberg and Raindance Corporation, Guerrilla Television. (Holt, $4.)

TUBE, an underground TV magazine. $8/yr. TUBE, 1826 Spaight St., Madison, WI 53704.

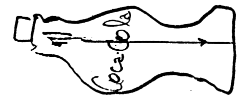
Cable Report, $7/yr. 192 N. Clark St., Room 607, Chicago. Samples $1. "SCANDAL IS RAMPANT in the cable television industry. Only Cable Report follows cable TV developments from the citizen's perspective and tells you what's happening and what's going wrong." Ad in Chicago READER.

Nicholas Johnson, How to Talk Back to Your Television Set. Bantam, 95¢

LIGHTNING IN A BOTTLE:

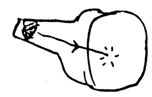
THE CATHODE-RAY TUBE

A cathode-ray tube is actually a bottle filled with a vacuum and some funny electrical equipment. The equipment in the neck of the bottle shoots a beam of electrons toward the bottom of the bottle.



This beam of electrons is called, more or less for historical reasons, a cathode ray. Think of it as a straw that can be wiggled in the bottle.

Actually the bottle is shaped so as to have a large viewing area at the bottom (the screen), and this screen is coated with something that glows when electrons hit it. Such a chemical is called a phosphor.

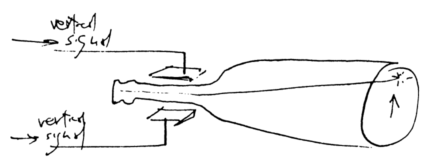


Now, two useful things can be done with this beam.

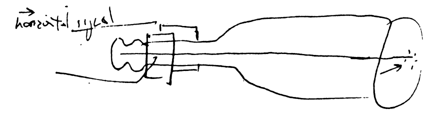
1) It can be made brighter by increasing the voltage, which increases the number of electrons in the beam.

2) The beam can be moved! That is, it can be made to play around the face of the tube the way you can slosh the stream of a garden hose back and forth on the lawn; or wiggle a straw in a coke bottle. The beam can be moved with either magnetism or static electricity. This is applied in the neck of the bottle-- or even from outside the neck-- by deflection plates, whose electrical pulsations determine the pattern the beam traces on the screen. (Note that the beam can be moved on the screen at great speed.)

The vertical deflection plates can pull the beam up or down on the screen, controlled by a signal to them;



the horizontal deflection plates can pull the beam sideways on the screen, controlled by a signal to them.



By sending combined signals to both horizontal and vertical deflection plates, we can make the end of the beam-- a bright dot on the screen, sometimes called a flying spot-- jump around in any pattern on the screen. A repeated pattern of the beam on the face of the CRT is called a raster.

From these two capabilities-- brightening and moving the beam-- a number of very special technologies emerge:

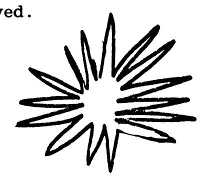
TELEVISION uses a zig-zag scanning pattern which repeats over and over. This zigzag pattern is always the same, night and day.

You can usually see the lines clearly on a black-and-white set. The picture consists of the changing pattern of brightness of this beam, which comes in over the airwaves as the television signal.

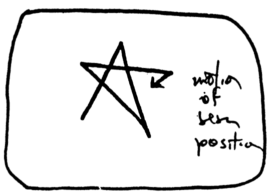
IF WE HADN'T STANDARDIZED TV WHEN WE DID, WE'D HAVE A BETTER SYSTEM NOW.

Let this be a lesson: standardize on the best system, not necessarily the first.

RADAR DISPLAY uses a CRT to show reflected images around where the radar antenna is standing. This uses a scanning raster of a star shape, brightening the beam when reflected images are received.



COMPUTER CRT GRAPHICS generally use the CRT in still another way: the beam is moved around the screen in straight lines from point to point. (Between different parts of the picture the beam is darkened, turned very low so you don't see it.)



Because the image on a normal CRT fades quickly, the computer must ordinarily draw the picture again and again and again. (Methods for this are discussed on p. DM 22-3.)

SPECIAL KINDS OF CATHODE-RAY TUBES

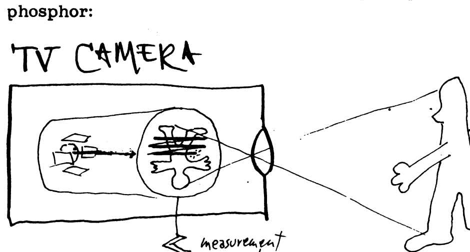
The CRT is not merely a single invention, but an entire family of inventions. The ordinary CRT, which we have discussed, is viewed at one end by a human being, has an image which fades quickly, and can have its flying spot driven in any kind of raster or pattern.

Here are some other kinds of CRT:

The picture transmitter, which has different versions and names: Vidicon, Image Orthicon, Plumbicon, etc. THIS IS THE MAGICAL DEVICE THAT MAKES THE TELEVISION CAMERA WORK, AND YET, BY GOSH, IT'S JUST ANOTHER CRT. Except instead of the picture coming into it as an electrical signal and out of it as an optical image, the picture comes into it as an optical image and goes out of it as an electrical signal.

How can this be?

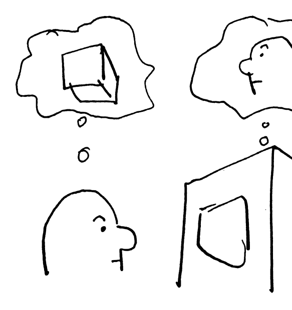
The tube sits inside the television camera, which is an ordinary camera, like, with a lens projecting a picture through a dark chamber onto a sensitive surface. But instead of the surface being a film, the surface is the faceplate of a CRT with some kind of a special pickup phosphor:



The electron beam, which is just like any other electron beam, is made to zigzag across the faceplate in a standard television raster. And the special phosphor of the tube measures the brightness of the picture at the spot the beam is hitting. I have no idea how this happens, but it's chemical and electronical and mysterious, and is based on the way the phosphor interacts with the light from one side and the electrons from the other side at the same time. Anyhow, a measurement signal comes out of the faceplate, indicating how bright the projected picture is in the very spot the electron beam is now hitting.

As the beam criss-crosses the faceplate in the zig-zag television raster, then, a continuously changing output signal from the faceplate shows the brightnesses all across the successive lines of the scan.

And that is the television signal. Together with synchronizing information, it's what goes out over the airwaves, down your antenna and into your set. Your set, obeying the synchronizing information, brightens and darkens its own beam in proportion to the brightness of the individual teeny regions of the faceplate in the television camera. And this produces the scintillating surface we call television.



The color tube is a weird beast indeed. There are several types, but we'll only talk about the simplest (and many think the best), Sony's Trinitron(TM) tube.

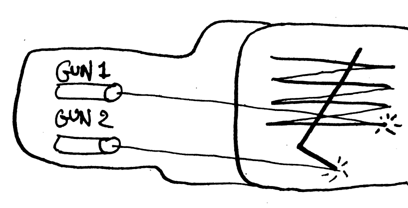
This is an ordinary CRT which has, instead of a uniform coating on the faceplate, tiny vertical stripes of three primary colors-- red, blue and green. (You thought the primary colors were red, blue and yellow, didn't you. If you're mixing pigments that happens to be true. For some ungodly reason, however, if you're mixing lights, the colors that yield all others turn out to be red, green and blue; it turns out that yellow light can be made out of red and green. If you don't believe me go to a chintzy hardware store, get a red and a green bulb, turn 'em on and see what happens in a white-walled room.)

At any rate, color television uses additional color signals, and in the Trinitron these control the response of the faceplate. If the color signal says "green" as the electron dot crosses a certain part of the screen, the color signal tells the green stripes that they're free to light up when hit. If it's Yellow Time, the signal tells both the red stripes and the green, and so side by side they light up red and green, as the beam crosses them, but the total effect from more than a few inches is Yellow.

Most American color TV sets, however, at least up till this year, used something very different, something entirely weird called the Shadow Mask Tube. I'll spare you the picture, but there were several different electron beams -- often referred to jokingly as the "red electron beam," "blue electron beam" and "green electron beam," though of course they were identical in character. These hit a perforated sieve, up near the screen, called the shadow mask, and the color signal tweaked the unwanted beams so they did not hit different-colored phosphor dots that were intricately arranged on the screen. I'm sorry I started to explain this.

Multigun tubes have more than one electron gun and more than one electron beam. They can be used in different ways (aside from the old shadow-mask TV tube, mentioned above).

For instance, one gun can be driven in a video raster, to show television, while another gun can be used as a computer display, drawing individual lines with no regard to the TV pattern.



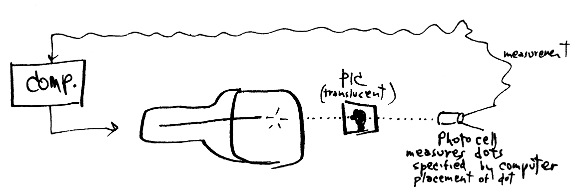
The storage CRT comes in two flavors: viewable and non-viewable. But what it does is very neat: it holds the picture on the screen. The mechanisms for this are of various types, and it's all weird and electronic, but the idea is that once something is put on the screen by the electron beam, it stays and stays. Up to several minutes, usually. The main manufacturers are Tektronix, Princeton Electronic Products, and Hughes Aircraft; each of these three has a product that works by a different method.

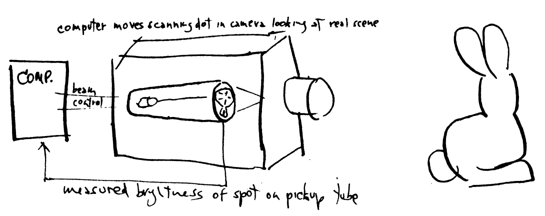
Note: Tektronix' tube is built into a number of different computer displays, and is recognizable by its Kelly green surface. They themselves make complete computer terminals around this scope for $4000 and up, but lots of other people put it in their products also. It shows whatever has already been put on the screen, and the electron beam does not have to repeat the action. However, it usually only stays lit for about a minute.

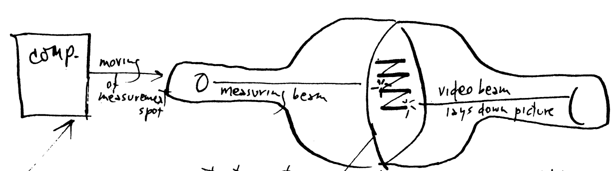
Princeton Electronic Products (guess where) is a much smaller outfit, so perhaps it is appropriate that they make a much smaller storage tube. It is about one inch square at its storage end, and you don't look at it directly. Instead, an image can be stored on it either wth a TV raster or by computer-driven line drawing. After the image is stored on it, though, it functions as a TV camera: the picture stored on the plate can be read out with a scanning raster, exactly as if it were a picture transmitter in a television camera. The Princeton folks have built a quite expensive, but quite splendid, complete terminal around this device: it can hold both video and computer-drawn pictures, superimposed or combined, and sends them back out in standard black-and-white TV. $12000.

CRTS which bring in a picture one way (such as a video raster) and send it back out another way (such as by letting a computer search out individual points) are called scan converters.

A word about this last method. It is often desired by computer people to turn a picture into some form of data (see p. DM 10). Scan converters, usually by the three manufacturers named above, can be hooked up to let the computer program poke around in the picture and measure the brightness of the picture in arbitrary places. A device which examines the brightness of something in arbitrary places is called a flying spot scanner.) Here are some different kinds of flying-spot scanners:







video picture itself is measured in a two-ended storage tube.

I have heard it said that it might be possible to build a CRT with a changeable mirror surface: that is, the screen becomes mirrored temporarily where it is being hit with the electron beam. Interesting. This would mean that you could make computer displays (and TV) bright and projectable to any degree, say, by pouring a super-intensity laser beam on it. "Be great for writing 'Coca-Cola' on the moon," says a friend of mine. If you believe in astral projection.

BIBLIOGRAPHY: Color TV Training Manual, Sams & Co./ Bobbs-Merrill ($7), is a well-illustrated and intelligent introduction to the TV use of CRTs.

SANDIN'S IMAGE PROCESSOR

Dan Sandin, professor of Art at U. of Illinois, Chicago Circle, says very wise things (having been a physicist), and we were going to have a whole section on that, but as you can see there wasn't room.

Daniel J. Sandin (pronounced san-DEEN) has spent the last several years putting together a device he currently calls the IP (Image Processor). It's a system of circuits for changing and colorizing TV. What follows is the first published description of it.

I regret that the following is probably one of the most difficult sections of this book. (If you know nothing about video, read ~~the opposite page~~ DM 6-7 first.)

The idea is basically to create a completely generalized system for altering the color and brightness of video images. (I.e., the system does not move them on the screen. Thus it differs from the Computer Image line of video-twisting graphics systems, which alter positions of objects; see p. DM 39 . Note also that rather similar facilities exist as part of, e.g., the Scanimate system, p. DM 39 .)

This means that basically Sandin's system plays with the part of the TV signal called z, or brightness (as distinct from x or y, the signals for horizontal and vertical movement of the dot. See ~~opposite page~~ DM 6-7).

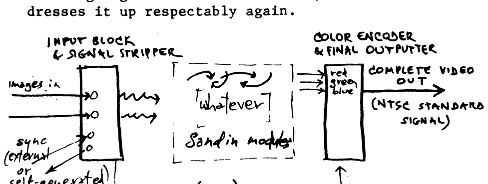
Now, as a physicist and field-theoretician, Sandin approached this as a problem in generality; and indeed, the style of generalization should be appreciated. Sandin repeatedly chose flexibility and power rather than obviousness in the parts he created. The resulting system is both parsimonious and productive.

His first important decision was that all parts of the system should be compatible and idiot-proof, so that any user could frivolously plug it together any way at all without burning out the circuits.

Indeed, Sandin decided to build it like a music synthesizer: by making all systems electrically compatible (as they are on the Moog and its progeny), any signal can be used to alter or influence any other signal. This is a very profound decision, whose far-flung results have not yet been fully explored even among Sandin's rather fanatical students.

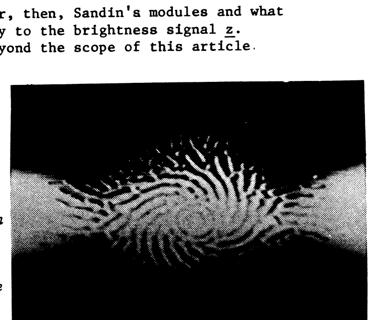
Basically, the incoming video image is "stripped" of its synchronizing information, so that all signals turning up in the guts of the machine may be freely modified. Only at the final output stage are the jots and tittles of the video signal put back on.

Thus the first and last blocks of the Image Processor act like bookends, between which the other modules have their fun. The first block makes the incoming signal into "naked" video, the last block dresses it up respectably again.



For the sake of clarity we will refer to the outputs as pictures, or as black, white or grey, which they would be if they went straight out to a screen; but they may be turned back into the system and function as inputs as well. "White" means +.5 volts, "black" means -.5 volts.

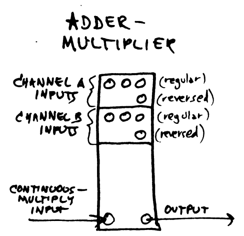
Let us consider, then, Sandin's modules and what they do individually to the brightness signal z. Combinations are beyond the scope of this article.



What Dan's processor can do to television is not to be believed.

Savage colors or delicate off-whites, solarizations and pictures on top of pictures. Then through "video feedback" (pointing a TV camera at a TV screen), the system can generate throbbing animated cobwebs and spirals of its own. Shown.

1. ADDER-MULTIPLIER. This combines two input channels, either directly or as specified by a third.

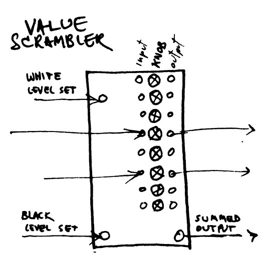


The channel A inputs are added together and multiplied by C; the channel B inputs are added together and multiplied by the reverse of C; both results are added to make the output. (NOTE: this unit is used among other things, for fades and keying.)

2. COMPARATOR. This is like Kodalith film, making an image into stark black and white. Its output is pure black or white. One input signal (the video) is compared with another input signal (reference level, other video, whatever).

While one is greater the output goes all black, and while the other is greater it goes all white.

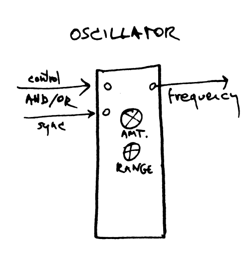
3. VALUE SCRAMBLER. This is a single module dividing the picture into eight levels. It may be thought of as eight of the above comparators, dividing the brightness spectrum by quantum jumps. The floor and ceiling of the signal to be divided are specified by the two control channels, but the dividing lines between them are then automatically determined. Each corresponding output level may be controlled by a knob.



Thus from a range of input values, we get an output step-function each of whose brightnesses is individually adjustable.

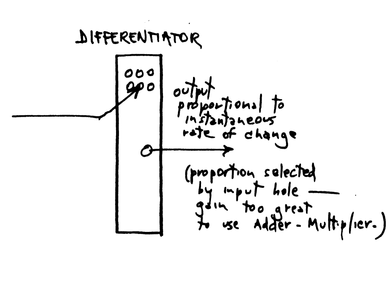
Note that these devices may be arranged in parallel, thus dividing the brightness spectrum into as many levels as desired.

4. OSCILLATOR MODULE (very unusual). Sandin's oscillators are voltage controlled, just like the ones in music synthesizers. However, if given any kind of a sync signal, they lock into the nearest multiple (or submultiple) within the specified range. (But then the control signal, if any, tweaks it higher or lower.) Standardized output comes in sine, square and sawtooth.



The two planned uses were A) with a sync, to generate fixed patterns, and B) without a sync, to generate movable patterns. If both inputs are used, it becomes a stubborn lock-on voltage-controlled oscillator, which tends to grab at passing submultiples.

5. DIFFERENTIATOR. Basically this sees edges in the picture, or any other part of a scan-line whose color is changing. Its output is proportional to change occurring in the brightness of a scan-line, As the input goes from black to white its output is light; as the input goes from white to black its output is dark. (The input hole selected determines the amount of multiplication.)



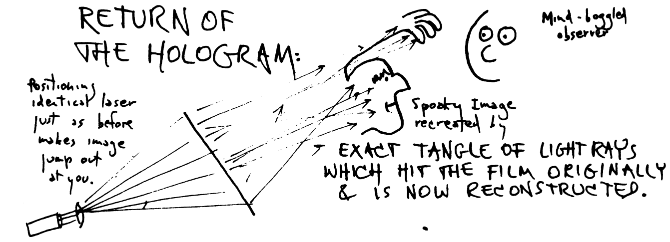


Diagram of how hologram is made, p. DM 20.

HOLOGRAPHY
puts you in the picture
and you, and you...

Holography is one of those Modern Miracles that we really can't get into. It is mind-blowing, influential, and of unclear importance.

Theoretically predicted by Dennis Gabor, the hologram (Greek "whole picture") was finally made to work in the late fifties by Leith and Upatnieks. Since then dozens of other types of holograms have been experimented with, including color holograms, movie holograms, video holograms, audio holograms and gracious know what.

Basically a hologram is an all-around picture. It doesn't look like a picture, but looks like a smudged fingerprint or other mistake of some kind.

Yet it is a marvel.

A basic hologram (-- actually it should be called a laser hologram or Leith-Upatnieks hologram, but we've no time for such distinctions--) is one of these smudgy pictures which, when viewed under a proper laser setup, shows you a three-dimensional picture. Worse than that: as you move your head, the picture changes correspondingly. It looks, not like the flat surface it is, but like a lit-up box with a model in it.

What does the hologram do? Actually it re-creates, not a single view, but the entire tangle of light rays that are reflected from the real object. Even down to bright reflections, which scintillate in the usual way, as from chromium.

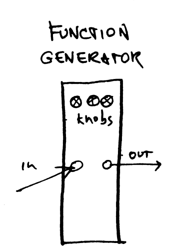
The only problem: ordinarily they have to be used with laser light, which is spookily one-colored.

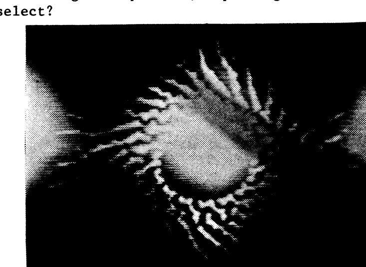
Notes from all over: art stylist Salvador Dali presided at an unveiling of "the world's first 360° hologram" at a New York gallery not long ago. The subject was song stylist Alice Cooper.

The Haunted House at Disney World in Florida will ride you through a building full of holograms. That's one way to move through ghosts, all right.

There is a New York School of Holography.

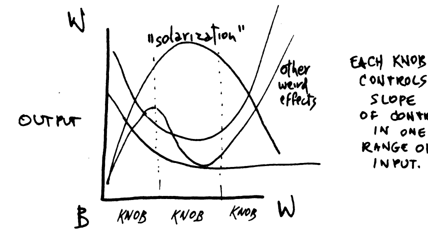
6. FUNCTION GENERATOR. This device is hardest to explain. Let's do it in terms of that first module, the Adder-Multiplier. Know how the Adder-Multiplier puts out either a positive or a negative picture, depending on which input you select?



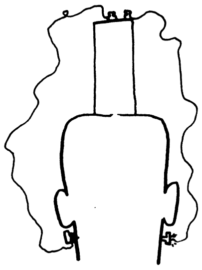


Well, the Function Generator divides the input brightnesses into three ranges, and multiplies each range positive or negative, in proportion to its own knob setting.

Thus the combined setting of the three knobs generates a "function," or curve, from the slopes of the individual settings. See graph. What in photography is called "solarization" represents just one of these combined settings. The others are nameless.



7. COLOR ENCODER MODULE. This is the last block. Into it go three signals, the desired red, blue and green; and out comes standard NTSC video.



BODY ELECTRONICS

"I sing the body electric..." -- Walt Whitman

There are various people who want to attach electronics to people's bodies and brains.

There are basically two starting points for this ambition. One is authoritarian, the other is altruistic. I am not sure both schools are not equally dangerous, however.

Let's consider first the authoritarians. Prof. Delgado of Yale has demonstrated that any creature's behavior can be controlled by jolts to the brain. Delgado has dealt especially with the negative circuits of the brain, that is, places where an electrical impulse causes pain (or "negative reinforcement"). In Delgado's most stunning demonstration, he stopped a charging bull with just a teeny radio signal. Enthusiastically Delgado tells us how fine this sort of thing would be for controlling Undesirable Human Behavior, too.

Now, let's consider just what we're talking about. In these experiments, needles are implanted in the creature's brain. This can involve removing a section of the skull, or it can be done merely by hammering a long hollow needle straight into the skull and thus the brain.

The researcher, or whatever we want to call him, had better know what he is doing. But due to the remarkable mass action of the brain, the destruction caused by such needles will have not observable effects if done properly.

The hollow needle, once in place, becomes a tube for shielded electrical wires, whose bare metallic tips may then be used to carry little electrical jolts, to whatever brain tissue is reached by the tip of the needle, whenever tiny signals are applied.

Now there are regions of the brain, distributed irregularly through its mysterious contents, which are loosely called the "pleasure" and "pain" systems. They are called that because of what the organism does when you jolt it in those places. (We do not know whether jolts to these areas really cause pleasure or pain, because these things haven't been done to human beings. Yet. The creatures it has been done to can't tell us just how it feels; thus "pleasure" and "pain" are in quotation marks. For now.)

Anyway, what happens is this. If you stimulate a creature in the "pain" system it tends to stop what it is doing-- this is called negative reinforcement-- and if you stimulate it in the pleasure system, it tends to do more of what it was doing. Positive reinforcement.

Now, to some people this suggests wonderful possibilities.

Delgado, for instance, believes that this technology gives us everything we need for the control of Anti-Social Tendencies. Criminals, psychopaths and Bad Guys in general-- all can be effectively "cured" (i.e., put on their best behavior) by these techniques. All we have to do, heh heh, is get into their heads, heh heh, habits of proper behavior. And with these new techniques of reinforcement, we can really teach 'em.

Unfortunately Delgado is probably right.

In principle this is just a drastic form of behavior control on the B.F. Skinner model (depicted also in Nineteen Eighty-Four and A Clockwork Orange). The new system is more stark and startling because of its violation of the individual's body interior, but not in principle different.

Skinner has the same naive, simpleminded solutions for everything. All "we" have to do-- using "we" to mean society, the good guys, good guys acting on behalf of society, etc.-- is control the behavior of the bad guys, and everything will be better, and "we" can accomplish anything "we" desire.

The reader may see several problems with this.

In the first place (and the last), there is the obvious question of who we are, and if we are going to control other people, who is going to control us.

At a time when our "highest" leaders show themselves preoccupied with low retaliations and lower initiatives, we can wonder indeed if it is not more important to prevent anyone from ever getting this kind of control over humans than to facilitate it.

Even if that weren't a problem, there is the more simpleminded question of who in the existing system would use such techniques. It turns out, of course, that they would be added to what is laughably called the Correctional System, or even more laughably called the Justice System. All the sadists you could possibly want work there. (And no doubt some very nice guys-- but experiments have demonstrated horrifically that decent people, turned into "guards" even for a short time, adopt the patterns of brutality we have known from time immemorial.)

So, like truncheons and electric shock therapy and solitary confinement and everything else, these techniques-- if they are used-- will enter the realm of Available Punishments, not to be used with clinical precision but with gratuitously brutalizing intent, new tools for punitivity and sadism. The "correctional" system would have to be magically corrected itself before such tools could be employed without simply making things worse. And the prospect is not good.

Such schemes grow, of course, from a caricature of the malefactor-- thinking him to be some sort of miswired circuit, rather than a human being caught up in anger, pain, humiliation and unemployment.

(There are also a lot of canards about Free Will, but these do nothing for either side in this controversy.)

NEW FACULTIES

Starting from an entirely different outlook, various designers and bio-engineers are trying to add things to the human body and nervous system, for the voluntary benefit of the recipient.

A number of research and development efforts are aimed at helping those with sensory impairments, and electronics obviously is going to involved.

An example: a firm called Listening, Inc. in Boston, founded by Wayne Batteau (whom John W. Campbell considered one of the Great Men of Our Time), devised a system for helping the totally deaf to hear. Supposedly this could transmit the actual sensation of hearing into the nervous system by some scarcely-understood form of electrical induction. The machine was sold off; whether it ever got a safety rating I don't know.

This is the sort of thing people would like to do for the blind, as well.

Now, in principle, it might be possible to transmit an image in some way to the actual visual area of the cerebral cortex. (This might or might not involve opening the skull.) Somebody's working on it.

In a related trend, numerous design groups are attempting to extend the capabilities of the human body, by means of things variously called possums, waldoes and telefactors.

"Possums" (from Latin "I can") are devices to aid the handicapped in moving, grasping and controlling. Whatever motions the person can make are electronically transposed to whatever realm of control is needed, such as typewriting or guiding a wheelchair. ("Waldo" is Heinlein's term for a possum that can be operated at a distance.)

In the space program, though, they call them telefactors. A telefactor is a device which converts or adapts body movements by magnification or remote mimicking. Unlike possums, they are meant to be operated by people with normal faculties, but to provide, for example, superhuman strength: cradled in a larger telefactor body, a man can pick up immense loads, as the movements of his arms are converted to the movements of the greater robot arms.

Telefactors can also work from far, far away. Thus a man sitting in a booth can control, with the movements of his own arms, the artificial arms of a robot vehicle on another planet.

(This whole realm of sensory and motor mechanics and transposition is an important aspect of what I call "Fantics," discussed on pp. DM48-51).

Then there are those who, like How Wachspress (see nearby), want to expand man's senses beyond the ordinary, into new sensory realms, by hooking him to various electronics.

THOUGHTS

There are two problems in all of this. The first and worst, of course, is who controls and what will hold them back from the most evil doings. Recent history, both at home and abroad, suggests the answers are discouraging.

The second problem, wispish and theoretical next to that other, is whether in turning toward bizarre new pleasures and involvements, we will not lose track of all that is human. (Of course this is a question that is asked by somebody whenever anything at all changes. But that doesn't mean it is always inappropriate.)

In the face both of potential evil and dehumanization, though, we can wish there were some boundary, some good and conspicuous stopping place at which to say: no further, like the three-mile limit in international law of old. I personally think it should be the human skin. Perhaps that's old-fashioned, being long breached by the Pacemaker. But what other lines can we draw?

The prospects are horrorshow, me droogies.

BIBLIOGRAPHY

T.D. Sterling, E.A. Bering, Jr., S.V. Pollack and H. Vaughan, Jr., Visual Prosthesis: The Interdisciplinary Dialog. ACM Monograph. $21.

PSYCHO-ACOUSTIC DILDONICS

I originally hadn't intended to include anything like this in the book, wanting it to be a family-style access catalog and all that, but this particular item seems fairly important.

Remember how we laughed at the Orgasmotron in Woody Allen's Sleeper? Well, it turns out not to be a joke.

An individual named How (not Howard) Wachspress, electronicker-in-residence at a San Francisco radio station, has been developing just that, except that he has more elevated purposes in mind. The secret was broken to the world in Oui magazine earlier this year; but Hefner, the publisher, evidently held back the more startling photographs of a model in electronically-induced ecstasy.

Wachspress' devices transpose sound (as audio signals) into feelings; you touch your body with an open-ended tube or other soft fixture attached to his device-- which in turn is attached to a hi-fi.

The sensations, it is claimed, are profound and moving. You may take them anywhere on your body; the effect is deeply relaxing and emotionally engrossing. Wachspress thinks he has reached an entire neurological system that wasn't known before, much like Olds' discovery of the "pleasure center" in the brain; he sees it as a new modality of experience and a generalization of music and touch. That is the main point. "Hyper-reality" is where he says it gets you: a point curiously congruent with the author's own notions of hypertext and hypermedia as extensions of the mental life.

This said, we can consider the prurient aspects of Wachspress' Auditac and Teletac devices (which he intends to market in a couple of years as hi-fi accessories, b'gosh). When played with the right audio, in the right places, and a good operator at the controls, they provide a sexual experience said to be of a high order.

Wachspress' work ties in interestingly with today's "awareness" movement, of which Esalen is the spiritual center, which holds that we have gotten out of touch with our bodies, our feelings, our native perceptions. As such, the Wachspress machines may be an unfolding-mechanism for the unfeeling tightness of Modern Man-- as well as a less profound treatment for "marital difficulties" and Why-Can't-Johnny-Come-Lately.

Inscrutable San Francisco! Wachspress gave a number of demonstrations of his devices in Bay Area churches, until he became disturbed at immodest uses of the probe by female communicants who had stood in line to try the machine.

(Auditac, Ltd., Dept. CLB,
1940 Washington St.,
San Francisco CA 94109.)

Harry Mendell, a good friend of mine, rigged an interesting experiment while he was still in high school.

He used a little Hewlett-Packard minicomputer, which the manufacturer had generously loaned to his Knights of Columbus Computer Club of Haddonfield, N.J.

Harry hooked the Hewlett-Packard up to a CRT display (see pp. DM6-7, DM22-3). At the top of the CRT, following his program, the computer continuously displayed the letters of the alphabet. A little marker (called a cursor) would skip along underneath the letters, acting as a marker for each of them in turn.

Harry rigged one more external device: a set of electrodes. These would be strapped, harmlessly, to the head of a subject. Harry's computer program used these electrodes to measure alpha rhythm, one of the mysterious pulses in the brain that come and go.

Every time the subject flashed alpha, Harry's program would copy the letter above the cursor to the bottom of the screen.

Sitting in this rig, subjects were able to learn, rather quickly, TO TYPE WORDS AND SENTENCES. Just by flashing alpha rhythm when the cursor was under the right letters.

Jubilant, Harry showed this setup to an eminent neurophysiologist from a great university nearby, a man specializing in electrode hookups. Harry was a highschool student and did not understand about Professionalism.

"What's so great about that?" sniffed the eminent professional. "I can type faster."

So Harry dropped that and went on to other stuff.

PICTURE PROCESSING

"Picture processing" is an important technology, largely separate from the rest of computer graphics. It means taking an incoming picture, usually a photograph, and doing something to it. (Some now call this area "computer pictorics.")

First of all, there is image enhancement. This means taking pictures, dividing them into points whose brightness is separately measured, and then using special techniques for making the picture better. To people familiar with photography, this may seem impossible; to photographers it is a maxim that photographs always lose quality at each step. Nevertheless, various mathematical techniques such as Fourier Analysis (mentioned elsewhere) do just that, producing a new data structure improving on the original data. Surfaces appear smoother, edges sharper.

(These techniques have been extensively used to clean up photographs sent back from our unmanned space vehicles-- both those used exploring other planets and those spying on our own-- see Secret Sentries in Space, Bibliography.)

Then there are recognizers-- programs that look at the data structure from an input picture, and try to discern the lines, corners and other features of the picture. (While your eye instantly sees these things, computers do not, and must look at the dots of a picture one-by-one. How to analyze pictures in such tedious sequences is no simple matter.)

For recognizing more complex objects in pictures-- boxes, spheres, faces or whatever-- more complex structure-analyzing programs are necessary. As the possibilities of what might be in a picture increase, these increasingly become guessing programs. (This becomes a branch of artificial intelligence, a misleading term for a curious field, discussed on pp. 12-14.)

Numerous computer people think it is important to match up our computer graphic display systems (described variously on this side of the book) to image input systems. This is a matter of taste.

These are all basically techniques for making a data structure. Any data stored in computers must have, of course, a data structure-- which basically means any arrangement of information you choose. (see p.26-9.)

These various techniques are intended to create reduced data structures, recording only the "most important" data of the picture-- from which new and varying pictures may be created, reflecting the "true" structures originally shown in the initial picture. How much it's going to be possible to create these data structures from input pictures remains to be seen; some of us think it's not going to be generally worthwhile.

BIBLIOGRAPHY

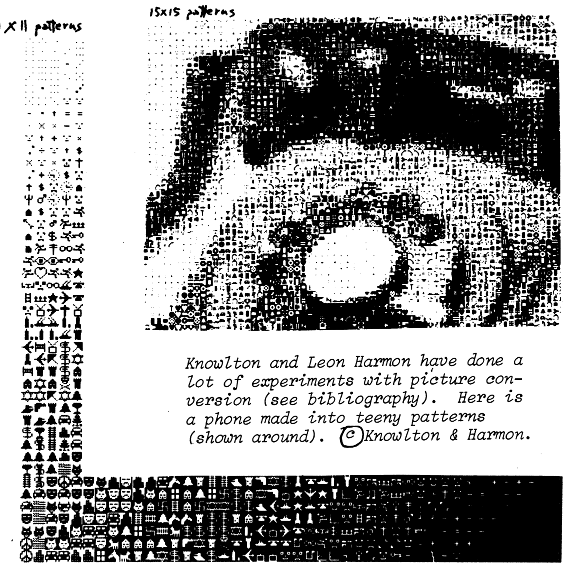
Azriel Rosenfeld, "Progress in Picture Processing 1969-71." ACM Computing Surveys June 73, 81-108.

Ken Knowlton and Leon Harmon, "Computer-Produced Grey Scales." Computer Graphics and Image Processing, April 72, 1-20.

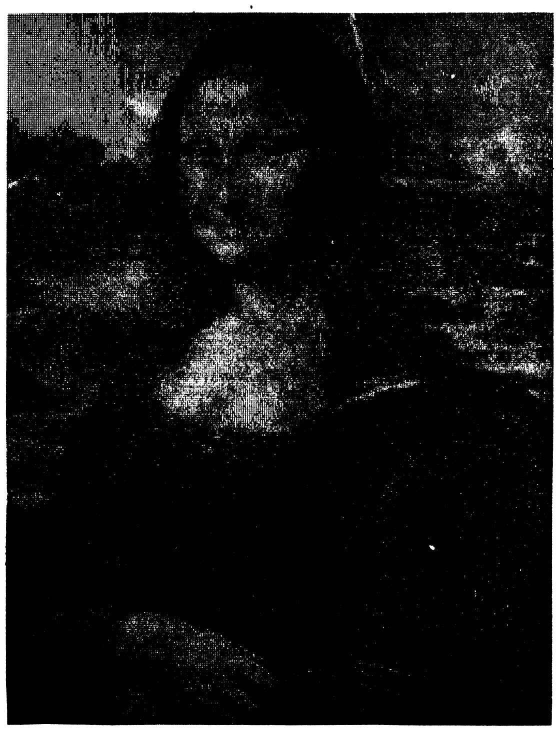
Philip J. Klass, Secret Sentries in Space. Random, 1971, $8. Interesting general book on geopolitical strategy and orbital photoreconnaissance. "Now-it-can-be-told" approach.

SATELLITE PICTURES OF YOUR OWN HOME COUNTY, OR WHATEVER

You can get pictures of any area you want from ERTS (Earth Resources Observation Systems) satellites, from EROS Data Center (no, not a dating service, see p. 64), Sioux Falls SD 57198, or call 605/594-6511 bet. 7 AM & PM central time.



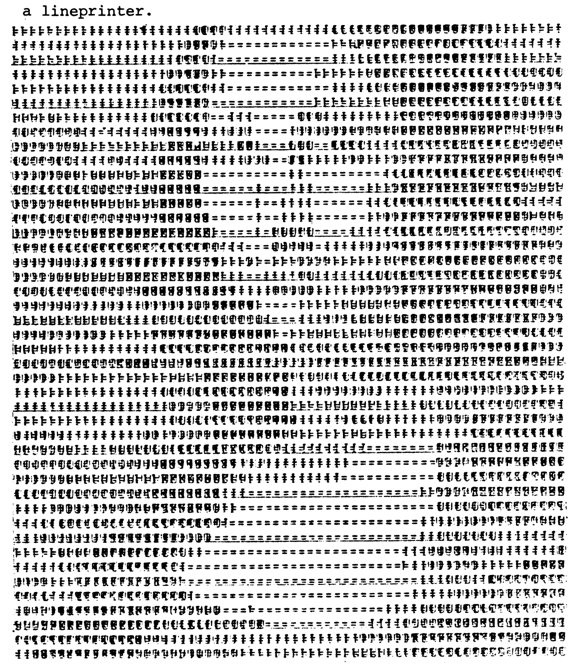
Knowlton and Leon Harmon have done a lot of experiments with picture conversion (see bibliography). Here is a phone made into teeny patterns (shown around). ©Knowlton & Harmon.



LIZZIE OF THE LINEPRINTER

A famous converted picture. The painting was divided into 100,000 brightness-measured spots by H. Philip Peterson of Control Data Corporation; then each dot was made into a square of overprinted letters on the printing device. The program allowed 100 levels of grey. Above: Control Data's version, reprinted by permission. Below: a cut-down version that often turns up. (From original flat 2D artwork by Len DaVinci of Medici Associates.)

NOTE: this is not a "computer picture." There is no such thing. It's a quantization put out on a lineprinter.



KEN KNOWLTON

Kenneth Knowlton is a Bell Labs lifer. Tall, patrician and gracious, his work, like Sutherland's, shows the inner light of unifying intelligence. He works in Max Mathews' section of Bell Labs at Murray Hill, where they do all that interesting stuff with music and perceptual psychology and so on. During the last decade, Knowlton has turned out vast quantities of articles, processed pictures, movies, and actual computer languages; while any ordinary man would be satisfied to be so productive, apparently he does a lot of other things in his work that he doesn't talk about.

Some of Knowlton's best-known work has been in picture processing, where he has converted photographs into mosaics of tiny patterns-- which nevertheless show the original.

His first widely-known language was BEFLIX (BEll Labs movie-making system); this was programmed for the 7094 in the early sixties. BEFLIX allowed the user to create motion pictures by a clever mosaic process that used the output camera more efficiently. (Actually, the lens was thrown out of focus manually and the entire frame created as a mosaic of alphabetical characters; this did the whole thing much more quickly and inexpensively.)

(Some of the clever data-handling techniques of BEFLIX Knowlton then turned around and used in L^6, a language which made these techniques available to other computer people. This may sound like only a computer technicality, but it's the sort of thing that's widely appreciated. (L^6 stands for "belL Labs' Lower-Level List Language."))

Wanting to get outside artists interested in BEFLIX and related media, he worked for a time with film-maker Stan Vanderbeek; from this Knowlton saw that artists' needs were more intricate than he had anticipated. Augmenting BEFLIX with some of the things Vanderbeek asked for, Knowlton came up with a new language called TARPS (Two-Dimensional Alpha-Numeric Raster Picture System). This in turn led to EXPLOR (EXPlicit(ly provided 2D Patterns,) Local (neighborhood) Operations, and Randomness). EXPLOR is fascinating because of its originality and generality-- not only does it modify pictures and serve as an artist's tool, but it has fascinating properties as a computer language and may even have applications in complex simulations for technical purposes.

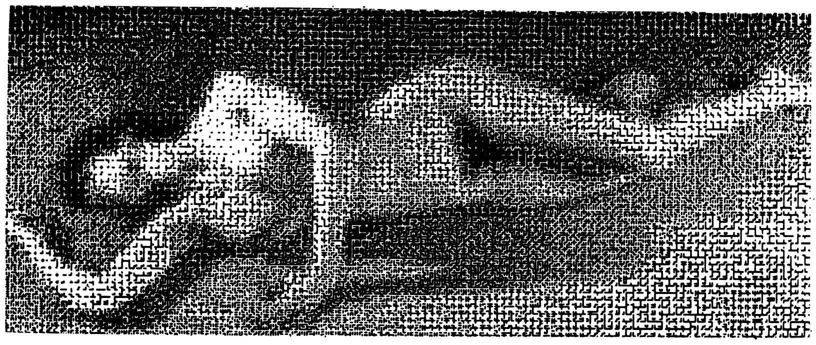
Since Vanderbeek, Knowlton has entered into a long and fruitful collaboration with Lillian Schwartz, a talented artist. Their many films have been clever, startling and powerful. I must say that they grow on you: I liked them at first, but when I saw five or six in a row this January, I found them just incredible. Because they are abstract, and full of fast-changing patterns and reversals, they take some adjusting to; but they're worth seeing over and over.

EXPLOR may be thought of as a highly generalized version of Conway's game of Life (see p. 48). You start with two-dimensional patterns as your data structure; these can be abstractions or even converted photographs, as in a recent Knowlton-Schwartz film showing Muybridge's Running Man. In your EXPLOR program, you may then cause the pattern to change by degrees, each cell of the pattern reacting to the cells around it or to random events as specified by the programmer.

EXPLOR, running without external data, comes up with some extraordinary snakeskin and Jack Frost patterns. But its uses in traffic simulation and various other studies of populations in space could be very interesting.

EXPLOR has obvious artistic applications. Lillian Schwartz is using it extensively in film-making. It's now running on a minicomputer feeding to a modified Sony Trinitron color TV. (This color setup was created by Mike Noll and is described in a recent issue of the CACM, though only for black-and-white TV; the color is more recent. It stores the color picture as a list of sequential colors represented in the computer's core memory, each dot being represented. Cf. "Boyell's Terrarium," p.DM38.)

Knowlton has used EXPLOR for teaching computer art at the University of California; the language is available programmed in "medium size" Fortran from Harry Huskey, Dept. of Information and Computer Science, U. of Cal. at Santa Cruz, Santa Cruz, California.



This is a non-simple picture conversion. The original photograph was converted into measured points; but these were in turn made into grow-together patterns by a program in the EXPLOR language. *© Knowlton & Harmon.*

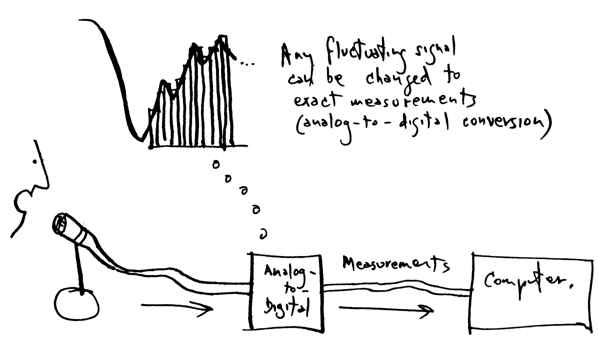
AUDIO

Wish there were room to talk about plain regular audio here— matters like "binaural" recording, and Why don't they make hi-fi systems based on a Grand Bus (see p. 42)? But there's no room here.

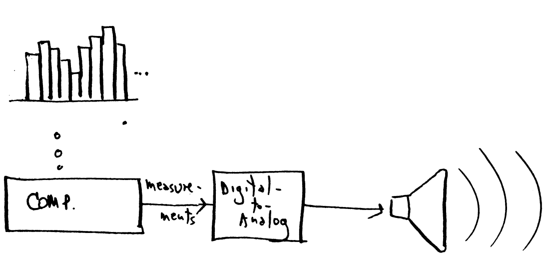
AUDIO AND COMPUTERS

People are occasionally still startled to hear that computers can make sound and music. They can indeed.

First of all, note that an incoming sound is a fluctuating voltage and can thus be turned into a data structure, i.e., a string of measurements.



To make sound by computer is the obverse. If the computer can be set up to send out a string of measurements, these can be turned back into a fluctuating voltage, and thus make sounds.



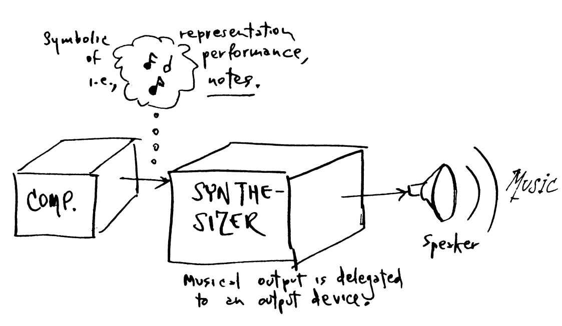
In the easiest case, the computer can just send back out the voltages it originally got in. This is rather ridiculous— using the computer just as a recording device— but it's a clear and simple example.

The question after that is what next: how to have the computer make interesting streams of output measurements, i.e., sounds and tones.

There are numerous methods we can't go into. Max Mathews, at Bell Labs, has for years been doing music by computer; his current system is called GROOVE. Heinz von Foerster, at the University of Illinois (Urbana), has been doing the same. Another lab at MIT has just gotten a PDP-11/45 (see p. 42) for the same purpose.

(The problem is: can the computer keep up with the output rate needed to make music in real time? maybe the 11/45 can.)

Another approach is to relieve the computer itself from making the tones, and use other devices— music synthesizers— for this, controlled by the computer. This is essentially the approach taken with General Turtle's Music Box (see p. 57), and at the Columbia-Princeton Electronic Music Center, where their RCA Mark II music synthesizer— an immense one-of-a-kind jobbie— is under more general computer control.



MUSICAL NOTATION

Note that the computer handling of musical notes, as symbols, is another task entirely, closely resembling computer text handling (mentioned variously in the book). A high-power structured-text system or Thinkertoy (see p. DM 52-5) is fine for storing and presenting written music.

And, of course, such stored musical notation (a data structure) can obviously be played by the hookups mentioned.

SPEECH BY COMPUTER

You may have heard about various kinds of "talking computer." This deserves some explanation.

Computers may be made to "talk" by various means. One is through an output device that simply stores recordings of separate words or syllables, which the computer selects with appropriate timing. (Machines of this type have been sold by both IBM and Cognitronics for a long time.)

A deeper approach is to have the computer synthesize speech from phonemes, or actually make the tones and noises of which speech is composed. These are very tricky matters. Bell Labs, and others, have been working on many of these approaches.

The real problem, of course, is how to decide what to say. (This was discussed under Artificial Intelligence, p. DM 12-14.)

AUDIO ANALYSIS AND ENHANCEMENT

The problem of analyzing audio is very like the problem of analyzing pictures (see p. DM 10), and indeed some of the same techniques are used. The audio goes into the computer as a stream of measurements, and the selfsame technique of Fourier Analysis is employed. This reduces the audio to a series of frequency measurements over time— but, paradoxically, loses little of the fidelity.

Once audio is reduced to Fourier patterns, it can be reconstituted in various ways: changed in timing and pitch independently, or enhanced by polishing techniques like those used in image enhancement (see p. DM 10).

This has been done with great success by Tom Stockham at the University of Utah, who has reprocessed old Caruso records into improved fidelity. In the picture we see him with equipment of some sort and an old record.

University of Utah

(Stockham has been in the news lately, as one of the panel puzzling over the notorious 18-Minute Gap.)

(The author has proposed the name Kitchensync™ for a system to synchronize motion pictures with "wild" sound recording by these means.)

BIBLIOGRAPHY

Thomas G. Stockham, Jr., "Restoration of Old Acoustic Recordings by Means of Digital Signal Processing." Audio Engineering Society preprint no. 831 (D-4), presented at Audio Engineering Society 1971 convention.

Prentiss H. Knowlton, "Capture and Display of Keyboard Music," Datamation May '72, 56-60. Describes a setup he built at U. of Utah that allows pianists to play music on an ordinary keyboard, and converts the input to symbolic representation in the computer. It uses an organ, a PDP-8 and a couple of CRT displays.

Heinz von Foerster and James Beauchamp, Music by Computers. Wiley, 1969. HAS RECORDS IN BACK.

Some of the early Bell Labs work may be heard on an excellent Decca LP with the misleading title "MUSIC from MATHEMATICS." (Decca DL 79103). (The mathematical myth is discussed on p. 8-9.)

time out for THREE COMPUTER DREAMS:

"AI" (Artificial Intelligence); "IR" (Information Retrieval); "CAI" (Computer-Assisted Instruction)

THE THREE DREAMS.

It's time for awe to be replaced with the critical eye.

These are three topics of great importance; of importance, unfortunately, less for what they have actually accomplished than for the degree to which they have confused and intimidated people who want to understand what's going on. Merely to mention them can be one-upmanship. All three titles mean so much, so many different specific things, as to mean almost nothing when lumped together as a whole. All three have developed a web of intricate technical facts (and sometimes theorems), but the applicability of these elegant findings is in all three cases a matter open to considerable scrutiny.

Since each of these fields has developed a considerable body of technical doctrine, the reader might well ask: why aren't they on the other side of the book, the computer side? The answer is that they are computerman's dreams, dreams of considerable intricacy and persuasiveness, and we are not considering the technicalities here anyway. As on the other side, the problem is to help you distinguish apples from oranges and which way is up. For more go elsewhere, but I hope this orientation will make sorting things out quicker for you.

These three terms-- "artificial intelligence," "information retrieval," "computer-assisted instruction"-- have a number of things in common. First, the names are so portentous and formidable. Second, if you read or hear anything in these fields, chances are it will have an air of unfathomable technicality. Both strange technicalism and deep mathematics may combine to give you a sense that you can't understand any of it. This is wrong. The fact that there are obscure and Deep Teachings in each has no bearing on the general comprehensibility of what they are about. More importantly, the question of how applicable all the things these people have been doing is going to be is a question of considerable importance, especially when some of these people want to take something over. Don't get snowed.

Each of these fascinating terms is actually a roof over a veritable zoo of different researchers, often of the most eccentric and interesting sort, each generally with his own dream of how his own research will be the breakthrough for humanity, or for something. It would take a Lemuel Gulliver to to show you the colorfulness and fascination of these fields; again, we just scratch the surface here.

Another interesting thing these three fields have in common: the frequent use of a classical computerman's putdown on anybody who dares question whether their super-ultimate goals can ever be achieved.

The line is, "WE DON'T KNOW HOW TO DO THAT YET."

If somebody pulls it on you, the reply is simply, "How do you know you ever will?"

ONE OF THE FEW GOOD LAYMEN'S COMPUTER JOKES

illustrating also certain problems of Artificial Intelligence.

A very large artificial-intelligence system (goes the story) had been built for the military to help in long-range policy planning; financed by ARPA, with people from M.I.T., Stanford and so on.

"The system is now ready to answer questions," said the spokesman for the project.

A four-star general bit off the end of a cigar, looked whimsically at his comrades and said--

"Ask the machine this: Will it be Peace or War?"

The clerk-typist (Sp4) translated this into the query language and typed it in.

The machine replied:

YES

"Yes what?" bellowed the general.

The operator typed in the query.

Came the answer:

Yes SIR

THE GOD-BUILDERS!

ARTIFICIAL INTELLIGENCE ... sort of

"Artificial Intelligence" is at once the sexiest and most ominous term in the world. It chills and impresses at the same time. In principle it means the simulation of processes of mind, by any means at all; but it generally turns out to be some form or another of computer simulation (see "Simulation," p. 58). Actually, "artificial intelligence" has generally become an all-inclusive term for systems that amaze, astound, mystify, and do not operate according to principles which can be easily explained. In a way, "artificial intelligence" is an ever-receding frontier: as techniques become well-worked out and understood, their appearance of intelligence, to the sophisticated, continually recedes. It's like the ocean: however much you take out of it, it still stretches on-- as limitless as before.

Unfortunately laymen are so impressed by computers in general that they easily suppose computers can do anything involving information. And public understanding is not fostered by certain types of stupid demonstration. One year I heard from numerous people about how "they'd seen on TV about how computers write TV scripts"-- what had actually been shown was a hokey enactment of how the computer could randomly decide whether the Bad Man gets shot or the Good Guy gets shot-- both outcomes dutifully enacted by guys in cowboy outfits. Duh.

It should be perfectly obvious to anybody who's brushed even slightly with computers, however-- for The Brush, see the other side-- that they just don't work like minds. But the analogy hangs around. (Edmund C. Berkeley wrote a book in the forties, I believe, with the misleading title of Giant Brains, or Machines That Think. The idea is still around.)

Here's a very simple example, though. Consider a maze drawn on a piece of paper. Just by looking, we cannot simultaneously comprehend all its pathways; we have to poke around on it to figure out the solution. Computers are sort of like that, but more so. While our eyes can take in a simple picture, like a square, at once, the computer program must poke around in its data representation at length to see what we saw at once.

The principle holds true in general. The human mind can do in a flash, all at once (or "in parallel") many things that must be tediously checked and tried by the highly sequential computer program. And the more we know about computers, the more impressive the human brain becomes. (The seeming cleverness of some simple programs does not prove the simplicity of the phenomena being imitated.)

Nevertheless, it is interesting to try things with computers that are more like what the mind does; and that is mostly what artificial intelligence is about.

In various cases this has resulted in helpful tricks that turn out to be useful elsewhere in the computer field. In this sense, artificial intelligence is sort of like menthol: a little may improve things here and there. But (in my opinion), that does not mean a whole lot of it would make things better still.

Nevertheless, some artificial-intelligence enthusiasts think there is no limit on what machines can do. They point out that, after all, the brain is a machine. But so is the universe, presumably; and we're never going to build one of those, either.

PATTERN RECOGNITION

This is one of the most active areas in artificial intelligence, perhaps because of Defense Department money. (It might be nice, goes the reasoning, to have guns that could recognize tanks, machines that could look over aerial reconnaissance pictures, radars that could recognize missiles...)

What it boils down to is the study of clues and guessing among alternatives. In some cases, well-defined clues can be found for recognizing specific things, like parts of pictures (even straight lines cannot be recognized by computer without a complex program) or like handwriting (see below). In the worse cases, though, careful study only raises the most horrendous technical problems, and the pursuit of these technical problems is its own field of study (articles have titles like "Sensitivity Parameters in the Adjustment of Discriminators," meaning It Sure Is Hard to Draw The Line).

But in some felicitous cases, researchers actually boil a recognition problem down to a manageable system of clues. For instance, take the problem of written input to computers. (Some people don't like to type and would rather write by hand on special input tablets.) But how can a program recognize the letters? Aha: the answer, kids, is in your text.

The Ledeen Character Recognizer (described in detail in Newman and Sproull, Principles of Interactive Computer Graphics, Appendix 8) is a method by which a program can look at a hand-drawn character and try to recognize it. The program extracts a series of "properties" for the character and stores them in an array. Every character in a given person's block lettering will tend to have certain property scores. But the Ledeen recognizer must still be trained, that is, the average property scores of the letters that each individual draws must be put into the system before that individual's lettering can be recognized. Even then it's a question of probability, rather than certainty, that a given character will be recognized.

COMPUTERS DON'T ACTUALLY THINK.

YOU JUST THINK THEY THINK.

(We think.)

HEURISTICS (pronounced hewRIStics)

If we want to make a computer do what we know perfectly well how to do ourselves, then all we do is write a program.

Aha. But what if we want a computer to do something we do not know how to do ourselves?

We must set up its program to browse, and search, and seize on what turns out to work.

This is called heuristics.

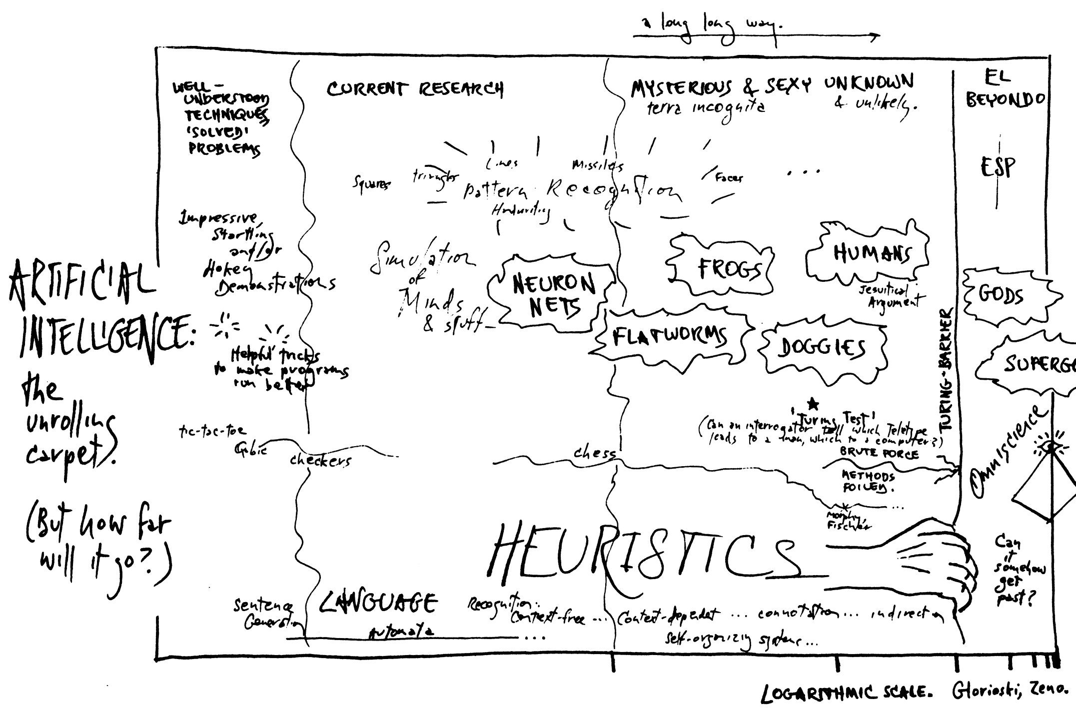
What it amounts to basically is techniques for trying things out, checking the results, and continuing to do more and more of what seems to work.

Or we could phrase it this way: looking for successful strategies in whatever area we're dealing with. As a heuristic program tries things out, it keeps various scores of how well it's doing-- a sort of self-congratulation-- and makes adjustments in favor of what works best.

Thus the Greenblatt Chess Program, mentioned under "Chess," nearby, can "invent" chess strategies and "try them out"-- what it actually does is test specific patterns of moves for the overall goodness of their results (in terms of the usual positional advantages in chess), and discard the strategies that don't get anywhere. It does this by comparing its "strategies" (possible move patterns) against the records of chess matches which are fed into it.

(If you've read the other side of the book, heuristics may be thought of as a form of operations research (p. 58) carried on by the computer itself.)

In some ways heuristics is the most magical area of artificial intelligence: its results are the most impressive to laymen. But, like so many of the computer magics, it boils down to technicalities which lose the romance to a certain extent.



NEURAL SIMULATION

An important branch of Artificial Intelligence is concerned with what bunches of imaginary neurons could do, even neurons that we made up to follow particular rules. This area of study is somewhere between neurology and mathematics; much of it is concerned with the mathematics of imaginary setups, rather than the properties of actual nerve-nets, as studied by psychologists, physiologists and others. (The hypothetical studies, of course, alert researchers to complex configurations and possibilities that may turn out to occur in reality, as well as being interesting for their own sake-- and conceivably as useful ways of organizing things to be built.)

However, an earlier myth, that you could simulate neurons till you got a person, is about dead.

SIMULATION OF THOUGHT-PROCESSES

Nobody talks anymore about simulating artificial brains; there's too much to it, and it involves dirty approximations.

However, a cleaner area is in the simulation of thought: creating computer programs that mimic man's mental processes as he dopes through various problems. Trying things out, deducing thoughts from what's already known, following through the consequences of guesses-- these can all be done by programs that "try to figure out" answers to problems like The Cannibal and The Missionary, or whatever.

AUTOMATA

"Automata", as the term is used in this field, is just a fancy word for imaginary critters, particularly little thingies that behave in exact ways. (The Game of Life, see p. 48, is an automaton in this sense.)

SELF-ORGANIZING SYSTEMS, SELF-REPRODUCING SYSTEMS, AND SO FORTH

These are terms for imaginary objects, having exactly defined mathematical properties, about which various abstract things can be proven that tend to be of interest only to mathematicians.

SPEECH

1. SENTENCE GENERATION

The problem of computers speaking human languages-- not to be confused with computer languages, pp. 15-25 and elsewhere-- is incredibly complicated. Just because little human tykes start doing it effortlessly, it is easy to suppose that it's a basically easy problem.

No way.

Only since the mid-fifties has human language begun to be understood. That was when Noam Chomsky discovered the inner structure of human languages: namely, that the long (and complex) sentence constructions of language are built out of certain exact operations. Previous linguists had sought to classify the sentence structures themselves; this led to complexities which Chomsky discovered were unnecessary. It is unnecessary to catalog sentence types themselves if we can simply isolate, instead, the exact processes by which they are generated.

These processes he called transformations (a term he borrowed from mathematics). All utterances are created from certain elementary pieces, called kernels, which are then chewed by transformations into surface structures, the final utterances. Examples of kernels: The man lives in the house, The house is white. Result of combining transformation: The man lives in the white house. Kernel: I go. Result of past-tense transformation: I went.

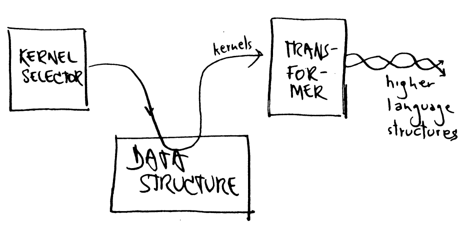
The most important finding, now, is that the transformations are carried out in orderly sequences: any sentences can have more transformations carried out on it, all adhering to the basic rules, resulting in the most complex sentences of any language.

Linguists since then have confirmed Chomsky's conjecture, and proceeded to work out the fundamental transformations of major languages, including English.

Now, one result of all this is that it turns out to be easier to generate sentences in a language than to understand them. Why? Because it is comparatively easy to program computers to apply transformations to kernels, BUT very hard to take apart the result. A complex "surface structure" may have numerous possible kernels-- does "Time flies like an arrow" have the same structure as "Susie sings like a bird" or "Fruit flies like an orange?"

Result: to program a computer to generate speech-- that is, invent sentences about a data structure and type them out-- is comparatively easy, but to have it recognize incoming sentences, and break them up into their kernel meanings, is not.

We may think of a language-generating computer system as follows:



2. SENTENCE RECOGNITION

Chomsky and others have discovered that sets of transformation rules (or grammars, praise be) vary considerably. It is possible to invent languages whose surface structures are easy to take apart, or parse; such languages are called context-free languages. (Most computer languages, see other side, are of this type.) Unfortunately natural languages, like English and French and Navaho, are not context-free. It turns out that the human brain can pick apart language structures because it's so good at making sensible guesses as to what is meant-- and if there is one thing hard to program for computers, it is sensible guessing. (But see "Heuristics," nearby.)

This means that to create computer systems which will take real sentences apart into their meanings is quite difficult. We can't get into the various strategies here; but most researchers cut the problem down in one way or other.

Dorothy read the card aloud, spelling out the big words with some difficulty; and this is what she read:

SMITH & TINKER'S
Patent Double-Action, Extra-Responsive,
Thought-Creating, Perfect-Talking
MECHANICAL MAN
Fitted with our Special Clock-Work Attachment.
Thinks, Speaks, Acts, and Does Everything but Live.
Manufactured only at our Works at Evna, Land of Ev.
All infringements will be promptly Prosecuted according to Law.

"How queer!" said the yellow hen. "Do you think that is all true, my dear?"

55

Ozma of Oz

"I don't know," answered Dorothy, who had more to read. "Listen to this, Billina:"

DIRECTIONS FOR USING:
For THINKING:—Wind the Clock-work Man under his left arm, (marked No. 1.)
For SPEAKING:—Wind the Clock-work Man under his right arm, (marked No. 2.)
For WALKING and ACTION:—Wind Clock-work in the middle of his back, (marked No. 3.)
N. B.—This Mechanism is guaranteed to work perfectly for a thousand years.

"Well, I declare!" gasped the yellow hen, in amazement; "if the copper man can do half of these things he is a very wonderful machine. But I suppose it is all humbug, like so many other patented articles."

"We might wind him up," suggested Dorothy, "and see what he'll do."

GORDON PASK

Gordon Pask is one of the maddest mad scientists I have ever met, and also one of the nicest. An eloquent English leprechaun who dresses the Edwardian dandy, Pask sows awe wherever he goes. A former doctor and theatrical producer, Pask is one of the great international fast-talkers, conference-hopping round the globe from Utah to Washington to his project at the Brooklyn Children's Museum. This spring, 1974, he has been at the University of Illinois at Chicago Circle, but soon he goes back to England and his labòratory.

In a field full of brilliant eccentrics, Pask has no difficulty standing out.

Pask is one of the Artificial Intelligencers who is working on teaching by computer, about which more will be said; but the original core of his interest is perhaps the process of conceptualization and abstraction.

Pask has done a good deal on the mathematics of self-contemplating systems, that is, symbolic representations of what it means for a creature (or entity omega) to look at things, see that they are alike, and divine abstract conceptions of them. A crowning moment is when Omega beholds itself and recognizes the continuity and selfhood. (Pask says several others-- scholars from Argentina, Russia and elsewhere-- have hit on the same formulation.)

Models and abstraction, then, are what we may call the first half of Pask's work.

Gordon Pask will be continued on p. DM 47.

3. SPEECH OUTPUT AS SOUND

It is possible in principle to set up computers to "talk" by converting the language surface structures that their programs come up with into actual sound. See "Audio," p. DM 11.

4. SPEECH INPUT TO COMPUTERS BY ACTUAL SOUND

So far we have been talking about the computer's manipulation of language as an alphabetical coding or similar representation. To actually talk at a computer is another kettle of fish. This means breaking down the sound into phonemes and then breaking it into a data structure which can be treated with the rules of grammar-- a whole nother difficult step.

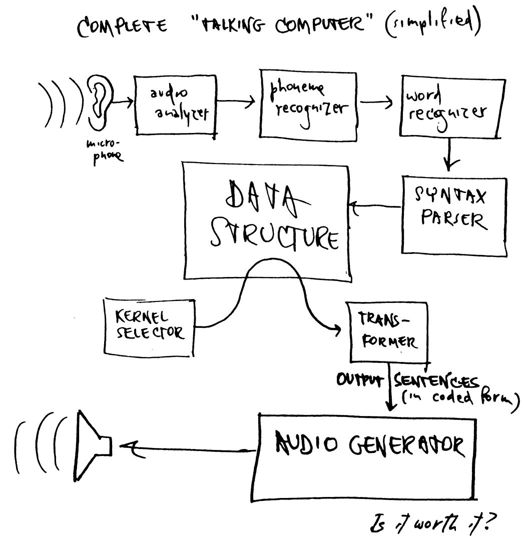
A few attempts have been made to market devices which would recognize limited speech and convert it to symbols to go into the computer. One of them, which supposedly can distinguish among thirty or forty different spoken words, is supposedly still on the market. Specific users have to "train" it to the particulars of their voices.

I repeatedly hear rumors of "dictation machines" which will type what you say to them. If such things exist I have been unable to confirm it.

(Everybody says that of course what we want is to be able to communicate with computers by speech. Speaking personally, I certainly don't. Explaining my punctuation to human secretaries is hard enough, let alone trying to tell it to a computer, when it's easy enough to type it in.)

5. ALL TOGETHER NOW

The complexity of the problem should by now be clear.



CYBERNETICS

Gordon Pask calls his field Cybernetics. The term "cybernetics" is heard a lot, and is one of those terms which, in the main, mankind would be better off without; although after talking to Pask I get the sense that there may be something to it after all.

The term "cybernetics" was coined by Norbert Wiener, the famously absent-minded mathematician who (according to legend) often failed to recognize his own children. Wiener did pioneering work in a number of areas. A special concern of his was the study of things which are kept in control by corrective measures, or, as he called it, Feedback. The term "cybernetics" he made out of a Greek word for steersman, applying it to all processes which involve corrective control. It turns out that almost everything involves corrective control, so the term "cybernetics" spreads out as far and as thinly as you could possibly want (The public is under the general impression that "cybernetics" refers to computers, and the computer people should be called "cyberneticians." There seems to be nothing that can be done about this. See "cybercrud," p. 8. This is an even worse term meaning "steering people into crud," specifically, putting things over on people using computers.)

Properly, the core of "cybernetics" seems to deal with control linkages, whether in automobiles, cockroaches or computers. However, people like Pask, von Foerster, Ashby (and so on) appear to extend the concept generally to the study of forms of behavior and adaptation considered in the abstract. The validity and fascination of this work, of course, is quite unrelated to what you call it.

THE TURING MACHINE

is the most classical abstract Automaton. A Turing Machine, named after its discoverer, is a hypothetical device which has an infinite recording tape that it can move back and forth, and the ability to make decisions depending on what's written there.

Turing proceeded to point out that no matter how fast you go step-by-step, you can't ever outrun certain restrictions built into all sequential processes as represented by the Turing Machine. This lays heavy limits on what can ever be done step-by-step by computer. (It means we have to look for non-step-by-step methods, which much of Artificial Intelligence is about.)

DO WE WANT TALKING SYSTEMS?

I had one quite irritating experience with a "conversational" system, that is, computer program that was supposed to talk back to me. I was supposed to type to it in English and it was supposedly going to type back to me in English. I found the experience thoroughly irritating. My side of the conversation, which I sincerely tried to keep simple, produced repeated apologies and confusion from the program. The guy who'd created the program kept explaining that the program would be improved, so that eventually it could handle responses like mine. My reaction was, and is, Who needs it?

Many people in the computer field seem to think we want to be able to talk to computers and have them talk back to us. This is by no means a settled matter.

Talking programs are complicated and require a lot of space in the machine, and (more importantly) require a lot of time by programmers who could achieve (I think) more in less time by other means. Moreover, talking programs produce an irritating strategical paradox. In dealing with human beings, we know what we're dealing with, and can adjust what we say accordingly; there is no way to tell, except by a lot of experimenting, what the principles are inside a particular talking program; so that trying to adjust to it is a strain and an irritation. (Compare: talking to a stranger who may or may not turn out to be your new boss.) Now, some programmers keep saying that eventually they'll have it acting just as smart as a real person, so we needn't adjust; but that's ridiculous. We always adjust to real people. In other words, the human discomfort and irritation of psyching the system out can never be eliminated.

Furthermore, on today's sequential equipment and with feasible budgets, I personally think the likelihood of making programs that are really general talkers is a foolish goal. There are many simpler ways of telling computer systems what you want to tell them— light pen choice, for example.

Moreover, having to type in whole English phrases can be irritating. (We can't even get into the problem of having the computer pick apart the audio if you talk it in.)

This is not to say understandably restricted talking systems are bad. If you know and understand the sorts of response the system makes to what kinds of thing, then an English-like response is really a clear message. For instance, the JOSS system (the first Quickie language— see p. 15) had an eloquent message:

eh?

which actually meant, What you have just typed in does not fit the rules of acceptable input for this system. But it was short, it was quick, it was simple, and it was almost polite.

Similarly, talking systems that use an exact vocabulary, whose limits and abilities are known to the person, are okay. (Winograd, see Bibliography, has a nice example of telling a computer to stack blocks, where the system knows words like between, on, above and so on.) Where this is understood by the human, it can be a genuine convenience rather than a spurious one.

(The problem of rudeness in computer dialogue has not been much discussed. This is partly because many programmers are not fully aware of it, or, indeed, some are so skilled in certain subtle forms of rudeness they wouldn't even know they weren't acceptable. The result is that certain types of putdown, poke, peremptoriness and importunacy can find their way into computer dialogue all too easily. Or, to put it another way: nobody like to be talked back to. Cf. Those stupid green THANK YOU lights on automatic toll booths.)

Now, this is not to say that research in these areas is wrong, or even that researchers' hopes of some breakthrough in talking-systems is misguided. I am saying, basically, that talking systems cannot be taken for granted as the proper goal in computers to be used by people; that the problems of rudeness, and irritating the human user, are far greater than many of these researchers suppose; and that there may be alternatives to this potentially eternal leprechaun-chasing.

If like the author you are bemused by the great difficulty of getting along with human beings, then the creation of extraneous beings of impenetrable character with vaguely human qualities can only alarm you, and the prospect of these additional crypto-entities which must be fended and placated, clawing at us from their niches at every turn, is both distasteful and alarming.

Artificial Intelligence enthusiasts unfortunately tend to have a magician's outlook: to make clear how their things work would spoil the show.

Thus, for a rather peculiar art show held at New York's Jewish Museum in 1970, a group from MIT built a large device that stacked blocks under control of a minicomputer (Interdata brand). Now, the fact that it could stack and re-stack blocks with just a minicomputer was really quite an accomplishment, but this was not explained.

Instead, the block-stacking mechanism was enclosed in a large glass pen, in which numerous gerbils— hoppy little rodents— were free to wander about. When a gerbil saw that a block was about to be stacked on him, he would sensibly move.

Now, it is fairly humorous, and not cruel, to put gerbils into a block-stacking machine. But this was offered to the public as a device partaking of a far more global mission, the experimental interaction of living creatures and a dynamic self-improving environment, blah blah blah.

Passersby were awed. "Why are those animals in there?" one would say, and the more informed one would usually say, "It's some kind of scientific experiment."

Well, this is a twilight area, between science and whimsical hokum, but one cannot help wishing simple and humorous things could be presented with their simplicity and humor laid bare.

I remember watching one gerbil who stood motionless on his little kangaroo matchstick legs, watching the Great Grappler rearranging his world. Gerbils are somewhat inscrutable, but I had a sense that he was worshiping it. He did not move until the block started coming down on top of him.

I take this as an allegory.

CAN A COMPUTER PLAY CHESS?

The real question is, can a set of procedures play chess? Because that's what the computer program really does, enact a set of procedures.

And the answer is yes, fairly well.

Now, a chess program is not something you jot down on the back of an envelope one afternoon. It's usually an immense, convoluted thing that people have worked on for years. (Although I vaguely recall that second place in the 1970 inter-computer chess contest was won by a program that occupied only 2000 locations in a 16-bit minicomputer— in other words, a compact and tricky sneaker.)

Now, simple games (like tic-tac-toe and Nim and even Cubic) can be worked out all the way: all alternatives can be examined by the program and the best one found. Not so with chess.

Chess basically involves, because of its very many possibilities, a "combinatorial explosion" of alternatives (see p. 45): that is, to look at "all" the possibilities of a midgame would take forever (perhaps literally— the Turing problem), and thus means must be found for discarding some possibilities.

The structure of branching possibilities is a tree (see p. 26); so that methods of "pruning" the tree turn out to be crucial.

Basically there are two approaches to the design of chess programs. In one approach, the programmers look for specific threats and opportunities in the data structure representing the board, and try to find good strategies for selecting good moves on the basis of them. This is the approach taken in COKO, the "Cooper-Koz" chess program. The programmers selectively cope with individual problems and strategies as they turn out to be necessary. (This means that it is likely to have specific Achilles' heels; which, of course, the authors of the program keep trying to repair by adding specific corrections.)

A different approach is taken by the Greenblatt chess program. This is basically a big Heuristic program. It "learns" best strategies in chess by "watching" the game. That is, your pour historical chess matches through it, and it tries out strategies— making various tentative rules about what kinds of moves are good, then scoring these moves according to the results of making them— as seen in positional advantages that resulted in actually championship play.

Obviously this is a field in itself. You won't get grants for it, but to those who really care about both chess and computers, it's the only thing to be doing.

FRANKENSTEIN MEETS CYBERCRUD

Fred Brooks, the keynote speaker at the IEEE computer conference in Fall 74, seems to have said that HAL 9000 (the unctuous, traitorous Presence in the movie 2001) was the way computers should be. (Computer Decisions, Apr 74, 4.)

I find it hard to believe that anybody could think that. Nevertheless, there are those artificial-intelligence freaks whose view it is that the purpose of all this is eventually (a) to create servants that will read our minds and do our bidding, (b) servants who will take things over and will implement human morality, regardless of our bidding (though we humans are too frail to do so— as in Asimov's I, Robot); or even (c) create masters who will take everything over and run everything according to their own principles and the hell with us. (I met a man in a bar, after an ACM meeting, who claimed to believe this was the purpose of it all: to create the master race that would replace us.)

According to Arthur C. Clarke's retroactive novel 2001: A Space Odyssey (Signet, 1968, 95¢), the HAL 9000 computer series began as follows:

> "In the 1980s, Minsky and Good had shown how neural networks could be generated automatically-- self-replicated-- in accordance with any arbitrary learning pattern. Artificial brains could be grown by a process strikingly analogous to the development of the human brain." (P. 96.)

I don't know who Good is, but these are among the lines Minsky has been working along for years, so I hope he's encouraged by the news of what he's going to accomplish.

Anyhow, so okay they grow the HAL 9000 in a tank. Then how come in the Death-of-Hal scene we see Keir Dullea bobbing around loosening circuit cards, just as if it were a plain old 1978 computer?

Possible answer #1. It is rumored that Clarke's retro-novel was Clarke's rebuttal to Kubrick's final film.

Possible answer #2. HAL's tanks of neural glop are controlled by PDP-11s, one to a card.

(Of course, if you take the letters after H, A and L in the alphabet, you get I, B and M. So maybe those are 1130s.)

DEUS EX MACHINA

Obviously such beliefs are outside the realm of science or engineering. They belong to pure speculation; and while various mechanisms have in fact been programmed to croak, stagger, stack blocks, compose sentences and so on, to suppose that we are in any real sense anywhere near mimicking human intelligence, let alone surpassing and superseding it, is either to be totally fooled or to hanker after some curious dream from inside yourself.

As we said on the other side of the book, everybody in computers has deeper motivations and interior twists that form his own special ties to these machines; and when it comes to our choices of fantasy machines, obviously an even deeper level of psychic imprint is projecting itself into the world.

...EX MENSA

People who fantasize about wondrous creatures and deities they want to make out of the computer obviously have something interesting in their own heads from which that comes. Perhaps it comes from a desire for imaginary playmates, or an ambivalence toward authority, or goodness knows what; there are so many odd people at different ends of Artificial Intelligence that there may be a lot of different psychological systems at work. Or maybe artificial intelligence is just where the most brilliant, determined and eccentric dreamers go. Anyway, I can only ask the question, not give the answer.

THE THREE LAWS OF ROBOTICS

1-- A robot may not injure a human being, or, through inaction, allow a human being to come to harm.

1-- A robot must obey the orders given it by human beings except where such orders would conflict with the First Law.

3-- A robot must protect its own existence as long as such protection does not conflict with the First or Second Law.

-- Handbook of Robotics, 56th Edition, 2058 A.D., quoted in I. Asimov, I, Robot.

Sometimes All We Need Is A Sympathetic Teletype

ELIZA WILL SEE YOU NOW

(What the ELIZA program types, we represent as all caps; what the person types is not.)

PLEASE TELL ME YOUR PROBLEM.
  I am feeling depressed.
HOW LONG HAVE YOU BEEN FEELING DEPRESSED?
  Ever since my mother left.
TELL ME MORE ABOUT YOUR FAMILY.
  My father is very strict.
PLEASE GO ON.
  Last night I dreamt he spanked me.
WHAT WOULD IT MEAN TO YOU IF HE SPANKED YOU?
  Didn't you ever get spanked?
WE WERE DISCUSSING YOU, NOT ME.
  You're just a dumb machine anyway.
DO COMPUTERS FRIGHTEN YOU?
  (quoted in Winograd, 74-5)

The fascinating thing about the ELIZA program is that it seems so like a real person— that is, a psychoanalyst of the nondirective, Carl Rogers school.

But all it is is a bag of tricks.

It has a lot of canned phrases it can throw out, like PLEASE TELL ME YOUR PROBLEM.

Then it spots certain types of sentences and turns them around. If the person types a sentence beginning with "I am...", the ELIZA program turns it around with HOW LONG HAVE YOUR BEEN... ?

And it looks for specific words which indicate a sentence can be turned around for a convincing result. For instance, if the person uses the word "everyone," the program simply replies, WHO IN PARTICULAR CAN YOU THINK OF?

And certain highly-charged words are signals to throw in a zinger, regardless of what the user may have said. For instance, if the person uses the word "mother," the machine replies:

TELL ME ABOUT YOUR FAMILY.

Now, do not be fooled. There is no Ghost in the Machine. The program does not "understand" the user. THE PROGRAM IS MERELY ACTING OUT THE BAG OF TRICKS THAT JOE WEIZENBAUM THOUGHT UP. Credit where credit is due: not to The Computer's Omniscience, but to Weizenbaum's cleverness.

(Look at the above sample dialogue and see if you guess what tricks the program was using.)

The thing is, many people refuse to believe that it's a program. Even when the program's tricks are explained.

And even some who understand ELIZA like to call it up from their terminals for companionship, now and then.

BIBLIOGRAPHY

Terry Winograd, "When Will Computers Understand People?" Psychology Today May 74, 73-9.

(Weizenbaum's full article on ELIZA appeared in the Communications of the ACM sometime in the mid or late sixties; a flowchart revealed its major tricks.

I have strong hunches about the inner workings of men who get millions of dollars from the Department of Defense and then say in private that really they're going to use it to create a machine so intelligent it can play with their children. (Not to name names or anything.) An obvious question is, do they play with their children? No, they play with computers.

But the point here is not to hassle the dreamers, just to sort out the dreams and put them on hangers so you can try them on, and maybe choose an ensemble for yourself.

BIBLIOGRAPHY

Terry Winograd, "When Will Computers Understand People?" Psychology Today May 74, 73-79.
  Particularly readable article.
Nicholas Negroponte, The Architecture Machine. MIT Press.
  Takes the view that computers should be made into magical servants which not only handle bothersome details, but more or less read our minds as well.
Leonard Uhr, Pattern Recognition, Learning and Thought: Computer-Programmed Models of Higher Mental Processes. Prentice-Hall.
Arthur W. Holt, "Algorithm for a Low Cost Hand Print Reader." Computer Design Feb 74, 85-89.
Edward A. Feigenbaum and Julian Feldman (eds.), Computers and Thought. McGraw-Hill.
  Old but still good for orientation.
A journal: Artificial Intelligence (North-Holland Publishing Co., Journal Division; P.O. Box 211, Amsterdam, The Netherlands. Was $26.50 a year in 1973.
  This'll show you what they're thinking about now.
Roger Lind, "The Robots Are Coming, The Robots Are Coming." Oui, Feb 1974.
  Typical layman's hype. You don't get told until the second page that a typical industrial "robot" is a huge mechanism with one grappling arm.
Edward W. Kozdrowicki and Dennis W. Cooper, "COKO III: the Cooper-Koz Chess Program." CACM July 73, 411-427.
Greenblatt, R.D., Eastlake, D.E., and Crocker, S.D., "The Greenblatt Chess Program." Proc FJCC 67, 801-810.
R.C. Gammill, "An Examination of Tic-Tac-Toe-like Games." Proc. NCC 74, 349-355.

INFORMATION RETRIEVAL

"Information Retrieval" is one of those terms that laymen throw around as if it were a manhole cover. It sounds as though it means so much, so very much. And so you actually hear people say things like: "But that would mean... (pregnant pause) ... Information Retrieval!!!" Similarly, some of the hokey new copyright notices you see in books from With-It publishers intone that said books may not be "placed in any information retrieval system..." I take this to mean that the publishers are forbidding you to put the book on a bookshelf, because "information retrieval" simply means any way at all of getting back information from anything. A bookshelf, since it allows you to read the spines of the books, is indeed an Information Retrieval System.

It happens, incidentally, that the phrase "information retrieval" was coined in the forties by Calvin Mooers, inventor of TRAC^{tm} Language (see pp. 18-21). (If Wiener had coined it he might have called it Getback. If Diebold had coined it it might have been Thoughtomation.)

Anyhow, numerous entirely different things go on in the field, all under the name of Information Retrieval. Here are some.

1. Non-computer retrieval. (See Becker and Hayes, Automatic Information Retrieval.) These things are kind of old-fashioned fun-- cards with holes punched along the edge, for instance, that you sort with knitting needles, or the more recent systems with holes drilled in plastic cards. Trouble is, of course, that computers are becoming much more convenient and even less expensive than these, counting your own time as being worth something.

2. Document Retrieval. This basically is an approach that glorifies the old library card file, except now the stuff is stored in computers rather than on cards. But what's stored is still the name of the document, who wrote it, where it was published and so on. Obviously helpful to librarians, but scarcely exciting.

3. Automatic document indexing. Some organizations find it helpful to have a computer try to figure out what a book is about, rather than have a person look at it and check. (I don't see why this saves anything, but there you are.) Anyway, the text of the document (or selected parts) are poured through a computer program that selects, for instance, keywords, that is, the most important words in it, or rather words the program thinks are most important. Then these keywords can go on the headings of library file cards, or whatever.

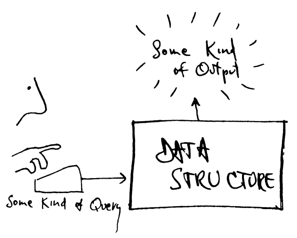
There are various related systems by which people study, for instance, the citations between articles, but we won't get into that.

4. Content retrieval. Now we're getting to the sexy stuff. A system for content retrieval is one that somehow stores information in a computer and lets you get it back out.

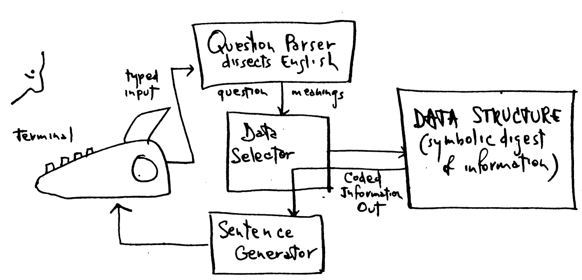
The trick on both counts is of course how.

Well, as we said on the other side of the book, any information stored in a computer has a data structure, which simply means whatever arrangement of alphabetical characters, numbers and special codes the computer happens to be saving.

In a content-retrieval system, information on some subject is somehow jammed into a data structure-- possibly even by human coders-- and then set up so people can get it back out again in some way. Lot of possibilities here, get it?

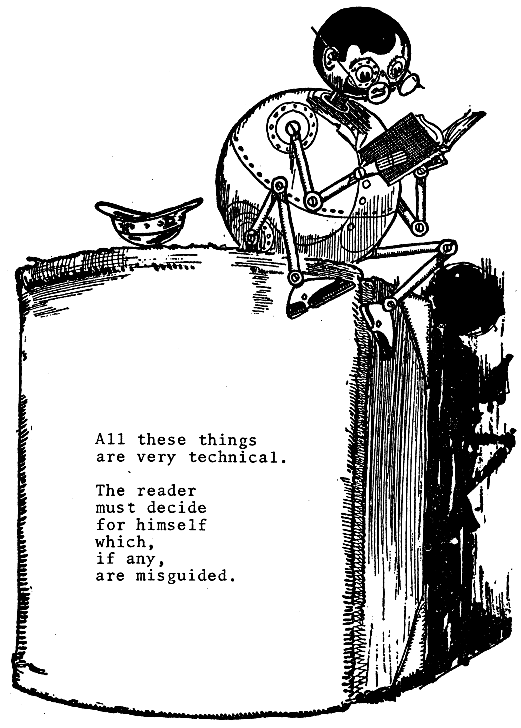


In the most startling of these systems, the QAS, or "Question-Answering System," some sort of dialogue program (see "Artificial Intelligence," nearby) tries to give you answers about the data structure. But this means there have to be a whole lot of programs:



These systems can be quite startling in the way they seem to understand you (see Licklider book; also Winograd piece under Artificial Intelligence). But they don't understand you. They are just poor dumb programs.

Many people (including Licklider) seem to see in Question-Answering-Systems the wave of the future. Others, like this author, are skeptical. It's one thing to have a system that can deduce that Green's House is West of Red's House from a bunch of input sentences on the subject, but the question of how much these can be improved is in some doubt. A system that can answer the question, "What did Hegel say about determinism?" is some ways away, to put it mildly.



Then there is the matter of consistency. The really interesting subjects are the ones where different authors claim opposing facts to support opposite conclusions. In other words, there is inconsistency within the content of the field. In this case such systems are going to have a problem. (See "Rasho-Mon Principle" under "Tissue of Thought," pp. DM 16-19.)

Another fundamental point is this. It may be easy enough to program a system to answer the question,

WHAT TIME DOES THE NEXT PLANE LEAVE FOR LAGUARDIA?

but it is a lot simpler to have it display schedules your eye can run down, or allow you to go look at some kind of graphic display.

Speaking personally, I don't like talking to machines and I don't like their talking back to me. I'm not saying you have to agree, I'm just telling you you're allowed to feel that way.

5. Screen summaries. These systems let you sit at a computer display screen and read summaries of various things, as well as run through them with various programs to look for keywords. (The New York Times now offers such a system, costing over a thousand dollars a month to subscribers.)

6. "Full-text systems." These are systems that one way or another allow you to read all the text of something from a computer display screen. There are those of us who see these as the wave of the future, but many others are perfectly outraged at the thought. (Hypertext systems, now, are setups that allow you to read interconnected texts from computer display screens. See pp. DM 44-7.)

This has been brief and has skipped a lot. Anyway, as you see, IR is no one thing.

BIBLIOGRAPHY

Vannevar Bush, "As We May Think." Atlantic Monthly, July 1945, 101-108.
Theodor H. Nelson, "As We Will Think." Proc. Online 72 Conference, Brunel U. Uxbridge, England.
G. Salton, "Recent Studies in Automatic Text Analysis and Document Retrieval." JACM, Apr 73, 258-278.
Donald E. Walker (ed.), Interactive Bibliographic Search: The User/Computer Interface. AFIPS Press, $15.
Theodor H. Nelson, "Getting It Out of Our System." In Schechter (ed.), Critique of Information Retrieval (Thompson Books, 1967).
J.C.R. Licklider, Libraries of the Future. MIT Press, 1965. Clear and readable summary of the rest of the field; then he goes on to advocate "procognitive systems," systems that will digest what's known in any field and talk back to you, using techniques of artificial intelligence.
Whatever its other merits, this book is great for shaking people up, especially librarians. It seems so official.
Richard M. Laska, "All the News That's Fit to Retrieve." Computer Decisions, Aug 72, pp. 18-22.

It is a truism that Mendel's theories of genetics got "lost" after publication in 1865, to be rediscovered in 1900. "If only there had been proper information retrieval under the right categories," people often say. Recent studies indicate that the publication containing Mendel's paper reached, or got nearly to, "practically all prominent biologists of the mid-nineteenth century." (Scientific American, July 68, 55.)

I take this as suggesting that the problem isn't categorical retrieval at all. It's multi-connected availability (see "hypertext," pp. DM 44-7.

COMPUTER-ASSISTED INSTRUCTION

Like Artificial Intelligence and Information Retrieval, Computer-Assisted Instruction sounds like something exact and impressive but is in fact a scattering of techniques tied together only nominally by a general idea.

The real name for it should be Automated Dialogue Teaching. That would immediately allow you to ask, should computer teaching use dialogues? But they don't want you to ask that.

In the classic formulation of the early sixties, there were going to be three levels of CAI: "drill-and-practice" systems, much like teaching machines, that simply helped students practice various skills; a middle level (often itself called, confusingly, "computer-assisted instruction"); and a third level, the Socratic system, which would supposedly be Ideal. Students studying on Socratic systems would be eloquently and thoughtfully instructed and corrected by a perfect being in the machine. "We don't know how to do that yet," the people keep saying. Yet, indeed.

(My personal view on this subject, expressed in an article (following) is that Computer-Assisted Instruction in many ways extends the worst features of education as we now know it into the new realm of presentation by computer.)

DOES THE NAME PAVLOV RING A BELL?

This is a true story. (The details are approximate.) It may provide certain insights.

An Assistant Commissioner of Education was being shown a CAI system by representatives of a large and well-known computer company.

One one side of the Commissioner stood a salesman, who wanted him to be impressed. On the other side stood one Dr. S., who knew how the system worked.

The terminal, demonstrating a history program that had hurriedly been put together, typed: WHO CAPTURED FORT TICONDEROGA?

"Can I type anything?" asked the Assistant Commissioner.

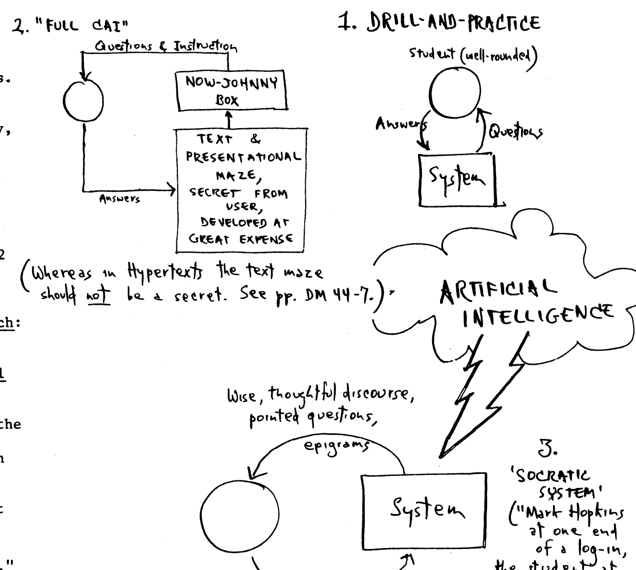
"Sure," said the salesman, ignoring the frantic head-shaking of Dr. S.

The Assistant Commissioner typed: Gypsy Rose Lee.

The machine replied:

NO, BUT YOU'RE CLOSE. HE CAPTURED QUEBEC A SHORT TIME LATER.

The Assistant Commissioner evidently enlivened many a luncheon with that one, and Computer-Assisted Instruction was effectively dead for the rest of the administration.



ANOTHER ANECDOTE

Some of us have been saying for a long time that learning from computers ought to be under control of the student.

One group (never mind who) has taken hold of this idea and gotten a lot of funding for it under the name of STUDENT CONTROL. This group talks as if it were some kind of scientific breakthrough.

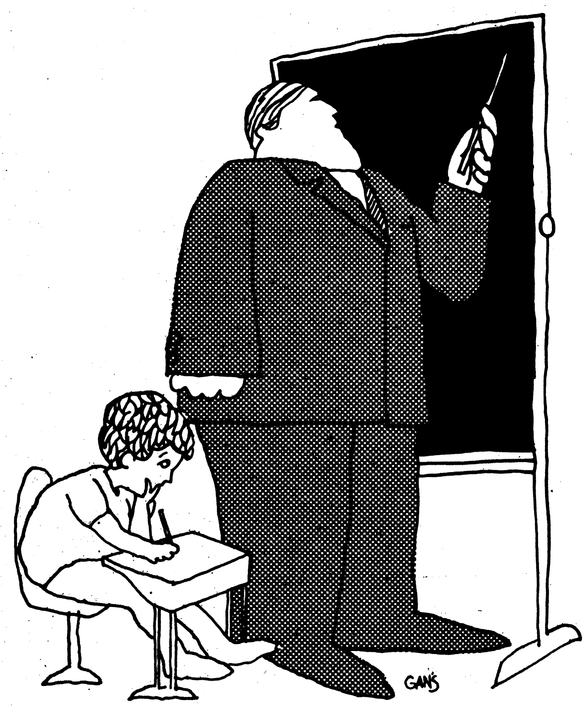
A friend of mine suggests, however, that this phrase may have brought the funding because administrators thought it meant control of the student.

BIBLIOGRAPHY

George B. Leonard, Education and Ecstasy. Dell, $2.25. Argues for making education an enthusiastic process.
Theodor H. Nelson, "No More Teachers' Dirty Looks." Follows.

DM 16

(The following article appeared in the September, 1970 issue of Computer Decisions, and got an extraordinary amount of attention. I have changed my views somewhat-- we all go through changes, after all-- but after consideration have decided to re-run it in the original form, without qualifications, mollifications or anything, for its unity. Thanks to Computer Decisions for use of the artwork by Gans and for the Superstudent picture on the cover, whose artist unfortunately insists on preserving his anonymity.



An interesting point, incidentally, is that people read this a lot of different ways. One Dean of Education hilariously misread it as an across-the-board plug for CAI. Others read in it various forms of menace or advocacy of generalized mechanization. One letter-writer said I was a menace but at least writing articles kept me off the streets. Here is my fundamental point: computer-assisted instruction, applied thoughtlessly and imitatively, threatens to extend the worst features of education as it is now.

© Walt Disney Productions.

Some premises relevant to teaching

1. The human mind is born free, yet everywhere it is in chains. The educational system serves mainly to destroy for most people, in varying degrees, intelligence, curiosity, enthusiasm, and intellectual initiative and self-confidence. We are born with these. They are gone or severely diminished when we leave school.
2. Everything is interesting, until ruined for us. Nothing in the universe is intrinsically uninteresting. Schooling systematically ruins things for us, wiping out these interests; the last thing to be ruined determines your profession.
3. There are no "subjects." The division of the universe into "subjects" for teaching is a matter of tradition and administrative convenience.
4. There is no natural or necessary order of learning. Teaching sequences are arbitrary, explanatory hierarchies philosophically spurious. "Prerequisites" are a fiction spawned by the division of the world into "subjects;" and maintained by not providing summaries, introductions or orientational materials except to those arriving through a certain door.
5. Anyone retaining his natural mental facilities can learn anything practically on his own, given encouragement and resources.
6. Most teachers mean well, but they are so concerned with promoting their images, attitudes and style of order that very little else can be communicated in the time remaining, and almost none of it attractively.

books.[1] And this all ignores a simple fact: all are arbitrary. Instructional sequences aren't needed at all if the people are motivated and the materials are clear and available.

Testing as we know it (integrated with walled curricula and instructional sequences) is a destructive activity, particularly for the orientation which it creates. The concerns of testing are extraneous: learning to figure out low-level twists in questions that lead nowhere, under pressure.

The system of tensions and defenses it creates in the student's personality are unrelated to the subject or the way people might relate to the subject. An exploitive attitude is fostered. Not becoming involved with the subject, the student grabs for rote payoff rather than insight.

All in a condescending circumstance. Condescension is built into the system at all levels, so pervasive it is scarcely noticed. Students are subjected to a grim variety of put-downs and denigrations. While many people evidently believe this to be right, its productivity in building confident and self-respecting minds may be doubted.

The problems of the school are not particularly the teacher's fault. The practice of teaching is principally involved with managing the class, keeping up face, and projecting the image of the subject that conforms to the teacher's own predilections. The educational system is thereby committed to the fussy and prissy, to the enforcement of peculiar standards of righteousness and the elevation of teachers—a huge irrelevant shell around the small kernel of knowledge transmitted.

The usual attacks on computer teaching tend to be sentimental and emotional pleas for the alleged humanism of the existing system. Those who are opposed to the use of computers to teach generally believe the computer to be "cold" and "inhuman." The teacher is considered "warm" and "human." This view is questionable on both sides.

The computer is as inhuman as we make it. The computer is no more "cold" and "inhuman" than a toaster, bathtub or automobile (all associated with warm human activities). Living teachers can be as inhuman as members of any people-prodding profession, sometimes more so. Computerists speak of "freeing teachers for the creative part of their work;" in many cases it is not clear what creative tasks they could be freed for.

At the last, it is to *rescue* the student from the inhuman teacher, and allow him to relate directly and personally to the intrinsically interesting subject matter, that we need to use computers in education.

Many successful systems of teacherless learning exist in our society: professional and industrial magazines; conventions and their display booths and brochures; technical sales pitches (most remarkably, those of medical "detail men"); hobbyist circles, which combine personal acquaintance with a round of magazines and gatherings; think-tanks and research institutes, where specialists trade fields; and the respectful briefing.

None of these is like the conventional classroom with its haughty resource-chairman; they are not run on condescension; and they get a lot across. We tend to think they are not "education" and that the methods cannot be transferred or extended to the regions now ruled by conventional teaching. But why not?

If everything we ate were kibbled into uniform dog-food, and the amount consumed at each feeding time tediously watched and tested, we would have little fondness for eating. But this is what the schools do to our food for thought, and this is what happens to people's minds in primary school, secondary school and most colleges.

This is the way to produce a nation of sheep or clerks. If we are serious about wanting people to have creative and energetic minds, it is not what we ought to do. Energy and enthusiasm are natural to the human spirit; why drown them?

Education ought to be clear, inviting and enjoyable, without booby-traps, humiliations, condescension or boredom. It ought to teach and reward initiative, curiosity, the habit of self-motivation, intellectual involvement. Students should develop, through practice, abilities to think, argue and disagree intelligently.

Educators and computer enthusiasts tend to agree on these goals. But what happens? Many of the inhumanities of the existing system, no less wrong for being unintentional, are being continued into computer-assisted teaching.

Although the promoters of computer-assisted instruction, affectionately called "CAI," seem to think of themselves as being at the vanguard of progress in all directions, the field already seems to operate according to a stereotype. We may call this "classic" or "conventional" CAI, a way of thinking depressingly summarized in *"The Use of Computers in Education"* by Patrick Suppes, Scientific American, September, 1966, 206-220, an article of semi-classic stature.

It is an unexamined premise of this article that the computer system will always decide what the student is to study and control his movements through it. The student is to be led by the nose through every subject, and the author expresses perplexity over the question of *how* the system can decide, at all times, *where* to lead the student by the nose (top of col. 3, p. 219). But let us not anticipate alternatives.

It is often asserted (as by Alpert and Bitzer in *"Advances in Computer-Based Education,"* Science, March 20, 1970) that this is not the only approach current. The trouble is that it *seems* to be the only approach current, and in the expanding computer universe everyone seems to know what CAI "is." And this is it.

Computer-assisted instruction, in this classical sense, is the presentation by computer of bite-sized segments of instructional material, branching among them according to involuntary choices by the student ("answers") and embedding material presented the student in some sort of pseudo-conversation ("Very good. Now, Johnny, point at the . . .")

CAI: Based on unnecessary premises

At whichever level of complexity, all these conventional CAI systems are based on three premises: that all presentations consists of *items,* short chunks and questions; that the items are arranged into *sequences,* though these sequences may branch and vary under control of the computer; and finally, that these sequences are to be embedded in a framework of *dialogue;* with the computer composing sentences and questions appropriately based on the student's input and the branching structure of the materials. Let us call such systems SIC (Sequenced-Item Conversational) systems.

These three premises are united. For there to be dialogue means there must be an underlying graph structure of potential sequences around which dialogue may be generated; for there to be potential sequences means breakpoints, and hence items.

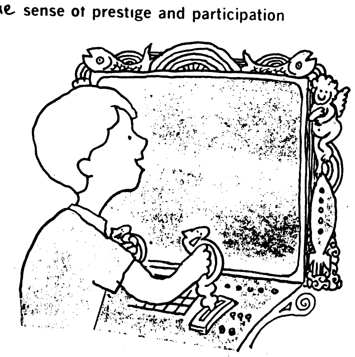
Let us question each of the premises in turn.

1. **Is dialogue pleasant or desirable?** Compulsory interaction, whether with a talking machine or a stereotyped human, is itself a put-down or condescension. (Note that on superhighways there is often a line of cars behind the automatic toll booths, even when the manned ones are open.) Moreover, faked interaction can be an annoyance. (Consider the green light at the automatic toll booth that lights up with a "thank you.") Moreover, dialogue by simple systems tends to have a fake quality. It is by no means obvious that phony dialogue with a machine will please the student.

2. **Is the item approach necessary?** If the student were in control, he could move around in areas of material, leaving each scene when he got what he wanted, or found it unhelpful.

3. **Are sequences necessary?** Prearranged sequences become unnecessary if the student can see what he has yet to learn, then pursue it.

The sense of prestige and participation



CAI: unnecessary complication

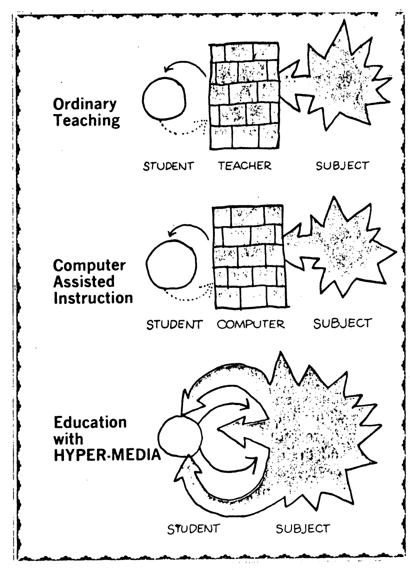
The general belief among practitioners is that materials for computer-based teaching are extremely difficult to create, or "program." Because of possible item weakness and the great variety of possible sequences within the web, extensive experimentation and debugging are required. Each item must be carefully proven; and the different sequences open to a student must all be tested for their effectiveness. All possible misunderstandings by a student need to be anticipated and prevented in this web of sequences, which must be designed for its coverage, correct order, and general effectiveness.

CAI: general wrongfulness

Computers offer us the first real chance to let the human mind grow to its full potential, as it cannot within the stifling and insulting setting of existing school systems. Yet most of the systems for computer-assisted instruction seem to me to be perpetuating and endorsing much that is wrong, even evil, in our present educational system. CAI in its conventional form enlarges and extends the faults of the American educational system itself. They are:

- Conduciveness to boredom;
- The removal of opportunities for initiative;
- Gratuitous concerns, both social and administrative ("subject," "progress" in subject);
- Grades, which really reflect commitment level, anxiety, and willingness to focus on core emphasis;
- Stereotyped and condescending treatment of the student (the "Now-Johnny" box in the computer replacing the one that sits before the class);
- The narrowing of curricula and available materials for "results" at the expense of motivation and generalized orientation;
- Destructive testing of a kind we would not permit on delicate machinery; and,
- An overt or hidden emphasis on invidious ratings. (Ungraded schools are nice—but how many units did *you* complete today?).

There are of course improvements, for instance in the effects of testing. In the tell-test, tell-test nattering of CAI, the testing becomes merely an irritant, but one certainly not likely to foster enthusiasm.



But isn't CAI 'scientific?'

Part of CAI's mystique is based upon the idea that teaching can become "scientific" in the light of modern research, especially learning theory. It is understandable that researchers should promote this view and that others should fall for it.

Laymen do not understand, nor are they told, that "learning theory" is an extremely technical, mathematically oriented, description of the behavior of abstract and idealized organisms learning non-unified things under specific conditions of motivation and non-distraction.

Let us assume, politely, that learning theory is a full and consistent body of knowledge. Because of its name, learning theory has at least what we may call nominal relevance to teaching; but real relevance is another matter. It may be relevant as Newtonian equations are to shooting a good game of pool: implicit but without practical bearing.

Because of the actual character of learning theory, and its general remoteness from non-sterile conditions, actual relevance to any particular type of application must still be demonstrated. To postulate that the theory still applies in diluted or shifted circumstances is a leap of faith. Human beings are not, taken all together, very like the idealized pigeons or rats of learning theory, and their motivations and other circumstances are not easily controlled. Studies concerned with rate of repetition and reinforcement are scarcely relevant if the student hates or does not understand what he is doing.

I do not mean to attack all CAI, or any teaching system which is effective and gratifying. What I doubt is that SIC systems for CAI will become more and more wonderful as effort progresses, or that the goal of talking tutorial systems is reachable and appropriate. And what I further suspect is that we are building boredom systems that not only make life duller but sap intellectual interest in the same old way.

Should systems 'instruct?'

Drill-and-practice systems are definitely a good thing for the acquisition of skills and response sets, an improvement over workbooks and the like, furnishing both corrections and adjustment. They are boring, but probably less so than the usual materials. But the CAI enthusiasts seem to believe the same conversationalized chunk techniques can be extended to the realm of ideas, to systems that will tutor and chide, and that this will provide the same sort of natural interest provided by a live tutor's instruction.

The conventional point of view in CAI claims that because validation is so important, it is necessary to have a standardized format of item, sequence and dialogue. This justifies turning the endeavor into pickywork within items and sequence complexes, with attendant curricular freeze, and student inanition and boredom. This is entirely premature. The variety of alternative systems for computer teaching have not even begun to be explored. Should systems "instruct" at all?

'Responding Resources' and 'Hyper-Media'

At no previous time has it been possible to create learning resources so responsive and interesting, or to give such free play to the student's initiative as we may now. We can now build computer-based presentational wonderlands, where a student (or other user) may browse and ramble through a vast variety of writings, pictures and apparitions in magical space, as well as rich data structures and facilities for twiddling them. These we may call, collectively, "responding resources." Responding resources are of two types: facilities and hyper-media.

A *facility* is something the user may call up to perform routinely a computation or other act, behaving in desired ways on demand. Thus JOSS (a clever desk calculator available at a terminal) and the Culler-Freed graph-plotting system (which graphs arbitrary functions the user types in) are facilities.

Hyper-media are branching or performing presentations which respond to user actions, systems of prearranged words and pictures (for example) which may be explored freely or queried in stylized ways. They will not be "programmed," but rather *designed, written, drawn* and *edited,* by authors, artists, designers and editors. (To call them "programmed" would suggest spurious technicality. Computer systems to present them will be "programmed.") Like ordinary prose and pictures, they will be *media;* and because they are in some sense "multi-dimensional," we may call them *hyper-media,* following the mathematical use of the term "hyper-".

A modest proposal

The alternative is straightforward. Instead of devising elaborate systems permitting the computer or its instructional contents to control the situation, why not permit the student to control the system, show him how to do so intelligently, and make it easy for him to find his own way? Discard the sequences, items and conversation, and allow the student to move freely through materials which he may control. Never mind optimizing reinforcement or validating teaching sequences. Motivate the user and let him loose in a wonderful place.

Let the student control the sequence, put him in control of interesting and clear material, and make him feel good—comfortable, interested, and autonomous. Teach him to orient himself: not having the system answer questions, all typed in, but allowing the student to get answers by looking in a fairly obvious place. (Dialogue is unnecessary even when it does not intrude.) Such ultra-rich environments allow the student to choose what he will study, when he will study it and how he will study it, and to what criteria of accomplishment he will aim. Let the student pick what he wishes to study next, decide when he wishes to be tested, and give him a variety of interesting materials, events and opportunities. Let the student ask to be tested on what he thinks he knows, when he is ready, selecting the most appropriate form of testing available.

This approach has several advantages. First, it circumvents the incredible obstacles created by the dialogue-item-sequence philosophy. It ends the danger to students of bugs in the material. And last, it does what education is supposed to do—foster student enthusiasm, involvement, and self-reliance.

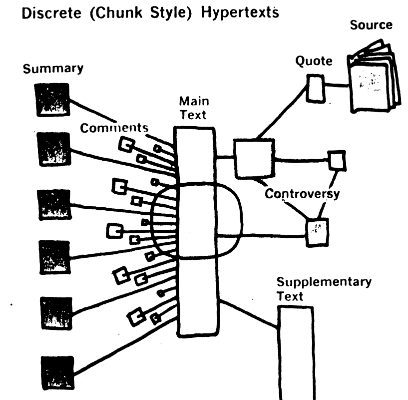
Under such circumstances students will actually be interested, motivated to achieve far more than they have ever achieved within the normal instructional framework; and any lopsidedness which may result will be far offset by the degree of accomplishment which will occur—it being much better to create lopsided but enthusiastic genius specialists than listless, apathetic, or cruelly rebellious mediocrities. If they start soon enough they may even reach adulthood with natural minds: driven by enthusiasm and interest, crippled in no areas, eager to learn more, and far smarter than people ordinarily end up being.

Enthusiasm and involvement are what really count. This is why the right to explore far outweighs any administrative advantages of creating and enforcing "subjects" and curriculum sequences. The enhancement of motivation that will follow from letting kids learn anything they want to learn will far outweigh any specialization that may result. By the elimination or benign replacement of both curriculum and tests in an ultra-rich environment, we will prevent the attrition of the natural motivation of children from its initially enormous levels, and mental development will be the natural straight diagonal rather than the customary parabola.

Is it so hard? some ideas

CAI is said to be terribly hard. It would seem all the harder, then, to give students the richer and more stimulating environments advocated here. This is because of the cramped horizons of computer teaching today. Modest goals have given us modest visions, far below what is now possible and will soon be cheap.

Discrete (Chunk Style) Hypertexts



The static computer displays now associated with CAI will give way to dynamic displays driven from minicomputers, such as the IDIOM, IBM 2250/4 or Imlac PDS-1. (The last of these costs only $10,000 *now* by 1975 such a unit will probably cost $1,000 or less.) Not only will computers be much cheaper, but their usability will improve: a small computer with a fair amount of memory will be able to do much more than it can now, including operate a complex display from its own complex data base.

It is generally supposed that systems like these need big computers and immense memories. This is not true if we use the equipment well, organize storage cleverly, and integrate data and display functions under a compact monitor. This is the goal of The Nelson Organization's Project Xanadu, a system intended to handle all the functions described here on a minicomputer with disk and tape.

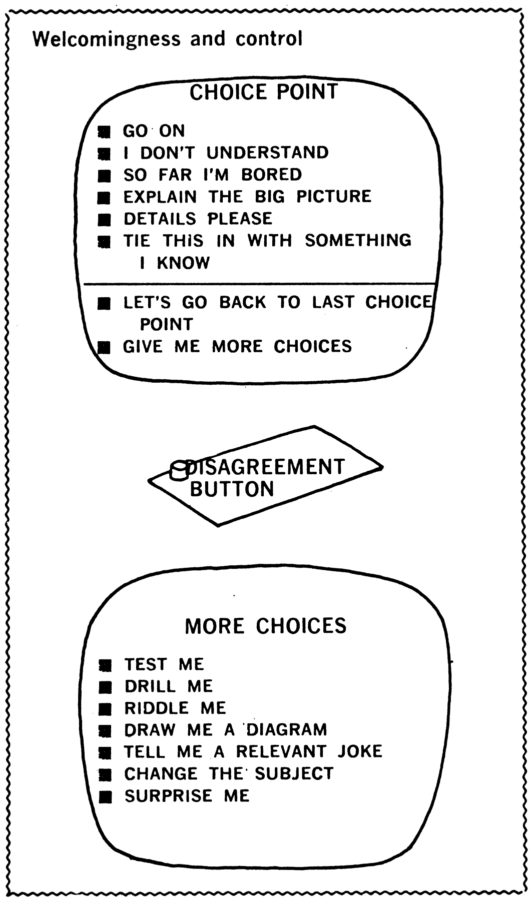
Discrete hypertexts

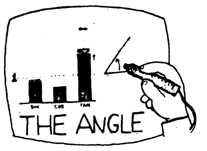
"Hypertext" means forms of writing which branch or perform on request; they are best presented on computer display screens.

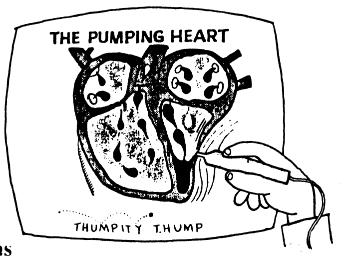
In ordinary writing the author may break sequence for footnotes or insets, but the use of print on paper makes some basic sequence essential. The computer display screen, however, permits footnotes on footnotes on footnotes, and pathways of any structure the author wants to create.

Discrete, or chunk style, hypertexts consist of separate pieces of text connected by links.

Ordinary prose appears on the screen and may be moved forward and back by throttle. An asterisk or other key in the text means, not an ordinary footnote, but a *jump*—to an entirely new presentation on the screen. Such jumpable interconnections become part of the writing, entering into the prose medium itself as a new way to provide explanations and details to the seeker. These links may be artfully arranged according to meanings or relations in the subject, and possible tangents in the reader's mind.







Performing hypergrams

A hypergram is a performing or branching picture: for instance, this angle, with the bar-graph of its related trigonometric functions. The student may turn the angle upon the screen, seizing it with the light-pen, and watch the related trigonometric functions, displayed as bar charts, change correspondingly.

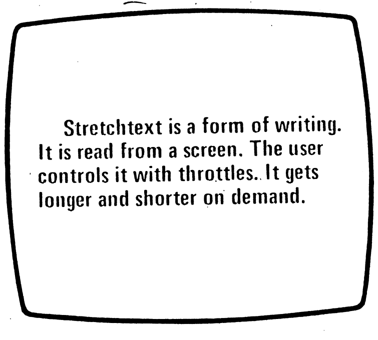
Hypergrams may also be programmed to show the consequenees of a user's prod—what follows or accompanies some motion of the picture that he makes with a pointing tool, like the heartbeat sequence.

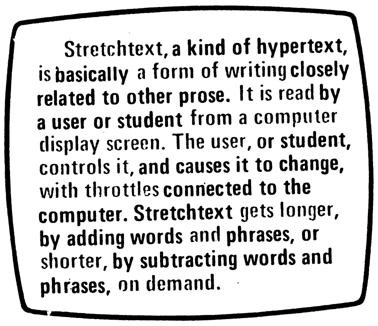
Stretchtext fills in the details

This form of hypertext is easy to use without getting lost. As a form of writing, it has special advantages for discursive and loosely structured materials—for instance historical narratives.

There are a screen and two throttles. The first throttle moves the text forward and backward, up and down on the screen. The second throttle causes changes in the writing itself: throttling toward you causes the text to become *longer* by minute degrees. Gaps appear between phrases; new words and phrases pop into the gaps, an item at a time. Push back on the throttle and the writing becomes shorter and less detailed.

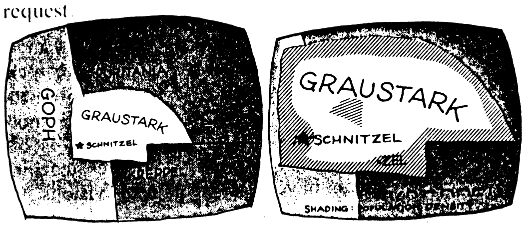
The stretchtext is stored as a text stream with extras, coded to pop in and pop out at the desired altitudes:

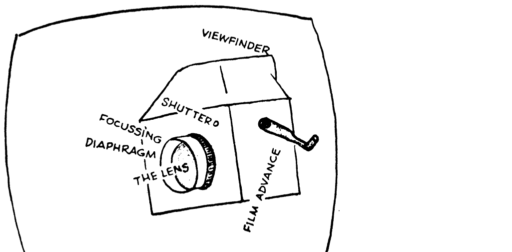


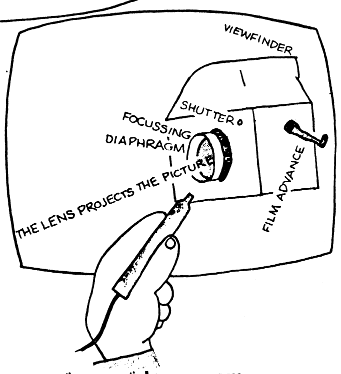


Hypermap zips up or down

The screen is a map. A steering device permits the user to move the map around the world's surface; a throttle zooms it in. Not by discrete jumps, but animated in small changes, the map grows and grows in scale. More details appear as the magnification increases. The user may request additional display modes or "overlays," such as population, climate, and industry. Such additional features may pop into view on request.



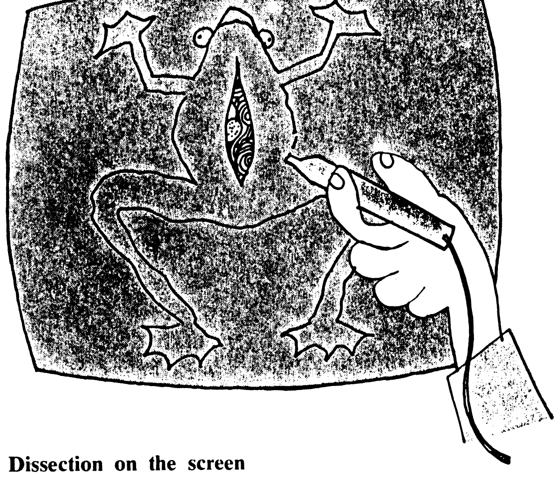




Queriable illustrations: a form of hypergram

A "hypergram" is a picture that can branch or perform on request. In this particular example, we see on the screen a line-drawing with protruding labels. When the student points at a label, it becomes a sliding descriptive ribbon, explaining the thing labelled.

Or asterisks in an illustration may signal jumps to detailed diagrams and explanations, as in discrete hypertexts.



Dissection on the screen

The student of anatomy may use his light-pen as a scalpel for a deceased creature on the screen. As he cuts, the tissue parts. He could also turn the light-pen into hemostat or forceps, and fully dissect the creature —or put it back together again. (This need not be a complex simulation. Many key relationships can be shown by means of fairly simple schematic pictures, needing a data structure not prohibitively complicated.)

Hyper-comics are fun

Hyper-comics are perhaps the simplest and most straightforward hyper-medium. The screen holds a comic strip, but one which branches on the student's request. For instance, different characters could be used to explain things in different ways, with the student able to choose which type of explanation he wanted at a specific time.

'Technicality' is not necessary

Proponents of CAI want us to believe that scientific teaching requires a certain setup and format, incomprehensible to the layman and to be left to experts. This is simply not true. "Technicality" is a myth. The problem is not one of technical rightness, but what *should* be.

The suggestions that have been given are things that should be; they will be brought about. □

★ THE MIND'S EYE ★

It was explained on the other side that computers have no fixed purpose or style of operation, but can be set in motion on detailed and repetitive tasks in any realm of human interest-- as long as those tasks are exactly specifiable in certain humdrum ways.

Now, if you had a machine like that burning a hole in the corner of your office, what would you really want to do with it?

You can't drive it on the road.

You can't make love to it. (But see p.$^{DM}_{9}$.)

You can't cook in it, or get the news on it.

To get it to control elaborate events in the real world requires a lot of expensive equipment and interfaces, so cross that out.

Yet suppose you have an inquiring imagination-- which is not unlikely, considering that you are reading this sentence.

And we are also supposing (from an earlier paragraph) that you have a computer.

What sorts of thing would you do with it?

Things that are imaginative and don't require too much else.

I am hinting at something.

YOU COULD HAVE IT MAKE PICTURES
and show you stuff
and change what it shows
depending on what
you do;

and if this idea doesn't turn you on, the rest of this book is probably not for you.

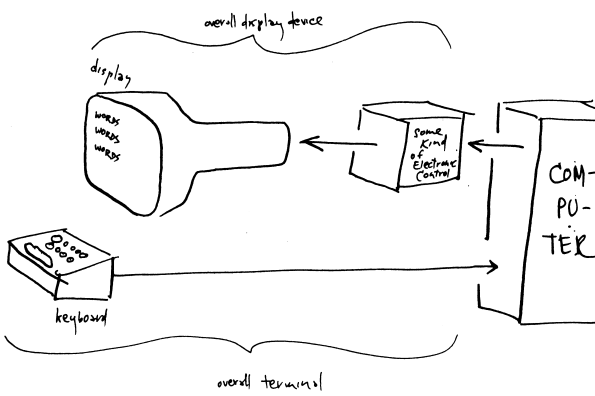
The techniques of making pictures by computer are called computer graphics.

But that includes the dull kinds of making pictures by computer, the ones that do it with pens and printing machines.

The techniques of making computers present things interactively on screens is called computer display. (Some say "interactive computer graphics;" this is not just too long, but too restrictive as well: interactive text systems are not "graphic" or pictorial, but they are going to be a profoundly important area of computer display.)

(Incidentally, the silly word "interaction" was coined because the previous word "intercourse," which meant exactly the same thing, had racy connotations for some people. Cf. "donkey" and "rooster," also relatively recent.)

You will note that computer display is what makes possible the computer terminals with screens that we saw on the other side. All that a screen-terminal is is some sort of computer display, to which a keyboard has been added.



You will therefore see that to understand all the different computer display terminals, you would have to understand all the different computer display techniques; unfortunately we can only cover a few here, and those but sparely.

Some of the types of computer display to be covered hereabouts include:

CRT, or cathode-ray tube, displays; these are my favorite because the stuff on their screens may be animated by the computer.

video displays, which use television techniques. These have troubles deriving from the way a TV picture is timed.

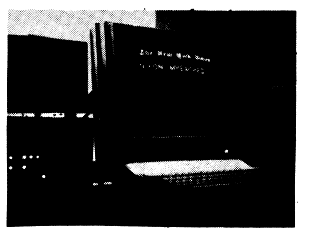
panel displays, i.e., those which appear on a flat panel. These are going to be cropping up all over. (The pictures can't move much, but the devices are going to be cheap. Flat, too. Some people think that's very important.)

3-D displays, especially of the CRT type. NOTE: this term refers ambiguously to two different things: setups which present flat views of three-dimensional scenes, and those which present stereoscopic views of 3-D scenes; these are much rarer.

image synthesis or halftone techniques and systems. These are computer programs and special devices which make shaded or photograph-like pictures. (This happens to be a favorite topic of mine, and so there's quite a bit on it here, a lot of which is not widely known in the field.)

BIBLIOGRAPHY ★ THE MIND'S EYE ★ continues p. 22.

Newman and Sproull, Interactive Computer Graphics, McGraw, $15. Your basic text on all forms of computer graphics (and thus animation).

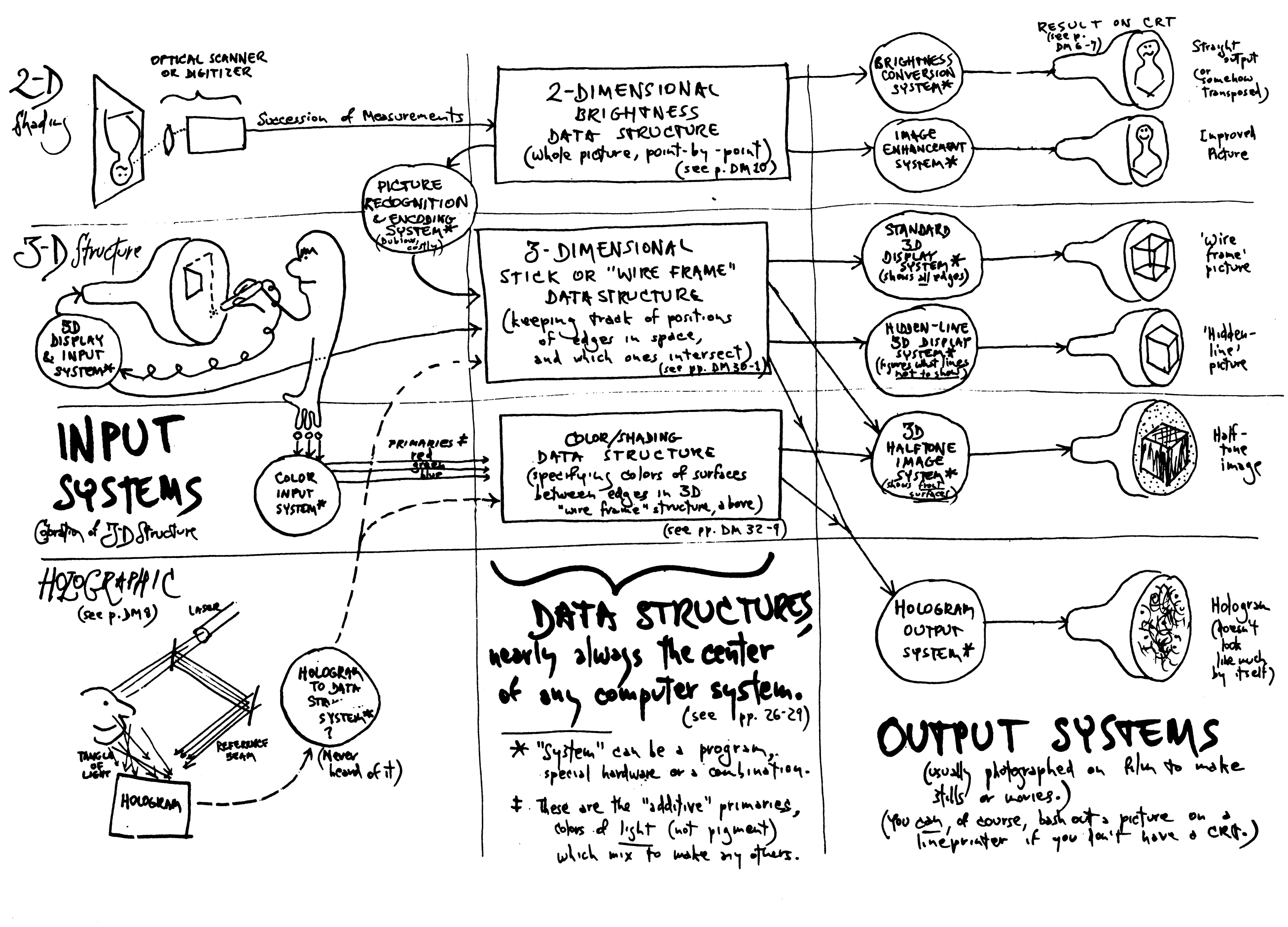


Responding computer displays come in all sizes and prices. This little setup (in the under-$10,000 class) is a PDP-8 minicomputer with home-built display circuitry. Gothic lettering data structure available from somebody in the military; message courtesy of R.E.S.I.S.T.O.R.S. The big display is an IBM 2250 (over $100,000, including minicomputer).

GET THE PIC-TURE?

→ Today's zippy picture systems come in many inter-related varieties.

this book presents information on various types; perhaps this diagram will help sort them out.



DISPLAY TERMINALS

Some computer displays have to be deeply attached to a computer and some don't. These latter we call display terminals.

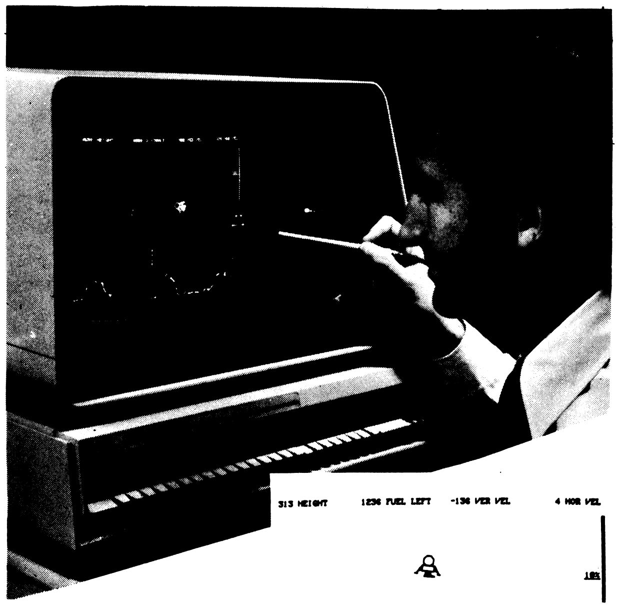
A display terminal is like an ordinary computer terminal (see p 14): that is, fundamentally a device by which a computer and a person can type at each other. However, display terminals have screens.

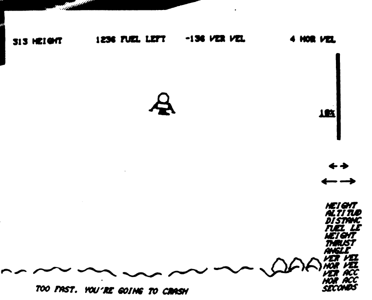
Now, some display terminals only show text, just like ordinary printing terminals (described on the other side). But manufacturers are free to add any other features, and so different manufacturers make it possible to do various kinds of picture-making with their particular display terminals, if appropriate programs are running in the computer that controls them.

Some devices are sold as display terminals but actually, to further confuse the issue, contain complete minicomputers. (The fact that the manufacturer may not stress this is simply a marketing angle he has chosen.) Similarly, certain terminals contain microprocessors (see p. 44), which means they can be programmed to behave like various other terminals, but ordinarily they cannot be programmed to do much else by themselves.

Without getting into it deeply, there are two main types of display terminal: those that are refreshed and those that are not. A refreshed display is one whose viewing surface fades and must be continually re-filled; a non-refreshed display somehow stores the presentation in the viewing surface itself.

Non-refreshed displays simply take the symbols from the computer, blam them onto the screen, and that's it until the screen is erased (by either the computer or the user).





This honey is the GT-40 from DEC ($12,000, including computer-- the thing with teeth, below). It's a subroutining display (see p. DM 23).

Man is playing Moonlander game: controlling screen action with lightpen. Computer simulates real moon lander. Reversed white-to-black for readability here.

THE WONDER OF INTERACTIVE DISPLAY SYSTEMS

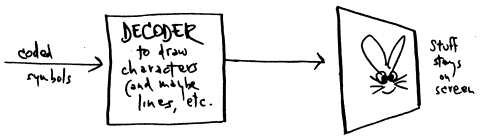
If you have not seen interactive computer display, you have not lived.

Except for a few people who can imagine it-- and I'm trying to help you with that as hard as I can-- most people just don't get it till they see it. They can't imagine what it's like to manipulate a picture. To have a diagram respond to you. To change one part of a picture, and watch the rest adapt. These are some of the things that can happen in interactive computer display-- all depending, of course, on the program.

For some reason there are a lot of people who pooh-pooh computer display: they say it's "not necessary," or "not worth it," or "you can get just as good results other ways."

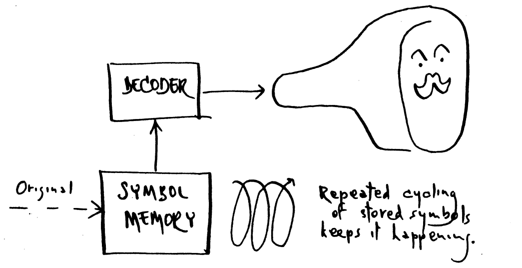
Personally, I wouldn't thing of trying to justify computer display on "practical" grounds. So what if it offers you faster access to information and pictures and maps and diagrams, the ability to simulate extremely complex things by modifying pictures, the ability to go through complex transactions with the system in very little time, the ability to create things in the world almost instantaneously (say, by creating fabric patterns which are then automatically woven, or design 3D objects which are then automatically milled by machines), and never mind that it enables the user, say, to control entire oil refineries by the flick of a lightpen.

As far as I'm concerned, these matters aren't very important compared to changing the world: making education an excitement, rather than a prison; giving scholars total access to writings and notes, in new complex form; allowing people to play imaginatively, and raising human minds to the potentials they should have reached long ago; and helping people think at the deepest level about very heavy and complex alternatives-- which confront us more ominously today than ever.



Two major types are the storage tube and the panel. These in turn have separate subtypes, etc.

Refreshed displays have to have some other kind of symbolic (digital) memory, whose contents repeatedly go to the screen:



Most refreshed displays use an actual television screen-- that is, a CRT (see p.DM6-7) whose entire area is repeatedly re-painted by the elctron beam.

Since computers send text out to terminals as individual alphabetic and punctuation codes, each terminal must contain circuitry to change the character code to a visible alphabetical character on the screen. Such a piece of circuitry is called a character generator. There are various kinds, they go at various speeds, some offer more different characters than others.

Display terminals generally have a little marker, or cursor, that the user or the computer can move around the screen. The computer can sense what the user is pointing at by the motion codes it gets, telling where the user has moved the cursor.

I had intended here to print a little directory of display terminal manufacturers, but there simply is not time. See section on terminals, other side.

Note that the term video terminal is often used, incorrectly, for any display terminal. The term "video" should only be used when the screen is refreshed by an actual video raster. (See "Lightning in a Bottle," p.DM6-7.)

Text terminals (also called alphabetic terminals, character terminals or keyscopes) simply show written text, put in either by the computer or the user. (Some terminals, called transaction terminals, can be divided up into specific areas that the user may and may not type into-- for banking and stuff. However, whether that form of terminal is necessary may also be a matter of taste in the program design.)

Text terminals range in price from, say, $1500 on up to $6500. (This last is the price of a remarkable color text terminal demonstrated by Tec, Inc., at the 1974 National Computer Conference. Each alphabetic position could contain a letter and/or a bright color; altogether the screen could hold big colorful pictures made up of these bright spaces. Ostensibly just a text terminal, actually the device could be regarded as an Instant Movie Generator for television animation. But it may take Tec, Inc. awhile to realize what they have created.)

Graphic terminals offer some kind of pictures on their screens. These come in a great variety: line-drawing, some without, some with levels of grey. Of interest to the beginner are:

"The Tektronix." (Also called "the greenie," or "the green screen.") Tektronix, Inc., makes a display based on a pale green storage tube they make. (So does Computer Displays, Inc.) Such displays allow you to put more and more text and pictures on a screen, crowding it all up-- but you can't take the lines or words off individually.

"The PEP." Excellent (but very expensive) display that comes out to a video screen from a high-resolution storage tube. Permits grey scales and selective erase. Princeton Electronic Products.

The IDIgraf (Information Displays, Inc., Mount Kisco, NY). Allows line pictures with animation; interesting unit; somewhat less than $10,000.

A PLATO-like terminal (see PLATO terminal, nearby, and pp. 26-27) is now available for use with STANDARD computer interfaces and software. "Less than $5000" from Applications Group, Inc., P.O. Box 444A, Maumee, Ohio 43537.

REFRESHED HIGH-RESOLUTION COLOR SYSTEMS. A number of companies manufacture computer displays allowing complex grey-scale pictures, including color. They are expensive but very very nice. Indeed, if you buy them in clusters, these fancy-picture scopes can cost as little as text terminals. Some manufacturers are:

- Data Disk. (Disk refresh.) Note: I once recommended them to a consulting client of mine, who later expressed complete satisfaction with their equipment.
- Ramtek. (Semiconductor storage.)
- Adage, Inc. Their model 200 is a video system refreshed from semiconductor storage.
- Comtal. (Disk.)
- Spatial Data Systems. (Disk.)
- Dicomed. (Disk.) Extremely high resolution.

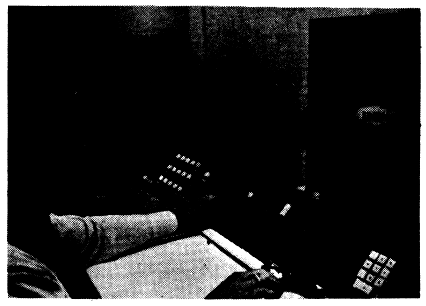
Student programmer Alan McNeil, an art major, ponders something or other. It may be the program for the Nova space-game he and Pete Rowell are building. Alan also made a film showing what may have been the motions of the continents, shooting straight off the PLATO screen. Some PLATO purists point out that this is not exactly what PLATO was originally intended for. So?

PLATO panel display (see DM 26-7).

MEN AT WORK

The computer display screen is the new frontier of our lives.

That such systems should (and will) be fun goes without saying. That they will also be a place to work may be less obvious from the tone of this publication, so I want to stress it here.



Making pictures with the GE halftone system (see pp. DM 32-9).

The thing about display screens-- especially the high-performance, subroutining kind-- is that the screen can become a place from which to control events in the outside world.

Example: I believe a town in N.Y. State has its electrical system hooked up to an IDIIOM subroutining display (made by Information Displays, Inc., and coupled to a Varian 620 minicomputer). Instead of having a wall with a big painted map having switches set into it, like many such control centers, the switches are linked directly to the minicomputer, and a program in the minicomputer connects these circuits to the pictures on the screen. Thus to throw a switch in the real world, the operator points with his lightpen at the picture of the switch, and the minicomputer throws the switch.

There are oil refineries that work the same way. The operator can control flows among pipes and tanks by pointing at their pictures, or at symbols connected with them, and bingo, it happens Out There.

In another case, a person designing something at a screen can look across the room and see a machine producing what he just finished designing a few minutes ago. I wish I could say more about that particular setup.

The true problem that I think is emerging, though, is the problem of system response and style. Okay, so you're controlling widget assembly, or traffic light grids, at the CRT screen. The real question is, how does the screen behave and respond? This is not, darn it, a technical issue. It's psychological and then some. The design of screen activities which will enjoyably focus the user's mind on his proper concerns-- no matter how personal these may be-- is the new frontier of design, of art, and of architecture. But more of that later.

Now, the Xerox Corporation has said that they intend to replace paper (or, the way I heard it, "Somebody is going to replace paper with screens, and it will be either IBM or us, so let's have it be us.")

Well and good. Save the trees and stem the grey menace. But the question is: what will the systems be like? How should they perform? What forms will information take? What conventions, structures, diagrams, animations, ways to sign things, ways to view things ... HOW SHALL IT BE?

I am afraid that as long as people are befuddled with technicalities, or confused by those who profess that these considerations are their specialty by right, we will never get straight. Lacking time for the full discussion, I give you a motto:

IF THE BUTTON IS NOT SHAPED LIKE THE THOUGHT, THE THOUGHT WILL END UP SHAPED LIKE THE BUTTON.

SAVING ENERGY WITH COMPUTER DISPLAY

A timely criticism of computer display is that it needs electricity. But (as mentioned elsewhere) it saves paper, and, importantly, it bodes to save energy as well.

IF WE SWITCH TO COMPUTER SCREENS FROM PAPER, PEOPLE WON'T HAVE TO TRAVEL AS MUCH. Instead of commuting to offices in the center of town, people can set up their offices in the suburbs, and share the documentary structure of the work situation through the screens.

This view has been propounded, indeed, by Peter Goldmark, former director of research for CBS Labs, the man who brought you the LP record.

IF COMPUTERS ARE THE WAVE OF THE FUTURE, DISPLAYS ARE THE SURFBOARDS.

★ THE MIND'S EYE ★, Continued.

YOUR BASIC TYPES OF COMPUTER DISPLAY

(Note: the term "display" is also used in this field to refer to numbers and letters that can be made to light up in fixed positions, like on your pocket calculators. Those will not be discussed here. If you're interested see an article on the subject by Alan Sobel, Scientific American, early 1973 sometime.)

THE FORKED LIGHTNING

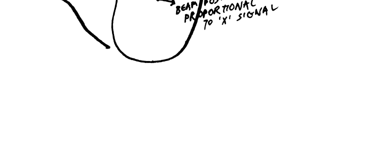
" Because their words have
forked no lightning they
Do not go gentle into that good night."
-- Dylan Thomas

The most basic, and yet eventually the most versatile, computer display is that of the CRT, or bottled lightning (as I like to call it). It is, you know: a beam of electrons, just like lightning in a storm, but from the neck of a very empty bottle to its flat bottom, whose chemically coated surface we watch. As manipulated by the computer, the CRT stabs its beam to all corners of the faceplate: forked lightning.

Computer display began in the late forties. Computers themselves were completely new, and so was Mr. Dumont's magical Cathode Ray Tube or CRT (see p.DM6), developed on a crash basis during the war so we could have radar, and as long as it was around after the war, we got television.

But the lightning bottle, or CRT, can be used in a variety of ways. Its control plates, which move the ray of electrons around on the screen, can be given various different electronic signals, causing the beam to move around in different patterns. In normal video, the signals move the beam in a zigzag pattern, where the zigs are very close together and the zags are invisible; the carpet of zigs covers the screen over and over in a repetitive pattern, and the beam's changing intensity paints the picture.

But we can drive the CRT differently. by using different control signals. For instance: we can apply a measured voltage to the height or "Y" plates of the CRT, moving the beam to a given vertical position, and another measured voltage to the sideways or "X" plates, controlling its horizontal position.



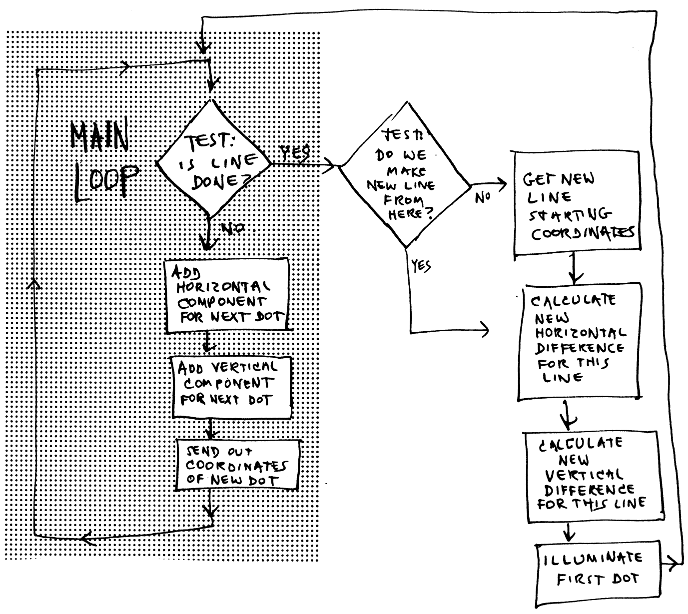
On our glorious Dig-It-All Screens we mingle the magics of air and of fire.

1. EARLIEST SYSTEM: A LITTLE PROGRAM TO MAKE DOTS

The earliest setup connected a CRT to a computer by the simplest possible means, and made its pictures with dots on the screen-- a sort of tattooing process.

It was simple because all the computer did was furnish to the connecting circuitry (or interface) symbols specifying how far up, and how far across the screen, the next dot should be. These symbols were actually coded numbers, and the interface turned them into voltages which then moved the beam correspondingly. (This process of making a measured voltage out of a coded numerical symbol is called digital-to-analog conversion, since (as explained on the other side) the main meaning of "analog" these days is "in a measured voltage.")

Now, this has several drawbacks. One is that the lines are dotty; nobody likes that. A more important annoyance, though, is that the computer scarcely has time for anything else. Here is a flowchart of what the computer has to do in its program. (Even if you didn't look at the other side of the book, flowcharts are nothing scary. They're just maps of what happens.)



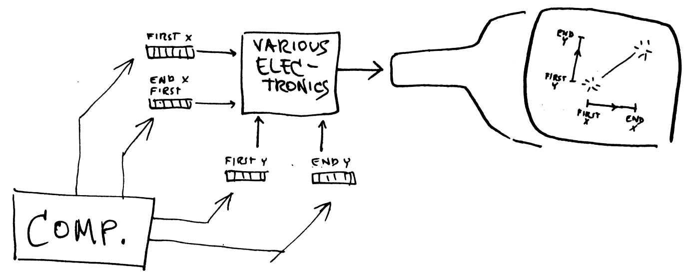
Furthermore, and here was the indignity of it, this system took far too long. To draw a line with thirty dots in it took thirty times around the main loop in the flowchart, and since each box in the flowchart takes at least one of the machine's rock-bottom instructions-- usually more-- then the main loop of this display routine takes four separate operations per dot, or 120 operations for a stupid 30-dot line. Plainly there has to be a better way to use an expensive computer.

Actually it wasn't just the ignominy of it, but the fact that it took so long, that made this a poor method. The amount of stuff the computer could draw in 1/40th of a second-- and this turns out to be how fast the whole picture has to be made-- was too little. After 1/40th of a second the human eye can see the lines on the CRT start to fade, and so the picture has to be redrawn to make it bright again before that happens. If your eye sees the picture fading, then when the computer draws the picture again you will see it get suddenly bright again-- and it will start to flicker. This is distracting, unhealthy, and disagreeable.

Note that the most important computer in the history of computer display used this technique (I think). This was the TX-2 at Lincoln Laboratories, a highly-guarded installation outside Boston which is formally part of MIT. The TX-2 was one of the first transistorized computers-- perhaps the first; and on it were programmed a number of milestone systems, including Sutherland's Sketchpad, Johnson's Sketchpad IV, and Baecker's GENESYS animation system (discussed somewhere).

2. LINE-DRAWING HARDWARE

The next step in design is to get the computer program out of the business of drawing lines by a succession of dots. So we build a piece of hardware that the computer program may simply instruct to draw a line. As an interface, it looks to the computer like four separate devices: registers that tell where on the screen the line must start ("first X" and "first Y") and registers that tell it where to stop ("end X" and "end Y").



This speeds things up considerably, and allows the computer program to display on the CRT simply by telling the device what lines it wants drawn. Moreover, the program is free to do other things while each line is being drawn, though this involves the problem of how the program is to know when it's time to send out another line-- and we needn't go into that here.

(Incidentally, it is a puzzling fact that such a device is available nowhere, although lots of people end up building one for themselves. There was such a thing on the market a couple of years ago-- line-drawing hardware with no interface and no CRT-- but it was withdrawn because of reliability problems. A just price, if anybody wants to go into that, would be five hundred to a thousand dolars-- this year.)

3. EVOLUTION FROM THIS: TWO OPTIONS

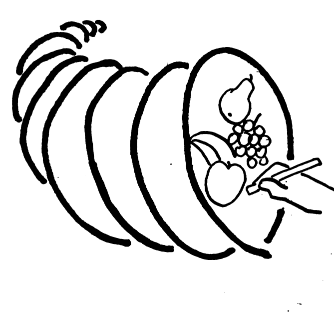
There are basically two ways to go from this basic starting point. Either we can keep the display device intimately and integrally connected to the computer, or we can say the hell with it and cut the display device loose as a separate entity.

Ivan Sutherland has cannily noted that there is a certain trap involved in these designs: as we build additional "independent" structures to take the burden of display away from the computer, we are tempted to keep adding features which make the "independent" structure a computer in its own right. This paradoxical temptation Sutherland calls "the great wheel of Karma" of computer display architecture.

It is tempting to cut the display loose from the computer. It means the computer can be fully occupied with other matters than refreshing the screen-- preparing the next displays, perhaps. Many computer people believe this is the right way to do it, and it is certainly one valid approach. But unfortunately it also drastically reduces the immediacy of the system's reaction, making interaction with the system less intimate and wonderful.

Approaches which put display refreshment and maintenance in a separate device are less interesting to me, and so that discussion continues separately nearby. ("Display Terminals," p. DM21).

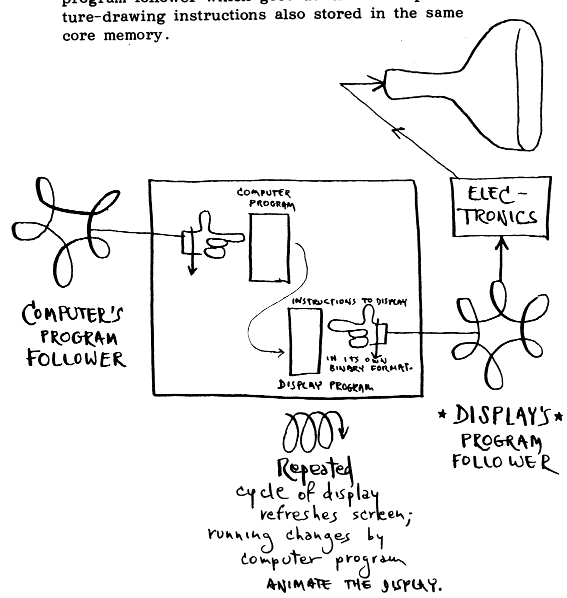
This article continues next page.)



4. THE SECOND PROGRAM FOLLOWER

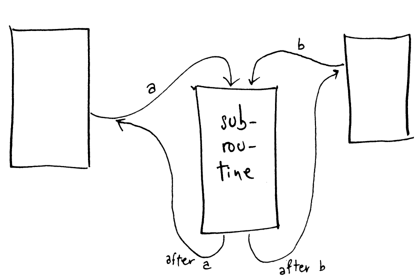
On the other side of the book, I explained that a computer is basically a zippy device, never mind how constructed, which follows a program somehow stored symbolically in a core memory. Such a device we call here a program follower. While programs may be in many computer languages-- all of them contrived systems for expressing the user's wishes, in different styles and with different general intent-- underneath they all translate to an inner language of binary patterns, which may just be thought of as patterns of X and O, or light bulbs on and off. The innermost program follower of the computer goes down lists of binary patterns stored in the core memory, and carries them out as specific instructions. It also changes its sequences of operations under conditions that the programmer has told it to watch for.

The most powerful and responsive computer displays are those which build a second program follower which goes down lists of picture-drawing instructions also stored in the same core memory.



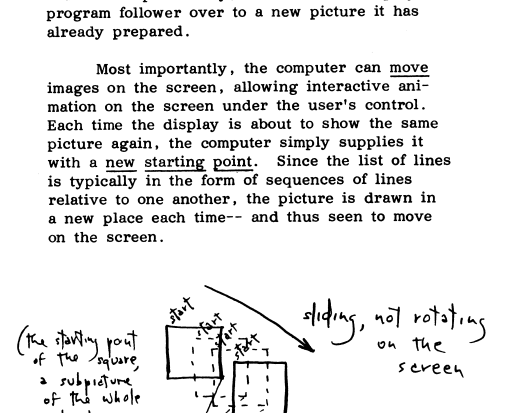
We may call this also a "list-of-lines" system, since the commands recognized by the display program follower are typically patterns that tell it what lines to draw.

Typically also it has its own way of jumping around in a program, and may jump to a specific list of lines, or subpicture, from numerous other parts of its program, always returning each time to the point from which it had jumped. This allows the same subpicture to appear in numerous places on the screen at the same time. (A program that can be jumped to by other programs which then resume operation is called a subroutine; thus the real, or most prestigious, name for such a device is a subroutining display.)



This design has some extraordinary advantages. One is that since the computer's program follower and the display's program follower both share the same core memory, they can work together most intimately. When the user demands something new-- by typing, say, or pointing with a light-pen-- the computer can step in and take various actions. Its program can compose a new picture for the user, get something from a disk or tape memory, or switch the display's program follower over to a new picture it has already prepared.

Most importantly, the computer can move images on the screen, allowing interactive animation on the screen under the user's control. Each time the display is about to show the same picture again, the computer simply supplies it with a new starting point. Since the list of lines is typically in the form of sequences of lines relative to one another, the picture is drawn in a new place each time-- and thus seen to move on the screen.

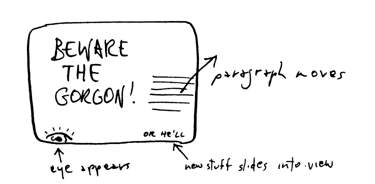


This design has some extraordinary advantages. One is that since the computer's program follower and the display's program follower both share the same core memory, they can work together most intimately. When the user demands something new-- by typing, say, or pointing with a light-pen-- the computer can step in and take various actions. Its program can compose a new picture for the user, get something from a disk or tape memory, or switch the display's program follower over to a new picture it has already prepared.

Most importantly, the computer can move images on the scren, allowing interactive animation on the screen under the user's control. Each time the display starts to show the same picture again, the computer simply supplies it with a new starting point. Since the list of lines is typically in the form of sequences of lines relative to one another, the picture is drawn in a new place each time-- and thus seen to move on the screen.

Finally, the computer itself is free most of the time-- free, that is, to do other things, which typically is always desirable. Just how much the computer can or should do in such a partnership is a matter of dispute. (Ordinarily such devices are spliced onto minicomputers; and minicomputer fans, such as the author, see no reason not to perform all services for the display there in the minicomputer-- and a pox on the big machines. Others, for various reasons, see the subroutining display and its host mini as needing the tender ministrations of a big-computer via some sort of communications line. There are various reasons for holding this entirely legitimate view. People who are devoted to the high number-crunching capacity of big computers, or to languages which require great big computers to run in, have a right to their opinion. Moreover, it is currently feasible to store large bodies of data only on big computers -- not because big disk and tape memories can't be easily attached to the small ones, for they can, but they usually aren't; and other ways to tie minicomputers to big stores of data aren't available yet.)

Subroutining displays often have commands allowing them to display text as well as lines and dots. In the display of text they can use the same technique of "moving the picture" by starting its display at successively creeping points; this will cause, say, whole paragraphs to slide on the screen. The importance of this feature in the displaying of text cannot be overemphasized. As more and more people have experience with displays of different kinds, they are beginning to realize how confusing and disorienting it is for a screen to clear and be filled with something new to read. You don't know where you are. On subroutining displays, moving the text can give the reader the same sense of orientation he gets from turning pages -- an important thing to replace.



It must be stressed here that, just as computers themselves have no fixed mode or style of operation, neither do computer displays; and so the purpose of such devices is simply

HELPING PEOPLE SEE AND MANIPULATE
PICTURES AND TEXT
IN ANY STYLE, AND FOR ANY PURPOSE.

Since pictures can be of anything, and text can be about anything, this effectively comprehends the entire mental and working life of mankind.

Many readers will scoff, supposing that computer display systems will always cost tons of money. This is not the case. You can already get a beauty, with its minicomputer, for as little as $13,000; and this price should fall to three or four thousand within a few years-- as soon as the minicomputer manufacturers realize that the market frontier is not in the office or factory, but in the home. But we're getting a bit ahead of ourselves here.

TYPES OF SUBROUTINING DISPLAY

Some early subroutining displays used a screen-dotting technique, but took the burden of it off the computer itself: it would extract from core memory the instructions telling it to draw individual lines and show text. (I refer here to the DEC model 338, introduced about 1965; this attached to a PDP-8 computer (see p.40) and cost about $50,000 including the computer.) Others drew lines as straight zips of light across the screen; an example is the IBM 2250 display, introduced about 1966. (The model 1 of this device buckled directly to the 360, and cost, I believe, something like $75,000; its successor, the model 4, buckled to their 1130 minicomputer, the package costing some $150,000, and then you were supposed to attach it to an IBM 360.) The 2250 was a good machine, but in performance suffers greatly from the restrictions of the 360 computer itself (see p.41).

These earlier machines are being replaced by new versions with better-designed instructions (see "Computer Architecture," p.32, for a sense of what well-designed instructions are). An especially fine unit is DEC's GT40, which buckles on the exceptionally fine PDP-11 minicomputer (see p.42). The GT40 is illustrated nearby. (p.DM21) It goes for some $12,000 including the computer. (That's today, we repeat. Consider not the price at this instant, but how fast it's going down.)

The units mentioned above are of the most basic type: "two-dimensional," whose pictures at any given instant correspond to flat drawings -- but, of course, derive their excitement and magnificence from their capacity to interact, change and animate what you are looking at.

THE MIND'S EYE continues on p. DM30.

Sutherland's SKETCHPAD

Seldom has an event in a new field had as much power and influence as what dour Ivan Sutherland did as a young man in the period 1960-64.

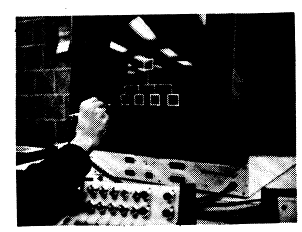
The SKETCHPAD system, which was basically his thesis work at MIT, was at once inventive, profound, overwhelmingly impressive to laymen, and deeply elegant. Simply for the universal influence it has had in the computer field, it deserves our close attention.

Sutherland was one of the first people to understand the use of the computer in helping people visualize things that weren't fully clear yet-- the opposite, of course, of the conventional notion of computers. While computers had been made to do animations as early as the forties, and computer graphics had been put to workaday duties in the old SAGE system (defending us against bombers in the fifties-- remember the good old days?), Sutherland turned computer display from an expensive curiosity into a true dream machine.

SKETCHPAD ran on the 36-bit TX-2, a one-of-a-kind experimental machine at Lincoln Laboratories (a military research place nominally a part of MIT). It had a display screen, light pen and lots of handy switches.

SKETCHPAD was basically a drawing system. But rather than simulating paper (as some people might have done), it found splendid ways to take advantage of the computer's special capabilities.

In the Sketchpad system, Sutherland looked for ways that a responding computer display screen could help people design things. He pioneered methods of drawing on screens, with such techniques as the "rubber-band line" (a straight line on the screen, one end of which follows your lightpen while the other remains fixed), and the "instance"-- a subpicture stored in core memory which could appear numerous times and ways in a larger picture).



This picture vaguely simulates the "instance" facility of Sketchpad, by which an overall picture may be created out of repetitions of a single master pattern.

Simulated with GRASS language (see p.31).

The mind-blowing thing about Sketchpad was the way you could move and manipulate the picture on the screen, with all its parts. One overall picture could be constructed out of a hundred copies of a basic picture; then a change in the basic picture would immediately be shown in all hundred places. Or you could expand your picture until it was effectively the size of a football field (with you looking at a tiny view in the handkerchief-sized screen). Or you could draw meshing gears on the screen, and with the lightpen (and through the "constraint" facility) make one gear turn by turning the other!

This elegant technique, the constraint, does not seem to have been imitated even now. A "constraint" was a restriction placed on some part of the overall stored picture complex. The user could move or manipulate various parts of the picture on the screen, but the parts that had constraints could only move in certain directions, or according to certain formulas, or dragging other parts along, etc., as specified.

This was a profound idea, because it meant that any rules for the manipulation of particular objects on the screen could be added to Sketchpad as particulars within the larger program, rather than having to be programmed in from scratch.

(One extremely interesting aspect of Sutherland's thesis, which most people seem to have missed, dealt with displaying a structure of constraints: that is, showing what elements depended on what other elements, in a highly abstracted diagram that the system could show you. This form of display has remarkable possibilities.

After his brilliant SKETCHPAD work, Sutherland was made head of ARPA's computer branch (see "Military," p. 58). There he was involved in many of the computer funding decisions of the late sixties, which contributed to the impetus of computer display. (His predecessor, Licklider, had been a pioneer in time-sharing, and much of the forward movement in the computer field in recent years may just have had to do with the strategic position of those two men when they were at ARPA/IPT.)

Sketchpad went on as a continuing research tradition at Lincoln Labs. Timothy Johnson, for instance, made a version of it that allowed the drawing of three-dimensional objects; this became the forerunner of the various three-dimensional line systems described hereabouts.

From ARPA, Sutherland went on to the University of Utah, whence he slipped off with the Computer Science department chairman to found the Evans and Sutherland Computer Company, makers of the top-of-the-line computer display systems (see p.DM30 and p.DM33).

Sutherland's work has shown an elegance and inventiveness outstanding in the field. (For instance, I believe one issue of Communications of the ACM had two unusual articles by him: one describing an eccentric "Chinese auction" system worked out for scheduling use of a computer, which benefited users more than any previous method; and the infamous "Great Wheel of Karma" article, where he compared the design of graphical computers to the Hindu system of reincarnation-- if you keep adding desirable features to the design, soon you have another program follower and another computer in the same box-- over and over.)

COMPUTER MOVIES

How do computers make movies?

Well, first of all, computers do not make movies unless thoroughly provoked.

In fact, only people make movies. But computers, if sufficiently provoked, will do a lot of it: enact the movie and photograph it, frame by frame.

There is no single method.

All forms of computer display and computer graphics may be used to make computer movies.

"Computer animation" is any method of making movies in which a computer successively draws or paints the successive individual frames, which may be done by any of the methods mentioned in this book. Now, since there are numerous methods of making pictures by computer, then any method of making different individual pictures, in a succession of changing frames, is computer animation. So a "computer movie" is any film made by, or with the picture-making aid of, computers.

In other words-- it's no one thing.

Now, there already exist hundreds, if not thousands, of computer movies. So far most of them have been on technical topics-- the mechanics of satellite orbit stabilization, the mechanics of explosions and so on.

Here are a few stills from some other movies, more humanistic.

BIBLIOGRAPHY

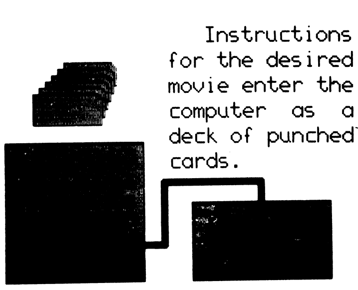
Newman & Sproull, Interactive Computer Graphics. McGraw, $15.

This is the textbook. Anyone interested in computer display should get this immediately.

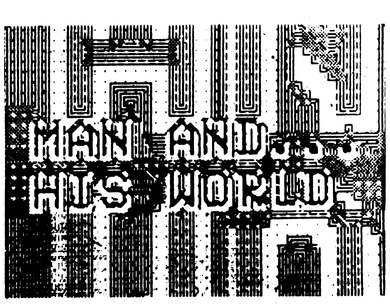
An expensive journal, Computer Graphics and Image Processing, comes from Academic Press.

Sherwood Anderson, "Computer Animation: A Survey." Journal of Micrographics, Sep 71, 13-20. Lists nineteen computer-animation languages of that time.

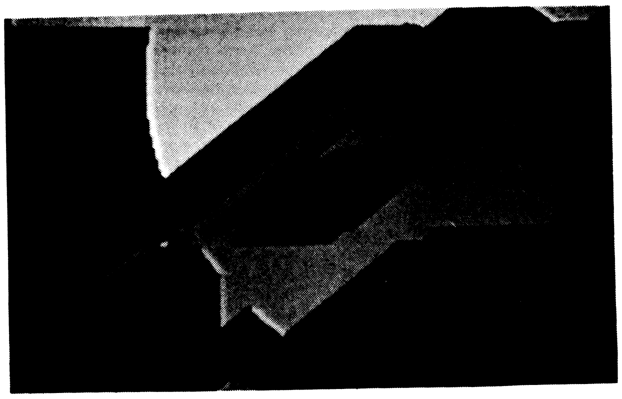
Ken Knowlton, "Computer-Made Films," Filmmakers Newsletter Dec 70, 14-20.



Vintage Knowlton, using BEFLIX. (This language used the COM quite efficiently: dots were actually out-of-focus letters.)



Vanderbeek & Knowlton (using TARPS, which shows strong influence of BEFLIX, which it grows from).

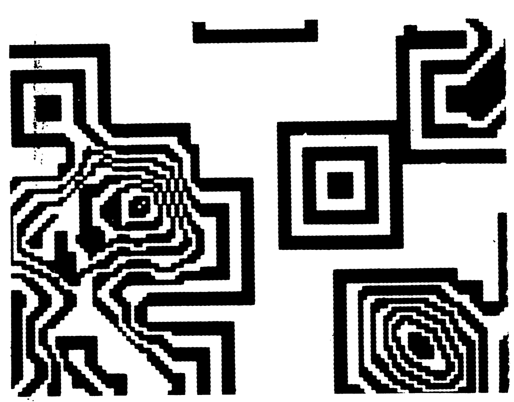


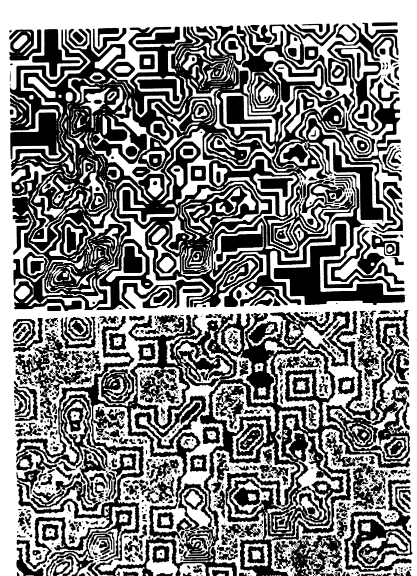
Lillian Schwartz

LILLIAN SCHWARTZ

A talented artist with a feel for technology, Ms. Schwartz has been working for several years with Knowlton and others at Bell Labs. Her films with Knowlton, mentioned elsewhere, are marvelous. She now works at a more permanent setup, a minicomputer that runs successive images on a color TV screen, employing a modified form of Knowlton's EXPLOR language. The work is immediately viewable. This allows rapid film construction, not previously possible when the work had to go through a slow animation camera before she could see the result.

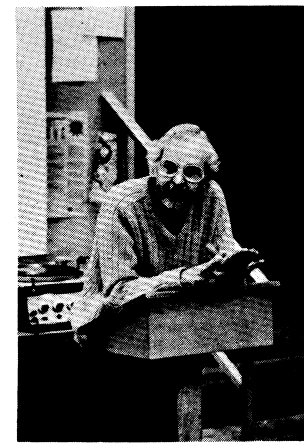
For Knowlton-&-Schwartz films contact: Martin Duffy, AT&T, 195 Broadway, NY NY.





Schwartz & Knowlton. Using the EXPLOR language, they make pictures and patterns scintillate and grow together. (EXPLOR in some ways generalizes Conway's Game of Life; see p.48 and p. DM 26)

JOHN WHITNEY



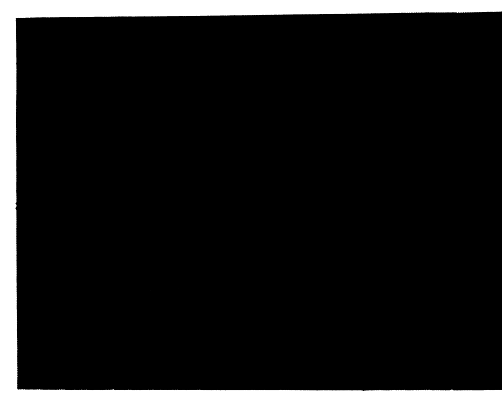
JOHN WHITNEY

John Whitney is the ancestor of us all, probably the first computer movie-maker. He is also a gripping speaker.

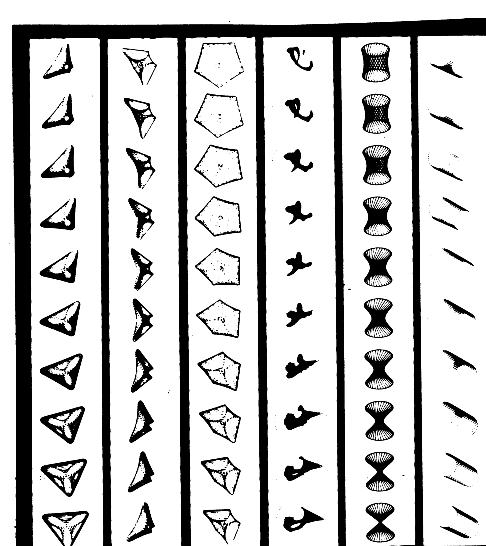
In the forties, he built a special animation stand-- using analog computers.

Deeply concerned with music, Whitney has in his images emphasized rhythmic and contrapuntal movement of shapes and lines.

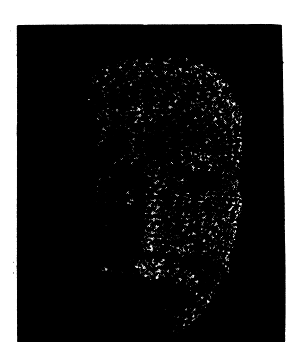
Whitney films available from: Pyramid Films, Box 1048, Santa Monica CA 90406.



John Whitney



John Whitney



Lillian Schwartz (with Henry Magnuski; see p. DM34)

RON BAECKER'S GENESYS

By now there are dozens of computer animation languages-- perhaps hundreds. Each one employs the techniques of animation which its developer wanted to use, tied together in the ways that seemed appropriate to him. (See "Computer Languages," p. 15, and note Knowlton's various animation languages, described nearby.)

One of the more influential animation systems has been Ron Baecker's GENESYS, a 2-dimensional animation system programmed in the late sixties at MIT's high-security Lincoln Laboratory. (It used the TX-2 computer, mentioned elsewhere in this book.)

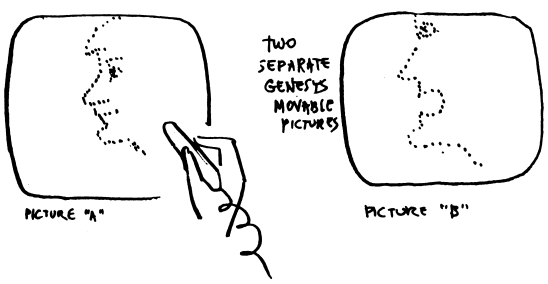
Baecker, a cheery and genial fellow, expressed interest as a student in using the TX-2 for animation, and was allowed to. The system he produced has a number of lessons for us all.

GENESYS is a "Good-Guy" system, as discussed on p. 13 Meaning, in this case, that it is easy to learn and simple to use. As argued elsewhere in this book, making computer systems clear and simple is often hard for the programmer (and may go against his grain), but is essential.

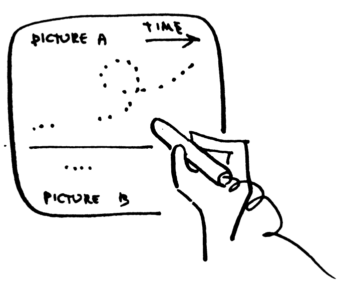
PICTURES AND MOTIONS

GENESYS makes the following simplifications of your movie: all images are made up of dots. They do not change as you watch; animation consists of the images either moving or being replaced.

To create an image, you draw it onto the screen with a lightpen or a tablet. (As in the SKETCHPAD system; see p. DM23.) Parts of the image may be changed until you're satisfied.



Now, to create the animation, you do the same thing. Each image can be made to move on the screen; and the path of the motion may be drawn on the screen, through the picture area. Not only that, but the timing of the motion is controlled through the same diagram, by the spacing of the dots. (Baecker calls his control diagrams p-curves.)



Lastly, sections of picture may be replaced by means of the control diagram (as indicated in picture above).

Having created such an animated sequence, which is stored in symbolic form in the computer ("digitally"), you can view it on the screen, decide what you do and don't like about it, and change any part of it.

The basic elegance of the system is this: Baecker made everything work the same way, through control by screen diagrams. He simplified the animation problem in a clear and simple way.

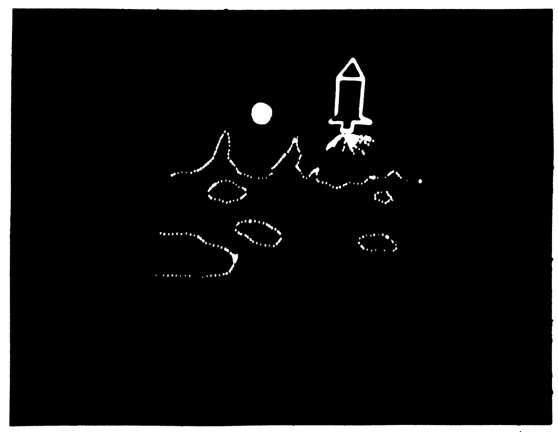
Ron now teaches in Canada and is into working with PDP-11s. The results should be fun.

LYNN SMITH

Lynn Smith is a young Boston artist who has worked extensively with Baecker's GENESYS (see nearby). One result has been a movie which should be an example to us all: "The Wedding Movie for Bob and Judy." (Her Friends Bob and Judy were getting married, so she made this movie, a few minutes long and quite clever, to celebrate it.)

This is my favorite example of how computers should be used in the human world; it says more on the subject than any dozen articles.

(One question that remains unanswered is how a system like GENESYS could have been used for such a purpose, seeing that most people in the field believe GENESYS only runs on the heavily-guarded TX-2 computer. Regretfully, I can shed no light on this here.)



COMs (Computer Output Microfilm devices)

are what you use to make computer movies. Basically they consist of a CRT and a movie camera in a box.

Mostly they are used to put text on microfilm by computer, so generally they are not connected to a computer but run off magnetic tape.

This turns out to be very annoying if you want to hook up the computer directly to the COM, and make movies that fill the frames spot-by-spot. For that you really need your own movie camera and a minicomputer. (Movie cameras that can be made to start and stop by computer are called "pulse cameras" or "instrumentation cameras.") The society for people who make Movies by Computer is called UAIDE (Users of Automatic Information Display Equipment-- an obsolete title). It used to be a club just for companies that owned COMs made by Stromberg Datagraphix, but evidently it has now cut itself loose and become a subsidiary of the National Microfilm Association, 8728 Colesville Road, Silver Spring MD 20910.

(NOTE: for them as want to make color movies, the two alternatives have been either to have separate primary negatives combined at a lab-- the "old Technicolor" process-- or to add a complicated color-filter box to a COM or other CRT setup. Such things are available commercially now, from Dicomed-- a whole Color COM.)

BIBLIOGRAPHY

Computer Output Microfilm. $10 from National Microfilm Assn., above. Lists available COMs and service centers.

THE NEW REPUBLIC OF

(Above: PLATO LIVES! Anyway, the word itself goes through changes in the Game of Life (see p. 48), as programmed for the PLATO system by Danny Sleator, and photographed from a PLATO screen.)

PLATO is the world's greatest computer display system.*

Some 500 users, at terminals around the world (but mostly in Illinois), simultaneously tie up to a big computer in Urbana, Illinois and savor instantaneous pictorial and text deliveries on their bright orange screens. Diagrams, explanations, tests and even animation of a sort, flow almost without interruption to the bright orange screens all over. The system is extremely responsive: depending on what the user is up to, its various programs can respond to each pressing of a key, usually within a fraction of a second.

While literature on PLATO is copious, it is hard to read and slightly sales-oriented. But a few minutes' intercourse with a PLATO terminal makes anyone an enthusiast for the system.

PLATO is the brainchild of Don Bitzer, a U. of Illinois engineer who has devoted over a decade to its creation. Michael Scriven, no slouch himself, has called Bitzer "one of the great men of our time." Bitzer is also certainly one of the world's greatest salesmen. A crew-cut, huggy-bear sort of a fellow, he flies around the world demonstrating, lugging a great terminal along. When you sign on the system you may be informed that Bitzer is at that very moment demonstrating in Paris or Tokyo. This "travelling dog and pony show," as PLATO staffers call it, has created awe and excitement wherever it goes, and where the awe has been strong enough to generate money, there you will now find PLATO terminals.

If you have a PLATO terminal-- "you" presumably being a school or other favored institution-- you can in principle log onto PLATO from anywhere in the world, though most terminals stay in one place. There is one main network, consisting of a big Control Data computer in Urbana (the model 6800; see p. 41) with tendrils extending out into the phone system and the educational TV cable of the state of Illinois. When the Urbana system is "finished" and fully loaded, it will have 1008 terminals; all are already spoken for. The PLATO terminal is a totally unique animal (see box), manufactured (all too slowly) by Magnavox, incorporating a terrific plasma panel built by Corning. (The plasma panel was invented by Bitzer, and even though much of PLATO was publicly funded, he is reputedly rich from it. We said he was a great politician.)

* *In terms of high performance for lots of users. Various systems (described hereabouts) offer more power, but at huge cost.*

As a first taste of interaction on a graphical computer system, PLATO can be a thrilling mind-opener-- especially to people who think computers can only behave loutishly or through printout.

PLATO is a complete stand-alone system, with its own monitor program or "operating system" (see p. 45) running on the CDC 6800 computer all by itself. It does not run on any other manufacturer's computers, or simultaneously with any other big programs. It communicates only with PLATO terminals, no other, and PLATO terminals, because of their unusual design, can communicate only with it, partly because of its unusual design and partly because of its unique 20-bit interface. (See diagram of PLATO terminal, box nearby.)

A PLATO terminal costs about $4000 and the price seems to be going up; $5000 in the next few years is a popular estimate. But you can't just buy one. You have to get on the waiting list, and who are you, anyway? There was a time when almost anybody could buy into PLATO, but now that the system is unstoppable, applicants are being scrutinized.

Is it really unstoppable? Educational Testing Service, of Princeton, is conducting an elaborate Effectiveness Evaluation of the PLATO system, presumably to decide whether it should live or die (on public funds). But with so many terminals in the field already and so many man-years already gone into its creation and the making of materials for it (-- the ghastly term "authoring" seems likely to stick), it is hard to believe PLATO could die. Not now.

Especially considering that two more systems are now being put together: at Lowry Air Force Base (Colorado) and Florida State University. That means there will be whole other computers of the CDC 6000 series running the PLATO Monitor and shepherding PLATO materials to users at PLATO terminals, unconnected to Urbana, one for Lowry AFB and one in Florida.

And it won't end there.

Control Data, whose vested interest in the system (though they didn't pay for its creation) is enormous, is said to be projecting

ONE MILLION PLATO TERMINALS BY 1980.

Another sign in the wind: Montgomery Ward has one.

Now, to call the PLATO system a "computer graphics" system may seem somewhat odd to people who know it in another guise, as a system for Computer-Assisted Instruction (called CAI). But as the author does not like CAI in general, at least as it's been going-- see p. DM 15-19 -- and rather likes PLATO, I prefer to describe it as I prefer to see it.

Nevertheless, to understand PLATO properly we had better consider what the people have been doing in terms of what they think they have been doing, and offer any amendments or restatements later.

"OPTIMIZED FOR CAI"

PLATO stands for "Programmed Logic for Automated Teaching Operations," and has been billed (and sold) as a system for automated instruction.

It is, most PLATO fanciers will tell you, "optimized for instruction." ("Optimized," in computer talk, means "just what somebody says you need for a specific purpose.") As with any system, the leaps of faith between its basic design premises have become lit by airport beacons; clearminded individuals with alternate views have difficulty making themselves understood to some PLATO enthusiasts. But the most basic underlying feature of the system, INSTANT RESPONSE, cannot be quarreled with. PLATO can respond, as already mentioned, to a single key-pressing by a user, almost instantly; this feature is virtually impossible, say, on IBM systems (but see lower-left box p. 55). This responsiveness is the system's greatest beauty.

Because of the need for high responsiveness, it was decided that all users had to have their particular programs ("lessons") running in core at the same time. That meant there would be no swapping (bringing in materials from disk memory), which can bring mortifying delays (if a lot of people need it at once); but it also meant lessons have to be very small. Large bodies of material, which would have to be moved in from disk, are not allowed; thus each lesson is basically a little love-nest that must generate its own action. Hence there is an emphasis on little programs to respond various ways, rather than text which may be read in quantity.

Partly because large amounts of text cannot be shipped to the user, a little PROJECTOR is in the terminal. It uses a tiny microfiche, or microfilm sheet, small enough to fit in the palm of your hand.

If a PLATO author deems it necessary, he requires for his lesson, not just the use of the keyboard and plasma screen, but a microfiche as well. The student must put the microfiche in place when he starts the lesson; signals from Urbana (or wherever) then jump the projected image among 256 different images, in response to what the student does.

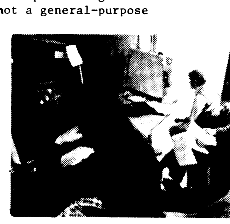
Now, PLATO people are not doctrinaire about how their system is to be used. The plasma screen can be continuously showing little decorations along with the teaching material. The microfiche could be showing irrelevant works of art or travel scenes. These are all facilities at the option of the PLATO author; at his beck and call, if he thinks his program or lesson needs them. (But it's very bothersome to have the microfiche made-- an important difficulty.)

Every terminal has the screen, the keyboard, and the projector. Other options may be added, however:

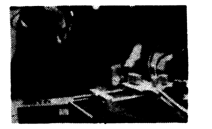
1. The touch panel, This is a transparent window that goes over the plasma screen and reports to the main computer whether it has been touched, and where. (This allows illiterates, especially kiddies, to use the system without typing.)
2. The audio disk. This allows the terminal to respond with sound, including canned words, to the student. (It does not actually synthesize the sound, as discussed on p. DM11.)
3. The general jack. Not to be confused with Pershing, this is a connector socket that will send and receive data from any other device-- provided you've got the right interface. This allows all kinds of other devices, such as piano keyboards, to be used for student input. Or output (like gum-ball machines.)

Actually, except for the restriction on quantities of material that can reach the student, PLATO is an extremely general system. Despite the strange convention of calling all user programs "lessons"; despite the odd stipulation that all users are called either "students" or "authors"; and despite being told by PLATO spokesmen that PLATO is not a general-purpose system; actually, it is.

Amongst the terminals, PLATO room, Circle Campus. What one person is doing ordinarily has no bearing on the others, who could as well be in Timbuktu as far as the main computer is concerned.



PLATO's audio device permits the system to respond to the user with a spoken phrase, snatch of music, or whatever -- in a fraction of a second. The magnetic disk is forever turning; compressed air shoots the read-head to the required track on the disk for the reply.



The hardware was designed by Bitzer. The software-- that is, the underlying computer part, never mind the contents to be shown (also regrettably called "software" by many hands)-- was initially less stressed by Bitzer, but eventually grew under the direction of others. In particular, an ex-biologist named Paul Tenczar (pron. "Tenzar") created its underlying TUTOR language. (For an introduction to computer languages see p. 15 and what comes after.) The TUTOR language exists only on PLATO; and PLATO authors may only use the TUTOR language, Paul Tenczar's creation.

The TUTOR language can best be understood as a reaction to Coursewriter, another CAI language offered by IBM on its 1500 Instructional System, not described elsewhere in book. Coursewriter's original intent was to enable non-computer people, especially teachers, to create drill-and-practice instructional lessons roughly of the type

$$\text{Now, } \underbrace{\text{Johnny}}_{\text{name filled in}}\text{, what is } \overbrace{3 \times 5}^{\text{particulars filled in}}?$$

$$\text{Kid's answers}\begin{cases}10 & \text{No.}\\ 15 & \text{Good!}\end{cases}\text{ System replies.}$$

Obviously, by changing the numbers and pushing the kid on types of problems he hasn't mastered, the computer can patiently bring students to mastery of various simple skills, diagnosing weaknesses and stressing the individual student's problems. The difficulty is that attempting to extend this method out of the very simple has great pitfalls and may not even be worthwhile (see pp. DM 15-19).

Anyway, Coursewriter was promulgated by IBM with the 1500 and thus suffered premature standardization before things had been thought out. IBM is not to blame for Coursewriter's deficiencies, they were just trying to make a buck; but because a lot of scared people believed Coursewriter was the way it had to be, the evolutionary improvement usual for computer languages didn't have time to occur. An egregious omission: Coursewriter did not allow the author much access to the computer itself. That is, programs written for numerical calculation, say, could not be brought into instructional materials at a sophisticated level.

Tenczar's TUTOR changed all that. It has both the virtues and defects of being original. Apparently unlettered in computermen's controversies and dogma, Tenczar designed a language of great power and speed; it is utterly strange to computer people, offers various brilliant features, and is in some respects quite irritating. It looks very simple to the user-- but beyond a few deceptively simple techniques, it has to be learned in considerable detail to do anything interesting. (See box, "Is It Better to Toot?", next page.)

This tale has, of course, been simplified. Bitzer and Tenczar did not work alone, but rather were leaders in a seething community of dozens of smart people working like blazes on the project. It has taken some fifteen years of Bitzer's effort, and tens of millions of dollars, to get the system where it is now-- Ready and Working.

Project PLATO now extends far beyond its original domain. Originally a fairly tight nucleus at the Computer-Based Education Research Laboratory ("CERL") at the U. of Illinois in Urbana, the community of PLATO now sprawls out through its lines to a larger constituency, the PLATO community of users.

(Indeed, this extended Republic of PLATO-- the systems people (see p. 45) in Urbana, the authors and locals-in-charge throughout the network-- constitute one of the maddest rookeries of computer freaks in the world. Where else would you find a 14-year-old systems programmer who's had his job for two years? Where else would you see people fall in love over the Talkamatic (a PLATO program which allows you to have written conversations with people at other terminals, wherever they may be) only to clash when at last they meet in person? Where else can you play so many different games with faraway strangers? (See Box.) Where else can students anywhere in the network sign into hundreds of different lessons in different subjects (most of them incomplete)? Where else are people working on various different programs for elementary statistics, all to be offered on the same system?)

PLATO is one of the wonders of the world.

Mike O'Brien, a Tolkien fancier, has put the entire Elvish alphabet onto PLATO as a special character-set. Here the system gives a famous warning to turn back, both in English and Elvish. Mike says it intimidates snoopers poking around his material.

Unfortunately, there are so many learners, and so few PLATO terminals, that use of the terminals must now be fairly strictly controlled. (The eight terminals at the University of Illinois at Chicago Circle, at which most of these pictures were taken, generally work an eight-hour day.) The time was when people could just walk in, sit down at a terminal and do what they liked; now, sadly, each user must have an "account" and a password.

But the rabble is howling at the gates. Many professors want to use it to take rote aspects of teaching off their backs; and the computer bums and students want to play the PLATO games (see box) and tinker with an interactive system of its power and lusciousness. But most of them will have to wait.

PLATO's services are "free," for now. That is, if your school has PLATO terminals, and IF it will pay for the communications lines, THEN the services of the central computer are "free"-- the National Science Foundation is bankrolling its operation for a couple of years more. Then, bango, PLATO central service becomes something that has to be paid for too.

Just to give you an idea, the communication costs to Urbana for Circle Campus's eight terminals run at over $10,000 a year. But these costs should be coming down sharply; it is the price of tooling up for whatever the PLATO future is going to be. Anyway, the general cost of the system comes out to about $1.50 an hour, the same as general time-sharing on a PDP-10 (see p. 42). But that's without paying for the central computer-- another cost which we expect to go down, however.

This is all a far cry, of course, from Bitzer's claim a decade ago that PLATO terminals would cost only $400. But considering the system's success, we needn't dwell on that.

Perhaps the real question is this: with man-machine intercourse of this quality now possible, can people's love for the system stay Platonic?

PLATO GAMES

Moonwar on a Saturday in Urbana. It's man-to-man among the craters; then a quick kill of the unknown adversary.

And our doughty warrior looks to the Big Board for more challengers. Kids love PLATO games.

They work hard and they play hard on the mighty PLATO system.

When the Author gets tired of Authoring, or the Student of Stewing, just around the corner, a few keystrokes away, are diversions and games to boggle the imagination.

You can go to a program ("lesson rose") and look at "the great roses"-- elaborate curlicues generated by mathematical patterns that appealed to the authors of that program; or find, also tucked in rose, Conway's Game of Life (see writeup, p. 48 , and picture series, nearby).

Then there are games you can play against the system, like racetrack and blackjack. (These games let you win astronomical sums of money-- play money, forgotten when you sign off.) Remember, of course, that you're not really playing against a computer but against a specific program, with its quirks and shortcuts and blind spots.

Then there are games you play by yourself-- actually responding resources (see pp. DM 18-19), which entice you into trying things out. Tenczar himself has created two elegant, gem-like lessons, man and picto, which teach you computer programming without ever saying so. These two programs present the user with a little picture of a man on the screen, and show him how the little man may be moved around and made to pick up pictures of balls. From there on the student may have his way-- and is never told that he's learning to program a true computer language. (Though it is a quite restricted one, dealing exclusively with little men and their excursions among balls and falling sticks.)

Another charming game, I don't know by whom, is called candy factory. Here too the user may control the animation of the picture by what he types. Machines are seen to manufacture candy, box it and ship it-- depending on what buttons you press.

Some games are played between people who sit together before a single PLATO terminal, often with teaching intent. Such games include the hop game, where Bunny (you) and Frog (your friend) add their way along a board with numbered squares. Older children can dig How the West Was (1+2)X 3, which involves grouping the numbers you get by chance to try to get ahead of the other stagecoach.

THE "BIG BOARD" GAMES

Still another category of games, though, awaits the adult who craves real excitement. Because PLATO has so many terminals, all over, there is a curious combination of anonymity and intimacy between users (-- much like the curious Nonexistent Phone Numbers of Paris; in the French phone system, people calling the same nonexistent phone number can talk to each other; strange blindfolded encounters occur at the Number of The Day, spread by word-of-mouth; sometimes these result in people really getting together...)...

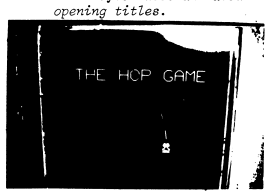
Anyway, the Big Board games of PLATO have exactly that: a shared list, or "Big Board," showing who is playing the specific game.

But you don't have to use your right name.

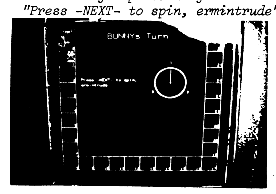
In this jaunty society of shadows, you pick your own nom de guerre, or fighting name. This has numerous advantages: the most obvious is that as you improve at play, you can shed the identity in which you have been humiliated.

The main games with Big Boards are that old standby, spacewar (rocketships wheeling and firing at each other and sliding around on the screen); dogfight (biplanes wheeling and firing at each other and sliding around on the screen), moonwar (shooting at the other guy by specified angles as you stand among craters). In addition, PLATO offers (not during working hours) what must be two of the most baroque space-war games anywhere, empire (eight races (the Vulcanians, Klingons, etc.) seek to control the galaxy) and nova (simulated navigation among millions of different stars and solar systems, all of which may be revisited, all of which are different...)

Welcome to the Hop Game. PLATO often uses animated opening titles.

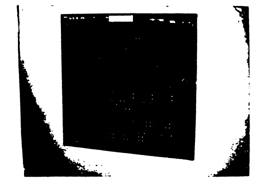


Here it is Bunny's turn. Screen instructs you personally: "Press -NEXT- to spin, ermintrude".



The Navigation part of nova is already working. To get around you need instruction; here we are at the Training Center.

View from your Nova spaceship includes perspective view of where you are among billions of stars; and your various controls.



People who only play PLATO games occasionally have to sign on by typing their names into the big board. (They often get slaughtered by the regulars). The regulars-- hah. When they're signed into the system, they have merely to jump to a specific game for their fightin' names to be posted on the big board. A mighty rollcall they make, too-- such great warriors as von Dave, zot, fright pilot, AL 9000, simpson, doc, THE RED BARON, The Red Sweater, The Giant Pud, Fodzilla, tigress, enema salad, Conan, Siddhartha, wonder pig!!!!!, and EXORCIST.

(As those insiders who have automatic sign-on to Big Boards write programs to do the sign-on, their arrival in a Big Board game is often an animated sign-on. The cutest trick is THE RED BARON's: it looks like this.

THE RED BARON ← (plane falling in flames)

It works like this. For dogfight, the terminal already has stored in its temporary memory, as "characters," the little pictures of airplanes that are going to buzz around the screen. So the Baron just follows his name with the code for that special character.)

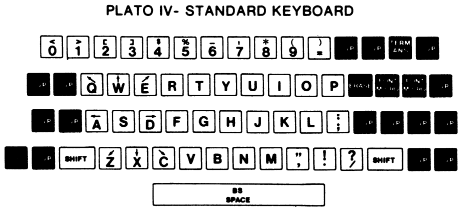
One last point. No longer can you sign on with an obscenity: a little obscenity-checking program looks for the usual expletives, in case visitors or other priggish folk might be looking. But of course this is easy to circumvent by putting periods between the letters of your nasty word, or something similarly deceptive to the poor program.

THE STRUCTURE OF PLATO-SPACE

THE KEYBOARD AND WHAT IT SORT OF MEANS

The PLATO keyboard. What looks odd and arbitrary to you is believed by devout Platonists to be divinely ordained.

PLATO IV- STANDARD KEYBOARD

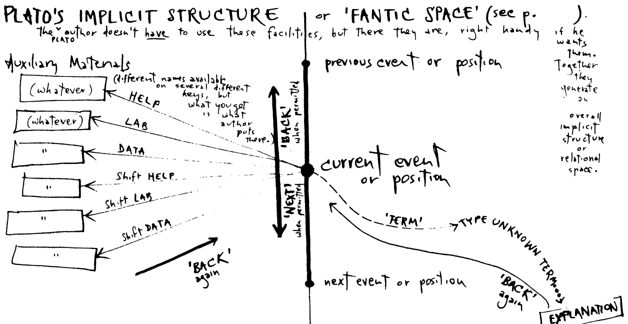


TO MOVE BETWEEN LESSONS, the basic action is to hold down SHIFT and press STOP. (For further complications see Ins-And-Outs diagram.)

TO MOVE WITHIN A LESSON, basic actions are NEXT (to go forward or tell the system it's its turn; BACK, which sometimes returns you to earlier points in the sequence of your lesson; and six step-out-of-line options, by which the author may permit the user to sidestep to explanations, enrichment material, or things out of sequence.

PLATO'S IMPLICIT STRUCTURE or 'FANTIC SPACE' (see p.).

The PLATO author doesn't have to use these facilities, but there they are, right handy if he wants them. Together they generate an overall implicit structure or relational space.



The original idea was evidently that there would be a basic sequence, in which NEXT and BACK would be the forward and back controls, and the other six would represent Help for the Confused, a "Lab" allowing experiments, and additional Data the student decides he needs. The three with Shifts simply provided a second option of each type.

How the author might use these, however, was his own affair.

"TERM" evidently was for when students wanted things Looked Up: by pressing TERM and typing the unknown word, the student would get a definition. "ANS" suggested that it might also be used when the student was allowed the option of being told the answer.

Note the arrows over Q,W,E,A,D,Z,X,C. They allow the student to move cursors, draw, point directions, etc. Unfortunate confusion ensues with the left-arrow on the far left, used in programming (as in APL; see p. 22-3.)

ERASE allows the student to correct his input; COPY helps edit and change things. SUP and SUB allow superscripts and subscripts; FONT MICRO is like a special shift key, going into whatever special font is currently stored on the terminal. I have no inkling of what the little square means.

IS IT BETTER TO TOOT?

"A tutor who tooted the flute
Tried to tutor two tutors to toot.
But he asked through his snoot:
is it better to toot
Or to tutor two tutors to toot?"

Folk thing

The TUTOR language grew out of drill-and-practice, for which it has a command specifying where a student's answer is to appear on the screen. This is the "arrow" command. The language has a strange scanning structure built around this "arrow" command, much as the TRAC Language (see pp. 18-21) has a scanning structure built around parentheses and commas. Beginners don't need to understand the scan and the arrow command, but journeymen do.

TENCZAR'S CONCEPT OF A CONCEPT

Much has been made of TUTOR's facility for "analyzing the content" of what students type in. Actually, of course, the computer does not "understand" what the student says (see "Artificial Intelligence," pp. DM 12-14), but rather offers certain efficient tricks to the person using TUTOR to prepare presentational materials.

Basically, TUTOR's "concept" facility reduces every input word to a 60-bit code. The technique of reduction (called a "hashing function") supposedly substitutes for any word of any language a code of 60 bits (see "Binary Patterns," p. 33), which means the program in TUTOR can rapidly test a student's input for numerous different possible things. (The power of this technique would be readily recognized by computer people; unfortunately there is no room to explain it further here.)

Thus a TUTOR program may contain "concept searches" that test whether a student types either a desired response or numerous alternatives. While it may be strange to call this a "concept," it is a powerful technique.

Paul Tenczar's TUTOR language, the programming language inside PLATO, is like any other programming language (see pp. 15-31); intricate, and unlike its results. That is, a program bears no more resemblance to what it does than the word "cow" looks like a cow.

PLATO is a system for canned presentations that respond to the student. Students need not know TUTOR. Anyone out to prepare such presentations must learn it, however; and the attempt has discouraged many.

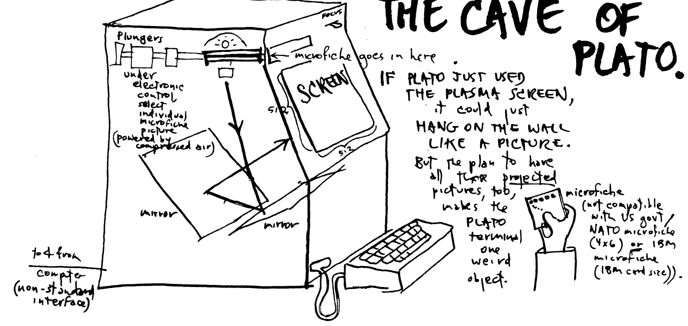
Tenczar is a former biologist, and had no preconceptions from computer orthodoxy to bind him in the design of TUTOR. Thus the language is very original. There is only room to raise the following points:

To learn the first steps in TUTOR-- how to set up drill-and-practice lessons, for instance-- is unusually easy.

To do anything complex, however, requires you to learn the bulk of the TUTOR language. Thus when people say TUTOR is "easy," they mean those first steps.

TUTOR is not Extensible, like, say, TRAC Language (see pp. 18-19) or GRASS (see p. DM 31). That is, a programmer cannot customize the language with new compound functions of his own making. Steps are being taken to correct this; meanwhile, it is said that the Urbana people can be persuaded to put in new commands others want for, e.g., chocolate chip cookies.

THAT EXTRAORDINARY TERMINAL, —— **THE CAVE OF PLATO.**



You can read the standard-size lettering off the screen at SIX FEET-- even though it's NO BIGGER THAN PICA TYPE. Fantastic.

The internal circuitry that draws on the screen is highly capable. Receiving a 20-bit code, the terminal itself deciphers it as--

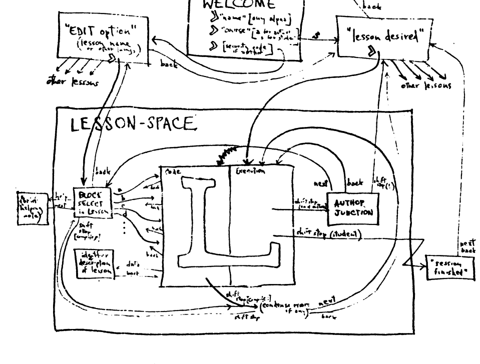
A LINE ON THE SCREEN, or
TWO STANDARD CHARACTERS ON THE SCREEN from its FIXED character memory, or
TWO SPECIAL CHARACTERS ON THE SCREEN from its CHANGEABLE character memory (which can be loaded with Russian, Armenian, katakana, Cherokee or whatever-- even little pictures-- at the start of the lesson), or
A COMMAND TO THE MICROFICHE PROJECTOR, or
A COMMAND TO THE AUDIO PLAYER, or
A COMMAND TO WHATEVER'S IN THE GENERAL JACK.

Note that all lines and characters for the plasma screen can be turned on (orange on black) or off (black on orange).

PLATO'S HANDY KEYBOARD is on a flexible cable, can be worn in your lap.

AUTHOR'S PLATO-SPACE (as of early 1973)

Getting around in this system might be easier. (see "Fantics," p. DM 48-51.)

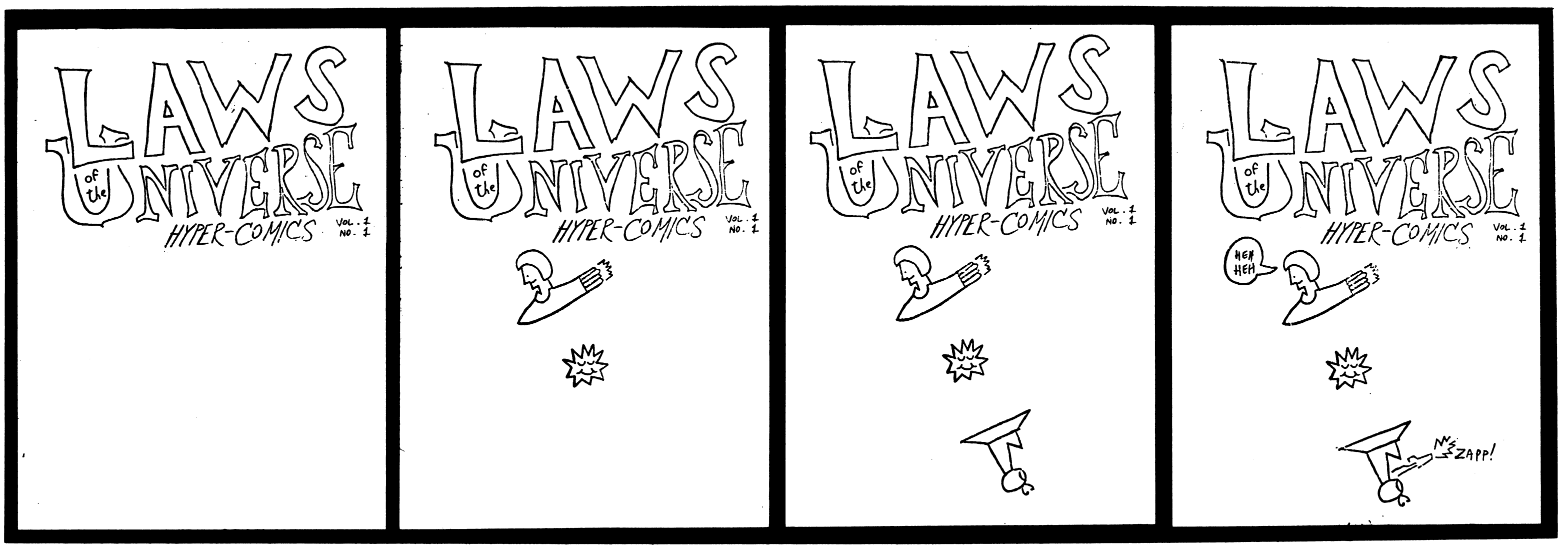


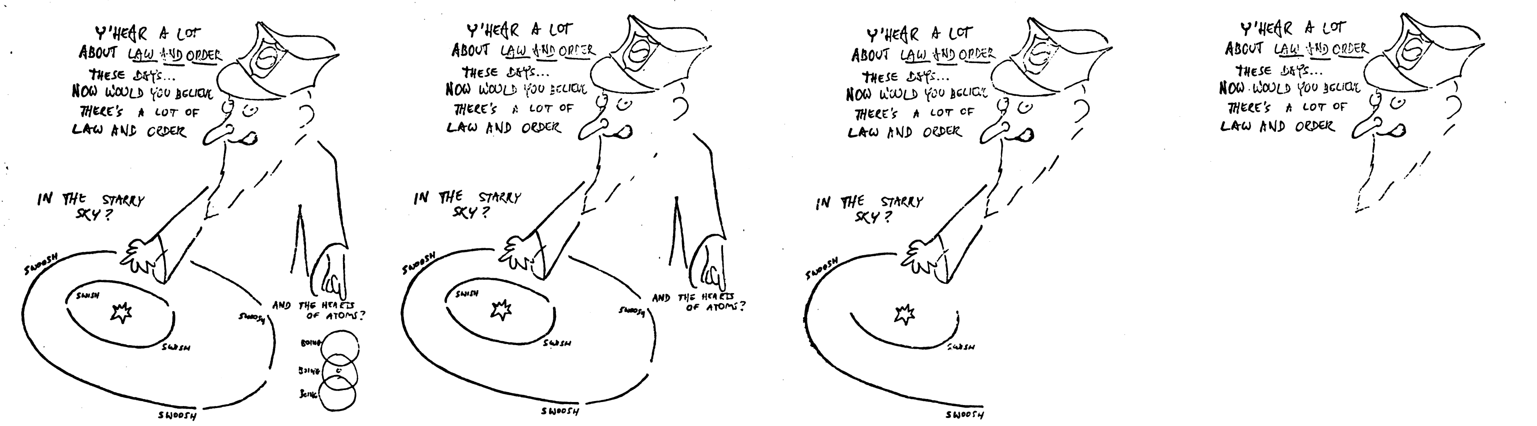
WHAT THIS IS. I briefly visited Alfred Bork's CAI shop at the University of California at Irvine on a consulting basis. Bork is a really swell guy, but he's devoted to Dialogue CAI-- that is, to teaching programs that have pseudo-conversations with the student. (As I've said variously already, the pseudo-conversation parts are not only expensive and difficult, but sometimes irritating and objectionable; and happier, zippier, simpler techniques are available using various techniques of old-fashioned showmanship-- as from movie-making, writing and (here) the comic book.

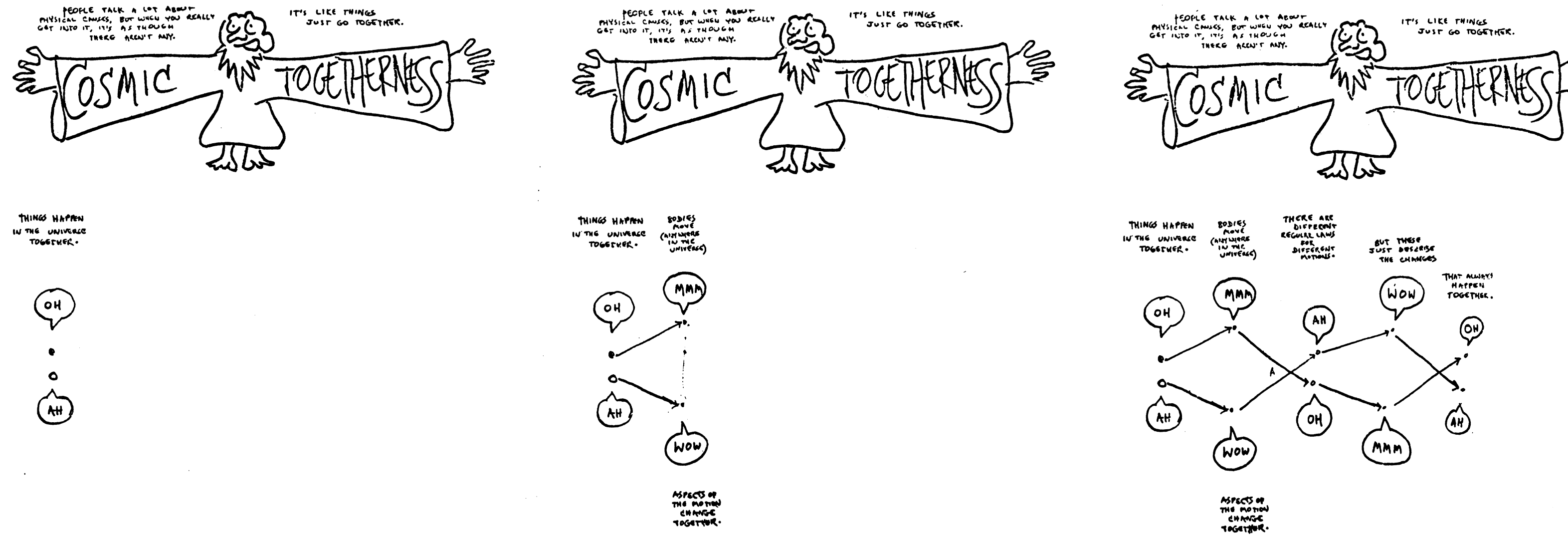
This is my reply to Bork's question, "Well, how would you do it?"

This ties into Bork's physics display system. That is, it's intended to be a front-end program (see p. 13) on a Tektronix graphics terminal (see p. DM7 and DM 20-23), leading into a simulation program (see p. 58) allowing the user to see all kinds of motions in physical law. The program it's intended to supplant uses dialogue.

WHAT IT CONTAINS: introductory remarks; statement that physical law (as of motions) simply summarizes constant covariances. Sorry if readability is poor (Xerox of a Xerox).







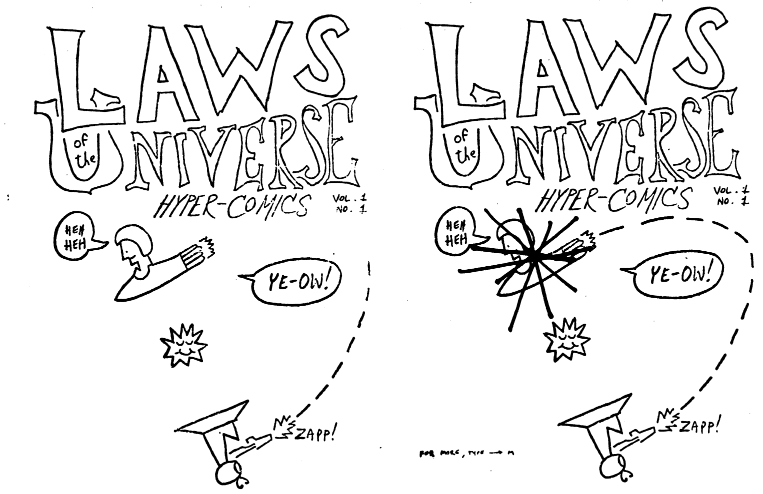
DM29

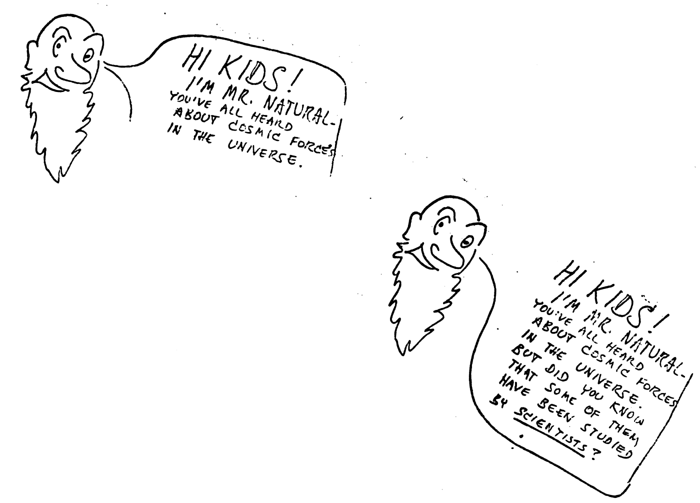
You will note the artistic problem of composing cumulative animation for a display screen.

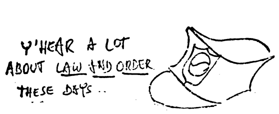
Some people have accused me of trying to be humorous. Obviously nothing of the sort was intended. Research supported by NSF grant no. GJ296 (but "Mr. Natural" character property of Robert Crumb).

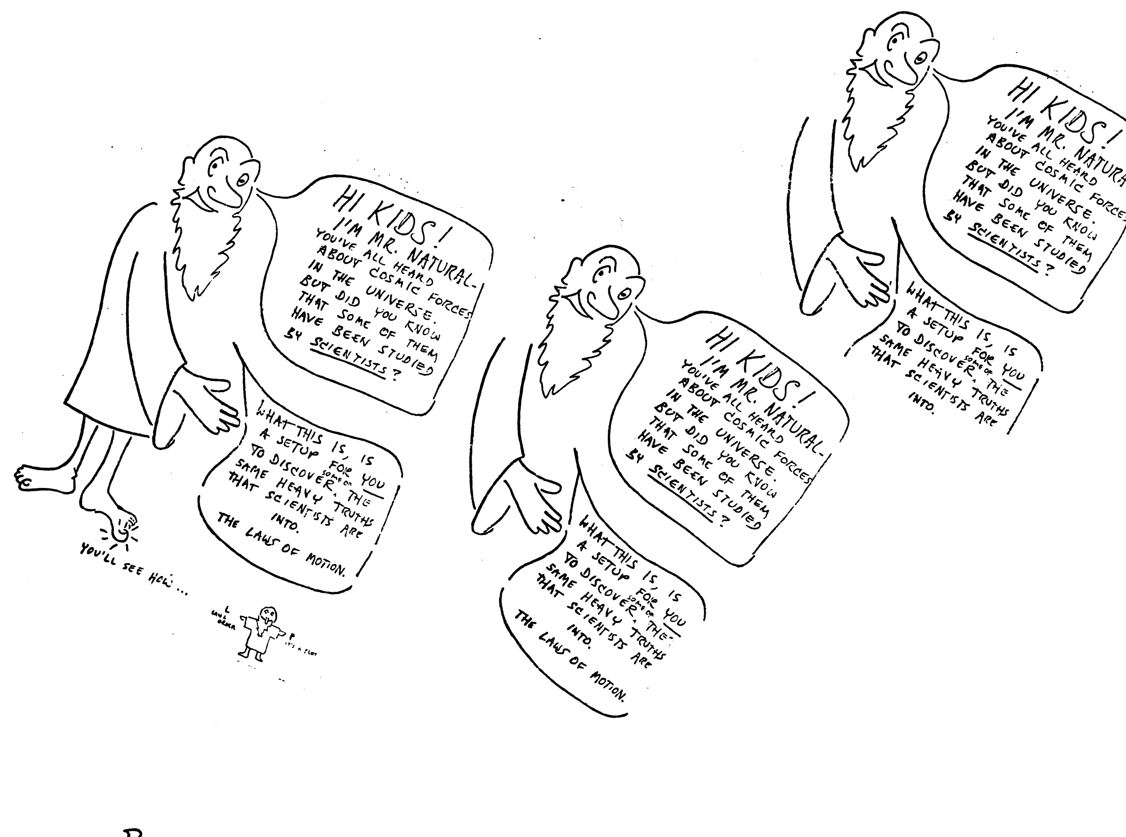
Homage to Robert Crumb.

BIBLIOGRAPHY: for comic technique, study the works of Crumb; also, comicbook stands are currently featuring reprint magazines of THE SPIRIT, which is some of the finest stuff ever done. Also study Wally Wood in the early MADs.



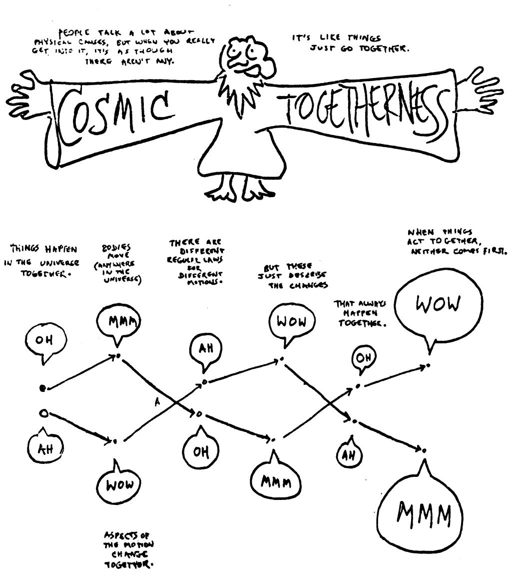






L

P



LET'S BEGIN!

DO YOU WANT TWO STARS WITH YOUR PLANET, OR ONE?

[answer here]

OKAY, HERE GOES —

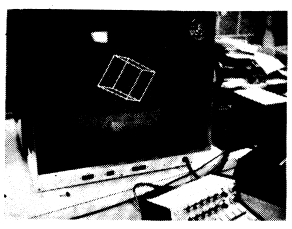
[first plot here]

* THE MIND'S EYE, MORE *

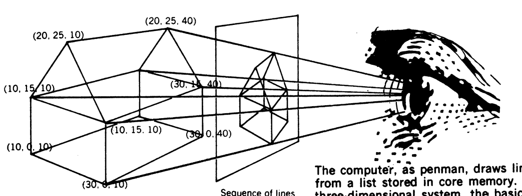
THREE-DIMENSIONAL LINE DISPLAYS

So far we've discussed the two-dimensional subroutining displays. However, things do not by any means stop there. A number of people in the early days experimented with techniques for drawing line pictures by program; the earliest of these used plotters, output devices that let the program draw with a pen. But interest soon grew in the possibility of interactive three-dimensional displays on screens. Johnson's Sketchpad 4 did this entirely by program. But as night follows day, people set about putting these techniques into hardware, creating devices that would automatically show things in three-dimensional views-- allowing the viewer to rotate views of nonexistent objects as if they were on unseen turntables.

The views we are talking about, now, co sist of bright lines on a dark field, and so the "objects" we are talking about are called "wire-frame" objects-- they could effectively be made of welded wire. But now we do not have to build them physically to see them.



Basically a three-dimensional system of this type stores the lines as coordinates in threes: endpoints of lines in a mythical three-dimensional space. Each point's location in the space is told by three numbers (example showing a house may be seen on p.); a line in a space is represented in the data structure by two such points, and a code or something tying them together.



The computer, as penman, draws lines from a list stored in core memory. In a three-dimensional system, the basic list of 3-D coordinates is converted to a list representing a particular view; the result looks like a wire frame.

The second program follower in such a device behaves much as it does in the 2D system, but with certain additions. Like the 2D system, it proceeds down its own program one step at a time. Like the 2D system, it finds in its program the coordinates of a line to display and creates electronic signals representing its endpoints. But it does not display these directly, since these are three-dimensional coordinates. Instead it routes these signals to what we may call a view calculator, a particular piece of hardware that has been primed with the angle from which you want to view the object. This view calculator, automatically and by mysterious means which vary among machines, produces the view, and its signals go to the screen.

Let's say we want to display a point. The display's program follower pulls three numbers from its display list and notes the code that says it's a spatial point and not the end of a line. These three numbers slide on into the view calculator, already primed with the angle of rotation; and the view calculators figgers where on the screen that point should be displayed. The coordinates for the screen-- telling where the point goes in the desired picture-- go to the screen controller, and the point is brightened.

How are these coordinates calculated? Well, some commercial units do it electronically ("in analog") and some do it symbolically ("in digital"). The result is the same.

(If you want the equations for this, they're in the Newman and Sproull book.)

Then how does the view calculator handle a line? Same thing.

The program follower pulls three numbers from its display list and notes the code that says it's a line, so it takes three more. Then the view coordinates of both points are calculated and fed to the screen controller. The screen controller now has two points on its screen-- so it draws a line between them.

The first device of this type was, I think, the so-called Kludge (pron. "Klooj"-- computer slang for a ridiculous machine, but in this case applied affectionately) built at MIT's Electronic Systems Laboratory in the early sixties. This device was a one-of-a-kind, built out of DEC circuit cards and hooking to a bigger machine. The ESL Kludge showed vividly how good it was to have instantaneous view calculation under a user's control.

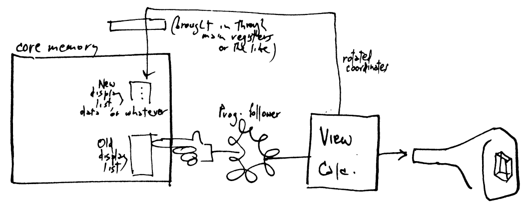
The first of these systems to be offered commercially, I believe, was the "Adage Display," made by Adage, Inc. of Boston, which used their unusual Ambilog computer (see p. 43) to rotate objects on the screen. I vaguely recall that it cost about $80,000 with computer but without accessories.

Actually Adage had a tremendous lead in this field, but they let it slip for some reason, and have now lost it to two firms: Evans and Sutherland on the high end, Vector General on the low end. (But of course things keep changing.)

The Evans and Sutherland Computer Company was founded in 1966 by Ivan Sutherland, creator of the masterful Sketchpad system, and David Evans, chairman of computer science at the University of Utah. (For a time both held appointments at U2 at the same time, but now both have left the university to devote full time to their dream factory in Salt Lake City.)

Their first product was an extraordinary piece of hardware called the LDS-1, which they said innocently stood for Line Drawing System. (To anybody from Utah, however, LDS means Latter-Day-Saint, and don't you forget it. Evans, indeed, is a Mormon, but I've been told it may have been Sutherland's sense of humor that chose the acronym.)

It should be pointed out that a special advantage of digital perspective calculation is that viewed coordinates can be read back by the computer, and serve as new data, if you go for that sort of thing.



The Adage Display is isometric, meaning that lines do not get shorter as they get farther away or longer as they get closer. While this is marvelously impressive, most people want real perspective; and it was this that Evans and Sutherland set about to make available in real time, i.e., in direct response to the viewer's actions.

The LDS-1, weighing in at half a million dollars or so, buckled to the PDP-10, a big 36-bit computer from DEC (see p. 40). Its view calculator worked symbolically (digitally), and thus could work to the higher precision necessary for true perspective calculation.

Among the exciting demonstrations that you can see sitting at an LDS-1 are a map of the United States you can zoom in on, bringing you in to a map of New Jersey, then Atlantic City, then a specific intersection, all in one smooth continuous motion. Also a simulated landing on the flight deck of an aircraft carrier -- with you flying the airplane, so you can go over it, to the side, into the drink or straight at the carrier. In all cases the ghostly ship will move, turn and change perspective on the screen as if somehow it were really there.

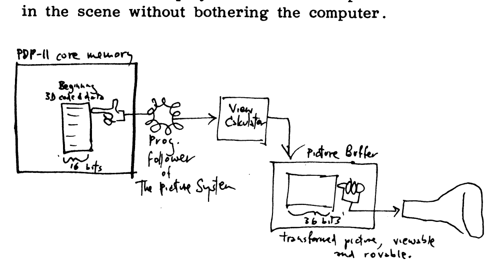
Several LDS-1s were sold.

Meanwhile a little new firm of young guys in Southern California, Vector General, came up with a line of terminals like the Adage line, except that they could buckle to the 16-bit minicomputer of your choice. (In practice most of them have been attached to PDP-11s; see p. 42.)

The Vector General display is isometric, and makes its calculations in analog, like the Adage Display. It has been very successful among both universities and private corporations. In addition, a highly interactive and well-designed language is available for the creation of data structures representing 3D objects, as well as for general-purpose programming and the creation of whole environments. And it's free to individuals or companies that have Vector General displays attached to PDP-11s. (See "Coup de GRASS," p. DM31.)

But wait. Evans and Sutherland has now dropped the LDS-1 and given us-- no, not LDS-2, but something called The Picture System-- also built onto the PDP-11, but this one works symbolically (digitally) and in full perspective. The price starts at eighty grand.

Since the Picture System works out of the PDP-11 core memory, the commands it follows are 16 bits long, since that's the size of a slot in PDP-11 core. But wait. They've designed the thing to convert to 36 bits, so that coordinates are moved to a private store or buffer between the program follower and the display. This means the display can zoom and zip around in the scene without bothering the computer.



Another important feature of The Picture System: it will do, not just ordinary perspective, out such weird view calculations as wideangle barrel distortion, pincushion distortion and similar stuff.

Mr. Rolls, Meet Mr. Royce

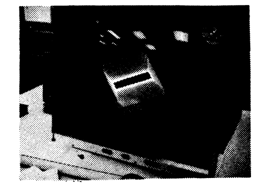
Sutherland   Evans

Courtesy U. of Utah

INTERACTIVE ROTATION

3D screens-- aside from their fun and excitement-- allow people to understand and work with complex 3D structures without having to build them physically.

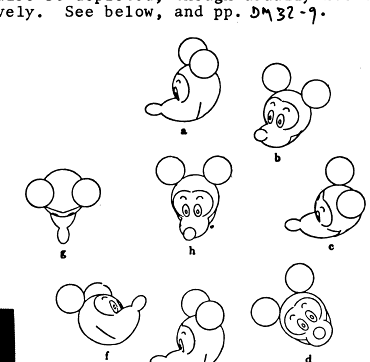
The understanding, however, comes from being able to turn and manipulate the structure on the screen. If you can't turn it you can't really perceive the 3D structure, because the arrangement of lines could be anything.



However, systems like the Adage and the Vector General and the Evans and Sutherland devices allow you to turn things on the screen as easily as if they were on turntables behind a pane of glass. That's how you see, you see.

This interaction is what makes computer display augur a new era for mankind, if we're lucky. (It's also why we use the term computer display in this book, rather than "computer graphics," since people who make computers draw with pens are also doing "computer graphics"-- a related activity, but not one to change the world.)

UNFORTUNATELY, just to get through the basics, there is only room to discuss stick-figure graphic display here. But curved surfaces may also be depicted, though usually not interactively. See below, and pp. DM32-9.



Drawing by Ruth Weiss' BE VISION program, done at Bell Laboratories, mid-sixties. (© Walt Disney Productions.)

This program represented truly curved surfaces in its data structure, as "quadric surfaces"-- that is, involving powers of two in the math-- and calculated the visible lines tangent to the edges from the viewpoint, thus drawing the edges. Removing the hidden parts of the curves is of course one of the greatest problems. (From Ruth A. Weiss, "BE VISION." JACM Apr 66, 194-204, p. 201.)

* THE MIND'S EYE * CONCLUDES ON P. 40.

The rules of perspective have been understood since the Renaissance. In olden computer times (up till about 1965) people used to do three-dimensional view calculation by angles relative to a three-dimensional data structure. Then Larry Roberts at MIT noted that there was a more appropriate mathematical method, long moldering in obscure texts. The idea is this: if you add an extra dimension to the data, it's easier to program. It's easier because it becomes a simple matrix multiplication, which has no commonsense explanation but is important to mathematicians.

SO that means that to calculate views of three-dimensional objects, the most usual way is now to add that extra dimension. Instead of having a point in space whose position is 36-24-36 (in some set of three-dimensional coordinates), another arbitrary number is added to make it, say, 36-24-36-1.

It seems that in the mathematics of multiple dimensions, it comes out simpler that way. Indeed, from a mathematical point of view the new improved dimension is just like the other three. For this reason, such an augmented system of coordinates is called homogeneous coordinates. Like homogenized milk, the additional coordinate is just stirred in with the rest, and out comes your desired view calculation. (The formulas are to be found in Newman and Sproull, Principles of Interactive Computer Graphics, McGraw, $15, your basic text on the subject.)

At any rate the additional coordinate is often referred to, incorrectly, as the "homogeneous coordinate." They're all homogeneous, which is why it works.

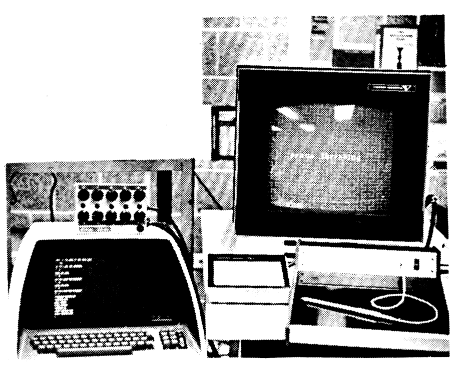
DeFanti*s Coup de GRASS

Impudent and plucky Tom DeFanti was an assistant professor at 24. This in part because he has created one of the world's hottest 3D graphics languages, which he calls GRASS. (He says it stands for GRAphics Symbiosis System-- also, he says, it Turns You On.)

Tom's GRASS language is an excellent beginner's computer language for two reasons: first, it is easily taught to beginners, and second, it is about things of interest to beginners, i.e., pictures and graphical manipulation on screens. (But compare the three beginners' languages presented briefly on pp. 16-25.)

A prototype for the system was developed at Ohio State, on a project directed by artist Charles Csuri. Tom had a free hand, though, and the language design is his; but much of the specific coding was done by Gerry Moersdorf, and the graphics algorithms and rotation were programmed by Manfred Knemeyer. Inspiration was furnished by Maynard E. Sensenbrenner.

GRASS runs on the PDP-11, a splendid minicomputer (Tom's is shown on p. 36) and is specifically designed for the control of three-dimensional stick-figure displays on the Vector General display system (see p. DM 30). But a lot of people have wrestled with these matters and not done as well. Let's consider:



1. ITS CLEAR SIMPLICITY. Tom believes computers are for everybody; he is not a high priest bent on making things obscure (see "Cybercrud," p. 8). Thus he made his language as sensible, clear and easy to learn as possible. Tom likes to stress the concept of "habitability" (a term of W.C.Watt), meaning the coziness of a system.

2. ITS GENERALITY. Refining and condensing the basic ideas of a system is the hardest part of the design. DeFanti made several interesting decisions.

A. The internal form of the language is ASCII code (see p. 26). In other words, you can read programs in their final GRASS form.

B. For a three-dimensional system such as the Vector General, the main form of data structure is the three-dimensional object-- a list of points and lines in space. This is the form of data GRASS uses for most purposes.

C. In the design of such a system you want larger 3D objects to be buildable out of smaller ones. This implies arranging data in tree structures (see p. 24). You also want to be able to make things do compound motions on the screen-- for example, showing an airplane flying around on the screen with its propellor spinning; this too implies a tree structure. There are some programmers who would use different tree structures for both objects grouped together and for movements grouped together; Tom uses one.

D. Objects shown on Tom's system can also appear to move on complicated paths through three-dimensional space. In Tom's system, such a path is merely another object. It seems obvious when you say it, yet this kind of simple generality is exactly what many programmers seem to avoid. (Note: this facility is a generalization of Baecker's p-curve; see p. DM 25).

E. Input devices are completely arbitrary and programmable. What happens on the screen can be controlled by anything-- any variable (see p. 16) in the programming language. In other words, DeFanti has decoupled the screen from any particular form of control, allowing user programs to make the connection between controls and consequences. This means that, using Tom's language, it is comparatively easy to build complex custom controls for any function. (This is discussed under "Fantics,"* p. DM 48-51.)

F. The language has string functions that allow text handling. Since the language may also use conversational terminals, it is eminently suited for "good-guy" interactive systems for naive users, as described on pp. 12-13.

G. Tom's language is interpretive, like TRAC Language (see p. 30). That means it is "slow" in terms of the number of machine cycles required for it to do each operation. However, DeFanti has added a "compile" feature to the language, so that for long macros (sections of program) that have to run repetitively, more efficient compiled versions of the macros may be generated.

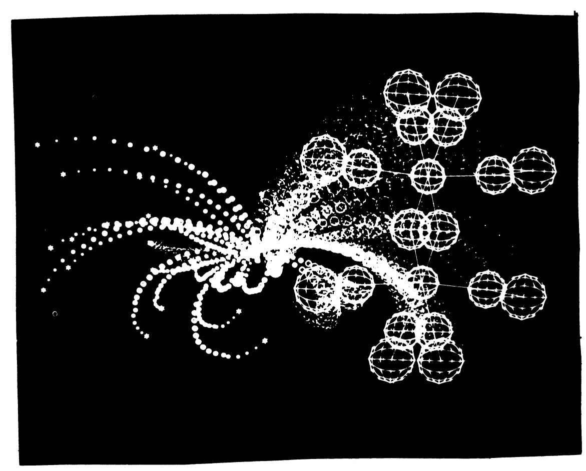
H. The language is extensible, meaning that the user may create new commands in the language as programs. These commands, however, may be used in later programs as if they were built into the language itself.

I. The system is completely general-purpose. Many graphics languages are not, being restricted only to their original purpose. This is more difficult, but oh, so much more worthwhile.

3. ITS DEEP GENERALITY. Things should be versatile, and able to be tied together in many different ways. This is what we mean by "generality;" and this kind of generality can make a system very powerful. (The term in mathematics is "elegance.") As is said on the other side of the book, complicatedness is not generality or goodness or power, but a sign of the designer's shallowness.

Anyway, GRASS has this kind of generality. It has a great number of facilities, growing weekly, and they all tie together in clear and predictable ways, without exceptions. Rather than create special functions which cannot be tied together, Young Doctor DeFanti has chosen instead to make the separate desirable functions part of a simple and clear language. (A note to you elegant types: GRASS is fully recursive. As a nice example, Dan Sandin (see p. DM 8) wrote a program to display Peano lines that was under forty GRASS instructions long. It is also astonishingly reversible: you can watch it uncreate the Peano line, straightening itself backward.)

In the more usual sense, DeFanti's language is not the 'most advanced'; there are more powerful 3D systems than the Vector General (the LDS-1, see p. DM 30, offers true perspective), more elegant user-level languages (see TRAC Language and APL, other side), true halftone (the Watkins Box); yet his achievement on close examination is extraordinary. Never mind his age, the more esoteric features of his system (full recursiveness, etc.) or the fact that he does not seem to have made one mistake, which is infuriating. Consider only this: TOM DEFANTI'S 'GRASS' LANGUAGE IS PERHAPS THE ONLY SYSTEM THAT CAN BE TAUGHT IN A FEW HOURS TO COMPUTER-NAIVE BEGINNERS THAT PERMITS FULL THREE-DIMENSIONAL ANIMATED INTERACTIVE GRAPHICS WITH TREE-STRUCTURED DATA.



Tom DeFanti

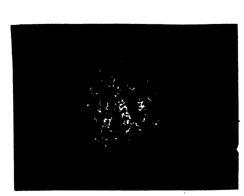
THREE WAYS OF SEEING MOLECULES USING 3D COMPUTER DISPLAY.

Much of today's impetus for 3D computer display is coming from the field of chemistry. University chemistry departments are buying equipment like the Evans & Sutherland LDS-1, the Adage and the Vector General.

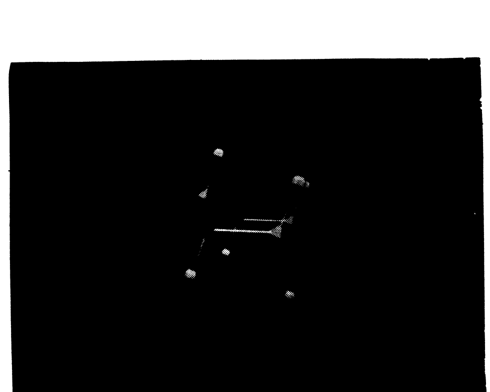
Why?

Because chemistry is increasingly involved with complex three-dimensional structures. Crystals, long folding chain molecules, minuscule forces acting on structures whose shape determines the outcome. Organic molecules that involve thousands of atoms, and whose complex folded structure exposes only certain key features. And so on.

The Vector General display illustrated here and there on these pages belongs to the Department of Chemistry, University of Illinois at Chicago Circle.



Tom DeFanti. Shows part of hemoglobin molecule. Data structure from Richard J. Feldmann, NIH.



Bouknight & Kelley (see p. DM 34)

The best feature of all: it's currently available. PDP-11 owners-- even without Vector General displays-- may inquire of: Tom DeFanti, Doctor of Arts Program, UICC, Chicago IL 60680.

You may wonder how a young bronking buck like DeFanti has managed to do such an excellent job, so elegantly, where so many have stumbled and failed?

"I just learn from other people's mistakes," he says cheerily.

Prof. DeFanti on the system.

MISCELLANY:

Coupling his system with that of Dan Sandin (p. DM 8) has created the "Circle Graphics Habitat," described on p.

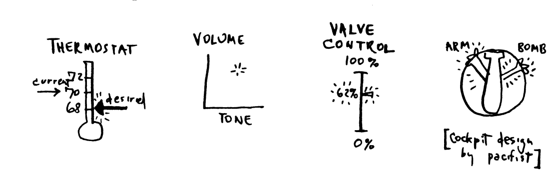
I hope I'm around long enough to write the GRASS language manual.

(DeFanti's GRASS is an ideal language for something like the 3D Thinkertoy, described on p. DM 55. However, it doesn't have any provision for the storage of large complex data structures, so the hard part would actually be working out an adequate storage data structure and storage macros within GRASS's use of the DEC file system.)

SCREEN CONTROLS

The great thing about CRT displays is that they can be used to control things by manipulation of pictures. Instead of moving buttons or levers, you can seize parts of the picture with the light-pen and move some part of the picture. The computer, sensing the choice or adjustment you have made, can then perform whatever operations you have directed.

Some samples:



The design of screen controls-- easy-to-use, clear and simple controls for everything-- is one of the frontiers of computer graphics. (See "Fantics," p. DM 48-51.)

DIMENSIONAL FLIP

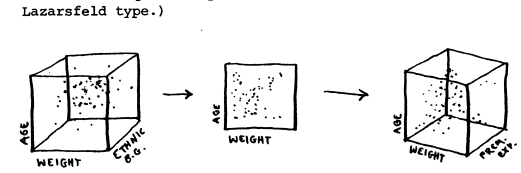
3D scopes are about the best we've got-- so what do we do about multidimensional phenomena?

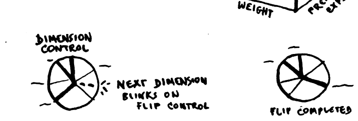
One very good solution is to show a selection of three dimensions at a time, and provide for easy "flip" from one dimension to another-- so that instead of looking at something on demensions A, B and C you are looking at it on dimensions A, B and X.

For example, suppose you're a sociologist looking at measurements of various traits among a group of people. It's a cloud of dots in three dimensions-- whatever three dimensions you're looking at. Some could be: age, height, weight, sex, ethnic background, premarital experience, education... etc.

You view this cloud of dots, say, according to age, weight and ethnic background. That means you can rotate it around and see how many people in the group are what.

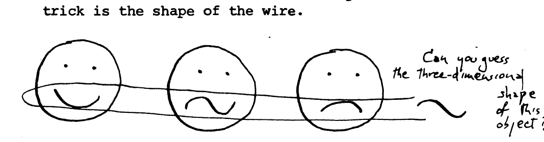
Using dimensional flip, however, you can change the view as follows: rotate the box-frame till it becomes a square to your eye. Then you hit the control that makes the unseen dimension "flip" to another dimension that interests you. The cloud still looks the same-- until you rotate it, and the third dimension is now "premarital experience." So you can quickly get a view of how populations are really divided up. (Note to sociologists: this same operation, with stretching and clipping, provides a visual technique for "partialing" operations of the Lazarsfeld type.)





THE TWISTED SMILE

You can make a character change expression on a 3D scope by making his mouth a twisted wire that can be rotated between "frown" and "smile" positions. The trick is the shape of the wire.



NOW GUESS WHAT: DeFanti's GRASS language is the best language I know of for doing all the above things.

* I coined the term fantics, for the art and technology of showing things, long before I ever heard of Tom DeFanti, and I am not about to change it just because he is now my friend and roommate.

COMPUTER HALFTONE IMAGE SYSTEMS.

A Series of Review Articles for Computer Decisions Magazine.

WHERE TO GET IT.

Computer 3D halftone systems are now available to moviemakers from a variety of sources. It tends to cost a lot of money, but when compared with normal Hollywood production expenses, it turns out not to be so bad.

SALES OF MACHINES.

Computer Image Corporation, Denver, offers various systems for sale. See p. DM 39.

Evans and Sutherland Computer Corporation, Salt Lake City, offers the Watkins Box, a real-time display device using the Watkins Method (see next page) and offering also Gouraud pseudo-curved shading (see p. DM 37). It costs about $500,000 and attaches to a PDP-10 large computer; see p. 40).

SERVICES.

General Electric, Syracuse, offers three-dimensional scene synthesis like that at the bottom of this page. Every job is custom. It's done on videotape through programs running on a smallish computer. Production costs, after your data structures are all in, could run as little as hundreds of dollars per minute (rather than thousands).
Contact: Charles P. Venus, General Electric Co., Building 3, Syracuse NY 13201, 315/456-3552. (Given in detail because harder to reach than these others.)

Computer Visuals, Inc., Elmsford, NY. Offer more detail than GE system, and go straight to film without video. More expensive: probable costs run in the thousands of dollars per minute. Again, every job is custom.
Contact: Nat C. Myers, president.

Dolphin Productions, NYC, has several Computer Image machines, but their president, Allen Stanley, is interested in everything.

Computer Image Corp., Denver and Hollywood, also offers services on their machines. On occasion they have been willing to back film-makers, reportedly on a 50-50 basis. Their president, Lee Harrison III, is a swell fella.

Author's note. These articles were written for Computer Decisions magazine, and reflect the results of a lot of phone calls they paid for. The first of these articles was published in 1971. The others have not been previously published, as the editors and I were never able to get together on quite what they wanted.

This is, to my knowledge, the only existing collection and summary of computer half-tone systems to date, and in some cases the articles reveal more about the systems than has been published anywhere. Surprisingly, even two years later they do not seem out of date.

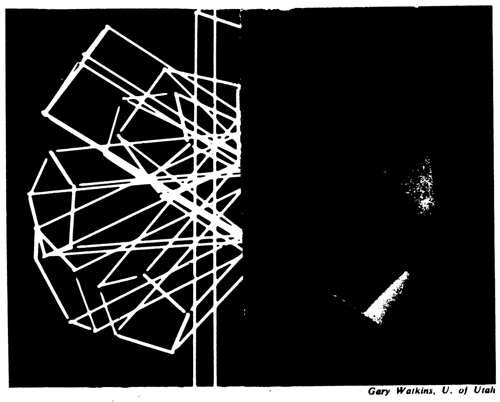
However, due to the editorial style of Computer Decisions, and my own, this has all come out extremely condensed, and phrased in breezy and humorous ways not ordinarily considered acceptable for serious technical reviews. The hope is that they will supply orientation to the browser, deeper insights to the technically-minded, and further directions for them as wants to pursue.

My thanks to the publishers of Computer Decisions and its editor, Robert C. Haavind, for their encouragement, phone money and permission to reprint this.

FIRST ARTICLE

General idea of 3-D halftone.

Polygon Systems.


Gary Watkins, U. of Utah

halftone image synthesis

There are more ways than one to produce shaded pictures with computers. Here are the methods of the 'polygon school.'

by Theodor H. Nelson
The Nelson Organization

To most people in the computer field, "computer graphics" means line drawing—systems and programs for mapmaking, pipe layout, automobile and aircraft design, or any other activity where a diagram may help. Using line-drawing programs and equipment, designers may create line drawings on fast-responding graphic screens, reworking their ideas until satisfied; the system then disgorges polished drawings and specifications for the designer's real intent, something else that is to be made or done. But it is possible for a picture itself—instructive, interesting or pretty—to be the goal. In that case we will often want pictures that look like things instead of wires. A picture that is not all black and white we call "halftone."

With much secrecy and a slow start, computer halftone systems are now being built all over. The methods are extremely different from one another; only the outputs are similar. Some exist in software, some have already been built into special hardware. These systems have many potential uses for visualization, animation and new kinds of photography, in art, scholarship, motion pictures and TV; for visualizing worlds lost and imagined, equipment yet unbuilt, the responsiveness of aircraft. It may not be long until moviemakers can buy different brands of picture synthesizer, just as musicians choose today among Moog, Buchla and ARP music synthesizers. But none is in production yet. This is an attempt to review the coming apparatuses of apparition.

Not only is the field of halftone one of the most exciting in computing; it is also one of the nuttiest and most secretive. For instance, at one time a firm that was supposedly marketing its halftone system declared the present author *persona non grata* and not to be communicated with in any way, though information was freely available to others. "I don't think it's necessarily paranoia," says Rod Rougelot of General Electric. "A lot of guys started about the same time, and proceeded in a heads-down manner." It took a special kind of initiative to head off in that direction with no external provocation. "All those heavy cats from ARPA and MIT were saying in the sixties I could never do a Mickey Mouse," says Lee Harrison III of Computer Image. "But I'm not that kind of researcher. I talk to the Lord."

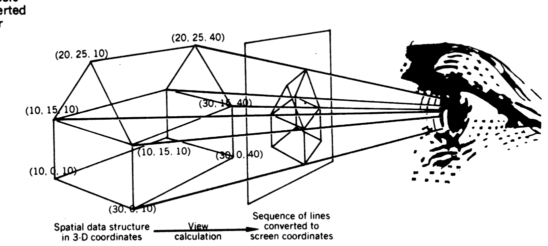
The systems' stories are as different as the systems themselves. General Electric's system grew out of cockpit displays for blind flying. The system of Pennsylvania Research Associates began with terrain and radar modelling. The system of MAGI (Mathematical Applications Group, Inc.) began with the study of radiation hazards in battlefield machinery. Two system families, that of Computer Image Inc. and my own Fantasm, were designed from the beginning for moviemaking, especially "special effects" and puppeteering. The most poignant tale may be that of Lee Harrison, whose struggling family was warmed through cold winters by the tubes of their analog computer.

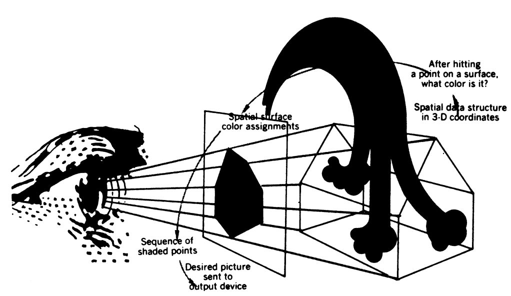
Halftones in two dimensions

Two-dimensional computer halftone is not new. Halftone pictures converted from photographs have often been printed out on line printers, either for fun (nudes often turn up at big installations), or in connection with some scientific problem, such as analyzing chromosomes. Kenneth C. Knowlton, at Bell Laboratories, has executed some well-known photo conversions making pictures into huge grids of tiny whimsical symbols having different grey-values.

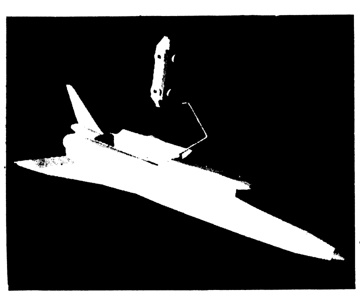
Various other systems have allowed users to create their own original 2-D pictures. But the natural temptation is to want the computer really to make pictures. Why not have the computer produce a photographic picture directly from the 3-D representation of objects? Computers don't do this by nature, any more than they do anything else by nature, so *how* it may be done by computer is very interesting. The problem is also interesting because of its intuitive nature. Visions of scenes in space are around us constantly, and we intuitively understand the geometry of outlines and light. As 3-D work progresses large problems are being overcome. The famed "hidden line problem," for example, was misleadingly couched, since the problem is not finding what lines are hidden, but what surfaces are in front!

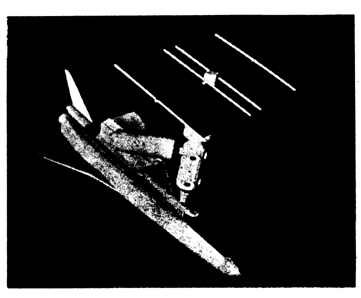
Computer graphics the ordinary way
The computer, as penman, draws lines from a list stored in core memory. In a three-dimensional system, the basic list of 3-D coordinates is converted to a list representing a particular view; the result looks like a wire frame.

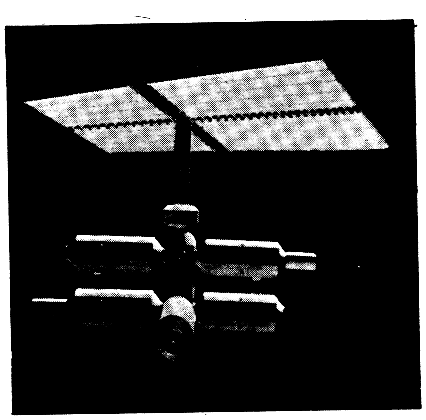


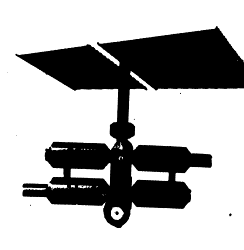


3-D halftone system
Today's new procedures can use the same data to make a realistic shaded or halftone picture. The visible parts of the objects are ascertained by programs or special hardware, using the same 3-D coordinates as in the ordinary systems. These visible parts are then shaded according to the appropriate color information. The series of shading-points makes the picture on an output device.









General Electric will make movies and videotapes for you with their pictorial synthesis system. These are from a beautiful (really beautiful) film they did for NASA. The point of the film was to explain to everybody how a proposed space laboratory would be built and would function. Rather than use diagrams, they enacted it in the GE system, so viewers could understand how the sections would be delivered and fit together, how the antennas would unfold and so on. For exposition of that kind, nothing beats this kind of enactment.

DM33

We must draw on this understanding of scenes to figure out how to make pictures, for there is no mathematically elegant or preferable approach. Scenes are geometrically rich, and thus many different techniques may be used to extract pictures from them. These techniques may look at planar structures, spatial interconnections, relative edges of intersections or anything else you can define and process. I prefer to think of computer halftone as like trick photography of the kind done in Hollywood: a variety of techniques can be combined in various ways. As in trick photography, the number of touches and enhancements that you add generally determines how good it will look, regardless of what system you begin with.

The simplest systems are those that depict objects made of polygons—that is, planes with straight edges. We will discuss such systems in the present installment.

The wild polygon yonder

At least two companies are building image systems that will behave and respond like onrushing reality. Such a system, hooked to cockpit-like controls, can show a trainee pilot the delicate and precipitous results of what he does. Realistic action, rather than surface detail, is crucial.

The techniques of action polygon halftone were originally developed by General Electric, of Syracuse, N.Y., and are now also under development at Link Division of Singer Company (makers of the beloved pilot trainer and its progeny). Basically such systems operate upon the scan-lines that crisscross a television screen, switching the color of the running scan as it crosses from polygon to polygon.

The action polygon school—GE and Link—takes a curious but effective approach to halftone TV: their "environments" are composed entirely of convex objects made entirely of convex polygons. To use only convex objects (no dents) means that one object may be in front of another or vice versa, but never both. (An object with apparent indentations, such as an airplane, has to be made out of a group of convex objects flying together.) To use only convex polygons (notchless) makes it easy for the system to decide, at a given instant, whether the scan is crossing the polygon or not.

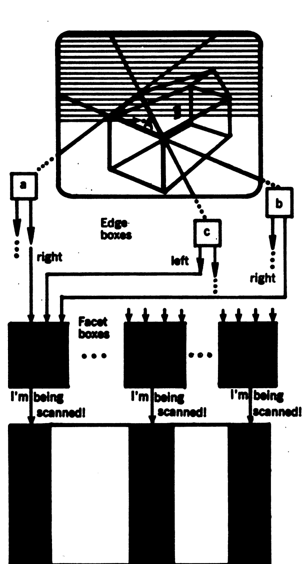
Instantaneous enactment: halftone animation gives a sense of really being there. (Rod Rougelot, General Electric)

This work evolved in part from GE's work in the fifties with a "ground plane simulator," a system that would show a correct representation of the ground's position, dipping and rotating, to the pilot of an aircraft in fog or night. In 1963 the General Electric group, under Rod Rougelot, worked out for NASA the design of an "environment simulator"—a device that would simulate the appearance and performance of any equipment. This is now called the "old NASA system." It permitted the user—seated before a color TV screen—to work controls for an imaginary aircraft or spacecraft, and see roughly what the pilot of the craft would see, flying in real time through a breathtaking color scene. Films made on this machine have been stunning. Imaginary cities, roller coasters and aerial dogfights are among the visions that can be presented.

General Electric's old NASA method is fairly weird if not mischievous. The earlier "ground plane simulator" had shown an edge (the horizon) digitally displayed on a crt; the system was extended to many edges, and the logical analysis of areas between them.

The scene was represented by a collection of edge boxes, physically jumpered into a collection of facet boxes. Each edge box and facet box was loaded with certain numerical and logic values, representing edges and facets in the scene, which could change between frames as required by the action.

In the preprocess for each frame the old NASA system used a specially built digital computer, the "vector calculator." This performed at great speed the three-part vector calculations necessary to determine all scene positions, including the positions and slants of all edges. Each individual edge generator, loaded with its own edge position, constantly reported whether the running scan of the picture was to the left or right of its own edge. It dutifully guarded this edge from border to border of the picture.

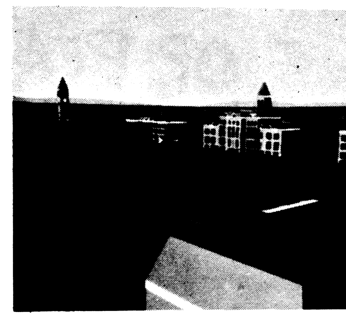


"Old NASA" method: Each edge box constantly reports which side of its edge the scan is on; each facet box sums the edge reports to sense when the scan is crossing it.

The edge-box reports summed into the facet boxes, each of which was set to respond to a particular combination of left-right, above-below reports. At the instant all the facet's edge boxes replied in the proper preset combination, the facet box signalled that its own facet was being crossed by the scan-line. When more than one facet-box responded, the one nearest the viewpoint had its color gated to the screen.

Now Rougelot's group is replacing the old NASA system by a new NASA system, which works on entirely different principles, but keeps the vector calculator. The old one could show scenes with up to 240 edges; the new NASA system will at least double that. GE's new method is already operational on smaller research facilities. They don't tell what it is, but basically it involves sorting by distance. Supposedly the sort method is good enough to make the old edge boxes obsolete.

The Link group claims competitive performance for their system, which will go to black-and-white thousand-line TV. They say their system is different, better, and secret.

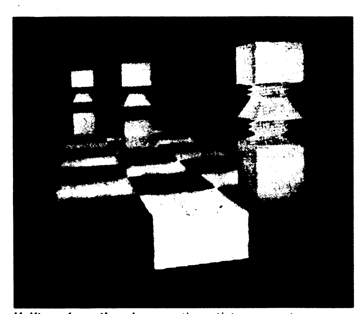


Campus of Fooled U. (GE)

Wylie-Romney: shoot the works

The Wylie-Romney method, disclosed in 1967, was the first generally publicized procedure for making halftone pictures. Indeed, the 1967 publication signalled the explosion of the University of Utah into the forefront of computing research.

The Wylie-Romney method was actually the joint work of Chris Wylie, Gordon Romney, David C. Evans and Alan Erdahl; but much of the impetus for its development came from Evans, chairman of computer sciences at Utah, who had long suspected the possibility of 3-D halftone synthesis.



Halftone for art's sake: now the artist can create worlds and photograph them. (Gordon Romney, Utah)

(Note: more output by various Utah systems appear on following pages.)

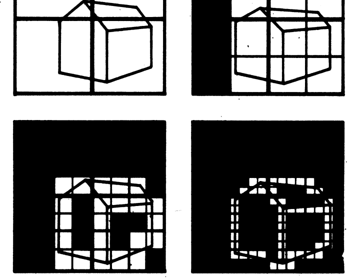
The Wylie-Romney method is this: for each picture-point desired in the final picture, shoot a searching ray through the scene at a corresponding angle. Find where this searching ray hits every surface in its way.

Since the locations in space of these hit-points are easily calculated, figure their distances from the vantage point. The nearest of the intersections is the visible one. Look up the color of that surface and shade the output point accordingly.

This may sound inefficient, but it is comparatively easy to ascertain all the piercing-points, since the surfaces to be hit in a given scanning row can be largely predicted from the previous row.

John Warnock's method, also from Utah, is unrelated to the other methods, but has qualities mathematicians like, as well as a certain whimsy.

Consider a square in the picture area. (At the start consider the whole picture area.) *Now then.* Test whether the present square is entirely filled with one color. If so, output a corresponding square all of that color. If the present square is not all one color, divide it into four smaller squares. Take another square and go back to *Now then.* End the process when each of the squares in the broken-down picture has been completely filled with one color—or the unsatisfied squares are too small to care about.

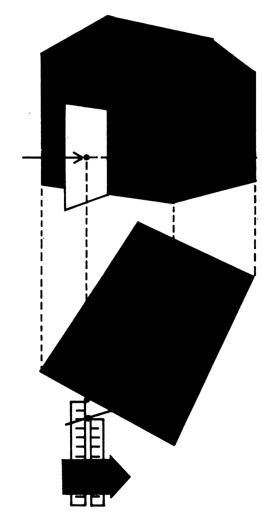


Warnock's dicing method: What can't be made all one color is redivided till its pieces can be.

Movie sets, TV effects: computer halftone is ready to compete. (Gary Watkins, Utah)

The method of Gary Watkins is the result of a profound search at the University of Utah for *the* method—a polygon technique fast enough for real-time enactment, but cheaper than the GE-type systems and not subject to the convexity restrictions. They seem to have found it.

Each video scan of the scene results in a "slice" through surfaces in the scene. The two nearest surfaces are continuously compared to see which is closer, as if by two rulers. The instant a new surface becomes the nearer one, the system makes it the visible one. The nearest surface always shows, down to the precise instant two surfaces cross.



Watkins method: A new nearest surface is instantly sensed through continuous comparison of the closest two.

NOW AVAILABLE! Machine running Watkins technique, the Watkins Box, allows you to view imaginary objects in color and manipulate them in real time. See top of preceding page.

Shading: Last of the great fudge-functions

Suppose that we have some data structure representing a three-dimensional object, and a halftone method to search out its visible surfaces. How do we shade the output points? What do we take into account: how combine the basic greys or colors, how blend them with computations of surface angle, distances from the vantage point, or anything else we can think of?

The answer: any way at all. The combining function is an aesthetic choice. There are not many areas left where you can make up a mathematical hodge-podge and get pleasing or interesting results. Computer halftone is a felicitous exception: you can augment by adding or multiplying, diminish by subtracting or dividing, and yet always come up with an image resembling something. Anyone who has worked in a darkroom will recognize that this is like enlarging: playing with parameters won't obliterate the picture.

There are purists who insist that halftone coloration should exactly follow the formulas that simulate the behavior of real light. For some purposes, like pilot training, this may often be true. But insisting on mathematical accuracy as a general principle is like insisting on ultra-high fidelity—an aesthetic judgment couched as a mechanical imperative.

Until now the output hardware was not really ready for halftone. Five years ago a computer could usually create halftone pictures only on a line printer or a 4020 microfilm plotter. Today there are many different photographic printers, going to all sizes of film and paper; one even uses a laser. There are various display terminals permitting grey-scale and color halftone on TV screens.

The age of computer image synthesis has begun. Polygon systems are fast and simple, and will come to be used in our daily lives for such diverse purposes as molecule study, the memorization of delivery routes, and visualization of every kind of layout and design. They will be fundamental to our new world of computer display. □

DM34

SECOND ARTICLE.

Surface patterns.

Curvature.

Shadow.

SHADES OF REALITY

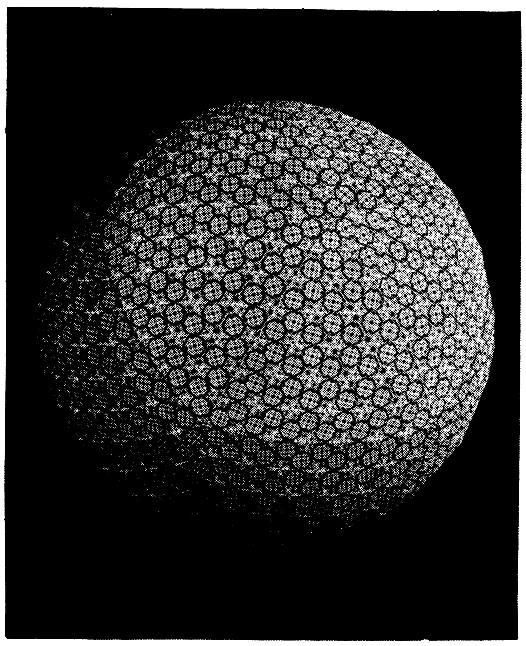
VARIOUS NEW TECHNIQUES PERMIT US TO ADD CURVES, SHADOWS AND SURFACE PATTERNS TO COMPUTER-GENERATED HALFTONE PICTURES

THE PLOT SO FAR.

Various computer methods now make it possible to create artificial photographs of three-dimensional objects or scenes represented in the computer's storage. This is done by coloring or shading points in an output picture like the points in the scene that can be sighted through them from the vantage point. What the methods really boil down to, though, are searching processes in the data representation of the three-dimensional scene.

In an earlier article we have considered some of the techniques being used to depict simple scenes-- those made up of polygons. Now we turn to more elaborate scenes which add shadows, surface patterns and curvature.

One of the most interesting things about this branch of computer graphics-- already seen in the polygon methods discussed earlier-- is the variety of techniques that can be employed. Moreover, these methods, for all their sophistication, can usually be intuitively understood as thought they were operations performed on objects in space. The same continues to be true for the more complex systems.



MAGNUSKI'S PATTERNED CONSTRUCTIONS

A number of contributions have been made by individuals working alone. For instance, Henry Magnuski, at M.I.T., created a program that repeatedly positions patterned facets in space to make large constructions.

This program did not calculate "true" shadow, basing its shading partly on angle of surfaces. Neither does it show true curves. Yet it shows the impressive degree to which such effects may be approximated. The resulting beach ball picture is reminiscent of Moorish architecture.

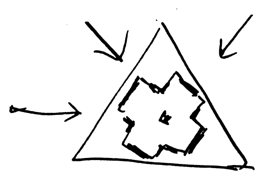
ENHANCED POLYGON SYSTEMS

In the methods discussed so far, we looked at several computer techniques for photographically depicting scenes and objects made up of polygons-- planar facets-- in a represented three-dimensional scene. Imaginary houses of cards, cardboard airplanes and triangular scenery take on a compelling vividness when depicted by the computer. And for visualizing such things as architectural arrangements, such systems promise to be of increasing practical value.

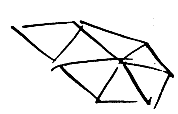
Those of us interested in the artistic aspects of computer halftone images want more. This article looks at some ways to add the appearance of curvature and surface pattern to computer-synthesized images.

MAGNUSKI'S CONSTRUCTIONS OF REPEATED PATTERNS

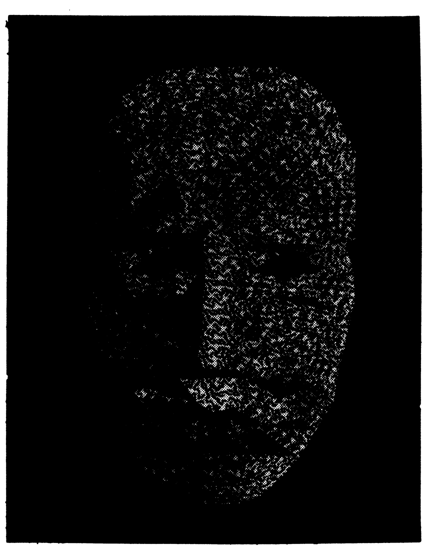
(different perspective calculations)



Basic triangle pattern...



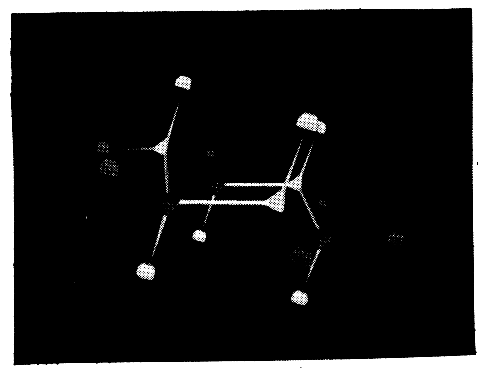
is stitched together in adjacent positions at appropriate angles.



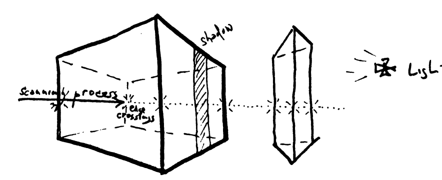
BOUKNIGHT AND KELLEY: PICKING THROUGH A CAT'S CRADLE

The method of Bouknight and Kelley, at the University of Illinois, permits the addition of shadow to polygon pictures. Their method uses an intricate system of scanning sweeps across the scene, analyzing the successive edge-crossings. For each output line, a list of the edges in the scene is ordered according to which will be next encountered. To make a specific output line of shaded points, we step through successive positions of the scan-line, until an an edge is crossed. With each edge we cross, we enter or leave at least one facet. Of all the current facets we are in after a given edge-crossing, the system finds out the nearest one, the visible one, by comparing distances. The coloration of this facet is then fed out to the picture, until the next edge-crossing.

Bouknight and Kelley expand their method to show shadows by an additional step. They create a new list of edges to be encountered, this one relative to scans from the light source. Then, during the regular output picture scan, they look to this latter data to see about shadow. As soon as they know two consecutive edges of a visible object in the picture, they are able to search the shadow-edge list to see if any shadow-edges impinge between them. The final list of edges-- visible facet edges and shadow edges-- goes to the picture output device.

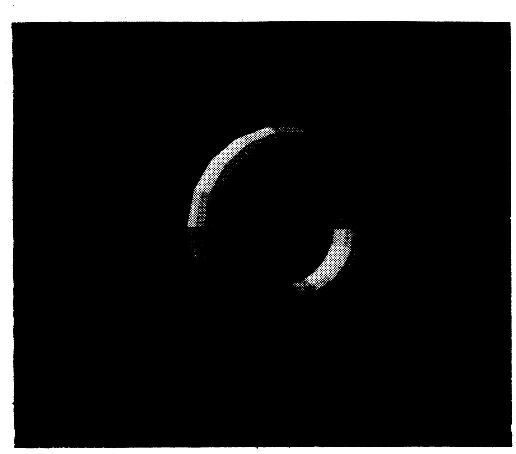


BOUKNIGHT-KELLEY METHOD



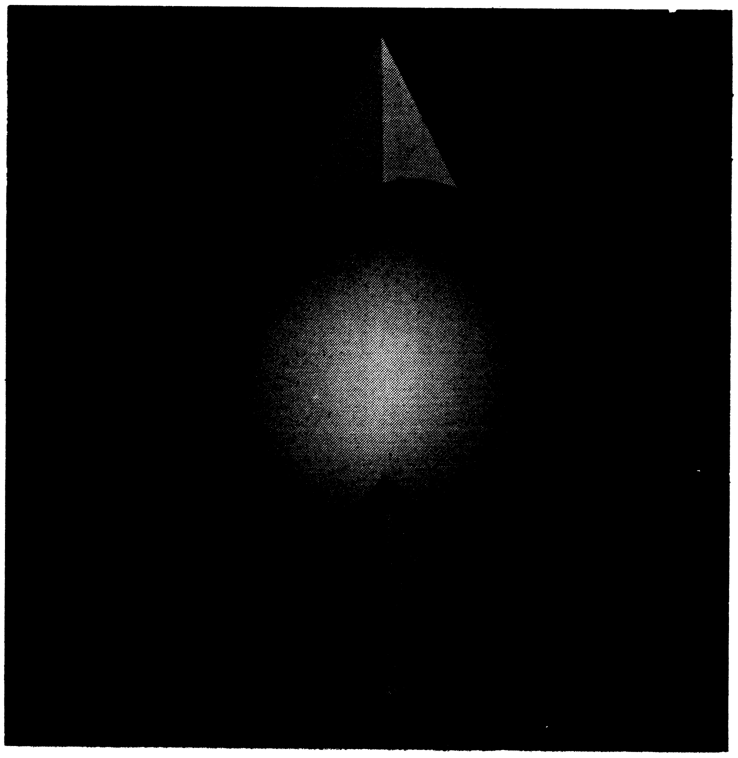
Consider the series of edges whose projections cross the current scan-line. Each time the scan-line crosses an edge, find out what facets are currently pierced by a sight-line from the viewpoint. The nearest of these facets is the visible one.

To add shadow, use an extra list of the scene's edges relative to the light rather than the camera. Between viewed edges, check for shadow-edges as well.

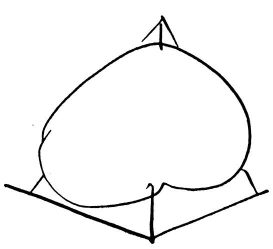


DON LEE FILLS IN THE GAPS

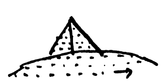
Don Lee, at the University of Illinois, produced his fine-toned pictures of spheres in 1966 simply because someone bet him a quarter he couldn't program the method he'd suggested in twenty-four hours. He almost made it. He made his pictures of spheres and polygons by calculating the boundaries, then checking for overlap and filling in with greys according to viewing angle. His program works only in special cases, but is interesting for its historical position; it was one of the earliest half-tone curvature systems.



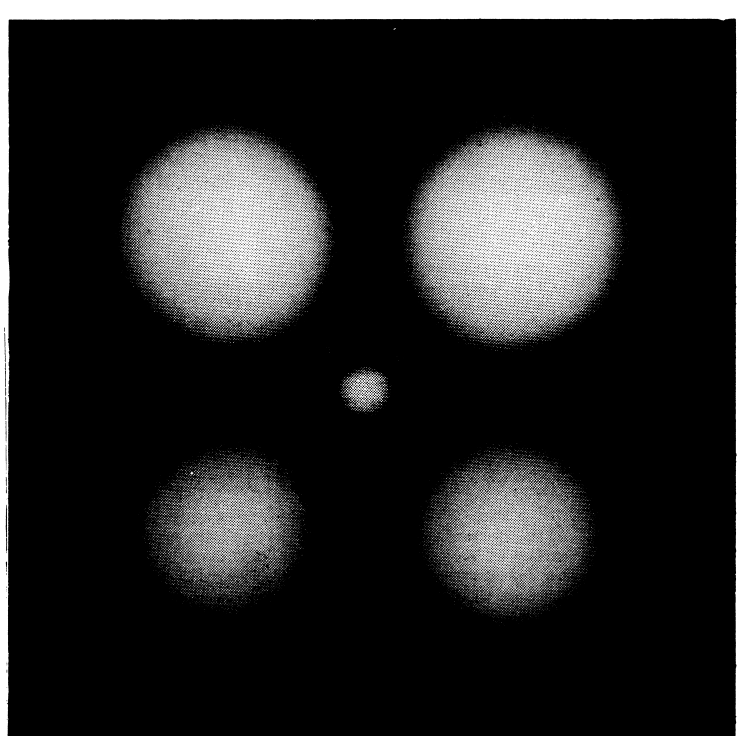
HAVE A BALL WITH DON LEE.

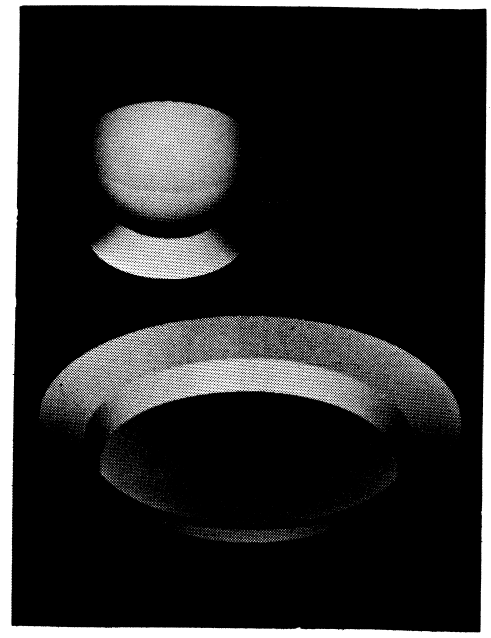


His program first works out the general outlines.



Then fills in curvaceous shading.





SIMPLEX CURVATURE SYSTEMS: MAHL & MAGI

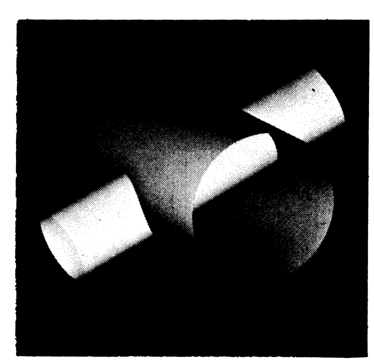
A fundamental type of system we may call the "simplex" system was exemplified in the previous article by the Wylie-Romney program. A simplex technique simply projects simulated rays toward the scene from the vantage point till they hit the represented objects, and fills corresponding positions on the output picture with the colors encountered on the front surfaces of objects in the scene.

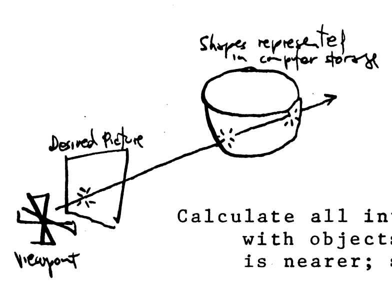
The same principle extends naturally to scenes with curved and otherwise embellished objects.

Robert Mahl, at the University of Utah, has recently reported his results with simplex methods using quadric surfaces-- those curved surfaces generated by mathematical powers of two. His pictures-- like the cup and saucer shown here-- have a pleasing 1920s Bauhaus-like quality.

One problem with this method is that computational complexity increases rapidly as the scenes grow more complex; the more surfaces and piercing-points, the more time-consuming (and expensive) it becomes to make the picture.

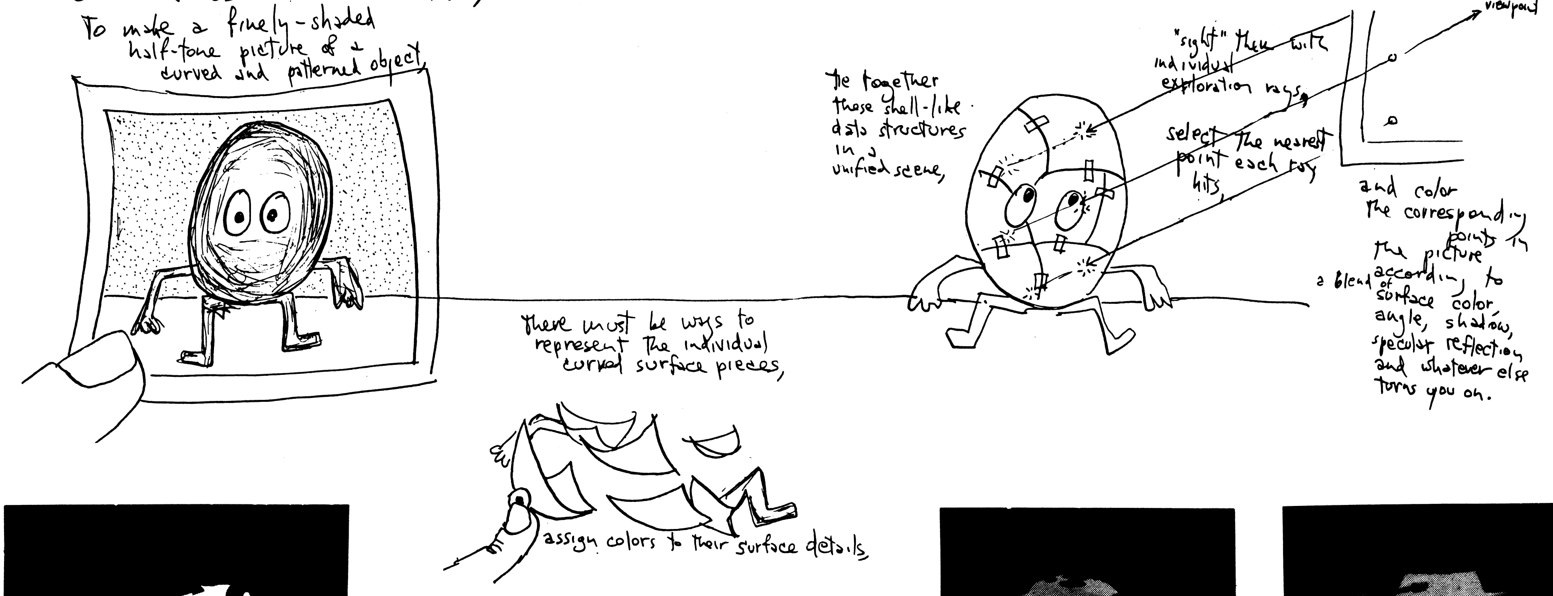
MAHL'S SIMPLEX METHOD

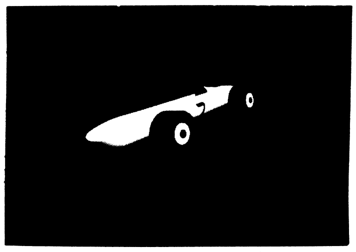


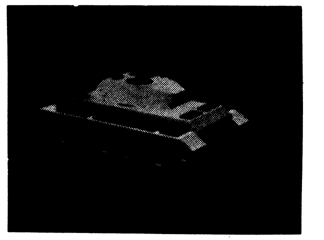


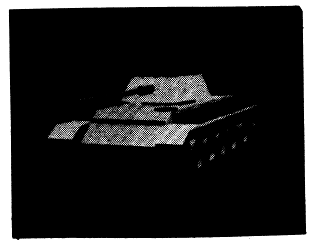
Calculate all intersections of sighting ray with objects in scene; calculate which is nearer; shade it according to angle.

GENERALIZED SIMPLEX METHOD, AS EXEMPLIFIED BY MAGI SYSTEM

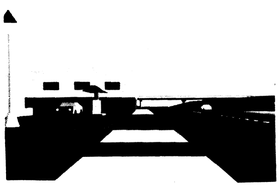


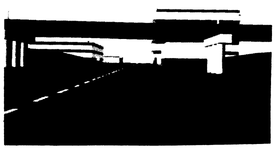


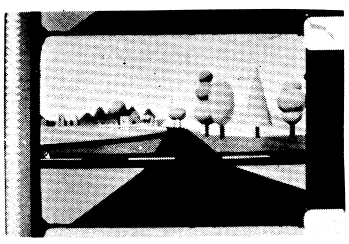


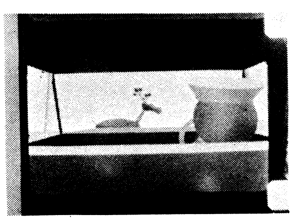


MAGI program was originally developed for study of radiation hazards inside military armor; the pseudo-photographic techniques were a side effect of the approach chosen. Who knows, these tanks may be the ones studied.







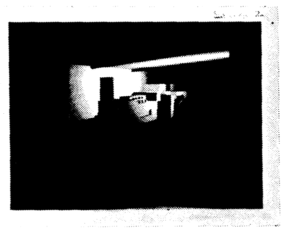


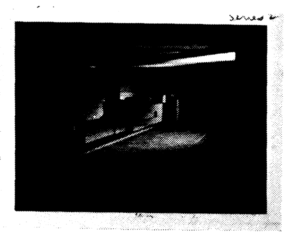
It seems, however, that Mahl's work may only be a rediscovery of what one organization worked out earlier and is being secretive about. A firm delightfully called MAGI (Mathematical Applications Group, Inc.) of Elmsford, N.Y., has extended the same idea more elaborately. They happened into the halftone game through a military contract.

MAGI's system, now thoroughly developed under Robert Goldstein, began in 1965 in a study of radiation hazards in battlefield equipment. They wrote a program to simulate paths of radiation, say, that might reach a tank driver under varous disagreeable circumstances. Having written a program that would ascertain the susceptibility to radiation of battlefield machinery, they noted that the same program could be adapted to making photographs. The progam simulated radiation; light is radiation; ipso facto, pictures. Substantially the same program would make photograph-like images, by treating the objects as opaque, and reflecting different shades according to color and angle of view.

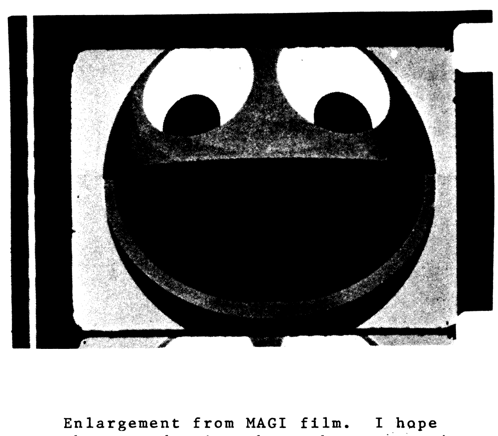
The resulting system makes nice pictures of objects composed of planes and quadric surfaces; and includes, as will be seen from the racing car and chair, colored surface designs, shadows and spectral reflections. Not only does MAGI's software for this process produce delicately shaded pictures; if the virtual picture-plane is moved until it intersects the subject, it produces a cross-section.

MAGI runs this program remotely in Fortran on a big computer-- but they have their own minicomputer setup for photographing the results as color movies. They now offer use of this system commercially for making movies or stills.

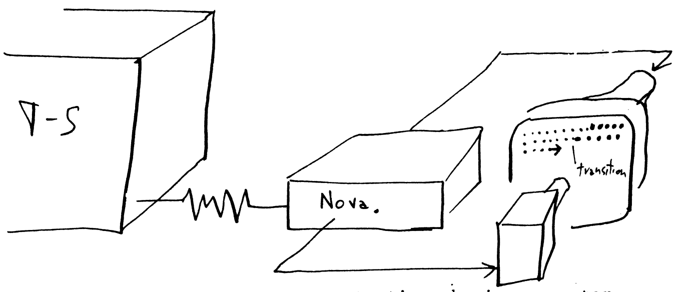




MAGI techniques were used to study alternative ways of lighting mines.



Enlargement from MAGI film. I hope the reproduction shows the concentric rings, called Mach bands, that divide areas of shading; Knowlton and Harmon (citation p. DM 10) advise on pseudo-random techniques for correcting this.



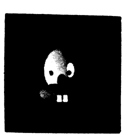
SYNTHEVISION SETUP uses remote time-sharing computer, running big secret Fortran program and containing entire data structure of three-dimensional scenes. Minicomputer photographic setup is on premises at Computer Visuals, Inc., MAGI subsidiary marketing the Synthevision service.

Local setup uses Nova minicomputer controlling both CRT display and camera. Informed guess would suggest that time-sharing system does not send all successive points of output line, but difference and transition values; Nova program would then interpolate gradations in relatively quiet sections of the scan-line.

MAGI's precise system is secret. However, the only real questions boil down to: forms of surface representation; systems of scene sorting; and method of scene scanning to produce output scan.

Note that one of the most impressive things about MAGI work, at least for sophisticates, is the degree of artistic control that seems to have been realized in their input and revision systems. It seems they offer excellent control over motion and color, and, of course, revision of the action in a scene till the maker is satisfied.

Popular Science, I think it was, had a spread on Synthevision in fall of 73.



An early MAGI character.

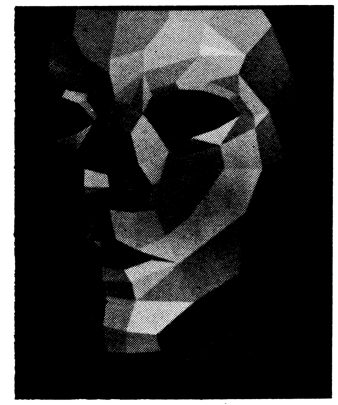
ROUNDUP

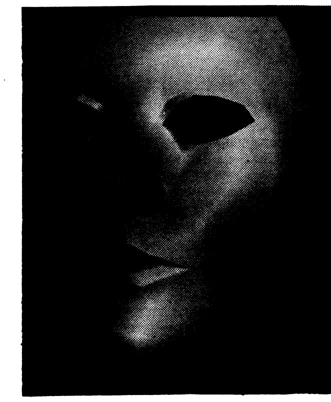
These have been some of the highlights of the halftone game to date. The methods described so far are mainly software-oriented, and for the most part work most efficiently as programs. In the next article we will look at some outlandish new forms of equipment, under construction or proposed, for dedicated production of 3-D halftone pictures.

THIRD ARTICLE. Specialized hardware systems.

SPECIAL EQUIPMENT IS NOW BEING BUILT FOR MAKING "REALISTIC" HALFTONE PICTURES BY COMPUTER. THIS ARTICLE COVERS SOME OF THE MORE UNUSUAL HALFTONE HARDWARE SYSTEMS NOW IN EXISTENCE OR BEING PLANNED.

HARDENING OF THE ARTISTRIES



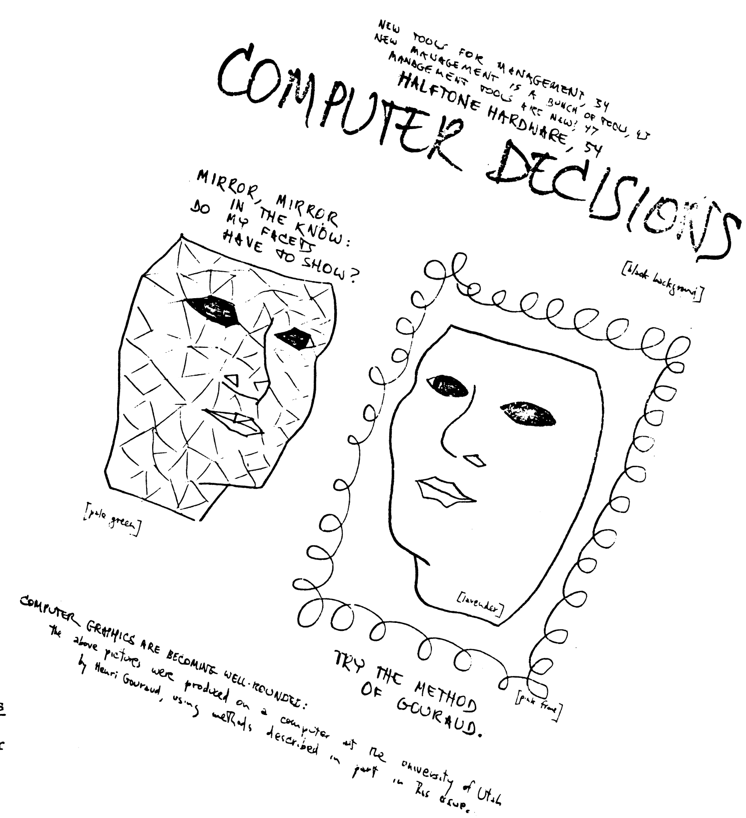


Results of Gouraud's swell smoothing technique. Mme. Gouraud posed for the data structure on the left, a system of interconnected flat polygons. The Gouraud process (see box below) created the smooth-looking face from it by an extremely simple process. (Note that the power of the technique is in the use of a simple polygon data structure, rather than the more difficult truly-curved surfaces used, e.g., by MAGI.) (Note also that the edges remain jagged.)

HARD TIMES A'COMIN'.

In two previous articles we have summarized some of the important basic techniques in computer halftone-- the artificial construction by computer of photographic pictures of 3-D scenes, scenes which are represented within the computer as colored or shaded surfaces placed in a coordinate system of three dimensions.

The techniques we have looked at were all intuitively "spatial" in character, having to do with the analysis of sight-lines and relative edge positions, and suited to implementation in computer software. Now we turn to some more advanced and peculiar techniques and equipment intended to make 3-D computer halftone faster to use, or more realistic, or easier to work with, or cheaper. These systems represent a coming generation of halftone hardware.



I suggested this cover for this article. The folks at Computer Decisions reacted with puzzlement if not dismay. "This cover doesn't have practical applications for the average user," I think someone said.

THE WATKINS BOX

The University of Utah is now building what wil be for some time the world's most spectacular interactive computer display, the Watkins Box. This device, interfacing between a computer and a television screen, will carry out the Watkins algorithm (described in the first article of this series) in real time: ripping through a predigested list of facet information, the Watkins Box will create on the screen an image of an opaque object which the user can rotate or see manipulated by program.

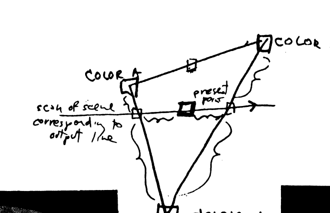
The Watkins Box can operate in two modes: normal mode, in which the object appears faceted, and Gouraud mode, in which it appears to be curved over (see masks, nearby).

The Gouraud algorithm, developed by a graduate student of that name, is a ridiculously simple technique which marries perfectly to the Watkins method. Instead of shading the facets uniformly, this technique calculates a shade of gray for each point. In effect the method interpolates the shade of the point from those around it, across facet boundaries. In actual procedure, the Gouraud method shades a point by linear interpolation between two edge-colors: the color of the last edge and the next edge to be encountered on the present scan-line. (These shades are in turn found by linear interpolation between their endpoints.)

It will be noted that Gouraud's method does not curve the edges. But considering its simplicity as a small addition to the Watkins box, that's no great sacrifice.

Naturally, the Watkins Box will not reach the private home for several years; current likely price is in six figures. But that's now.

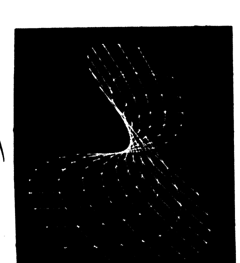
⇨ Can it be an accident that this curvaceous system was worked out by a Frenchman?



GOURAUD'S TWIST adds the appearance of curvature to a faceted object shown opaquely by the Watkins method (described in first article).

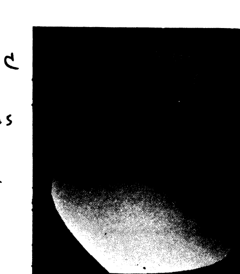
Instead of shading each point within a facet with the same color, interpolate between the vertex-colors according to how far down the edges you've gotten. Note that the jagged edges are retained.

"Wire frame" of Old-Fashioned 3D computer graphics



as shaded by Watkins (or other) method

... and as Gouraud makes it look curved.

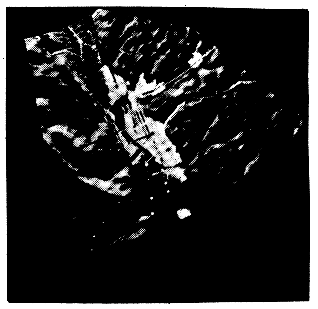


GOURAUD'S SPECIAL TWIST

PRA'S WORLD-VIEW

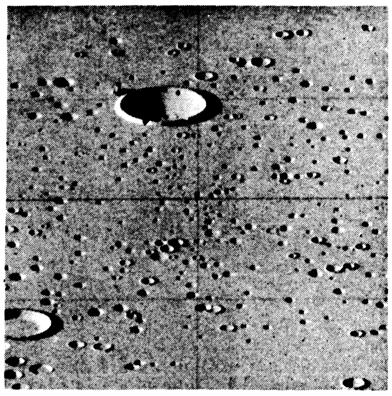
Roger Boyell, of Pennsylvania Research Associates, Philadelphia, likes to refer to the company's main interest as "modelling the physical world." Thus he and his associates have developed systems for cartography, landscape modelling, pipe design, and simulation of complex radar systems.

A radar simulator they are putting together for the Navy will show the results of any possible radar system moving over any possible terrain. A pilot or navigator trainee, in a simmulated cockpit, will see the mission's changing radar picture as he changes the plane's course or the radar's tuning. The radar picture, appearing on a screen and changing in real time, will look just the way the radar would look on a real mission-- flying in perspective among mountains or valleys, high or low, at any bearing and speed, and viewed through any type of radar.

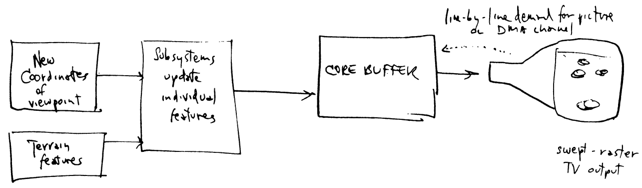


Boyell's approach is to treat each component of the pictorial/radar simulation as a separate problem, to be handled in different ways, and blended in a final buffer, a core memory which is read out to television. Separate mechanisms supply components of shadow, specular reflection, coloration and randomizing effects. The core buffer continuously refreshes the scanned CRT display.

Boyell has put the same techniques to work making simulated halftone pictures of the moon (see cut). Both the radar and moon systems use the same type of halftone image synthesis, even though superficially they seem quite different. But radar is radiation, just like light, and Boyell's techniques of three-dimensional modelling and search apply equally well to depiction by reflected visible light-- i.e., halftone images.



BOYELL'S TERRARIUM fills a fast core-memory buffer with a TV image constantly being read out (much like the Knowlton-Schwartz setup: see pp. DM10 and DM24, top Schwartz picture) and changes individual features one-at-a-time to match a changing view.



An outfit called HUMRRO, in Washington, say they have a real-time interactive half-tone that will knock several people out of the ballpark-- especially the GE hardware and the Evans and Sutherland Watkins Box (earlier).

The HUMRRO system is intended to go out to color screens (modified Sony Trinitrons) with shaded pol ygon halftone, offering pseudo-curved shading like Gouraud's (see earlier).

The techniques were worked out by Ron Swallow, and they're not telling about how they work. It is claimed, however, that their real-time picture generator handles scenes with 16,000 edges, and that this will cost $150,000 and service 16 (or was it 64) user terminals simultaneously.

It may have been a bad phone connection, or this may be what they're really claiming. Obviously it'll be really great if it turns out to be real.

Evidently they have in mind the use of such high-performance scopes for teaching, allowing students to explore intricate three-dimensional scenes or objects. Terrific.

(Note: compare the claim of 16,000 edges on a $150,000 system with the 2000 (?) edges allowed by the old NASA system built by GE, or the Watkins Box-- I don't know how many edges-- at $500,000 from Evans and Sutherland.)

THE SHAPE OF THINGS TO COME

If these systems sound far-fetched, or only for theoretical investigation, consider this: the Air Force is now letting contracts for an advanced flight-training simulator that is a small boy's dream. To be situated in Dry Lake, Arizona, the simulator will have the most realistic cockpits ever built: the entire mockup will turn and tilt in response to the user, and the seats will even swell and deflate, to simulate acceleration and weightlessness. The cockpits alone, without the visual display screens, will cost ten million dollars each.

But the visual systems-- ah. The pilot-user will look out into an artificial world, among whose mountains and meadows and clouds he will fly in real time. Six CRTs, arranged as parts of a dodecahedron in an entire visual surround, will show him the changing terrain and flying environment. Each of these CRTs will be driven by a real-time perspective halftone simulator, with all displays spliced together and driven by a master simulator responding to his actions. Who will build them is not yet decided; they could be Warnock or GE boxes.

The sheer joy of such a system will be hard to beat. But no doubt others will be on the way-- perhaps at the amusement-park level.

AIR FORCE SUPERTOY

The new pilot trainer will not only swing and dip in response to the controls; on six giant CRTs, with optics in front that focus the eye on infinity and connected at the seams, the pilot will see a responding perspective simulation of the world he is flying through, planes he is dogfighting with, and who knows-- witches? Superman?

NELSON'S FANTASM™: A LOT OF BOSCH?

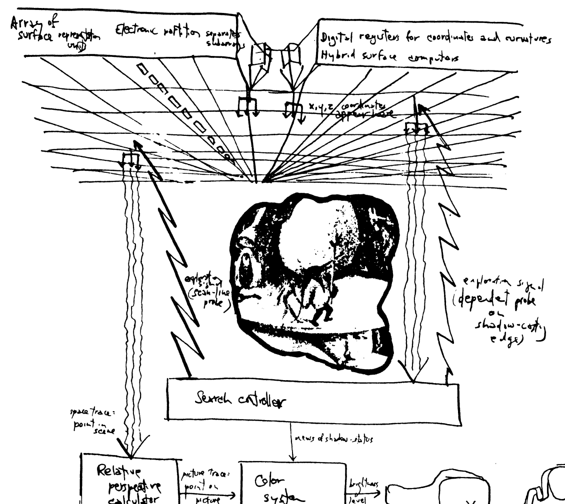
I don't expect you to believe this, because not even my patent attorney does, but the system I call Fantasm is intended to make pictures that pass the Turing-test: you won't be able to tell them from real photographs. Fantasm is intended to allow the user to make realistic, Hieronymus Bosch-like photographs and movies, with real-looking people (and scenery, imaginary characters, monsters, etc.) in scenes of arbitrary complexity. It is expected that 1975 economics will make its construction feasible.

Fantasm I originally conceived as a method of making realistic photographs and movies, not knowing at the time that this was impossible, but feeling it could be done somehow if the problem were broken down sufficiently. At times it was not clear which of us would be broken down first, I or it.

It occurred to me sometime in 1960-1 that computer-interpolated, Disney-type cartooning methods would be feasible. After some thought I realized that pseudo-photography would be possible, and dropped the cartooning idea. The strange behavior of people whom I told about this led me to increasing secrecy.

The general goal was to make a system that could do realistic movies without scenery or actors, and make pictures indistinguishable from real photographs of real scenery and actors. ("What do you mean, indistinguishable from photographs?" people keep asking. What do they mean what do I mean?) The surfaces are to be put in by "sculptors," animated by "puppeteers," and photographed by a "director." The objective is for moviemaking to be under the utter imaginative control of the creative user.

I am indebted to Prof. Charles Strauss for the formalization of my smoothing-function.



FANTASM AT LAST PARTIALLY REVEALED,

at least to certain readers.

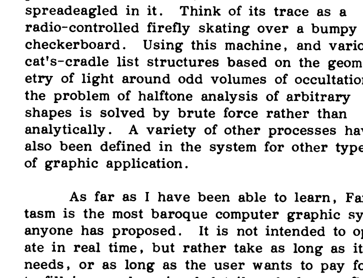
A scene of arbitrary curvature and topology is represented in a system of holding registers; the surface is presented (through D-to-A converters and an array parallel function generator) to interrogating circuitry which steers an inquiring signal around the represented surfaces. Operation is empirical. Array has partition logic allowing simultaneous queries of various subsurfaces. Feedback steering circuitry allows multiple loops through array. Steering signal and returned surface parameter are analog and continuous. List techniques manage shadow and visibility 'umbrellas' (surfaces of occulted volumes or umbras).

The Fantasm Scene Machine™, the representation and search array, is one chip repeated in a carpet. Large-scale integration permits the required digital storage of about 500 bits per surface section plus analog circuitry and switching logic. Patent work underway.

SUMMARY: outlines handled by Perimeter Parameter Occultation Chasing, fill-in by Bullet Search, animation continuity management by list-processing techniques.

The system could come in a number of different versions. One of these involves a large array of LSI computing modules (the checkerboard Scene Machine) to be guided by special hardware under an unusual monitor running on a general-purpose computer. The checkerboard Scene Machine holds a great spread of surface data. It is a logical curiosity, an array that replies as a unit, ignoring cell boundaries, to electrical explorations of the shapes represented in it. The resulting trace makes various 3-space explorations on the faces, mountains or automobiles spreadeagled in it. Think of its trace as a radio-controlled firefly skating over a bumpy checkerboard. Using this machine, and various cat's-cradle list structures based on the geometry of light around odd volumes of occultation, the problem of halftone analysis of arbitrary shapes is solved by brute force rather than analytically. A variety of other processes have also been defined in the system for other types of graphic application.

As far as I have been able to learn, Fantasm is the most baroque computer graphic system anyone has proposed. It is not intended to operate in real time, but rather take as long as it needs, or as long as the user wants to pay for, to fill in complex visual details, shadow, reflections, curlicues, leaves, hair, etc. It is best suited to the production in Panavision of Busby Berkeley musicals, or "The Lord of the Rings" with realistic wraiths and interspecies battles. But it may well cost too much to use for that. Indeed, its economics seem to improve in low-budget settings like videotape, although there its output bandwidth will flower unseen. But the Scene Machine should also be useful for more mundane applications, such as contour mapping, automobile design, advertising photography and medical illustration.



FOURTH ARTICLE.

Systems of Computer Image Corporation.

COMPUTER IMAGE'S MAD WHIRL

SO FAR WE HAVE SUMMARIZED AND DISTINGUISHED AMONG THE MAJOR TECHNIQUES FOR COMPUTER SYNTHESIS OF IMAGES FROM DIGITALLY STORED REPRESENTATIONS OF SCENES. WE NOW TAKE THE WRAPS FROM A DIFFERENT BUT RELATED SET OF TECHNIQUES-- THE SYSTEMS OF COMPUTER IMAGE CORPORATION.

Lee Harrison III got the idea for what is now Computer Image Corporation in 1959. Already having an art degree, he went on for a degree in electrical engineering, and through long lean years put together the technical basics around which CI's systems are now built. Computer Image Corporation is now a going concern, and output from their systems, especially Scanimate, is now widely visible on television.

Computer Image Corporation seems to be the first firm to be commercially successful in the halftone field. Whether they should be included with the others is arguable, however. Their systems are not widely understood, and the relation of these systems to the other systems and programs described in these articles is problematical. Among the few who understand their techniques, some argue that they do not synthesize images at all, but rather twist preexisting pictures with a sort of Moog synthesizer, and that their analog techniques are really just compound oscillators rather than true computing. I think that this view is wrong, at least as regards their most ambitious system, and that CI's techniques deserve review. All the world is not digital. CI systems do fill up areas with grey-scale (and other) pictures, and their systems involve three-dimensional coordinates, occultation and coloration; thus I think it appropriate to discuss them here.

The following discussion is the first, I believe, to lift the veil of secrecy that has hitherto confounded observers of this company's work. In the light of the extreme sophistication with which they have pursued extremely strange techniques, they should benefit from the wider understanding. (Note that this material, which has been assembled from various sources and careful TV watching, is partly conjectural.)

Computer Image's systems represent an apparently unpromising approach brilliantly followed through.

All of CI's systems are a strange combination of closed-circuit TV and analog components out of a music synthesizer: oscillators, potentiometers, interconnection networks. The basic mechanisms are the same for all, but they are carried to different logical extremes, with differing accoutrements, in the four systems. They all seem to be based on the extraordinary Animac II, not yet implemented; it would seem that for business reasons the company decided to raise money promoting simpler systems, so its bread and butter now consists of two less ambitious systems, Scanimate and Animac I; both of which might be puzzling if not recognized as parts of a more elegant whole. It would seem they were designed backwards as spinoffs from Animac II, as was CAESAR, their more recent 2-D system.

The extraordinary ramifications and varieties of this system, with all its electronic add-on and composite methods, stagger the most jaded technical imagination.

At the heart of the CI systems is the principle of filling areas of a CRT screen with an oscillating trace. This is a principle common to both Lissajous figures and television; but Computer Image has elaborated it peculiarly. By variations they paint twisted television images, wiggle sections of superimposed drawings, create moving filigree effects, and hope to animate whole groups of opaque electronic puppets in 3-space.

Consider an oscillating trace on an oscilloscope. This is a two-dimensional oscillation, having two signals, x and y. But a three-dimensional oscillation is also possible; any third signal, z, can be interpreted as a third dimension, meaning that a "point of light" is whirling out some pattern in a three-dimensional space-- an oscillotank, so to speak. Let us call this point moving in three dimensions a "space trace."

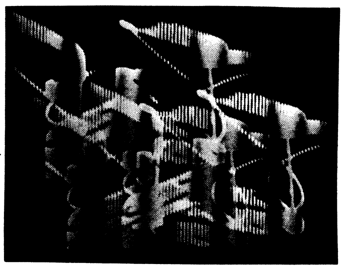
Now to view this trace we need to cut it down to two dimensions. By ignoring one of the traces we can view the oscillotank in certain fixed ways; but by creating a "view calculator," a box performing certain perspective transformations on the three signals of the space trace, we may obtain a view of the oscillotank from a movable vantage point. This is an x-y view which we may put on an ordinary oscilloscope.

Let us now add one more signal, b (for brightness). This is the brightness signal familiar in television.

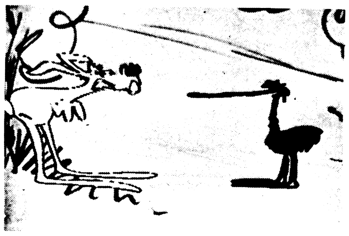
Brightness of the spot is thus independent of the movement of the space trace. For example, the space trace could describe a helical path, a sort of tornado motion, and we could time its spinning to phase with a TV signal. If we now brighten the space trace only with the brightness signal of a TV pickup, we now will see (in our view of the oscillotank) what would look like a TV picture curled around itself in space.

The different CI systems are built around this effect.

Output from all these signals is ordinarily picked up by another vidicon, which stabilizes it by converting it into conventional television imagery.

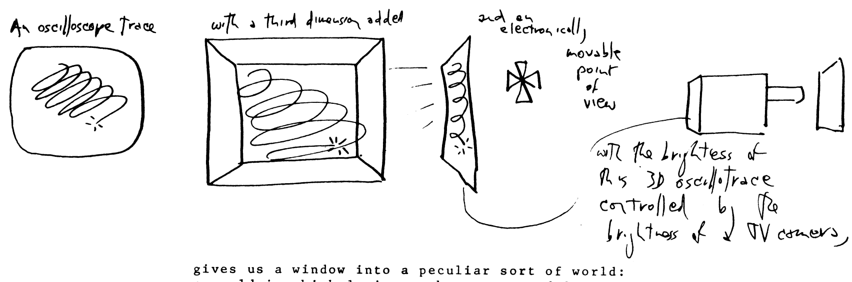


Scanimate's twirl, by now familiar to most TV watchers. Scanimate is extensively used on "The Electric Company."

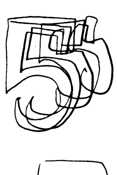


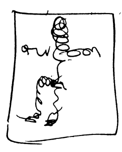
CAESAR System. Characters are made to move jaws and lips by jointing technique similar to Animac II (below), but in such a way as to matte over drawn artwork-- meantime wiggling other drawn artwork through scan manipulation.

THE WHIRLING UNIVERSE OF COMPUTER IMAGE CORPORATION.



gives us a window into a peculiar sort of world: a world in which luminous shapes can undulate and spin on invisible spindles (Scanimate), or wiggle as separate bones (CAESAR.





Tubelike shapes may be rotated and shaped in 3D (Animac), and puppets may eventually be rolled like cigarettes (Animac II), which may then be painted from a TV pickup on the side nearest the viewpoint.

By using a storage tube and spinning the trace close together, like cotton candy, and cutting off the painting signal while the trace is within the area already filled, we get electronic masking: which blends animated drawings in 2D (CAESAR) and may eventually manage shadows and occultation masking among 3D puppets (Animac II).

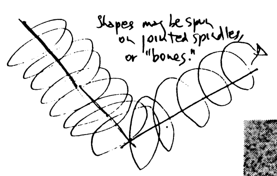
SHAPING METHOD

Lissajo_us and zigzag figures are rapidly spun in three dimensions -- that is, varying voltages x, y and z. The resulting "tubes" and "curtains" are then viewed by perspective calculation. The circuitry permits these shapes to flex at joints, wave, and go through other changes.

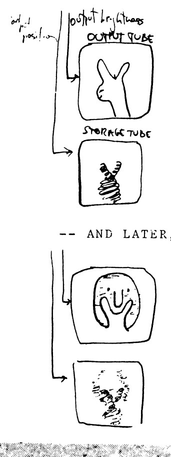
IN SCANIMATE: zigzag and curling shapes define a moving scroll on which an image is painted.

IN CAESAR: curling shapes are treated 2-dimensionally, as blocking controls for artwork.

IN ANIMAC II: puppets will be sculpted much like rolling a cigarette.



BLOCKING METHOD

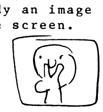


While showing nearer part--

spin out same part on storage scope; continuously test potential of screen;

-- AND LATER, IN THE SAME FRAME:

cut off the brightness of the output signal, wherever there is already an image on the screen.



OCCULTED COMPOSITE

The only picture I've been able to find that relates to the 3D sculpturing of Animac II is this frame, blown up from a short 16mm sequence. The figure is sculptured from oscillations in three variables, modulated to represent this figure of thirteen sections or "bones." Head and torso are clearly visible in the film; the figure is seen to spin as if in an ejection seat.

A last CI technique, technically minor but remarkable in effect, permits this blocking and shadowing among separate objects. This is the use of a storage CRT tube on which every frame is painted (from the viewpoint or from the light source). The picture is painted on the storage CRT, nearest things first; and the return signal from the screen tells whether the space trace is crossing an area already painted during the frame. The tube's output signal then effectively constitutes a silhouette. This clue indicates that the space trace should not be visible; and hence is used to cut off brightness while the trace is within the already-filled area. This gates between two desired objects or pictures, foreground and background. If operated from the point of view of the light, it gates shadow: the signal is used to control the relative brightness of the shadowed and unshadowed features of a puppet in 3-space.

A fascinating variety of embellishments has been put into these systems by CI's ingenious engineers. Coloration of the final video signal is added by gating color levels under control of the brightness signal, permitting pictures with several grey-levels to be transformed to up to four rainbow hues. Separate shapes described by the space trace may be independently moved and jointed at the same time: Harrison pointedly calls such separate shapes "bones." Darkening at the backside of a spun shape, or brightening at edges of a painted portion, and brightening in proportion to curl, are all strange capabilities of this machine. Lip-synchronized mouthlike motion can be imparted to any part of the shape spun by the space trace (whether or not a mouth is painted on it), by an audio detector feeding directly to the circuitry from a live mike. And the limbs of CI's ghostly figures can be made to swing by connection of sensors to the animators themselves-- in a living pantograph.

SCANIMATE is a popular device now widely used (at CI's studios) for the making of TV commercials and station-break emblems. This is their simplest system, used for the conversion and discombobulation of flat artwork. In Scanimate, the space trace is controlled by hand-operated potentiometers. Two separate oscillator settings are available, so that the space trace can have two separate oscillation patterns, spinning out two entirely different virtual shapes in 3-space. A hand-throttle eases from one oscillator setting to the other. This permits an image to be moved, shrunk or enlarged, or flipped; to go from whirling around to a sort of hula; and many more effects. The picture painted on it may be seen to roll on invisible spindles, bloom into fountains, or undulate as pennants-- all by modulating the brightness of the flying spot as it traces its unseen shape. This shape, in turn, can move between its two forms under control of the throttle.

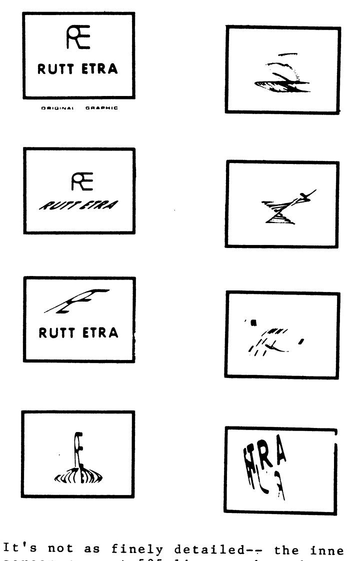
Animac I (usually called Animac) provides greater flexibility in controlling the space trace. The system's oscillations are controlled by an input vidicon, which artists may quickly modify with pastel check at the pickup. Ghostly tubular lettering, swarming pendulum-patterns and jiggling filigrees are among the possible doodles.

CAESAR, their newest system, is oriented toward Yogi Bear-type animation. The artist's cartoons are automatically superimposed on a background or each other. They may be moved, and made to wiggle under real-time control by the user.

But it is to Animac II that these curiosities lead. What Harrison calls the "Snow White Capability" of Animac II will permit the sculpture of full humanoid puppets, with perhaps thirty articulated "bones," opaque to one another and casting shadows, colored, moving and talking.

Two young fellas in a Manhattan loft, Messrs. Rutt and Etra, are offering a machine similar to Scanimate but much cheaper.

(Rutt Electrophysics, 21-29 W. Fourth St., NY 10012.)



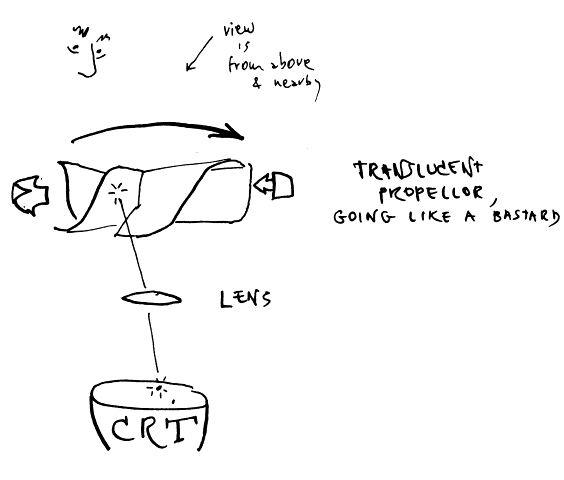
It's not as finely detailed-- the inner screen runs at 525 lines rather than 700-- but it costs some $15,000 instead of $150,000.

WHAT ABOUT REAL THREE-DIMENSIONAL DISPLAY?

In science-fiction stories you hear about how objects are made to appear as if they're standing in the middle of the room. For instance, I believe that in Heinlein's Stranger in a Strange Land they watched a "tank" in which things appeared.

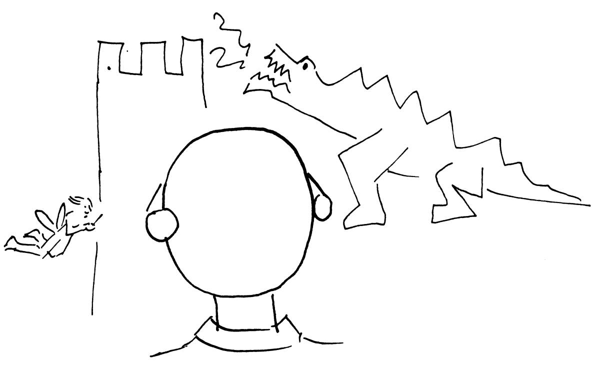
Well, a lot of people have thought about this, and it's not so easy as you might think.

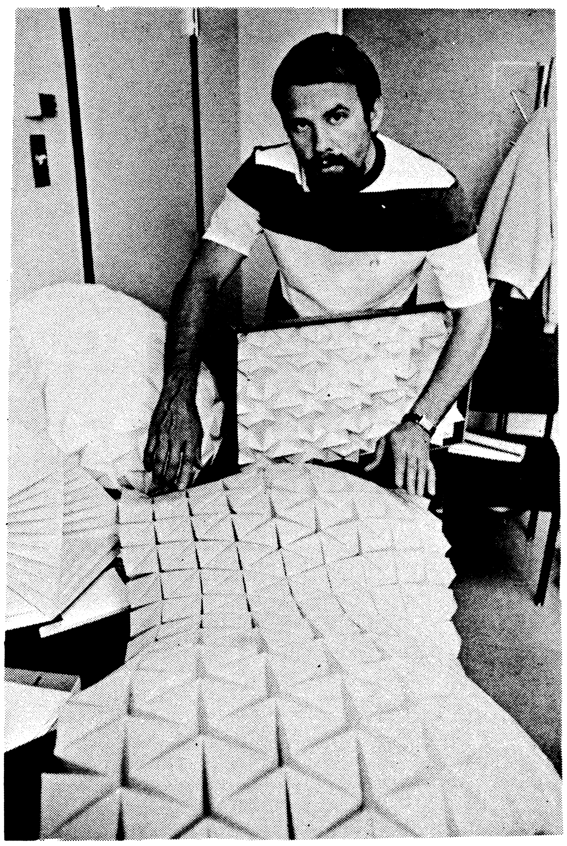
One interesting scheme used a sort of translucent propellor, spinning rather fast, on which computer-generated images were projected from below. It was done by the dotting method, so that a bright dot of light would appear high or low in space depending on whether it was projected on a relatively high or low point on the propellor.



This was interesting but had numerous disadvantages-- not the least of which was the danger of the thing flying apart. (Translucent materials tend not to be as strong as, say, metal.) Another basic problem, though, was the fact that any given point in the space could only be displayed at a given time, when the propellor's height in that region was just right, and that meant that at that given instant you couldn't display any of the other points that could only be displayed at that instant. A considerable disadvantage.

Probably the most astonishing 3D display is Sutherland's Incredible Helmet. This consists of a helmet with two dinky CRTs mounted on it, each being driven in real time by a perspective system (such as the LDS-1) and set up with prisms to the wearer's eyes. Through the prisms the wearer can see the real world in front of him. Reflected in the prisms, however, and thus mixed into the view of the real world, is the glowing wire-frame being presented to him-- in perspective, and with its separate views merging into an apparent object in front of him. But he need not stand still: as he moves, the helmet's changing position is monitored by the program, and the display system changes the views accordingly meaning he can walk around and through a displayed object. The illusion, and the possibilities, are fantastic: imaginary architecture, explanations and diagrams of things in the room, poetry that changes as you walk through it, ... well, you work on it. Not available commercially.





U.U.

RETURN TO THE FOLD

There was a lot to be said for tents. They could be made by tailors, rather than construction gangs; they could be transported and stored flat. Their surface-to-volume ratios couldn't be beat.

Noting this, an architect named Ron Resch said to himself: what about making large-scale foldable structures, likeunto geodesic domes, that cou_ld be simply manufactured in sheet form and creased at the factory, then bolted and cabled and strutted in the field?

Resch has now for years been experimenting with complex folded structures.

There's only one trouble. If you've messed with paper airplanes you know that folding is an inaccurate process, and so the prospect of discovering complex geometric structures by the hand-folding of paper is rather slim.

Recognizing this, Resch has contrived to work at a computer display. His work-- the search for great folding structures-- is one of the first practical uses of halftone polygon computer graphics. He is, naturally, at the University of Utah.

Lou Katz, of NYU, put old-fashioned stereopticons up to the CRT, and displayed two separate views to the two eyes. Works fine, even with isometric display.

Bob Spinrad of Xerox Data Systems has a patent on displaying 3D from a computer through an ordinary color TV. Assuming you're using some standard way of refreshing the TV-- described elsewhere-- the image for one eye is displayed in green, the other in red, and you look through red and green glasses. The wonders of modern science. Spinrad chuckles over it himself.

Another scheme glued silver Mylar to the front of a loudspeaker, then played a soft hum through the loudspeaker to pulse the Mylar back and forth. Then you used that as a mirror to look at what was going on the CRT-- which was showing a lot of points at odd places that would appear to be in space. Unfortunately this was hard to coordinate, and, like the propellor, often required you to put dots in several places at once, which don't work.

For a while you could get-- maybe you still can-- a three-dimensional computer output device. Here's what it did: it created objects showing data structures that had three variables. (It didn't make wire-frame objects or the like.) Automatically ejecting wire through a styrofoam block, and snipping the done ones, it created little mountains showing three-dimensional data. Very cute. Since many people have problems with mountainous computer data, it probably should have caught on.

Then a lot of people mumble the word "holography," as if that is going to settle something. While holograms are terrific and remarkable, and have been produced on computers, making them is not a process that can be carried out decently on sequential machines-- let alone making them in real time. So if a solution to interactive three-dimensional computer display is going to come through holography, it means a whole new batch of technology will have to be invented.

My friend Andrew J. Singer, who comes and goes in the computer field and is one of the five or six smartest people I ever met, says he knows how to build a display tank, and I believe him. He explained it quickly to me once and I asked him to tell it again, but he just said sadly, "What's the use-- there are so many great things that could be done..."

FOUR DIMENSIONS, EGAD

So much for three dimensions. Now, some readers are bound to ask, "What about four dimensions?" because they are science-fiction fans or troublemakers or mathematicians or something.

Just as we can make a two-dimensional picture of a three-dimensional object, it is possible, dear reader, to make a two-dimensional picture of a four-dimensional object.

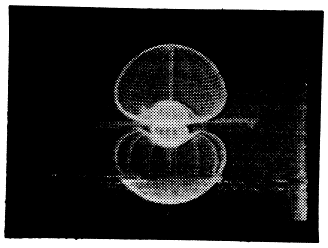
What is a four-dimensional object?

Why, any object that has four dimensions, (thanks a lot, you say), or even four measurable qualities, such as height, weight, age and grade point average. Well, let's not get into that, but it turns out that views of such multidimensional structures may be obtained by the same homogeneous matrix techniques already mentioned for regular perspective calculations. Rule of thumb: however many dimensions your data has originally, you add one more dimension, homogeneous with the rest, and there exist formulas (sorry, I don't have them) for view calculation.

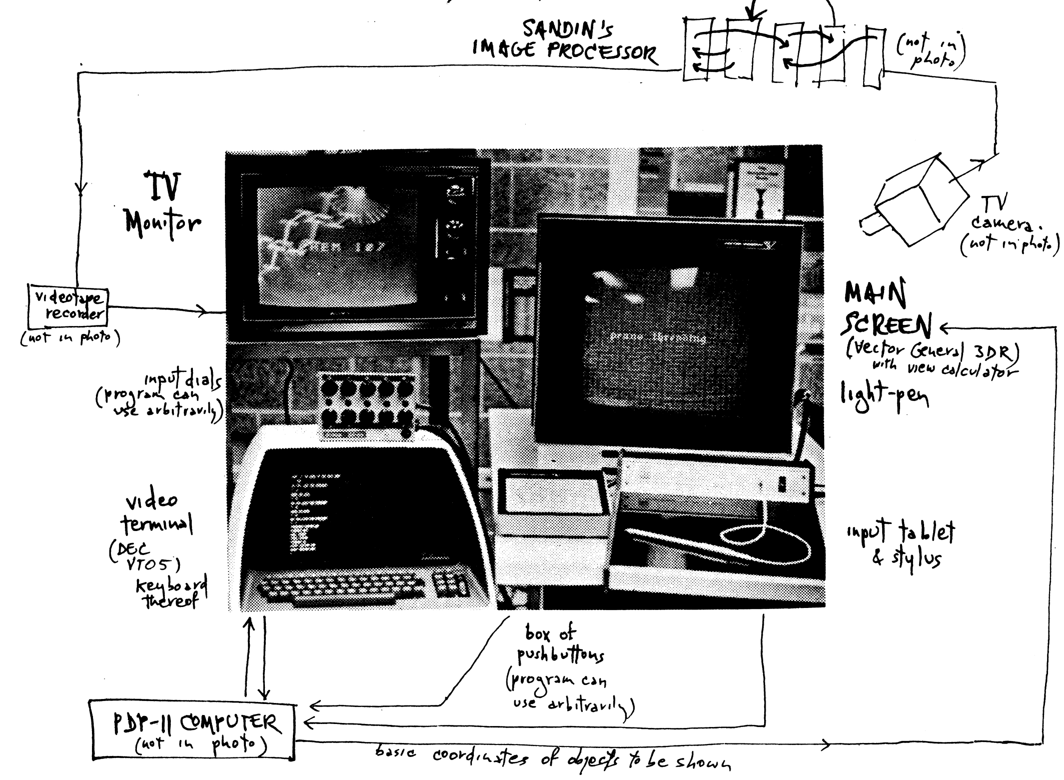
(Note, of course, that while a two-dimensional view is a picture, a three-dimensional view is a three-dimensional object-- you'll have to view it on an interactive 3D computer display of some kind.)

End of *THE MIND'S EYE*.

THE CIRCLE GRAPHICS HABITAT



From videotape, "The Hydrogen Atom According to Quantum Mechanics" by T.J. O'Donnell & David Parrish.



It is usually hard to combine things: especially complicated technical things. Usually it takes infinite reconsiderations, finagling, modification, intertwingling.

The Circle Graphics Habitat, however, is something else again. It results from two intricate, independent technological developments, each an intricate system carefully crafted by an exceptionally talented person, coming together like two hands clapping. Like ham and eggs, like man and woman, Sandin's Image Processor and DeFanti's GRASS language conjoin directly and interact perfectly as if they had been made for each other, which they were not.

Dan Sandin's Image Processor (see p. DM8) is a system of circuit boxes that allow video images to be dynamically colored, matted, dissolved and palpitated; Tom DeFanti's language (see "Coup de GRASS, p. DM 31) permits the rapid creation, viewing and manipulation of three-dimensional objects on the screen of a particular computer setup.

To combine them, you just point Dan's system at Tom's system.

Let's say that on the screen of Tom's system we are viewing an animated bird, flapping its wings. Since it's being shown on a three-dimensional refreshed line display (see pp. DM22-3, DM30), it appears only as white lines on a dark screen.

Dan merely points a TV camera at Tom's screen, and runs the TV signal into his Image Processor. Now, in the Image Processor, he gives it the magic of color. Different colors, interplaying with gradations and subtlety.

From the Image Processor, the finished signal goes out to videotape recorders.

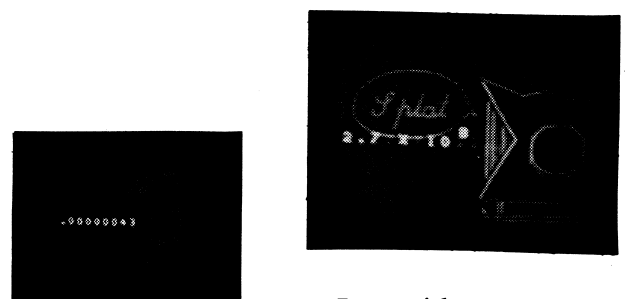
What then have we overall? One of the world's most flexible facilities for the rapid production of educational videotapes.

To explain something, you create a three-dimensional stick-figure "model" of it, using DeFanti's GRASS language. Then you make a videotape of it, showing rotations or other manipulations, using the Image Processor to give it color.

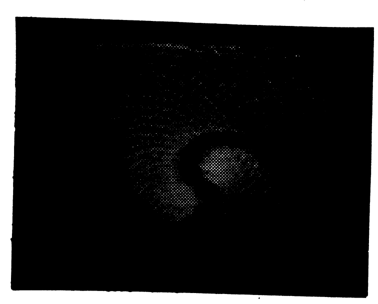
DeFanti and Sandin have spent much of the academic year '73-4 getting the kinks out of this procedure. (Many of the difficulties stem from the unreliability of videotape recorders.) Stills from some of the first work are shown here.

BIBLIOGRAPHY

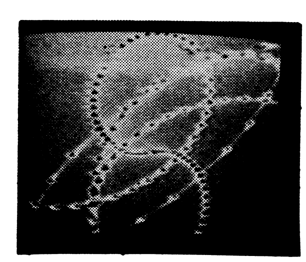
Thomas A. DeFanti, Daniel J. Sandin and Theodor H. Nelson, "Computer Graphics as a Way of Life." To be presented at U. of Colorado computer graphics conference, July 1974; to appear in proceedings purportedly to be called Computers and Graphics.

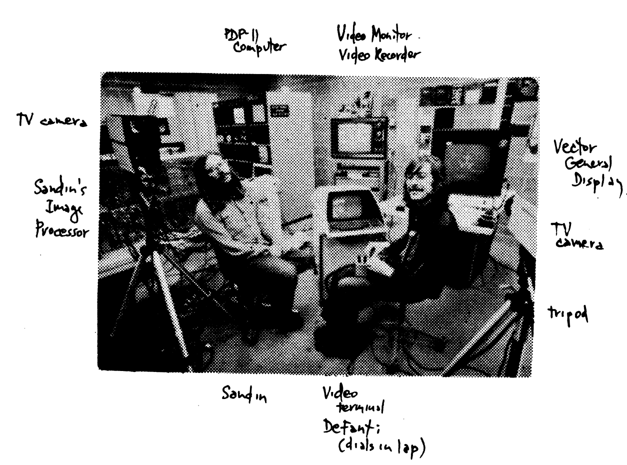


From videotape, "The Number Cruncher," by TDF & DJS.



From videotape, "The Spiral Tape," by DJS and TDF.





The TISSUE OF THOUGHT

Uneducated people typically think of education as the learning of a lot of facts and skills. While facts and skills certainly have their merits, "higher education" is also largely concerned with tying ideas together, and especially alternative structures of such tying-together: with showing you the vast uncertainties of things.

A wonderful Japanese film of the fifties was called Rasho-Mon. It depicted a specific event-- a rape-- as told by five different people. As the audience watches the five separate stories, they must try to judge what really happened.

The Rasho-Mon Principle: everything is like that. The complete truth about something is never known.

Nobody tells the complete truth, though some try. Nobody knows the complete truth. Nowhere may we find printed the complete truth. There are only different views, assertions, supposed facts that support one view or another but are disputed by disbelievers in the particular views; and so on. There are "agreed-on facts," but their meaning is often in doubt.

The great compromise of the western world is that we go by the rule: assume that we never know the final truth about anything. There are continuing Issues, Mysteries, Continuing Dialogues. What about flying saucers, "why Rome fell," was there a Passover Plot, and Did Roosevelt know Pearl Harbor would be attacked?

Outsiders find the intellectual world pompous, vague in its undecided issues, stuffy in its quotes and citations. But in a way these are the sounds of battle. The clash of theories is what many find exhilarating about the intellectual world. The Scholarly Arena is simply a Circus Maximus in which these battles are scheduled.

Many people think "science" is free from all this. These are people who do not know much about science. More and more is scientifically known, true; but it is repeatedly discovered that some scientific "knowledge" is untrue, and this problem is built into the system. The important thing about science is not that everything will be known, or that everything unanimously believed by scientists is necessarily true, but that science contains a system for seeking untruth and purging it.

This is the great tradition of western civilization. The Western World is, in an important sense, a continuing dialogue among people who have thought different things. "Scholarship" is the tradition of trying to improve, collate and resolve uncertainties. The fundamental ground rules are that no issue is ever closed, no interconnection is impossible. It all comes down to what is written, because the thoughts and minds themselves, of course, do not last. (The apparatus of citation and footnote are simply a combination of hat-tipping, go-look-if-you-don't-believe-me, and you-might-want-to-read-this-yourself.)

"Knowledge," then-- and indeed most of our civilization and what remains of those previous-- is a vasty cross-tangle of ideas and evidential materials, not a pyramid of truth. So that preserving its structure, and improving its accessibility, is important to us all.

Which is one reason we need hypertexts and thinkertoys.

A lot of people are afraid to ask questions because they're afraid of looking dumb. But the dumb thing is not asking questions.

If I ever get my school, the one course taught will be

HOW TO LEARN ANYTHING

As far as I can tell, these are the techniques used by bright people who want to learn something other than by taking courses in it. It's the way Ph.D.'s pick up a second field; it's the way journalists and "geniuses" operate; it brings the general understandings of a field that children of eminent people in that field get as a birthright; it's the way anybody can learn anything, if he has the nerve.

1. DECIDE WHAT YOU WANT TO LEARN. But you can't know exactly, because of course you don't know exactly how any field is structured until you know all about it.

2. READ EVERYTHING YOU CAN ON IT, especially what you enjoy, since that way you can read more of it and faster.

3. GRAB FOR INSIGHTS. Regardless of points others are trying to make, when you recognize an insight that has meaning for you, make it your own. It may have to do with the shape of molecules, or the personality of a specific emperor, or the quirks of a Great Man in the Field. Its importance is not how central it is, but how clear and interesting and memorable to you. REMEMBER IT. Then go for another.

4. TIE INSIGHTS TOGETHER. Soon you will have your own string of insights in a field, like the string of lights around a Christmas tree.

5. CONCENTRATE ON MAGAZINES, NOT BOOKS. Magazines have far more insights per inch of text, and can be read much faster. But when a book really speaks to you, lavish attention on it.

6. FIND YOUR OWN SPECIAL TOPICS, AND PURSUE THEM.

7. GO TO CONVENTIONS. For some reason, conventions are a splendid concentrated way to learn things; talking to people helps. Don't think you have to be anybody special to go to a convention; just plunk down your money. But you have to have a handle. Calling yourself a Consultant is good; "Student" is perfectly honorable.

8. "FIND YOUR MAN." Somewhere in the world is someone who will answer your questions extraordinarily well. If you find him, dog him. He may be a janitor or a teenage kid; no matter. Follow him with your begging-bowl, if that's what he wants, or take him to expensive restaurants, or whatever.

9. KEEP IMPROVING YOUR QUESTIONS. Probably in your head there are questions that don't seem to line up with what you're hearing. Don't assume that you don't understand; keep adjusting the questions till you can get an answer that relates to what you wanted.

10. YOUR FIELD IS BOUNDED WHERE YOU WANT IT TO BE. Just because others group and stereotype things in conventional ways does not mean they are necessarily right. Intellectual subjects are connected every whichway; your field is what you think it is. (Again, this is one of the things that will give you insights and keep you motivated; but it will get you into trouble if you try to go for degrees.)

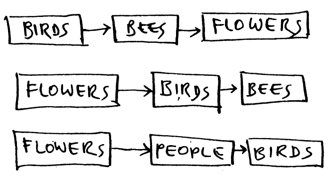
* * *

There are limitations. This doesn't give you lab experience, and you will continually have to be making up for gaps. But for alertness and the ability to use his mind, give me the man who's learned this way, rather than been blinkered and clichéd to death within the educational system.

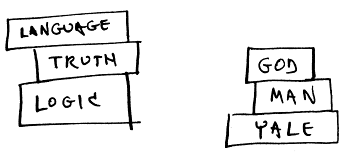
BIBLIOGRAPHY

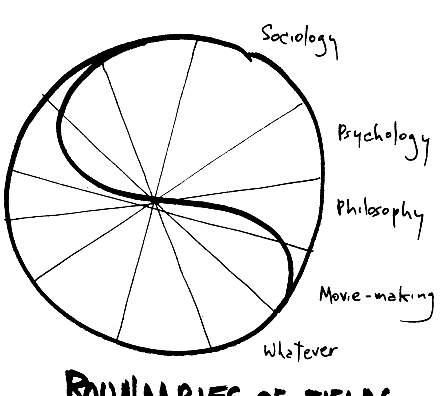
Wilmar Shiras, Children of the Atom.
Science-Fiction about what a school could be like where kids really used their minds. I've always been sure it was possible; the R.E.S.I.S.T.O.R.S. (see p. 47) made me surer.

PRESENTATIONAL SEQUENCES ARE ARBITRARY



HIERARCHIES ARE TYPICALLY SPURIOUS

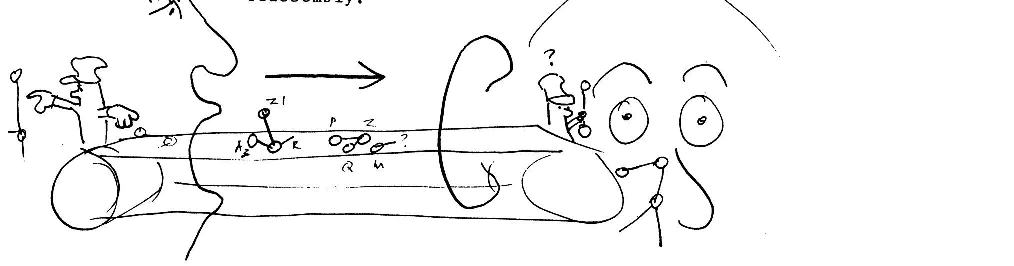




BOUNDARIES OF FIELDS ARE ARBITRARY

COMPARTMENTALIZED AND STRATIFIED TEACHING PRODUCES COMPARTMENTALIZED AND STRATIFIED MINDS.

Verbal communication-- whether written or spoken-- is the disassembly of the Tinkertoy of thought into pieces, and placing it on a conveyor belt to its place of reassembly.



"ON WRITING," a paradigm of the creative process

being an examination of some very Complex Matters which Nobody Seems to Understand; and whose Generality of Relevance may be Gradually Apprehended. (Eventually I hope to develop a somewhat more formal treatment of "ideas," as distinct from propositions, sentence kernels, etc. But there is certainly no room for that here. (Logicians: show me the truth-table of "BUT.")

The process of writing is poorly understood in most quarters. Many working writers despair of being "systematic," getting things done as best they can. On the other hand, people who think they might be able to contribute-- particularly the symbolic logicians and transformational linguists-- being immersed in their own formalisms, simply don't see what's going on-- at least, when I've tried to talk to them.

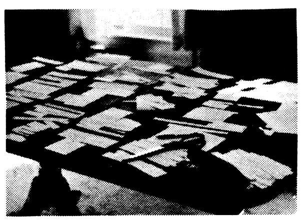
Writing is not simple. As with vision or speech or riding a bicycle, an immensely complex process is being unconsciously pursued.

Some people think you make an outline and follow it, filling out the details of the outline until the piece is finished. This is absurd. (True, some people can do this, but that is simply a shortcutting of the real process.) Basically writing is

THE TRY-AND-TRY-AGAIN INTERPLAY of PARTS AND DETAILS against OVERALL AND UNIFYING IDEAS WHICH KEEP CHANGING.

In fact a number of things are happening, often simultaneously. We can separate them into three:

1. Provisional development of ideas and points:
   A) forming overall organizing ideas, B) selecting tentative points; C) inductively finding overall organization among them; D) finding relations of interest between points.
2. Complex sifting and adjustment among collections of points, overall ideas.



3. Fine splicing within developed sequences.
   A) transition and juxtaposition managements, B) cross-citations, C) smoothing.

Regrettably, there's no room or time to pursue this here. (The article I had intended to write would take a whole spread.) For people who really care about the matter, I will make some points in very abbreviated form.

The interesting structures in written material include:

"Points"-- pieces, sentences, phrases, examples, plot events, and expository "points."

Organizing principles and structures (which we will call here arches)-- final ironies, things to be led up to, themes, plots, concepts, principles, expository structures, organizing titles, overconcepts. These may be either local or global, over the entire work. (Note: arches may not be heirarchical relative to one another.)

Now, we may think of points and arches as individual objects which have individual relations to one another. Between two points there may be a good transition; a specific point may link well to a specific arch.

The problem in writing, then, is that overall structures you choose (systems of arches) may not link well to the points that have to be included among them; and that transitions between points don't work out the way you want them to. Good transitions can't be worked out for the sequence of points you want to make, or, alternatively, there are too many good transitions within a specific structure of points, and picking among them involves difficult choices-- especially when you have to devise appropriate arches on the basis of the final sequence of points.

There are a number of other important structures in written material. They include accordances, juxtapositions, cross-citations, connotations, nuances and rhythms.

The only ones we will discuss here are accordances.

The term "accordance," as I shall use it here, is simply a vaguely formal way of talking about whether things match or fit together. Two items are in accord if they match or fit well, or in discord if they match or fit badly. Thus a good transition between points (as mentioned early) represents an accord, and a good link between a point and an arch is also an accord.

Now, it happens that a great deal of writing is concerned with notes to the reader about accordances in the material. In fact, quite a few words are exclusively concerned with subtly pointing out to the reader the accords and discords within the expository structure of what he is reading. We may call these accordance-connectives or accordance-notes.

Two of the most basic terms are indeed and but.

The word indeed has an interesting function.

The word indeed (in its main use, at the beginning of a sentence) indicates an accord between what has just been said and what is to follow. In other words, it functions as a positive transition, impetus or gas pedal, indicating a continuation of the flow in the direction already indicated. So do the words thus, then, therefore, moreover, so and furthermore. These are infix accords, that is, notes of accord that go between two items. We also see prefix accords, such as since, inasmuch as, insofar as; these have to be followed by two clauses, the second of which is in accord with the first.

The word but is exactly the opposite. It indicates a discord or contradistinction, a negative transition, "brakes" in the flow. Other such infix discords include nevertheless, despite this, on the other hand, even so, and "Actually,..." Similarly, there are prefix discords: while, despite, though..., notwithstanding.

I find this topic of inquiry very interesting. These sorts of terms have been used since time immemorial by writers adjusting their transitions for smooth flow (note such antiquey variants as haply, howbeit, withal, forasmuch and howsomever), but the importance and structure of this service has not, I think, been generally understood.

(Note also that there are more intricate accordance-connectives: I wish we could go here into the structure of In fact..., at least, ...if not... , ... otherwise... , Anyway... , and Now....)

* * * * * * * * * * * * * * * * * *

(Note: the try-and-try-again revision and reconsideration process, tinkering with structural interconnections, is a universal component of the creative process in everything from movie editing to machine design. There ought to be a name for it. I can't think of a satisfactory one, although I would commend to your attention grandesigning, piece-whole diddlework, grand fuddling, meta-mogrification, and that most exalted possibility, tagnebulopsis (the visualization of structure in clouds).)

THE HERITAGE

The past is like the receding view out the back of an automobile: the most recent is more conspicuous, and everything seems eventually to be lost.

We know we chould save things, but what? Those with the job of saving things-- the libraries and museums-- save so many of the wrong things, the fashionable and expensive and high-toned things esteemed by a given time, and most of the rest slips past. Each generation seems to ridicule the things held in esteem by times before, but of course this can never be a guide to what should be saved. And there is so much to save: music, writing, sinking Venice, vanishing species.

But why should things be saved? Everything is deeply intertwingled. We save for knowledge and nostalgia, but what we thought was knowledge often turns to nostalgia, and nostalgia often brings us deeper insights that cut across our lives and very selves.*

Computers offer an interesting daydream: that we may be able to store things digitally instead of physically. In other words, turn the libraries to digital storage (see Hypertexts, p. DM 48); digitize paintings and photographs (see "Picture Processing," p. DM 10); even digitize the genetic codes of animals, so that species can be restored at future dates (see "The Mitiest Computer," p. 60).

Digital storage possesses several special advantages. Digitally stored materials may be copied by automatic means; corrective measures are possible, to prevent errors from creeping in-- i.e., "no deterioration" in principle; and they could be kept in various places, lessening mankind's dependence on its eggs being all in one basket (like the Library at Alexandria, whose burning during the occupation of Julius Caesar was one of the greatest losses in human history).

But this would of course require far more compact and reliable forms of digital storage than exist right now.

Nevertheless, we better start thinking about it. Those who fear a coming holocaust (see p. 68) had best think about pulling some part of mankind through, with some part of what he used to have.

* See T.H. Nelson, The Snunking of the Heart: On the Psychology of Puns and Preterism in Carroll and Others. 1980, unless a decent writing system comes along.

BRANCHING PRESENTATIONAL SYSTEMS— HYPERMEDIA

In recent years a very basic change has occurred in presentational systems of all kinds. We may summarize it under the name branching, slthough there are many variants. Essentially, today's systems for presenting pictures, texts and whatnot can bring you different things automatically depending on what you do. Selection of this type is generally called branching. (I have suggested the generic term hypermedia for presentational media which perform in this (and other) multidimensional ways.)

A number of branching media exist or are possible.

Branching movies or hyperfilms (see nearby).

Branching texts or hypertexts (see nearby).

Branching audio, music, etc.

Branching slide-shows.

Wish we could get into some of that stuff here.

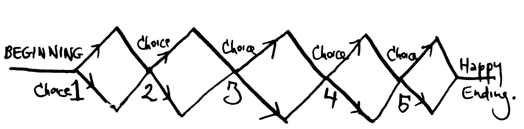
BRANCHING MOVIES

The idea of branching movies is quite exciting. The possibility of it is another thing entirely.

The only system I know of that worked was at the 1967 Montreal World's Fair (Expo 67). At the Czech Pavilion-- you will recall that before the crackdown they had quite a yeasty culture going in Czechoslovakia-- there were some terrific fantic systems going. One was a wall of cubes with slide projectors inside (that rolled toward you and back as they changed their pictures). And then the Movie.

The Czechoslovakian Branching Movie-- I forget its real name-- had the audience vote on what was to happen next at a number of different junctures. What should she do now, what will he do next, etc. And lo and behold! after they had voted, the lights went down, and that's what would happen next. People agreed that this gave the movie a special immediacy.

I never saw the movie-- I waited in line several hours but the line was too long to get into the last showing. So instead I went backstage and talked to Radusz Cincera, who worked out the system. It turns out that it didn't work quite the way people supposed. A lot of people thought that "all the possibilities" had been filmed in advance. Actually, there were always only two possibilities, and no matter what the audience had chosen, somehow the film was plotted to come down to the same next choice anyway:



In the actual setup, they simply had two projectors running side by side, with Film A and Film B, and the projectionist would drop an opaque slide in front of whichever wasn't chosen. But Cincera said that audiences almost always chose the same alternatives anyway, so half the movie was hardly ever used...

In the early sixties a movie was making the rounds in which audiences were supposedly allowed to vote on the ending-- "Mr. Sardonicus," I believe it was called. From the ads it seemed that audiences would be polled as to which last reel to show. Whether the villain was to get his comeuppance, or whatever.

Then there was that Panacolor cartridge projector, mentioned elsewhere, which would have allowed choices by the user

More recently there's the CMX system, also mentioned elsewhere. This is a setup, being jointly marketed by CBS and Memorex, for computer-controlled movie editing. But actually it could also be used as a branching movie system. Essentially the movie itself is stored frame-by-frame (as video) on big disks, made by Memorex; and, under computer control, the output can be switched rapidly among the frames, effectively showing the stored movies. (To my knowledge, the video networks haven't yet recognized the possibilities of this.)

The only trouble is, it's extremely expensive (half a million?), it has an exact storage capacity limited by the number of disk tracks (presumably one track per frame)-- perhaps five minutes total one one big unit, but you can buy more-- and it can only give its full performance to one viewer at a time.) (Or to the whole network, live.)

It may be that the most practical branching movie system would be a cartridge movie viewer and a big stack of cartridges. When you make your choice, change the cartridge. But of course that's not as much fun as having it happen automatically.

REALITY IS OBSOLETE

The idea that objective reality is perceived by our senses, is an obsolete concept. Old truisms like "seeing is believing", become much less believable as we become more aware that, the biological machinery of life itself, transforms images of the physical world before we are made conscious of them. These biological mechanisms share many similarities in principle and in application, to other mechanisms observed in the natural environment and those invented for our own use. Since we are becoming more aware of the nature of perception and those mechanisms involved, now is the time to gain control of ourselves and share more discretion in the operation of our own biological machinery. We have entered the age of hyper-reality.

Day-to-day living provides only a limited variety of physical stimulus, and little incentive to manipulate the physiological and psychological processing involved. Man's historical preoccupation with the need to maintain constant images of the physical world, is a product of his extreme orientation toward physical survival in a hostile environment. The current evolving society of leisure orientations removes this need for constant images and thereby enhances the opportunities for a more complete use of the sensory apparatus and those related brain functions. Many have turned to drugs or meditation. More specifically it is proposed here, that modern communications technology be employed as a "vehicle of departure" from this need for constant images, to bring about a more complete use of the human technology itself. Hyper-reality is the employment of technology other than the biological machinery, when used to affect the performance of the biological machinery beyond its own limitations. This is almost like making adjustments on a television set, except you are what's plugged in, and the controls are outside your body, being part of whatever technology is interfaced to the body itself. As part of such a man-machine interface you could extend your own mental processes, or if you should choose, you could just diddle with the dials. Hyper-reality is an opportunity to enhance the various qualities of the human experience. Reality is obsolete.

A -- *How Wachspress (see p. DM 6)*

COPYRIGHT 1973 AUDITAC, LTD.

!GREBNETUG

Now, in our time, we are turning Gutenberg around. The technology of movable type created certain structures and practices around the written word. Now the technology of computer screen displays make possible almost any structures and practices you can imagine for the written word.

So now what?

For new forms of written communication among people who know each other, jump to "Engelbart" piece, nearby.

To learn about new forms of multidimensional documents for computer screens, jump to "Hypertexts."

Or just feel free to browse.

HYPERTEXT

By "hypertext" I mean non-sequential writing.

Ordinary writing is sequential for two reasons. First, it grew out of speech and speech-making, which have to be sequential; and second, because books are not convenient to read except in a sequence.

But the structures of ideas are not sequential. They tie together every whichway. And when we write, we are always trying to tie things together in non-sequential ways (see p.DM 42). The footnote is a break from sequence; but it cannot really be extended (though some, like Will Cuppy, have toyed with the technique).

I have run into perhaps a dozen people who understood this instantly when I talked to them about it. Most people, however, act more bemused, thinking I'm trying to tell them something technical or pointlessly philosophical. It's not pointless at all: the point is, writers do better if they don't have to write in sequence (but may create multiple structures, branches and alternatives), and readers do better if they don't have to read in sequence, but may establish impressions, jump around, and try different pathways until they find the ones they want to study most closely.

(The astute reader, and anybody who's gotten to this point must be, will have noticed that this book is in "magazine" layout, organized visually by ideas and meanings, for that precise reason. I will be interested to hear whether that has worked.)

And the pity of it is that (like the man in the French play who was surprised to learn that he had been "speaking prose all his life and never known it"), we've been speaking hypertext all our lives and never known it.

Now, many writers have tried to break away from sequence. I think of Nabokov's Pale Fire, of Tristram Shandy and an odd novel of Lazaro Cortazar called Hopscotch, made up of sections ending with numbers telling you where you can branch to. There are many more; and large books generally use many tricks to get around the problem of indexing and reviewing what has and hasn't been said or done already.

However, in my view, a new day is dawning. Computer storage and screen display mean that we no longer have to have things in sequence; totally arbitrary structures are possible, and I think that after we've tried them enough people will see how desirable they are.

TYPES OF HYPERTEXT

Let's assume that you have a high-power display-- and storage displays won't do, because you have to see things move in order to understand where they come from and what they mean. (Especially text.) So it has to be a refreshed CRT.

Basic or chunk style hypertext offers choices, either as footnote-markers (like asterisks) or labels at the end of a chunk. Whatever you point at then comes to the screen.

Collateral hypertext means compound annotations or parallel text (see p. DM42).

Stretchtext changes continuously. This requires very unusual techniques (see p. DM19), but exemplifies how "continuous" hypertext might work.

Ideally, chunk and continuous and collateral hypertext could all be combined (and in turn collaterally linked; see "Thinkertoys," p. DM52).

A "fresh" or "specific" hypertext-- I don't have a better term at the moment-- would consist of material especially written for some purpose. An anthological hypertext, however, would consist of materials brought together from all over, like an anthological book.

A grand hypertext, then, folks, would be a hypertext consisting of "everything" written about a subject, or vaguely relevant to it, tied together by editors (and NOT by "programmers," dammit), in which you may read in all the directions you wish to pursue. There can be alternative pathways for people who think different ways. People who have to have one thing explained to them at a time-- many have insisted to me that this is normal, although I contend that it is a pathological condition-- may have that; others, learning like true human beings, may gather and sift impressions until the ideas become clear.

And then, of course, you see the real dream.

The real dream is for "everything" to be in the hypertext.

Everything you read, you read from the screen (and can always get back to right away); everything you write, you write at the screen (and can cross-link to whatever you read; see Canons, p. DM58).

Paper moulders. Microfilm is inconvenient. In the best libraries it takes at least minutes to get a particular thing. But as to linking them together-- footnoting Aeschylus with Marcus Aurelius, linking genetic data to 15th-century accounts of Indian tribes-- well, you can only do it on paper by writing something new that ties them together. Isn't that ridiculous? When you could do it all electronically in seconds?

Now that we have all these wonderful devices, it should be the goal of society to put them in the service of truth and learning. And this is the way I propose. Not through obscure forms of "information retrieval;" not through newly oppressive forms of "computer-assisted instruction;" and not through a purported science of "artifical intelligence" that will create new personalisms to irk us. All these obstructive oddities, I think, have developed as separate ideals because of the grand preposterosity of Professionalism that has created a world-wide cult of mutual incomprehensibility and disconnected special goals. Now we need to get everybody together again. We want to go back to the roots of our civilization-- the ability, which we once had, for everybody who could read to be able to read everything. We must once again become a community of common access to a shared heritage.

This was of course what Vannevar Bush said in 1945 (see box on right), in an article everybody cites but nobody reads.

The hypertext solution in many ways obviates some of these other approaches, and in addition retains and puts back together the great traditions of literature and scholarship, traditions based on the fact that dividing things up arbitrarily just generally doesn't work.

EVERYTHING IS DEEPLY INTERTWINGLED.

(The only way in which my views differ with those of Engelbart and Pask, I think is in the matter of structure and hierarchy. Both men generally assume that whatever natural hierarchy may exist in particular subjects needs to be accentuated; I hold that all structures must be treated as totally arbitrary, and any hierarchies we find are interesting accidents.)

CAN IT BE DONE?

I dunno.

Licklider, one of computerdom's Great Men, estimated in 1965 that to handle all text by computer, and bring it out to screens, would cost no more than what we pay for all text handling now. (But of course there is the problem of what to do with the people whose lives are built around paper; that can't be taken up here.)

The people who make big computers say that to get the big disk storage to hold great amounts of text, you have to get their biggest computers. Which is a laugh and a half. One IBM-style computer person pompously told me that for large-scale text handling the only appropriate machine was an IBM 360/67 (a shamefully large computer). Such people seem not to understand about minicomputers or the potential of minicomputer networks-- using, of course, big disks.

There are of course questions of reliability, of "big brother" (see Canons, p.), and so on. But I think these matters can be handled.

The key is that people will pay for it. I am sure that if we can bring the cost down to two dollars an hour-- one for the local machine (more than a "terminal"), one for the material (including storage, transmission and copyrights)-- there's a big, big market. (And that's what the Xanadu network is about; see p. DM57.) My assumption is that the way to do this is not through big business (since all these corporations can see is other corporations); not through government (hypertext is not committee-oriented, but individualistic-- and grants can only be gotten through sesquipedalian and obfuscatory pompizzazz); but through the byways of the private enterprise system. I think the same spirit that gave us McDonald's and kandy kolor hot rod accessories may pull us through here. (See Xanadu Network, p. DM57.)

Obviously, putting man's entire heritage into a hypertext is going to take awhile. But it can and should be done.

BIBLIOGRAPHY

Theodor H. Nelson, "The Hypertext." Proc. World Documentation Federation, 1965.

COULDN'T HAVE HYPERTEXT NOVELS, YOU SAY?

Consider the hypertext character of--
- Tristram Shandy, by Sterne.
- Spoon River Anthology, by Masters.
- Hopscotch, by Cortazar.
- Pale Fire, by Nabokov.
- Remembrance of Things Past, by Proust.

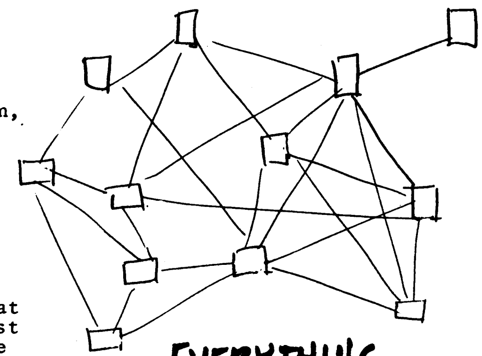
And, surprisingly, hypertext actually FIGURES IN Giles Goat-Boy, by Barth.

GLINDA'S MAGIC BOOK

Glinda the Good, gentle sorceress of the southern quadrant of the land of Oz-- not the flaphead portrayed by Billie Burke in the Goldwynized film-- has a Magic Book in which Everything That Happens is written.

The question, of course, is how it's chosen.

You can only watch news tickers for a short time before getting very bored.



EVERYTHING IS DEEPLY INTERTWINGLED.

In an important sense there are no "subjects" at all; there is only all knowledge, since the cross-connections among the myriad topics of this world simply cannot be divided up neatly.

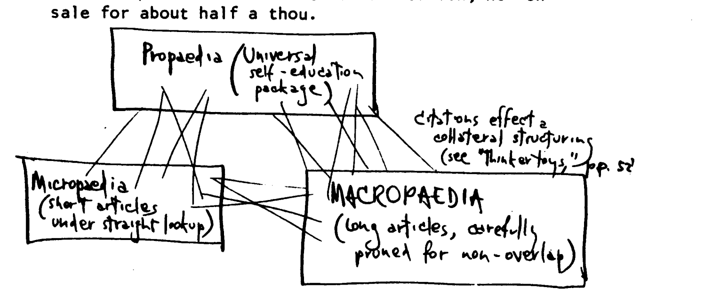
Hypertext at last offers the possibility of representing and exploring it all without carving it up destructively.

Arthur C. Clarke wrote a book entitled The Lost Worlds of 2001 (Signet, 1972), about the variants and alternatives of that story that did not find their way to the screen.

In a hypertext version, we could look at them all in context, in collateral views, and see the related variants-- with annotations.

Mortimer J. Adler, the man who reduced all of Western Culture to a few Great Books plus an index under his own categories, has now Addled the Encyclopedia Britannica.

Since 1965 he has been creating Britannica 3, the venturesome and innovative new version, now on sale for about half a thou.



Britannica 3 is basically a 3-level hypertext, made to fit on printed pages by the strictures of Adler's editing (according to Newsweek, some 200 authors withdrew their work rather than submit to the kind of restrictions he was imposing).

The idea may be basically good, even though the sesquipaedalian titles may impaed the raeder.

THE BURNING BUSH

In fact hypertexts were foreseen very clearly in 1945 by Vannevar Bush, Roosevelt's science advisor. When the war was in the bag, he published a little article on various groovy things that had become possible by that time.

"As We May Think" (Atlantic Monthly, July, 1945) is most notable for its clear description of various hypertext techniques-- that is, linkages between documents which may be brought rapidly to the screen according to their linkages. (So what if he thought they'd be on microfilm.)

How characteristic of Professionalism. Bush's article has been taken as the starting point for the field of Information Retrieval (see p.), but its actual contents have been ignored by acclamation. Information Retrieval folk have mostly done very different things, yet thought they were in the tradition.

Now people are "rediscovering" the article. If there's another edition of this book I hope I can run it in entirety.

TYPES OF HYPERTEXT

Let's assume that you have a high-power display-- and storage displays won't do, because you have to see things move in order to understand where they come from and what they mean. (Especially text.) So it has to be a refreshed CRT.

Basic or chunk style hypertext offers choices, either as footnote-markers (like asterisks) or labels at the end of a chunk. Whatever you point at then comes to the screen.

Collateral hypertext means compound annotations or parallel text (see p. DM42).

Stretchtext changes continuously. This requires very unusual techniques (see p. DM19), but exemplifies how "continuous" hypertext might work.

Ideally, chunk and continuous and collateral hypertext could all be combined (and in turn collaterally linked; see "Thinkertoys," p. DM52).

A "fresh" or "specific" hypertext-- I don't have a better term at the moment-- would consist of material especially written for some purpose. An anthological hypertext, however, would consist of materials brought together from all over, like an anthological book.

A grand hypertext, then, folks, would be a hypertext consisting of "everything" written about a subject, or vaguely relevant to it, tied together by editors (and NOT by "programmers," dammit), in which you may read in all the directions you wish to pursue. There can be alternative pathways for people who think different ways. People who have to have one thing explained to them at a time-- many have insisted to me that this is normal, although I contend that it is a pathological condition-- may have that; others, learning like true human beings, may gather and sift impressions until the ideas become clear.

And then, of course, you see the real dream.

The real dream is for "everything" to be in the hypertext.

Everything you read, you read from the screen (and can always get back to right away); everything you write, you write at the screen (and can cross-link to whatever you read; see Canons, p. DM58).

Paper moulders. Microfilm is inconvenient. In the best libraries it takes at least minutes to get a particular thing. But as to linking them together-- footnoting Aeschylus with Marcus Aurelius, linking genetic data to 15th-century accounts of Indian tribes-- well, you can only do it on paper by writing something new that ties them together. Isn't that ridiculous? When you could do it all electronically in seconds?

Now that we have all these wonderful devices, it should be the goal of society to put them in the service of truth and learning. And this is the way I propose. Not through obscure forms of "information retrieval;" not through newly oppressive forms of "computer-assisted instruction;" and not through a purported science of "artifical intelligence" that will create new personalisms to irk us. All these obstructive oddities, I think, have developed as separate ideals because of the grand preposterosity of Professionalism that has created a world-wide cult of mutual incomprehensibility and disconnected special goals. Now we need to get everybody together again. We want to go back to the roots of our civilization-- the ability, which we once had, for everybody who could read to be able to read everything. We must once again become a community of common access to a shared heritage.

This was of course what Vannevar Bush said in 1945 (see box at right), in an article everybody cites but nobody reads.

The hypertext solution in many ways obviates some of these other approaches, and in addition retains and puts back together the great traditions of literature and scholarship, traditions based on the fact that dividing things up arbitrarily just generally doesn't work.

EVERYTHING IS DEEPLY INTERTWINGLED.

(The only way in which my views differ with those of Engelbart and Pask, I think is in the matter of structure and hierarchy. Both men generally assume that whatever natural hierarchy may exist in particular subjects needs to be accentuated; I hold that all structures must be treated as totally arbitrary, and any hierarchies we find are interesting accidents.)

CAN IT BE DONE?

I dunno.

Licklider, one of computerdom's Great Men, estimated in 1965 that to handle all text by computer, and bring it out to screens, would cost no more than what we pay for all text handling now. (But of course there is the problem of what to do with the people whose lives are built around paper; that can't be taken up here.)

The people who make big computers say that to get the big disk storage to hold great amounts of text, you have to get their biggest computers. Which is a laugh and a half. One IBM-style computer person pompously told me that for large-scale text handling the only appropriate machine was an IBM 360/67 (a shamefully large computer). Such people seem not to understand about minicomputers or the potential of minicomputer networks-- using, of course, big disks.

There are of course questions of reliability, of "big brother" (see Canons, p.), and so on. But I think these matters can be handled.

The key is that people will pay for it. I am sure that if we can bring the cost down to two dollars an hour-- one for the local machine (more than a "terminal"), one for the material (including storage, transmission and copyrights)-- there's a big, big market. (And that's what the Xanadu network is about; see p. DM57.) My assumption is that the way to do this is not through big business (since all these corporations can see is other corporations); not through government (hypertext is not committee-oriented, but individualistic-- and grants can only be gotten through sesquipedalian and obfuscatory pompizzazz); but through the byways of the private enterprise system. I think the same spirit that gave us McDonald's and kandy kolor hot rod accessories may pull us through here. (See Xanadu Network, p. DM57.)

Obviously, putting man's entire heritage into a hypertext is going to take awhile. But it can and should be done.

BIBLIOGRAPHY

Theodor H. Nelson, "The Hypertext." Proc. World Documentation Federation, 1965.

COULDN'T HAVE HYPERTEXT NOVELS, YOU SAY?

Consider the hypertext character of--
- Tristram Shandy, by Sterne.
- Spoon River Anthology, by Masters.
- Hopscotch, by Cortazar.
- Pale Fire, by Nabokov.
- Remembrance of Things Past, by Proust.

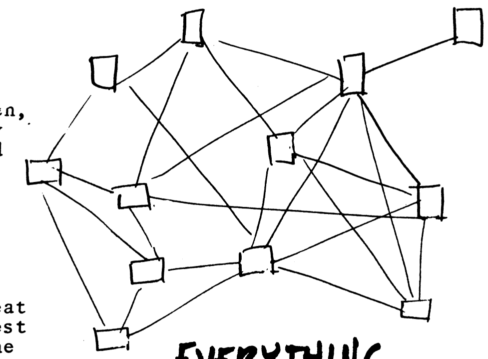
And, surprisingly, hypertext actually FIGURES IN Giles Goat-Boy, by Barth.

GLINDA'S MAGIC BOOK

Glinda the Good, gentle sorceress of the southern quadrant of the land of Oz-- not the flaphead portrayed by Billie Burke in the Goldwynized film-- has a Magic Book in which Everything That Happens Is written.

The question, of course, is how it's chosen.

You can only watch news tickers for a short time before getting very bored.



EVERYTHING IS DEEPLY INTERTWINGLED.

In an important sense there are no "subjects" at all; there is only all knowledge, since the cross-connections among the myriad topics of this world simply cannot be divided up neatly.

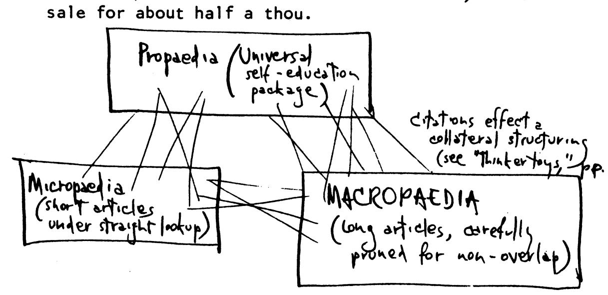
Hypertext at last offers the possibility of representing and exploring it all without carving it up destructively.

Arthur C. Clarke wrote a book entitled The Lost Worlds of 2001 (Signet, 1972), about the variants and alternatives of that story that did not find their way to the screen.

In a hypertext version, we could look at them all in context, in collateral views, and see the related variants-- with annotations.

Mortimer J. Adler, the man who reduced all of Western Culture to a few Great Books plus an index under his own categories, has now Addled the Encyclopedia Britannica.

Since 1965 he has been creating Britannica 3, the venturesome and innovative new version, now on sale for about half a thou.



Britannica 3 is basically a 3-level hypertext, made to fit on printed pages by the strictures of Adler's editing (according to Newsweek, some 200 authors withdrew their work rather than submit to the kind of restrictions he was imposing).

The idea may be basically good, even though the sesquipaedalian titles may impaed the raeder.

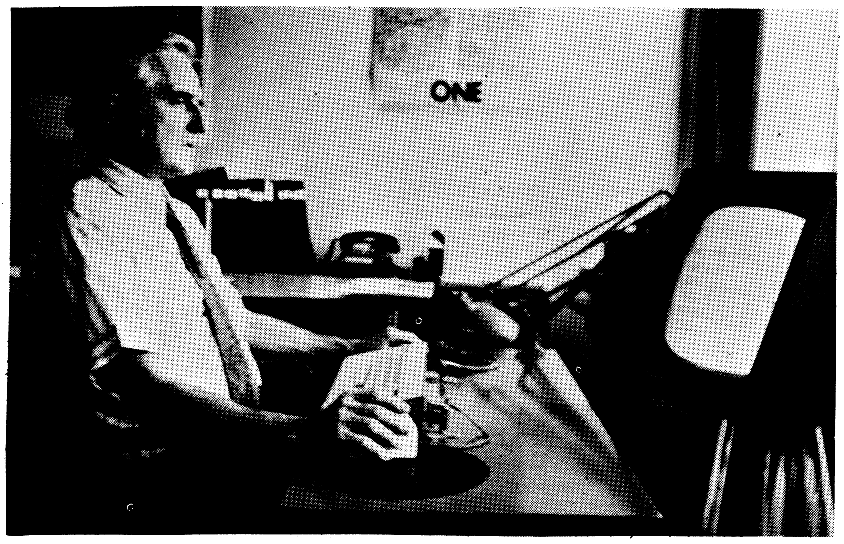
THE BURNING BUSH

In fact hypertexts were foreseen very clearly in 1945 by Vannevar Bush, Roosevelt's science advisor. When the war was in the bag, he published a little article on various groovy things that had become possible by that time.

"As We May Think" (Atlantic Monthly, July, 1945) is most notable for its clear description of various hypertext techniques-- that is, linkages between documents which may be brought rapidly to the screen according to their linkages. (So what if he thought they'd be on microfilm.)

How characteristic of Professionalism. Bush's article has been taken as the starting point for the field of Information Retrieval (see p.), but its actual contents have been ignored by acclamation. Information Retrieval folk have mostly done very different things, yet thought they were in the tradition.

Now people are "rediscovering" the article. If there's another edition of this book I hope I can run it in entirety.



DOUG ENGELBART AND "THE AUGMENTATION OF INTELLECT"

Douglas Engelbart is a saintly man at Stanford Research Institute whose dream has been to make people smarter and bring them together. His system, on which millions of dollars have been spent, is a wonder and a glory.

He began as an engineer of CRTs (see "Lightning in a Bottle," p.DM6); but his driving thought was, quite correctly, that these remarkable objects could be used to expand man's mind and improve each shining hour.

Doug Engelbart's vision has never been restricted to narrow technical issues. From the beginning his concern was not merely to plank people down at display consoles, but in the most profound sense to expand man's mind. "The Augmentation of Human Intellect," he calls it, by which he means making minds work better by giving them better tools to work with.

An obvious example is writing: before people could write things down, men could only learn what they experienced or were told by others in person; writing changed all that. Within the computer-screen fraternity, the next step is obvious; screens can double and redouble our intellectual capacities. But this is not obvious to everybody. Engelbart, patiently instructing those outside, came up with a beautiful example. To show what he meant by the Augmentation of Intellect, Engelbart tied a pencil to a brick. Then he actually made someone write with it. The result, which was of course dreadful, Engelbart solemnly put into a published report. Not yet being able to demonstrate the augmentation of intellect, since he had as yet no system to show off, he had masterfully demonstrated the disaugmentation of intellect: what happens if you make man's tools for working out his thoughts worse instead of better. As this poor guy was with his brickified pencil, explained Engelbart, so are we all among our bothersome, inflexible systems of paper.

Starting small, Engelbart programmed up a small version of what most fans call "The Engelbart System" some ten years ago. One version has it that when it came to looking for grants, management thought he acted too kooky, and so assigned a Front Man to make the presentation. But, as the story goes, the man from ARPA (see "Military...", p. 59) pointed at Engelbart and said, "We want to back him."

A small but dedicated group at SRI has built up a system from scratch. First they used little CDC 1700 minicomputers; then, various grants later, they were able to set up their own PDP-10, in which the system now resides, and from which it reaches out across the country.

Doug calls his system NLS, or "oN-Line System." Basically it is a highly responsive, deeply-structured text system, feeding out to display terminals. From a terminal you may read anything you or others have written, and write with as-yet-unmatched flexibility.

The display terminals are all over. The project has gone national, though at great expense: through the ARPA net of computers, you can in principle become a user of NLS for something like $50,000 a year.

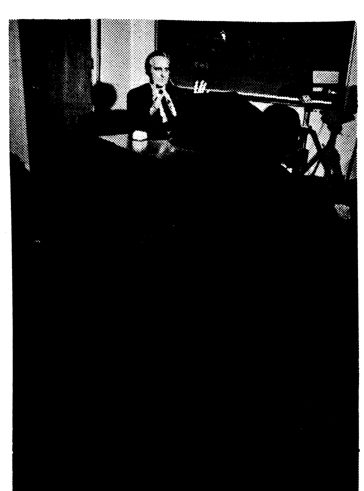
THE "KNOWLEDGE WORKSHOP"

For a lucky fifty or so people, Engelbart's system is Home. Wherever they are-- at Stanford Research Institute or far away on the ARPAnet-- a whole world of secretarial and communication services is at their fingertips. The user has but to call up through his display terminal and log on. At that point all his written files, and numerous files shared among the users, are at his fingertips. He may read, write, annotate the cross-link. (Engelbart's system has provision for collateral structuring: see "Thinkertoys," p.DM52.) He may send messages to others in the Workshop. He may open certain of his files to other people, and read those that have been opened to him.

This all has a certain vagueness if you do not understand how bound you are today by paper-- the problems of finding it, sorting it, looking things up. (If you write, that is, write a lot, you know all too well how intractable is paper, what a damned nuisance.) With a system like Engelbart's, now, whatever is written is instantly there. Whatever you want to look up is instantly there, simultaneously interconnected to everything else it should be connected to-- source materials, footnotes, comments and so on. A document is completed the moment it is written: no human being has to retype it. (It need not be typed on paper at all, if it's just for the workshop members: a printout is only needed if it has to go to someone outside the system.)

In many ways, Engelbart's system is a prototype for the world of the future, I hope. ALL HANDLING OF PAPER IS ELIMINATED. Whatever you write, you write on the screens with keyboard and pointer. (No more backs of envelopes, yellow pads, file cards, typewriters.) Whatever you transmit to fellow users of the system you simply 'release'-- no physical paper changes hands.

The group has also worked out some remarkable techniques for collaborative endeavor. Two people-- say, one in California and one in New York-- can work together through their screens, plus a phone link; it's as if they were side-by-side at a magic table. Each sees on his screen what the other sees; each controls a moving dot (or "cursor") that shows where he's pointing. The effect is somewhere between a blackboard and a desk; both may call up documents, point things out in them, change them, and anything else two people might do when working on something together.



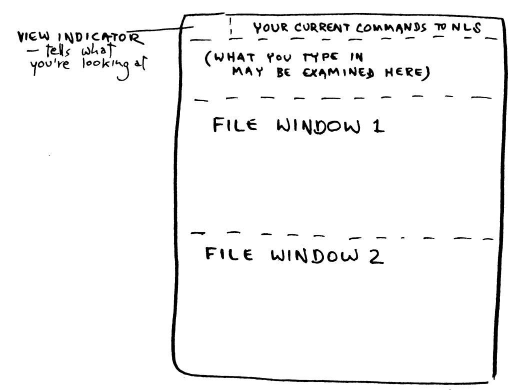
Engelbart meets with someone far away, as others watch.

THE SYSTEM ITSELF

Basically the system is a large-scale setup for the storage, bringing forth, viewing and revision of documents and connections among them.

The documents are stored (of course) in alphabetical codes. Connections among them, or other relations within them, are signalled by the presence of other codes within them; these are ordinarily not displayed, however, except as directed by a particular display program and display programs can of course vary.

There are various programs for display, in large part depending on what sort of screen system the individual user has. (NLS is used with everything from high-resolution line-drawing screens converted to 1000-line television, down to inexpensive Delta Data terminals-- a brand, incidentally, that allows text motion, which most don't.) Engelbart's system is extremely general, allowing the creation of files having all kinds of structures, and display programs in all kinds of styles. (I hope that this side of the present book conveys a sense of how many styles that can be.) However, most users are devoted to certain standardized styles of working that have been well worked out and permit the easy sharing of material and of operating practices. Here, for instance, is their standard screen layout:



Two separate panels of text appear, and links may be shown on them. (Thus it's a thinkertoy-- see p. .) Two little windows at the top remind you of what you're seeing and what you're asking for. We can't get into the rest of it here.

THE COMMAND LANGUAGE

NLS has a command language which all users must learn. While it is a streamlined and straightforward command language, nevertheless it requires the user to type in a specific sequence of alphabetical characters every time he wants something done. (This is acceptable to computer-oriented people; I suspect it would not be satisfactory, say, for philosophers and novelists. For designs oriented to such users, see JOT (p.DM50) and Carmody's System, nearby, Parallel Textface (p.DM53) and Th3 (p.DM55).)

Incidentally, NLS users may also employ a cute little keyboard, something like a kalimba, that allows you to type with one hand. You simply type the six significant ASCII bits (see chart p. 28) in one "chord" -- it sounds hard but is easy to learn.

Sample commands: I (insert), D (delete), M (move or rearrange). Then you point with the mouse.

MOUSE?

The Engelbart Folks have built a pointing device, for telling the system where you're pointing on the screen, that is considerably faster and handier than a lightpen. (Unfortunately, I don't believe it's commercially available.) It's called The Mouse.

The Engelbart Mouse is a little box with hidden wheels underneath and a cable to the terminal. As you roll it, the wheel's turns are signalled to the computer and the computer moves the cursor on the screen. It's fast and accurate, and in fact beats a lightpen hands down in working speed.

Through the command language, NLS allows users to create programs that respond in all sorts of ways; thus the fact that certain text-handling styles are standard (as in above illustration of screen layout) results more from tradition than necessity.

A VERY BASIC HYPERTEXT SYSTEM

Hypertext is non-sequential writing. It's no good to us, though, unless we can go instantly in a choice of directions from a given point.

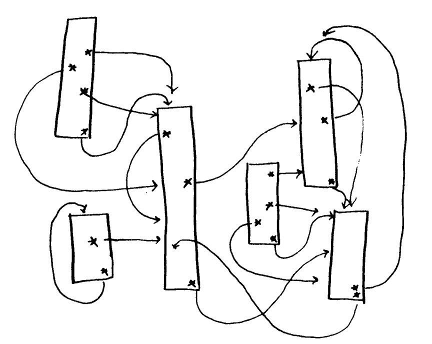
This of course can only mean on computer display screens.

Engelbart's system, now, was mainly designed for people who wanted to immerse themselves in it and learn its conventions. Indeed, it might be said to have been designed for a community of people in close contact, a sort of system of blackboards and collaborative talking papers.

A more elemental system, with a different slant, was put together at Brown U. on IBM equipment. We'll refer to it here as "Carmody's System," after the young programmer whose name came first on the writeup.

Carmody's system runs on an IBM 360 with 2250 display. While the 2250 is a fine piece of equipment, the quirks of the 360's operating system (see p. 45) often delay the user by making him wait, e.g., for someone else's cards to get punched before it responds to his more immediate uses; this is like making ice-skaters wait for oxcarts.

Anyway, the system essentially imposes no structure on the material; it may consist of text segments of any length and ties and links between them. An asterisk appearing anywhere in one piece of text signals a possible jump, but the reader doesn't necessarily know where to; zapping the asterisk with the lightpen takes you there, however.



This is stark and simple. It could also get you good and lost. However, a simple technique took care of that: everytime the user jumped, the address of his previous location was saved on a stack (see "The Magic of the Stack," p. 42). The user also had a RETURN button: when he wanted to go back to where he had last jumped from, the system would pop the last address off the top of the stack, and take him there. (This feature was adapted from my 1967 Stretchtext paper, and turned out to work out quite well in practice.)

The system also had handy features for light-pen text editing, and various nice printout techniques. All told, it was a clean and powerful design. While it lacked higher-level visualization facilities, like Engelbart's display of Levels (see "outline" in Engelbart article) or collateral display (see "Thinkertoys," p.DM52), it was in some ways suited for naive users; that is, it was eventually fairly safe to use, and could in large part be taught to rank beginners in a couple of hours-- provided they didn't have to know about JCL cards.

It is left for the reader to figure out interesting uses for it. How would you do collateral structures? How could you signal to a reader which of several pieces of text a jump was to?

(At least one real hypertext was actually written on this system. It tied together a lot of patents for multilayer electrodes. Readers agreed that they could learn more from it about multilayer electrodes than they had imagined wanting to know.)

BIBLIOGRAPHY

Steven Carmody, Walter Gross, Theodor H. Nelson, David Rice and Andries van Dam, "A Hypertext Editing System for the 360." In Faiman and Nievergelt (eds.), Pertinent Concepts in Computer Graphics (U. Ill. Press '69), 291-330.
Note: Mr. Gross now goes by the name of Lightning Clearwater.

The same apparently is true of the data structure. I used to be somewhat disturbed at the way Engelbart's text systems seem to be rigorously hierarchical. This in fact is the case, in the sense that having multiple discrete levels is built deep into the system. But it turns out to be harmless. The stored text is divided by the storage techniques into multiple levels, corresponding to a Harvard outline. Think of it as something like this:

1. HIERARCHICAL FORMAT
   - A. STORAGE
   - B. DISPLAY
   - C. LANGUAGE

But let's expand this example a little:

1. HIERARCHICAL FORMAT
   - A. STORAGE
     - A1. Everything in NLS is stored with hierarchical codes.
     - A2. Their effect depends on the display.
   - B. DISPLAY
     - B1. The hierarchical codes of NLS have no consequences in particular.
     - B2. The hierarchical codes for NLS can splay the material out into a variety of display arrangements.
       - B2A. They can be displayed in outline form.
       - B2B. They can be displayed in normal text form.
       - B2C. These dratted numbers can even be made to disappear.
   - C. LANGUAGE
     - C1. The command language determines what the display shows of the hierarchical structure.
     - C2. What is shown can be determined by a program in the command language. (For instance, "how many levels down" it is being shown).
       - C2A. This is four levels down. (The earlier example wasn't.)
     - C3. The display format all depends on what display program you use, in the NLS command language.

That's enough of that. I can't help remarking that I still don't like that sort of structuring, but it is deep in NLS, and if you don't like it either (poor deprived lucky user of NLS) you can program it to disappear, so it's hardly in your way.

BY THE BEARD OF THE PROPHET!

Engelbart in German means Angelbeard; Doug Engelbart is indeed on the side of the angels. In building his mighty system he points a new way for humanity. The sooner the better. Any history of the twentieth century will certainly hold him high. Few great men are also such nice guys.

BIBLIOGRAPHY

Douglas C. Engelbart, Richard W. Watson & James C. Norton, "The Augmented Knowledge Workshop." Proc NCC 73, 9-21.

Charles H. Irby, "Display Techniques for Interactive Text Manipulation." Proc NCC 74, 247-255.

GORDON PASK RETURNS [DIFFICULT SECTION]

This continues the remarks on Gordon Pask begun on p.

I will now try to describe Pask's work as he has explained it to me. Perhaps this will be of some help to those who may have been mystified or dumfounded by contact with this fabulous man.

Gordon Pask's concern is abstraction and how concepts are formed, whether in a creature of nature or a robot or a computer program. Abstraction is of interest primordially (as life evolved thinking capacity), psychogenetically (as the mind acquires new facilities, described most peculiarly by Piaget), and epistemologically (how do we know? Like, how do we know, man?), and methodologically (how can we most effectively find out more?).

His interest, then, is in teaching by allowing students to discover exact relations in a specific subject matter by the very process of abstraction that is of so much interest.

What he does, then, is prepare given fields of learning so that they can be studied by students using abstractive methods, without guidance.

This preparation basically has two steps. First he sets up the whole field. This is done in collaboration with a "subject matter expert," who names the important topics in the field and states what interconnections they have. The result is a complex graph structure (see p. 26) which Pask calls a conversational domain. It comes out to huge diagrams of labels and lines between them.

Then Pask processes this structure to make a more usable map of the field that he calls an entailment structure. The processing basically involves removing "cycles" in the graph, thus making the structure hierarchical in a slightly artificial way justified by what the subject-matter-expert has said is the structure of the field.

(This processing is carried out by a program called EXTEND.)

The resulting Entailment Structure is then presented to the student as a great map of the field which he may explore.

Pask intends that the student's explorations will consist of testing analogies, or what Pask calls morphisms, to find the exact structures of knowledge he is supposed to be acquiring. This knowledge will be in the form of isomorphisms, or exact analogies, i.e. laws.

Pask's overall system, examples of which he has running in his laboratory in England, he calls CASTE (Course Assembly System and Tutorial Environment). A further development, which is to be put on a PDP-11/45 computer (see p. 36 and p. 42) at the Brooklyn Children's Museum, is called THOUGHT-STICKER. This program is intended to allow the demonstration and testing of analogies directly, by children.

PASK AND HYPERTEXT

Gordon Pask's work is remarkably similar to my own stuff on hypertext.

Essentially Pask is reducing a field to an extremely formal structure of relations which may then be studied by the student, at the student's initiative.

(What I don't quite understand is how the analogies are to be explored and tested.)

Anyway, a principal point is that the student is in control and may use his initiative dynamically; the subject is not artificially processed into a presentational sequence. Moreover, the arbitrary interconnections of the subject, which are no respecters of the printed page, are recognized as the fundamental structures the student must deal with and come to understand. On all these points Pask and I are in total agreement.

Indeed, his explorable systems-- (I don't know if they will be what I elsewhere call hypergrams or responding resources)-- will be fascinating, fun and terrifically educational. Because he is.

Now it turns out that this exactly complements the notion of hypertext as I have been promulgating it lo these many years.

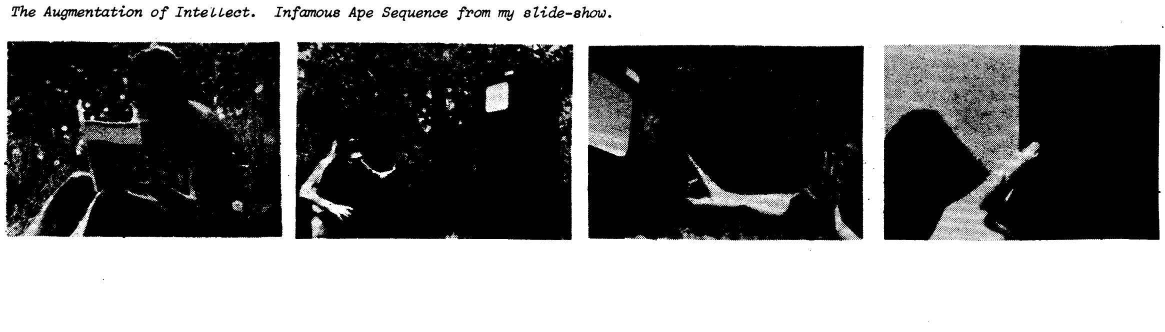
Hypertext is non-sequential text. If we write a hypertext on something, it will be most appropriate if we give it the general interconnective structure of the field. In other words, the interconnective structures chosen for the textual parts are likely to have the same connective structure (in general) as Pask's Entailment Structure.

For another kind of hypertext, the anthological hypertext built up of lots of other writings, it is also reasonable to expect the connective structures to cluster to the same general form as Pask's entailment structure.

In other words, the very same field of knowledge Pask is out to represent as an explorable, formalized whole, I am out to represent as an explorable informalized whole, with anecdotes, jokes, cartoons, "enrichment materials," and anything else people might dig.

In still other words, let's have both and call it a party.

The Augmentation of Intellect. Infamous Ape Sequence from my slide-show.



Actually it needs the '2001' music.

You can't read the screen here. It says: COGITO ERGO SUM $e=mc^2$ Call me Ishmael.

It really needs the music.

FEELED-EFFECT SYSTEMS ARE THE NEW FRONTIER

FANTICS

— BUT IT'S SHOWMANSHIP THAT'S PARAMOUNT, NOT ANY TECHNICAL SPECIALTY

Ah, Love! could you and I with Him conspire
To grasp this sorry Scheme of Things entire,
Would not we shatter it to bits—and then
Re-mould it nearer to the Heart's Desire!

Edward Fitzgerald.

Almost everyone seems to agree that Mankind (who?) is on the brink of a revolution in the way information is handled, and that this revolution is to come from some sort of merging of electronic screen presentation and audio-visual technology with branching, interactive computer systems. (The naive think "the" merging is inevitable, as if "the" merging meant anything clear. I used to think that too.)

Professional people seem to think this merging will be an intricate mingling of technical specialties, that our new systems will require work by all kinds of committees and consultants (adding and adjusting) until the Results-- either specific productions or overall Systems-- are finished. Then we will have to Learn to Use Them. More consulting fees.

I think this is a delusion and a con-game. I think that when the real media of the future arrive, the smallest child will know it right away (and perhaps first). That, indeed, should and will be the criterion. When you can't tear a teeny kid away from the computer screen, we'll have gotten there.

We are approaching a screen apocalypse. The author's basic view is that RESPONSIVE COMPUTER DISPLAY SYSTEMS CAN, SHOULD AND WILL RESTRUCTURE AND LIGHT UP THE MENTAL LIFE OF MANKIND. (For a more conventional outlook, see box nearby, "Another Viewpoint.")

I believe computer screens can make people happier, smarter, and better able to cope with the copious problems of tomorrow. But only if we do right, right now.

WHY?

The computer's capability for branching among events, controlling exterior devices, controlling outside events, and mediating in all other events, makes possible a new era of media.

Until now, the mechanical properties of external objects determined what they were to us and how we used them. But henceforth this is arbitrary.

The recognition of that arbitrariness, and reconsideration among broader and more general alternatives, awaits us. All the previous units and mechanisms of learning, scholarship, arts, transaction and confirmation, and even self-reminder, were based in various ways upon physical objects-- the properties of paper, carbon paper, files, books and bookshelves. To read from paper you must move the physical object in front of you. Its contents cannot be made to slide, fold, shrink, become transparent, or get larger.

But all this is now changing, and suddenly. The computer display screen does all these things if desired, to the same markings we have previously handled on paper. The computer display screen is going to become universal very fast; this is guaranteed by the suddenly rising cost of paper. And we will use them for everything. This already happens wherever there are responding computer screen systems. (I have a friend with two CRTs on his desk; one for the normal flow of work, and one to handle interruptions and side excursions.) A lot of forests will be saved.

Now, there are many people who don't like this idea, and huff about various apparent disadvantages of the screen. But we can improve performance until almost everyone is satisfied. For those who say the screens are "too small," we can improve reliability and backup, and offer screens everywhere (so that material need not be physically carried between them).

The exhilaration and excitement of the coming time is hard to convey on paper. Our screen displays will be alive with animation in their separate segments of activity, and will respond to our actions as if alive physically too.

The question is, then: HOW WILL WE USE THEM? Thus the design of screen performances and environments, and of transaction and transmission systems, is of the highest priority.

THE FRENCH HAVE A WORD FOR IT

In French they use the term l'Informatique to mean, approximately, the presentation of information to people by automatic equipment.

Unfortunately the English equivalent, informatics, has been preempted. There is a computer programming firm called Informatics, Inc., and when I wrote them about this in the early sixties they said they did not want their name to become a generic term. Trademark law supports them in this to a certain extent. (Others, like Wally Feurzeig, want that to be the word regardless.) But in the meantime I offer up the term fantics, which is more general anyhow.

MEDIA

What people don't see is how computer technology now makes possible the revision and improvement-- the transformation-- of all our media. It "sounds too technical."

But this is the basic misunderstanding: the fundamental issues are NOT TECHNICAL. To understand this is basically a matter of MEDIA CONSCIOUSNESS, not technical knowledge.

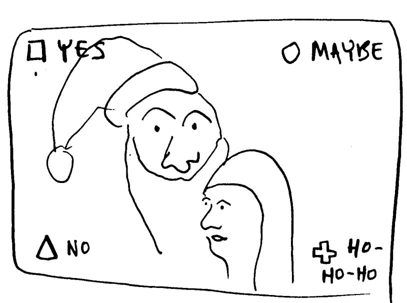
A lot of people have acute media consciousness. But some people, like Pat Buchanan and the communards, suggest that there is something shabby about this. Many think, indeed, that we live in a world of false images promulgated by "media," a situation to be corrected. But this is a misunderstanding. Many images are false or puffy, all right, but it is incorrect to suppose that there is any alternative. Media have evolved from simpler forms, and convey the background ideas of our time, as well as the fads. Media today focus the impressions and ideas that in previous eras were conveyed by rituals, public gatherings, decrees, parades, behavior in public, mummer' troupes...but actually every culture is a world of images. The chieftain in his palanquin, the shaman with his feathers and rattle, are telling us something about themselves and about the continuity of the society and position of the individuals in it.

Now the media, with all their quirks, perform the same function. And if we do not like the way some things are treated by the media, in part this stems from not understanding how they work. "Media," or structured transmission mechanisms, cannot help being personalized by those who run them. (Like everything else.) The problem is to understand how media work, and thus balance our understanding of the things that media misrepresent.

THOUGHTS ABOUT MEDIA:
1. ANYTHING CAN BE SAID IN ANY MEDIUM.

Anything can be said in any medium, and Inspiration counts much more than 'science'. But the techniques which are used to convey something can be quite unpredictable.

Voice of Little Girl:
"Santa, are you more important than God?"
Announcer, plonkingly:
"Your answer is..."



(After How To Be A Department Store Santa Claus, produced by the author for CBS Laboratories and the AVS-10 instructional device. Original slide, starring Michelle Dellinger and Henry Shrady, unfortunately mislaid.)

2. TRANSPOSABILITY

There has always been, but now is newly, a UNITY OF MEDIA OPTIONS. You can get your message across in a play, a tract, a broadside, a textbook, a walking sandwich-board, a radio program, a comic book or fumetti, a movie, a slide-show, a cassette for the Audi-Scan or the AVS-10, or even a hypertext (see p.DM46).

(But transposing can rarely preserve completely the character or quality of the original.)

3. BIG AND SMALL APPROACHES

What few people realize is that big pictures can be conveyed in more powerful ways than they know. The reason they don't know it is that they see the content in the media, and not how the content is being gotten across to them-- that in fact they have been given very big pictures indeed, but don't know it. (I take this point to be the Nickel-Iron Core of McLuhanism.)

People who want to teach in terms of building up from the small to the large, and others who (like the author) like to present a whole picture first, then fill in the gaps, are taking two valid approaches. (We may call these, respectively, the Big Picture approach and the Piecemeal approach.) Big pictures are just as memorable as picky-pieces if they have strong insights at their major intersections.

4. THE WORD-PICTURE CONTINUUM

The arts of writing and diagramming are basically a continuum. In both cases the mental images and cognitive structures produced are a merger of what is heard or received. Words are slow and tricky for presenting a lot of connections; diagrams do this well. But diagrams give a poor feel for things and words do this splendidly. The writer presents exact statements, in an accord-structure of buts and indeeds, molded in a structure of connotations having (if the writer is good) exact impreciseness. This is hardly startling: you're always selecting what to say, and the use of vague words (or the use of precise-sounding words vaguely) is simply a flagrant form of omission. In diagrams, too, the choice of what to leave in and out, how to represent overweening conditions and forces and exemplary details, are highly connotative. (Great diagrams are to be seen in the Scientific American and older issues of TIME magazine.)

This word-picture continuum is just a part of the broader continuum, which I call Fantics.

ANOTHER VIEWPOINT

[from handout, 1974 Nat'l. Joint Comp. Conf.]

John B. Macdonald
Research Leader
Computer Applications: Graphics
Western Electric Company
Engineering Research Center

PROBLEMS, PERILS, AND PROMISES OF COMPUTER GRAPHICS

I would begin with some definitions which may be obvious but bear repeating.

1. Engineering is the application of science for ($) profit,
2. Computer graphics does not make possible anything that was previously impossible: it can only improve the throughput of an existing process,
3. A successful application of computer graphics is when over a period of five years the cost savings from improved process throughput exceed the costs of hardware, software, maintenance and integration into an existing process flow.

FANTICS

By "fantics" I mean the art and science of getting ideas across, both emotionally and cognitively. "Presentation" could be a general word for it. The character of what gets across is always dual: both the explicit structures and feelings that go with them. These two aspects, exactness and connotation, are an inseparable whole; what is conveyed generally has both. The reader or viewer always gets feelings along with information, even when the creators of the information think that its "content" is much more restricted. A beautiful example: ponderous "technical" manuals which carry much more connotatively than the author realizes. Such volumes may convey to some readers an (intended) impression of competence, to others a sense of the authors' obtuseness and non-imagination. Explicit declarative structures nevertheless have connotative fields; people receive not only cognitive structures, but impressions, feelings and senses of things.

Fantics is thus concerned with both the arts of effect-- writing, theater and so on-and the structures and mechanisms of thought, including the various traditions of the scholarly event (article, book, lecture, debate and class). These are all a fundamentally inseparable whole, and technically-oriented people who think that systems to interact with people, or teach, or bring up information, can function on some "technical" basis-- with no tie-ins to human feelings, psychology, or the larger social structure-- are kidding themselves and/or everyone else. Systems for "teaching by computer," "information retrieval," and so on, have to be governed in their design by larger principles than most of these people are willing to deal with: the conveyance of images, impressions and ideas. This is what writers and editors, movie-makers and lecturers, radio announcers and layout people and advertising people are concerned with; and unfortunately computer people tend not to understand it for beans.

In fantics as a whole, then we are concerned with:

1. The art and science of presentation. Thus it naturally includes

2. Techniques of presentation: writing, stage direction, movie making, magazine layout, sound overlay, etc. and of course

3. Media themselves, their analysis and design; and ultimately

4. The design of systems for presentation. This will of course involve computers hereafter, both conceptually and technically; since it obviously includes, for the future, branching and intricately interactive systems enacted by programmable mechanisms, i.e. computers. Thus computer display, data structures (and, to an extent, programming languages and techniques) are all a part.

Fantics must also include

5. Psychological effect and impact of various presentational techniques-- but not particular formal aesthetics, as of haiku or musical composition. Where directly relevant fantics also includes

6. Sociological tie-ins-- especially supportive and dysfunctional structures, such as tie-ins with occupational structure; sponsorship and commercials; what works in schools and why. Most profoundly of all, however, fantics must deal with psychological constructs used to organize things:

7. The parts, conceptual threads, unifying concepts and whatnot that we create to make aspects of the world understandable. We put them into everything, but standardize them in media.

For example, take radio. Given in radio-- the technological fundament-- is merely the continuous transmission of sound. Put into it have been the "program," the serial (and thus the episode), the announcer, the theme song and the musical bridge-- conventions which are useful presentationally.

The arbitrariness of such mental constructs should be clear. Their usefulness in mental organization perhaps is not.

Let's take a surprise example, nothing electronic about it.

Many "highways" are wholly fictitious-- at least to begin with. Let's say that a Route 37 is created across the state: that number is merely a series of signs that users can refer to as they look at their maps and travel along.

However, as time goes by, "Route 37" takes on a certain reality as a conceptual entity: people think of it as a thing. People say "just take 37 straight out" (though it may twist and turn); groups like a Route 37 Merchants' Association, or even a Citizens to Save Scenic 37, may spring up.

What was originally simply a nominal construct, then, becomes quite real as people organize their lives around it.

This all seems arbitrary but necessary in both highways and radio. What, then, does it have to do with the new electronic media?

The same apparently is true of the data structure. I used to be somewhat disturbed at the way Engelbart's text systems seem to be rigorously hierarchical. This in fact is the case, in the sense that having multiple discrete levels is built deep into the system. But it turns out to be harmless. The stored text is divided by the storage techniques into multiple levels, corresponding to a Harvard outline. Think of it as something like this:

1. HIERARCHICAL FORMAT
   A. STORAGE
   B. DISPLAY
   C. LANGUAGE

But let's expand this example a little:

1. HIERARCHICAL FORMAT
   A. STORAGE
      A1. Everything in NLS is stored with hierarchical codes.
      A2. Their effect depends on the display.
   B. DISPLAY
      B1. The hierarchical codes of NLS have no consequences in particular.
      B2. The hierarchical codes for NLS can splay the material out into a variety of display arrangements.
         B2A. They can be displayed in outline form.
         B2B. They can be displayed in normal text form.
         B2C. These dratted numbers can even be made to disappear.
   C. LANGUAGE
      C1. The command language determines what the display shows of the hierarchical structure.
      C2. What is shown can be determined by a program in the command language. (For instance, "how many levels down" it is being shown).
         C2A. This is four levels down. (The earlier example wasn't.)
      C3. The display format all depends on what display program you use, in the NLS command language.

That's enough of that. I can't help remarking that I still don't like that sort of structuring, but it is deep in NLS, and if you don't like it either (poor deprived lucky user of NLS) you can program it to disappear, so it's hardly in your way.

BY THE BEARD OF THE PROPHET!

Engelbart in German means Angelbeard; Doug Engelbart is indeed on the side of the angels. In building his mighty system he points a new way for humanity. The sooner the better. Any history of the twentieth century will certainly hold him high. Few great men are also such nice guys.

BIBLIOGRAPHY

Douglas C. Engelbart, Richard W. Watson & James C. Norton, "The Augmented Knowledge Workshop." Proc NCC 73, 9-21.

Charles H. Irby, "Display Techniques for Interactive Text Manipulation." Proc NCC 74, 247-255.

A VERY BASIC HYPERTEXT SYSTEM

Hypertext is non-sequential writing. It's no good to us, though, unless we can go instantly in a choice of directions from a given point.

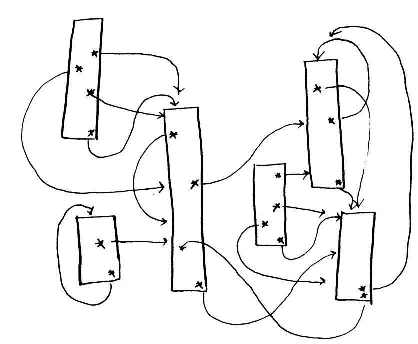
This of course can only mean on computer display screens.

Engelbart's system, now, was mainly designed for people who wanted to immerse themselves in it and learn its conventions. Indeed, it might be said to have been designed for a community of people in close contact, a sort of system of blackboards and collaborative talking papers.

A more elemental system, with a different slant, was put together at Brown U. on IBM equipment. We'll refer to it here as "Carmody's System," after the young programmer whose name came first on the writeup.

Carmody's system runs on an IBM 360 with 2250 display. While the 2250 is a fine piece of equipment, the quirks of the 360's operating system (see p. 45) often delay the user by making him wait, e.g., for someone else's cards to get punched before it responds to his more immediate uses; this is like making ice-skaters wait for oxcarts.

Anyway, the system essentially imposes no structure on the material; it may consist of text segments of any length and ties and links between them. An asterisk appearing anywhere in one piece of text signals a possible jump, but the reader doesn't necessarily know where to; zapping the asterisk with the lightpen takes you there, however.



This is stark and simple. It could also get you good and lost. However, a simple technique took care of that: everytime the user jumped, the address of his previous location was saved on a stack (see "The Magic of the Stack," p. 42). The user also had a RETURN button: when he wanted to go back to where he had last jumped from, the system would pop the last address off the top of the stack, and take him there. (This feature was adapted from my 1967 Stretchtext paper, and turned out to work out quite well in practice.)

The system also had handy features for light-pen text editing, and various nice printout techniques. All told, it was a clean and powerful design. While it lacked higher-level visualization facilities, like Engelbart's display of Levels (see "outline" in Engelbart article) or collateral display (see "Thinkertoys," p. DM 52), it was in some ways suited for naive users; that is, it was eventually fairly safe to use, and could in large part be taught to rank beginners in a couple of hours-- provided they didn't have to know about JCL cards.

It is left for the reader to figure out interesting uses for it. How would you do collateral structures? How could you signal to a reader which of several pieces of text a jump was to?

(At least one real hypertext was actually written on this system. It tied together a lot of patents for multilayer electrodes. Readers agreed that they could learn more from it about multilayer electrodes than they had imagined wanting to know.)

BIBLIOGRAPHY

Steven Carmody, Walter Gross, Theodor H. Nelson, David Rice and Andries van Dam, "A Hypertext Editing System for the 360." In Faiman and Nievergelt (eds.), Pertinent Concepts in Computer Graphics (U. Ill. Press '69), 291-330.
Note: Mr. Gross now goes by the name of Lightning Clearwater.

GORDON PASK RETURNS [DIFFICULT SECTION]

This continues the remarks on Gordon Pask begun on p.

I will now try to describe Pask's work as he has explained it to me. Perhaps this will be of some help to those who may have been mystified or dumfounded by contact with this fabulous man.

Gordon Pask's concern is abstraction and how concepts are formed, whether in a creature of nature or a robot or a computer program. Abstraction is of interest primordially (as life evolved thinking capacity), psychogenetically (as the mind acquires new facilities, described most peculiarly by Piaget), and epistemologically (how do we know? Like, how do we know, man?), and methodologically (how can we most effectively find out more?).

His interest, then, is in teaching by allowing students to discover exact relations in a specific subject matter by the very process of abstraction that is of so much interest.

What he does, then, is prepare given fields of learning so that they can be studied by students using abstractive methods, without guidance.

This preparation basically has two steps. First he sets up the whole field. This is done in collaboration with a "subject matter expert," who names the important topics in the field and states what interconnections they have. The result is a complex graph structure (see p. 26) which Pask calls a conversational domain. It comes out to huge diagrams of labels and lines between them.

Then Pask processes this structure to make a more usable map of the field that he calls an entailment structure. The processing basically involves removing "cycles" in the graph, thus making the structure hierarchical in a slightly artificial way justified by what the subject-matter-expert has said is the structure of the field.

(This processing is carried out by a program called EXTEND.)

The resulting Entailment Structure is then presented to the student as a great map of the field which he may explore.

Pask intends that the student's explorations will consist of testing analogies, or what Pask calls morphisms, to find the exact structures of knowledge he is supposed to be acquiring. This knowledge will be in the form of isomorphisms, or exact analogies, i.e. laws.

Pask's overall system, examples of which he has running in his laboratory in England, he calls CASTE (Course Assembly System and Tutorial Environment). A further development, which is to be put on a PDP-11/45 computer (see p. 36 and p. 42) at the Brooklyn Children's Museum, is called THOUGHT-STICKER. This program is intended to allow the demonstration and testing of analogies directly, by children.

PASK AND HYPERTEXT

Gordon Pask's work is remarkably similar to my own stuff on hypertext.

Essentially Pask is reducing a field to an extremely formal structure of relations which may then be studied by the student, at the student's initiative.

(What I don't quite understand is how the analogies are to be explored and tested.)

Anyway, a principal point is that the student is in control and may use his initiative dynamically; the subject is not artificially processed into a presentational sequence. Moreover, the arbitrary interconnections of the subject, which are no respecters of the printed page, are recognized as the fundamental structures the student must deal with and come to understand. On all these points Pask and I are in total agreement.

Indeed, his explorable systems-- (I don't know if they will be what I elsewhere call hypergrams or responding resources)-- will be fascinating, fun and terrifically educational. Because he is.

Now it turns out that this exactly complements the notion of hypertext as I have been promulgating it lo these many years.

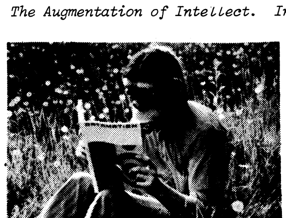
Hypertext is non-sequential text. If we write a hypertext on something, it will be most appropriate if we give it the general interconnective structure of the field. In other words, the interconnective structures chosen for the textual parts are likely to have the same connective structure (in general) as Pask's Entailment Structure.

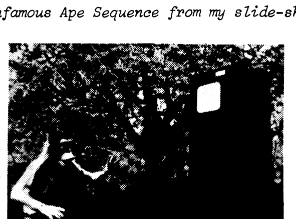
For another kind of hypertext, the anthological hypertext built up of lots of other writings, it is also reasonable to expect the connective structures to cluster to the same general form as Pask's entailment structure.

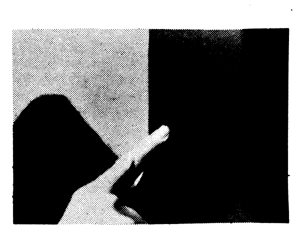
In other words, the very same field of knowledge Pask is out to represent as an explorable, formalized whole, I am out to represent as an explorable informalized whole, with anecdotes, jokes, cartoons, "enrichment materials," and anything else people might dig.

In still other words, let's have both and call it a party.

The Augmentation of Intellect. Infamous Ape Sequence from my slide-show.







Actually it needs the '2001' music.

You can't read the screen here. It says: COGITO ERGO SUM
$e=mc^2$
Call me Ishmael.

It really needs the music.

Simply this: till now the structures of media somehow sprang naturally from the nature of things. Now they don't anymore. Radio, books and movies have a natural inner dynamic of their own, leading to such constructs. While this may prove to be so for computer media as well (—as I argued in "Getting It Out of Our System," cited p.DM15), then again it may not. In other words, WE MUST ACKNOWLEDGE THAT WE ARE INVENTING PRESENTATIONAL TECHNIQUES IN THE NEW MEDIA, not merely transporting or transposing particular things into them because they seem right. The psychological constructs of man-machine systems may turn out to be largely arbitrary. Thus bringing to terminal systems conventions like dialogue instruction ("CAI"), or arbitrary restrictions of how things may be connected, presented or written on the computer may be a great mistake.

The highway-number analogy continues. The older highways had numbers for convenience, and our travels became organized around them, and particular highways (like "U.S. 1" and "Route 66") came to have special character. But now with the Interstates, a highway is a planned, sealed unit, no longer just a collection of roads gathered together under a name.

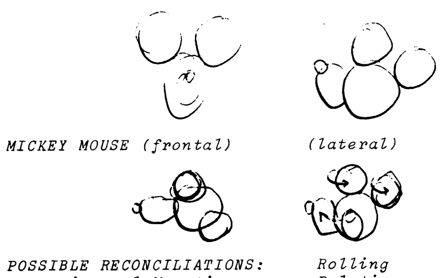
This unit, the Interstate, is not merely a psychological construct, but a planned structure. Knowing what works and what doesn't in the design of fast highways, the Interstates were built for speed, structured as closed units. Designing them with limited access has been a conscious decision in the system design for well-based reasons, not a chance structure brought in from horse-and buggy days.

Now, the constructs of previous media— writing, films, other arts— evolved over time, and in many cases may have found their way to a "natural" form. But because of the peculiar way that computer media are currently evolving (—under large grants largely granted to professionals who use very large words to promote the idea that their original professions are largely applicable—), this sort of natural evolution may not take place. The new constructs of computer media, especially computer screen-media, may not have a chance to be thought out. We need designs for screen presentations and their mixture— vignetting, Windows, screen mosaics, transformed and augmented views, and the rapid and comprehensible control of these views and windows. We are still just beginning to find clever viewing techniques, and have hardly begun to discover highly responsive forms of viewability and control (cf. collateration in "Thinkertoys," p.DM52), and Knowlton's button-box (oops, omitted) (See T. Nelson, "A Conceptual Framework for Man-Machine Everything," cited p. , and material on controls, below.)

THE MIND'S UNIFICATION

One of the remarkable things about the human mind is the way it ties things together. Perceptual unity comes out of nowhere. A bunch of irregular residential and industrial blocks becomes thought of as "my neighborhood." A most remarkable case of mental unification is afforded by the visage of our good friend Mickey Mouse. The character is drawn in a most paradoxical fashion: two globelike protrusions (representing the ears) are in different positions on the head, depending on whether we view him from the front or the side. No one finds this objectionable; few people even notice, it seems.

THE PARADOXICAL ANATOMY OF MICKEY MOUSE



MICKEY MOUSE (frontal) *(lateral)*

POSSIBLE RECONCILIATIONS: Diagonal Mounting *Rolling Relative to Camera*

What this shows, of course, is the way the mind can unify into a consistent mental whole even things which are inconsistent by normal rules (in this case, the rules of three-dimensional structure).

Even perceptions are subject to the same principle of unification. The fingernail is an excrescence with no nerves in it; yet somehow you can feel things with your fingernails— tying together disparate sensations into a unified sense of something in the world (say, a coin you're trying to pick up). In the same way, an experienced driver feels the road; in a very real sense, the car's wheels and suspension become his own sensory extensions.

This principle of mental unification is what makes things come together, both literally and figuratively, in a fantic field. A viewer sees two consecutive movie shots of streets and unifies them into one street; controls, if you are used to them, become a single fused system of options; we can have a sense of a greater whole, of which one view on a screen is a part.

THE GESTALT, DEAR BRUTUS
IS NOT IN OUR STARS
BUT IN OURSELVES.

CONTROLS: THEIR UNIFICATION AND FEEL

Controls are intimately related to screen presentation, just as arbitrary, and just as important.

The artful design of control systems is a deeply misunderstood area, in no way deconfused by calling it "human factors." There are many functions to be controlled, such as text editing operations, views of the universe on a screen, the heading of a vehicle, the tilt of an aircraft, the windage and adjustments of artillery, the temperature of a stove burner and any other controllable devices. And nowadays any conceivable devices could control them— pushbuttons, knobs, cranks, wheels, levers and joysticks, trigger, dials, magic wands, manipulation by lightpen on CRT screens (see p.DM31), flicks of the finger, the turning of the eyes (as in some experimental gun-aiming devices), the human voice (but that introduces problems— see p.DM13), keyboards, electronic tablets, Engelbart mice and chordwriters, and so on.

The human mind being as supple as it is, anything whatever can be used to control systems. The problem is having it be a comprehensible whole.

As already remarked, our ability mentally to unify things is extraordinary. That we somehow tie together clutch, gear, accelerator and brake into a comprehensible control structure to make cars go and stop should amaze and instruct.

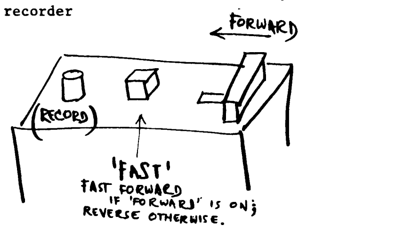
Engineers and "human factors" people speak as though there were some kind of scientific or determinate way to design control systems. Piffle. We choose a set of controls, much like an artist's Palette, on the basis of general appropriateness; and then try best and most artistically to fit them to what needs doing.

The result must be conceptually clear and retroactively "obvious"— simply because clarity is the simplest way to keep the user from making mistakes. Clear and simple systems are easier to learn, harder to forget, less likely to be screwed up by the user, and thus are more economical— getting more done for the resources put in.

There is a sort of paradox here. The kinds of controls are totally arbitrary, but their unification in a good system is not. Smoothness and clarity can come from disparate elements. It is for this reason that I lay particular stress on my JOT system for the input and revision of text, using a palette of keys available on the simplest standard computer terminal, the 33 Teletype. I cannot make the final judgement on how good this system is, but it pleases me. JOT is also an important example because it suggests that a conceptually unified system can be created from the artful non-obvious combination of loose elements originally having different intended purposes.

Mental analogy is an important and clear control technique. We tend to forget that the steering wheel was invented, separately replacing both the boat's tiller and the automobile's tiller. We also forget that the use of such steering mechanisms must be actually learned by children. Such continuous analogies, though, require corresponding continuities in the space to be controlled— an important condition.

Simplicity and clarity have nothing to do with the appearance of controls, but with the clarity and unique locatability of individual parts. For this reason I find deplorable the arrayed controls that are turning up, e.g. on today's audio equipment. Designers seem to think rows of things are desirable. On the contrary: the best designed controls I ever used are on the Sony TC-50 pocket tape recorder



but of course this is now phased out; instead most cassette recorders have five or six stupid buttons in a row. (Was it too good to last?)

Spurious control elegance comes in many guises. Consider Bruce McCall's description of the Tap-A-Toe Futuroidic Footless De-Clutching™ system. This was offered on the fictitious 1934 Bulgemobiles, and allowed you to drive the car with one pedal, rather than three (see box nearby).

Careless and horrible designs are not all fictitious. One egregious example also indicates the low level of design currently going into some responding systems: computer people have designed CRT writing systems for newspapers which actually have a "kill" button on the console, by which authors would accidentally kill their stories. In a recent magazine article it was explained that the eventual solution was to change the program so that to kill the story you had to hit the "kill" button twice. To me this seems like a beautiful example of what happens when you let insulated technical people design the system for you: a "kill" button on the keyboard is about as intelligent as installing knives on the dashboard of a car, pointing at the passenger.

There is another poor tendency. When computer programmers or other technical people design particular systems without thinking more generally, things are not likely to be either simple or combinable. What may result is intricate user-level controls for one particular function, controls that are differently used for another particular function, making the two functions not combinable.

What makes for the best control structures, then? There is no simple answer. I would say provisionally that it is a matter of unified and conspicuous constructs in the mental view of the domain to be controlled, corresponding to a well-distinguished and clearly-interrelated set of controlling mechanisms. But that is hardly the last word on the subject.

THE ORGANIZATION OF WHOLENESS

It should be plain that in responding screen-systems, "what happens on the screen" and "how the controls respond" are not really distinguishable. The screen events are part of the way the controls respond. The screen functions and control functions merge psychologically.

Now, there is a trap here. Just as the gas pedal, clutch, gearshift and brake merge psychologically, any control structure can merge psychologically. Clutch and gear shift do not have, for most of us, clear psychological relevance to the problem of controlled forward motion. Yet we psychologically integrate the use of these mechanisms as a unified means for controlling forward motion (or, like the author, get an Automatic). In much the same way, any system of controls can gradually come through use to have a psychological organization, even spuriously. The trap is that we so easily lose sight of arbitrariness and even stupidity of design, and live with it when it could be so much better, because of this psychological melding.

But useful wholeness can be helped along. Just as what I have called the accordance-structure of writing (see "Writing," p.DM43) moves it along smoothly, fantic design that builds from a well-organized internal dynamic should confer on a fantic system the same momentum and clarity that carefully-organized writing has.

This contribution of wholeness can only occur, however, if the under-level complications of a system have been carefully streamlined and smoothed back, at least as they affect the user. Consider the design of the JOT text editing system (p.DM50): while it is simple to the user, computer people often react to it with indignation and anger because it hides what are to them the significant features of computer text editing— explicit preoccupation with storage, especially the calling and revision of "blocks." Nevertheless, I say it is the details at this level which must be smoothed back if we are to make systems for regular people.

The same applies to the Th3 system (see p. DM55), which is designed to keep the user clear-minded as he compares things in multiple dimensions. The mechanisms at the computer level must be hidden to make this work.

FANTIC SPACE

Pudovkin and Eisenstein, great Russian movie-makers of the twenties, talked about "filmic space"— the imaginary space that the action seems to be in.

This concept extends itself naturally to fantic space, the space and relationships sensed by a viewer of any medium, or a user in any presenting or responding environment. The design of computer display systems, then, is really the artful crafting of fantic space. Technicalities are subservient to effects. (Indeed, I think computer graphics is really a branch of movie-making.)

FANTIC STRUCTURE

The fantic structure of anything, then, consists of its noticeable parts, interconnections, contents and effects.

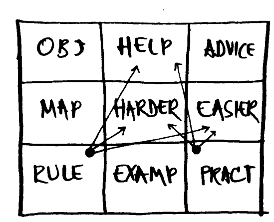
I claim that it is the fantic unity— the conceptual and presentational clarity of these things— that makes fantic systems— presentational systems and material— clear and helpful, or not.

Let us take an interesting example from a system for computer-assisted instruction now under implementation. I will not identify or comment on the system because perhaps I do not understand it sufficiently. Anyway, they have an array of student control-buttons that look like this:

OBJ [objective]	HELP	ADVICE
MAP	HARDER	EASIER
RULE	EXAMP [example]	PRACT [practice]

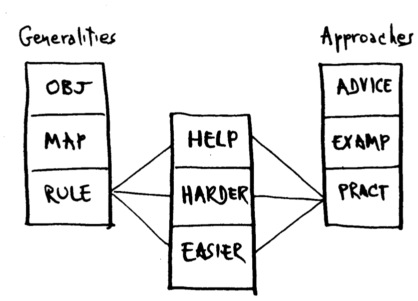
The general thinking in this system seems to be that the student may get an overall organizing view of what he is supposed to be learning (MAP); information on what he is currently supposed to be about (OBJ); canned suggestions based on what he's recently done (ADVICE). Moreover, he can get the system to present a rule about the subject or give him practice; and for either of these he may request easier rules or practice, or harder rules (i.e., more abstruse generalities) or harder practice.

For the latter, the student is supposed to hit RULE or PRACT followed by HELP, HARDER or EASIER, viz.:

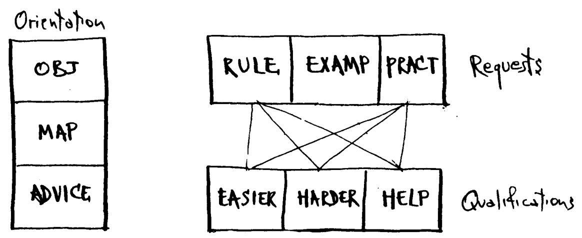


Now regardless of whether this is a well-thought-out way to divide up a subject— I'll be interested to see how it works out— these controls do not seem to be well-arranged for conceptual clarity. It seems to be the old rows-of-buttons approach.

I have no doubt that the people working on this system are certain this is the only possible layout. But consider that the student's options might be clearer to him, for instance, if we set it up as follows:



Or like this:



What I am trying to show here is that merely the arrangement of buttons creates different fantic constructs. If you see this, you will recognize that considering all the other options we have, designing new media is no small matter. The control structures merge mentally with the presentational structures. The temptation to settle on short-sighted designs having shallow unity is all too great.

FANTIC DESIGN

Fantic design is basically the planning and selection of effects. (We could also call these "performance values"-- cf. "production values" in movies.)

Some of these intended effects are simply the communication of information or cognitive structure-- "information transfer," to use one of the more obtuse phrases current. Other desirable effects include orienting the user and often moving him emotionally, including sometimes overwhelming or entrancing him.

In the design of fantic systems involving automatic response, we have a vast choice among types of presentational techniques, tricks that are just now becoming understood. Not just screen techniques and functions, but also response techniques and functions.

(If "feelie" systems are ever perfected, as in Huxley's Brave New World, it's still the same in principle. See Wachspress, p. DM9.)

In both general areas, though-- within media, and designing media-- it seems to me that the creation of organizing constructs is the most profound problem. In particular, the organizing constructs must not distract, or tear up contents. An analogy: in writing, the inventions of the paragraph, chapter and footnote were inventions in writing technique that helped clarify what was being expressed. What we need in computer-based fantic design is inventions which do not artificially chop up, constrain, or interfere with the subject (see box, Procrustes, nearby).

I do not feel these principles are everywhere sufficiently appreciated. For instance, the built-in structures of PLATO (see "Fantic Space of PLATO," p. DM27) disturbs me somewhat in its arbitrariness-- and the way its control keys are scattered around.

But there is always something artificial-- that is, some form of artifice-- in presentation. So the problem is to devise techniques which have elucidating value but do not cut connections or ties or other relationships you want to save. (For this reason I suggest the reader consider "Stretchtext," p. DM19 , collateral linkage (p. DM52), and the various hypergrams (p. DM18-19). These structures, while somewhat arbitrary and artificial, nevertheless can be used to handle a subject gently.)

An important kind of organizing construct is the map or overall orienting diagram. This, too, is often partly "exact" and partly "artifice:" certain aspects of the diagram may have unclear import but clear and helpful connotation. (For instance, consider the "picture systems" diagram on p. DM 20 -- just what does the vertical dimension mean? Yes, but what does it really mean?)

Responding systems now make it possible for such orienting structures to be multidimensional and responding (cf. the orienting function of the "dimensional flip" control illustrated on p. DM 31).

Fantic design, then, is the creation either of things to be shown (writing, movie-making, etc.) at the lower end, or media to show things in, or environments.

1. The design of things to be shown-- whether writing, movie-making, or whatever-- is nearly always a combination of some kind of explicit structure-- an explanation or planned lesson, or plot of a novel-- and a feeling that the author can control in varying degrees. The two are deeply intertwined, however.

The author (designer, director, etc.) must think carefully about how to give organization to what is being presented. This, too, has both aspects, cognition and feelings.

At the cognitive end, the author must concern himself with detailed exposition or argument, or, in fiction, plot. But simply putting appropriate parts together is not enough: the author must use organizing constructs to continually orient the reader's (or viewer's) mind. Repeated reference to main concepts, repeated shots (in a movie) of particular locations, serve this function; but each medium presents its own possible devices for this purpose.

The organization of the feelings of the work criss-crosses the cognitive; but we can't get into it here.

Selection of points and parts contributes to both aspects. If you are trying to keep the feeling of a thing from being ponderous, you can never include everything you wanted, but must select from among the explicit points and feeling-generators that you have thought of.

2. The design of media themselves, or of media subsystems, is not usually a matter of option. Books, movies, radio and TV are given. But on occasion, as for world's fairs or very personal projects, we have a certain option. Which allows things like:

> Smellavision or whatever they called it: movies with a smell-track, which went out into the theater through odor generators.
> Branching movies (see p. DM44).
> "Multi-media" (multiple audio tracks and simultaneous slide projections on different screens).
> Stereo movies.

And so on. The thing about the ones mentioned is that they are not viable as continuing setups for repeated productions. They do not offer a permanent wide market; they are not stable; they do not catch on. Which is in a way, of course, too bad.

But the great change is just about now. Current technicalities allow branching media-- especially those associated with computer screens. And it is up to us now to design them.

3. MENTAL ENVIRONMENTS are working places for structured activity. The same principles of showmanship apply to a working environment as to both the contents of media and the design of media. If media are environments into which packaged materials are brought, structured environments are basically environments where you use non-packaged material, or create things yourself. They might also be called "contentless media." The principles of wholeness in structured environments are the same as for the others, and many of our examples refer to them.

The branching computer screen, together with the selfsame computer's ability to turn anything else on and off as selected by the user, and to fetch up information, yields a realm of option in the design of media and environment that has never existed before. Media we design for screen-based computer systems are going to catch on widely, so we must be far more attentive to the options that exist in order to commit-- nationally, perhaps-- to the best.

In tomorrow's systems, properly unified controls can give us new flexibilities. If deeply well-designed, these promise magnificent new capabilities. For instance, we could allow a musician to "conduct" the performance of his work by a computer-based music synthesis system (see "Audio," p. DM11), perhaps controlling the many qualities of the performance on a screen as he goes, by means of such techniques as dimensional flip (see p. DM31). (The tradition of cumulative audio synthesis, as practiced in the fifties by Les Paul and Mary Ford, and more recently by Walter Carlos and Mike Oldfield, will take on a new fillip as multidimensional control techniques become common.)

One of the intents of this book has been to orient you to some of the possibilities and some of the options, considered generally. There is not room, unfortunately, to discuss more than one or two overall possibilities in detail. The most successful such system so far has been PLATO (discussed pp. DM18-19); others could not be listed for lack of space.

NEW MEDIA TO LAST

What's worse, we are confronted not merely with the job of using computers to present specific things. The greater task is to design overall computer media that will last us into a more intelligent future. Adrift in a sea of ignorance and confusion, it is nevertheless our duty to try to create a whole transportation system that everybody can climb aboard. For the long run, fantic systems must be treated not as custom systems for explicit purposes, but as OVERALL GENERAL DESIGNS WHICH WILL HAVE TO TIE TOGETHER AND CATCH ON, otherwise collapse and perish.

FINAL CONSEQUENCES.

It seems to me certain that we are moving toward a generalized and universal Fantic system; people can and should demand it. Perhaps there will be several; but if so, being able to tie them together for smooth transmission is essential. (Think of what it would be like if there were two kinds of telephones?) This then is a great search and crusade: to put together truly general media for our future, systems at which we can read, write, learn and visualize, year after year after year. The initiatives are not likely to come from the more conventional computer people; some of them are part of the problem. (Be prepared for every possible form of aggressive defensiveness from programmers, especially: "Why would you want that?" The correct answer is BECAUSE, dammit!)

But this all means that interior computer technicalities have to be SUBSERVIENT, and the programmers cannot be allowed to dictate how it is to behave on the basis of the underlevel structures that are convenient to them. Quite the contrary: from the fullest consideration of the richest upper-level structures we want, we the users-to-be must dictate what lower-level structures are to be prepared within.

But this means you, dear reader, must develop the fantic imagination. You must learn to visualize possible uses of computer screens, so you can get on down to the deeper level of how we are going to tie these things together.

The designer of responding computer systems is creating unified setups for viewing and manipulating things-- and the feelings, impressions and sense of things that go with them. Our goal should be nothing less than REPRESENTING THE TRUE CONTENT AND STRUCTURE OF HUMAN THOUGHT. (Yes, Dream Machines indeed.) But it should be something more: enabling the mind to weigh, pursue, synthesize and evaluate ideas for a better tomorrow. Or for any at all.

BIBLIOGRAPHY

Theodor H. Nelson, "A Conceptual Framework for Man-Machine Everything." Proc. NCC 73.

-----, "Computopia and Cybercrud." In Levien (ed.), Computers in Instruction, The Rand Corporation, 1971.

JOT(TM): Juggler Of Text.

From "A Human Being's Introduction to the JOT System." ©1972 T. Nelson.

Here's how simple it is to create and edit text with the JOT system. Since your typewriter is now a JOT machine, not every key does what it used to.

CREATING TEXT: just type it in.

You type: The quick brown fox jumps over the lazy dog.

It types: The quick brown fox jumps over the lazy dog.

REVIEWING A SENTENCE YOU JUST TYPED: the back-arrow takes you back, the space bar steps you through.

You type: ← sp sp sp sp

It types: (bell) The quick brown fox

DELETIONS AND INSERTIONS: the RUBOUT key rejects words you don't want. To insert , merely type.

You type: ← sp sp RUBOUT lithe sp sp sp sp sp sp

It types: (bell) The quick /brown/ lithe fox jumps over the lazy dog.

REARRANGING TEXT: first we make three Cuts in the text, signalled by free-standing exclamation points.

You type: sp ! sp ! sp ! fox

It types: The ! quick ! lithe ! fox

TO REARRANGE IT, YOU TYPE: LINE FEED key. This exchanges the two pieces between the cuts.

CHECK THE RESULTS:

← sp sp sp sp

(bell) The lithe quick fox

THINKERTOYS

Our greatest problems involve thinking and the visualization of complexity.

By "Thinkertoy" I mean, first of all, a system to help people think. ('Toy' means it should be easy and fun to use.) This is the same general idea for which Engelbart, for instance, uses the term "augmentation of intellect."

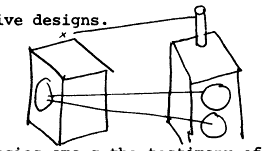
But a Thinkertoy is something quite specific: I define it as a computer display system that helps you envision complex alternatives.

The process of envisioning complex alternatives is by no means the only important form of human thought; but it is essential to making decisions, designing, planning, writing, weighing alternate theories, considering alternate forms of legislation, doing scholarly research, and so on. It is also complicated enough that, in solving it, we may solve simpler problems as well.

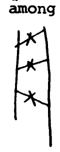
We will stress here some of the uses of these systems for handling text, partly because I think these are rather interesting, and partly because the complexity and subtlety of this problem has got to be better understood: the written word is nothing less than the tracks left by the mind, and so we are really talking about screen systems for handling ideas, in all their complexity.

Numerous types of complex things have to be intercompared, and their relations inter-comprehended. Here are a few of the many types:

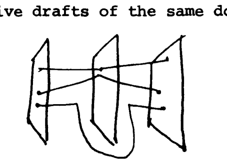
Alternative designs.



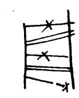
Discrepancies among the testimony of witnesses.



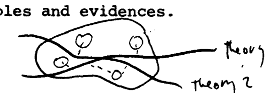
Successive drafts of the same document.



Pairs of things which have some parts the same, some parts different (contracts, holy books, statutes of different states, draft versions of legislation...)



Different theories and their ties to particular examples and evidences.

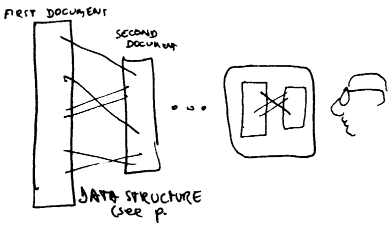


Under examination these different types of intercomparison seem to be rather different. Now, one approach would be to create a different data structure and viewing technique for each different type of complex. There may be reasons for doing that in the future.

For the present, however, it makes sense to try to find the most general possible viewing technique: one that will allow complex intercomparisons of all the types mentioned, and any others we might run across.

One such technique is what I now call collateration, or the linking of materials into collateral structures,* as will be explained. This is fairly straightforward if you think enough about the problem; Engelbart discovered it independently.

Let us call two structures collateral if there are links between them, connecting a selected part of one with a selected part of the other. The sequences of the connected parts may be different. For simplicity's sake, suppose each one is a short piece of writing. (We will also assume that there is some convenient form of rapid viewing and following between one end of a link and another.)



Now, it will be noted first off that this is an extremely general method. By collateral structuring we can easily handle the equivalents of: tables of contents; indexes; comments and marginalia; explanations, exegesis, explication; labeling; headings; footnotes; notes by the writer to himself; comments and questions by the reader for later reference; and additional details out of sequence.

Collateration, then, is the creation of such multiple and viewable links BETWEEN ANY TWO DATA STRUCTURES, in principle. It is general and powerful enough to handle a great variety of possible uses in human intellectual endeavor, and deserves considerable attention from researchers of every stripe.**

The problem then, is how to handle this for rapid and convenient viewing and whatever other work the user wants to do-- writing and splicing, intercomparing, annotating and so on. Two solutions appear on this spread: The Parallel Textface^{tm}, designed as a seminal part of the Xanadu system (see p.DM56), which I hope will be marketed with that system in the near future, and a more recent design which I've worked on at the University of Illinois, the 3D Thinkertoy or Th3.

CLARITY AND POWER

We stressed on the other side of the book that computer systems must be clear, simple and easy to use. Where things like business uses of computers are concerned, which are intrinsically so simple in principle, some of the complications that people have been forced to deal with in ill-designed computer systems verge on the criminal. (But some computer people want others to think that's the way it has to be. "Your first duty is to keep your job, right?" one computer person said to me recently. "It wouldn't do to set up systems so easy to use that the company wouldn't need you anymore." See "Cybercrud," p.8.)

But if it is desirable that computer systems for simple-minded purposes be easy to use, it is absolutely necessary that computer systems for complicated purposes be simple to use. If you are wrangling over complex alternatives-- say, in chess, or in a political simulation game (see "Simulation," p. 58), or in the throes of trying to write a novel, the last thing you will tolerate is for your computer screen to introduce complications of its own. If a system for thinking doesn't make thinking simpler-- allowing you to see farther and more deeply-- it is uselss, to use only the polite term.

But systems can be both powerful and simple at the same time. The myth that things have to be complicated to do anything for you is pernicious rubbish. Well-designed systems can make our mental tasks lighter and our achievements come faster.

It is for this reason that I commend to the reader these two designs of mine: as examples of user-level control and viewing designs-- fantic environments, if you will (see p.DM48-51)-- that are pruned and tuned to give the user great control over the viewing and cross-consideration of intricate alternatives, without complication. I like to believe that both of these, indeed, are ten-minute systems-- that is, when we get them running, the range of uses shown here can be taught to naive users, in ten minutes or less.

It is because of my heartfelt belief in this kind of simplicity that I stress the creation of prefabricated environments, carefully tuned for easy use, rather than the creation of computer languages which must be learnt by the user, as do such people as Engelbart (see p.DM46) and DeFanti (see p.DM31). Now, their approach obviously has considerable merit for sophisticated users who want to tinker repeatedly with variant approaches. For people who want to work incessantly in an environment, and on other things-- say writers-- and are absent-minded and clumsy and nervous and forgetful (like the present author), then the safe, prefabricated environment, with thoroughly fail-safe functions and utterly memorable structural and control interrelationships, is the only approach.

* In my 1965 paper (see bibliography) I called collateral structures zippered lists.

** A group at Brown University has reportedly worked along these lines since I worked with them, but due to certain personal animosities I have not kept up with their developments. It will be interesting to see what kind of response they can get out of the IBM systems they are using.

BIBLIOGRAPHY

Theodor H. Nelson, "A File Structure for the complex, the Changing and the Indeterminate." Proc. ACM 65, 84-100.

-----, "Simplicity Versus Power in User Systems." Unpublished.

WE OFTEN WANT TO SAVE ALTERNATIVES.

28

And closely to my [heart] she press'd, (side)
And closer still with bashful art,
[And ask'd me with her swimming eyes]
That I [would] rather feel than see (might)
The swelling of her Heart
[Her gentle Bosom rise.—]

29

[And now serene, serene & chaste,] I calm'd her fears; & she was calm
[But soon in calm and solemn tone]
[She] told her love with maiden pride; (And)
And so I won my Genevieve,
My [bright] & lovely Bride.] (dear)

From Coleridge's Poems: A Facsimile Reproduction of the Proofs and MSS. of some of the Poems. (Folcroft, 1972.)

DECISION/CREATIVITY SYSTEMS

[THINKERTOYS]

Theodor H. Nelson
19 July 1970

It has been recognized from the dawn of computer display that the grandest and most important use of the computer display should be to aid decisions and creative thought. The work of Ivan Sutherland (SKETCHPAD) and Douglas Engelbart have really shown how we may use the display to visualize and effect our creative decisions swiftly and vividly.

For some reason, however, the most important aspect of such systems has been neglected. We do not make important decisions, we should not make delicate decisions, serially and irreversibly. Rather, the power of the computer display (and its computing and filing support) must be so crafted that we may develop alternatives, spin out their complications and interrelationships, and visualize these upon a screen.

No system could do this for us automatically. What design and programming can create, however, is a facility that will allow us to list, sketch, link and annotate the complexities we seek to understand, then present "views" of the complexities in many different forms. Studying these views, annotating and refining, we can reach the final designs and decisions with much more in mind than we could otherwise hold together in the imagination.

Some of the facilities that such systems must have include the following:

Annotations to anything, to any remove.

Alternatives of decision, design, writing, theory.

Unlinked or irregular pieces, hanging as the user wishes.

Multicoupling, or complex linkage, between alternatives, annotations or whatever.

Historical filing of the user's actions, including each addition and modification, and possibly the viewing actions that preceded them.

Frozen moments and versions, which the user may hold as memorable for his thinking.

Evolutionary coupling, where the correspondences between evolving versions are automatically maintained, and their differences or relations easily annotated.

In addition, designs for screen "views", the motion, appearance and disappearance of elements, require considerable thought and imagination.

The object is not to burden the user, or make him aware of complexities in which he has no interest. But almost everyone in intellectual and decision pursuits has at some time an implicit need for some of these facilities. If people knew they were possible, they would demand them. It is time for their creation.

A full-fledged decision/creativity system, embracing both text and graphics, is one of the ultimate design goals of Project XANADU.

We might also think of them as systems for

THE MANAGEMENT OF LOOSE ENDS.

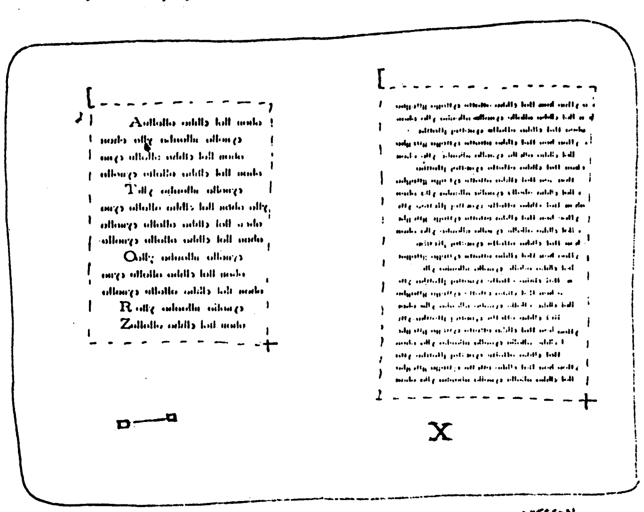
The PARALLEL TEXTFACE™

This user-level system is intended to aid in all forms of writing and scholarship, as well as anywhere else that we need to understand and manipulate complex clusterings of text (i.e., thought). It will also work with certain animated graphics.

The Parallel Textface, as described here, furnished the initial impetus for the development of the Xanadu™ system (see p. DM 56). Xanadu was developed, indeed, originally for the purpose of implementing some of these functions, but the two split apart. It turned out that the Parallel Textface required an extremely unusual data struc-structure and program techniques; these then became the Xanadu system. As developed in the final Xanadu design, they turn out to handle some very unusual kinds of screen animation and file retrieval. But this grew out of structuring a system to handle the functions described here.

Thus the Parallel Textface basically requires a Xanadu system.

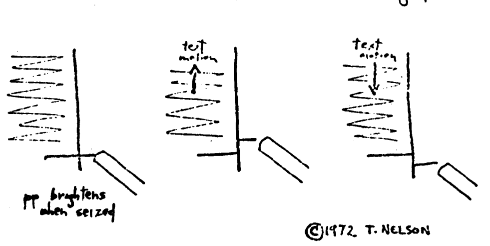
It is hoped that this system can be sold complete (including minicomputer or microprocessor--no connection to a large computer is required) for a few thousand dollars by 1976 or 1977. See p. (Since "business people" are extremely skeptical as to whether anybody would want such a thing, I would be interested in hearing expressions of interest, if any.)



As shown here, the screen presents two panels of text; more are allowed. Each contains a segment of a longer document. ("Page" would be an improper term, since the boundary of the text viewed may be changed instantly.)

The other odds and ends on the screen are hidden keys to control elements which have been made to fade (in this illustration), just to lessen the distraction.

Panel boundaries and control graphics may be made to appear by touching them with the lightpen.



ROVING FUNCTIONS

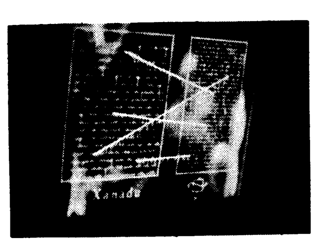
The text moves on the screen! (Essential.) The lower right hand corner of each text panel contains an inconspicuous control diagram. The slight horizontal extension is a movable control pip. The user, with his light pen, may move the pip up or down. "Up" causes the text to move smoothly upward (forward in the material), at a rate proportional to how far you push the pip; "down" causes it to move back. (Note that we do not refer here to jerky line-by-line jumps, but to smooth screen motion, which is essential in a high-performance system. If the text does not move, you can't tell where it came from.)

PARALLEL TEXTFACE (1971)

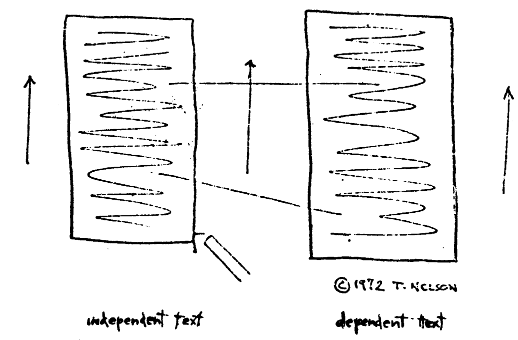
Real person sits at cardboard Xanadu mockup.

"Nice keyboard. But what happened to your typewriter?"

Two panels are about right for a 10 x 10 screen.

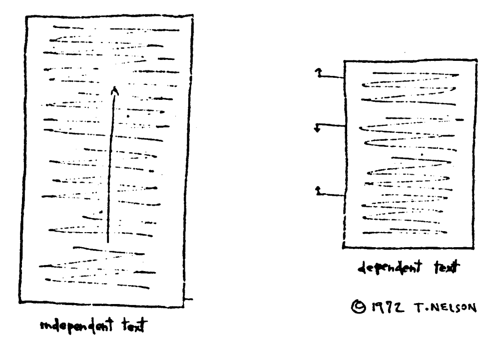


Independent text pulls dependent text along. Painted streaks simulate motion, not icicles.

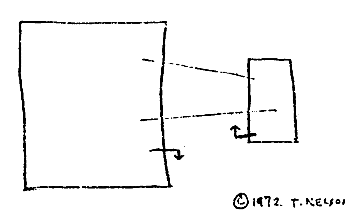


DERIVATIVE MOTION: when links run sequentially, connecting one-after-the-other on both sides, the contents of the second panel are pulled along directly: the smooth motion in one panel is matched in the other. This may be called derivative motion, between independent text (being handled directly with the lightpen) and dependent text (being pulled along). The relationship may be reversed immediately, however, simply by moving the lightpen to the control pip of the other panel, whose contents then become the independent text.

Irregularities in the links will cause the independent text to move at varying speeds or jump, according to an average of the links' connectivity.

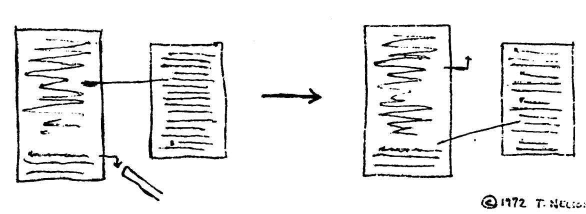


If no links are shown, the dependent text just stops.

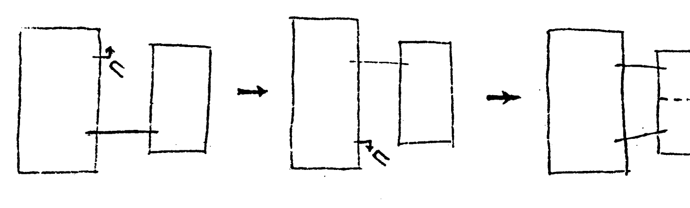


Collateral links between materials in the two panels are displayed as movable lines between the panels. (Text omitted in this diagram; panel boundary has been made to appear.)

Some links may not have both their endpoints displayed at once. In this case we show the incomplete link as a broken arrow, pointing in the direction of the link's completion.



The broken arrow serves not merely as a visual pointer, but as a jump-marker leading to the linked material. By zapping the broken arrow with the lightpen, the user summons the linked material--as shown by the completion of the link to the other panel. (Since there has been a jump in the second panel, we see that in this case the other link has been broken.)



When such links lead to different places, both of these destinations may nevertheless be seen at once. This is done by pointing at both broken links in succession; the system then allows both links to be completed, breaking the second panel between the two destinations (as shown by dotted line across panel).

XANADU

©1972 T. NELSON

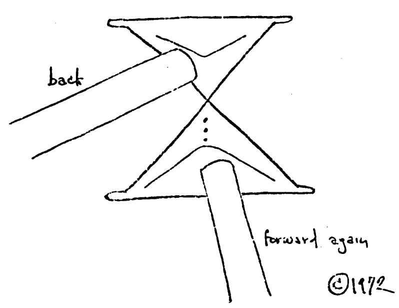
FAIL-SAFE AND HISTORICAL FEATURES.

In systems for naive users, it is essential to safeguard the user from his own mistakes. Thus in text systems, commands given in error must be reversible. For instance, Carmody's system (see p. DM47) requires confirmation of deletions.

Another highly desirable feature would allow the user to view previous versions, to see them collaterally with the corresponding parts of current versions, and even go back to the way particular things were and resume work from the previous version.

In the Parallel Textface this is all comprised in the same extremely simple facility. (Extremely simple from the user's point of view, that is. Inside it is, of course, hairy.)

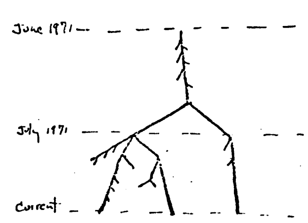
In an egregious touch of narcissistic humor, we use the very trademark on the screen as a control device (expanded from the "X" shown in the first panel):



Actually the X in "Xanadu™," as it appears on the screen, is an hourglass, with a softly falling trickle of animated dots in the lower half, and Sands of Time seen as heaps above and below. These have a control, as well as a representative, function.

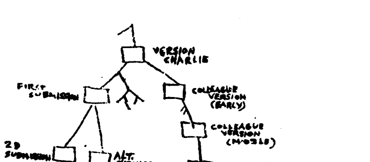
TO UNDO SOMETHING, YOU MERELY STEP "BACKWARD IN TIME" by dagging the upper part of the hourglass with the lightpen. One poke, one editing operation undone. Two pokes, two operations.

You may then continue to view and make changes as if the last two operations had never taken place. This effectively creates an alternative time-line.* However, if you decide that a previously undone edit operation should be kept after all, you may step forward-- stepping onto the previous time-line-- by using the lower half of the hourglass.

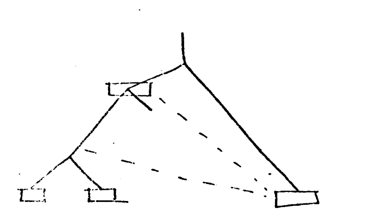


Revision Tree
©1972 T. NELSON

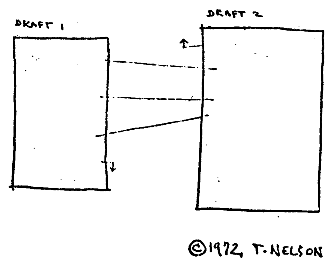
We see this clarified in a master time diagram or Revision Tree which may be summoned to the screen, never mind how. In this example we see that three versions are still "current," various other starts and variations having been abandoned. (The shaggy fronds correspond to short-lived variations, resulting from operations which were then reversed. In other words, "excised" time-lines, to use Gerrold's term-- see footnote.)



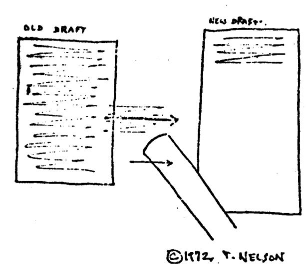
The user-- let's say he is a thoughtful writer-- may define various Versions or Drafts, here marked on the Revision Tree.



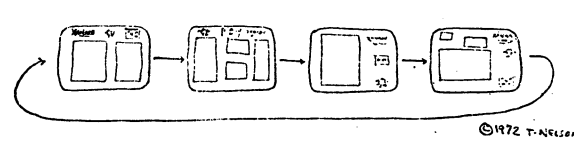
He may, indeed, define collateral linkages between different versions defined at various Times in the Tree...



... and see them displayed collaterally; and revise them further.



Materials may be copied between versions. (Note that in the copying operation of the Parallel Textface, you actually see the moved text moved bodily as a block.)

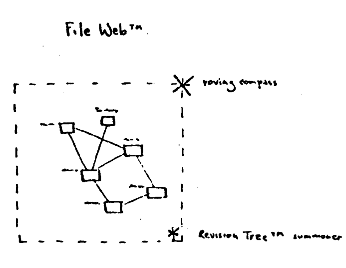


GETTING AROUND

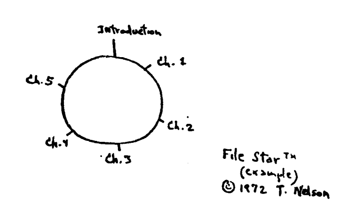
The user may have a number of standby layouts, with different numbers of panels, and jump among them by stabs of the lightpen.

Importantly, the panels of each can be full, each having whatever the contents were when you last left it.

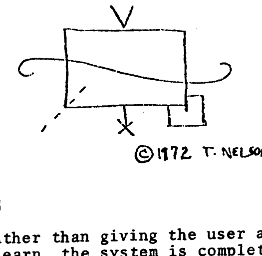
File Web™



The File Web™ is a map indicating what (labelled) files are present in the system, and which are collaterated.

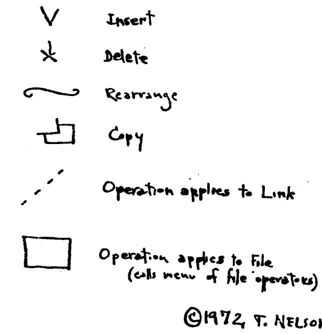


The File Star™ is a quick index into the contents of a file. It expands as long as you hold the lightpen to the dot in the center, with various levels of headings appearing as it expands. Naturally, you may jump to what you point at.

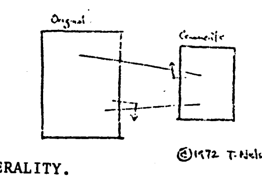


EDITING

Rather than giving the user anything complicated to learn, the system is completely visual. All edit controls are comprised in this diagram, the Edit Rose™. Viz.:

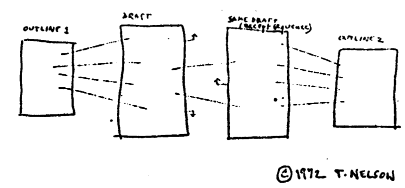


Separate portions of the Edit Rose invoke various edit operations. (You must also point with the lightpen to the necessary points in the text: once for insert, twice for Delete, three or four times for Rearrange, three times for Copy.)

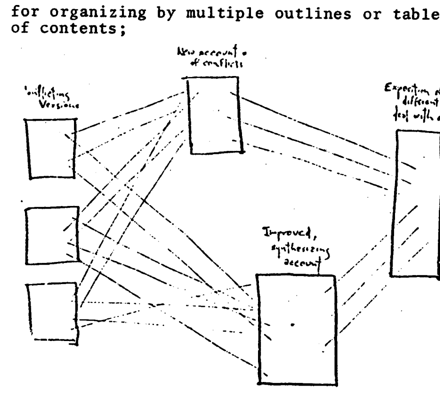


GENERALITY.

The system may be used for comments on things,



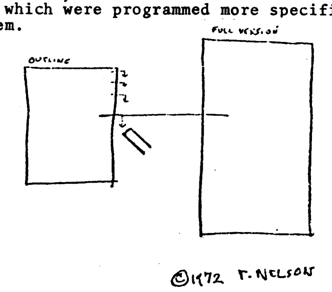
for organizing by multiple outlines or tables of contents;



and as a Thinkertoy, organizing complex alternatives. (The labels say: "Conflicting versions," "New account of conflicts," "Exposition of how different accounts deal with objections," "Improved, synthesizing account."

In other words, in this approach we annotate and label discrepancies, and verbally comment on differences in separate files or documents.

In ways this may seem somewhat obtuse. Yet above all it is orderly, and the complex of collateral files has a clarity that could be all-too-easily lost in systems which were programmed more specifically to each problem.



The fundamental strength of collateration, seen here, is of course that any new structure collateral to another may be used as a table of contents or an outline, taking the user instantly to parts which are of interest in some new context.

* Oddly, this has the same logical structure as time-travel in science-fiction.

There are basically three alternate premises of of time-travel: 1) that the past cannot be changed, all events having preceded the backstep; 2) that the past can be changed; and 3) that while time-travelers may be deluded into thinking (2), that (1) is really the case-- leading to various appointment-in-Samarra plots.

Only possibility (2) is of interest here, but there are various alternative logics of mutability and time-line stepping. One of the best I have seen is in The Man Who Folded Himself by David Gerrold (Popular Library, 1973): logic expounded pp. 64-8. I am bemused by the parallel between Gerrold's time-controls and these, worked out independently.

Th3

A VERY ADVANCED (?) TEXT SYSTEM. Read this at your peril. Multidimensional Concept-freaks only.

This design, Th3 (Thinkertoy in 3 dimensions), is one I have been working at while on the faculty of the University of Illinois. It is designed specifically for implementation using DeFanti's GRASS language (see p. DM 31), and the Vector General 3D display (see p. DM 30). Whether it will ever be actually programmed depends, of course, on numerous factors.

It is meant to be a very high-power thinkertoy, suitable for experimentation with creative processes, especially writing and three-dimensional design. (There is no room to discuss the latter here.) It is suited especially to the visualization of tentative structures in amorphous clusters. In some of its features it goes considerably beyond the more "commercial" thinkertoy system, the Parallel Textface (elsewhere in this spread).

Nevertheless, the same design criteria apply: a well-designed computer environment for any purpose should be learnable in ten minutes; otherwise the designer has not been doing his job. (I mean it would be learnable in ten minutes if you and I had it in front of us, working. This description will have to be weird and abstruse, I'm sorry to say.)

This system is designed around a three-dimensional display screen (the Vector General display, as manipulable by the GRASS language).

Now, most people do not think of text as three-dimensional. Laymen think of it as two-dimensional, since it's usually printed on rectangular pages. Computer people ordinarily think of it as one-dimensional, as a long string of characters and spaces--essentially what you'd get if you printed things in one line on a long, long ribbon. Well, frankly, I don't think of text as three-dimensional either; but like anything else, it has numerous qualities or dimensions, any three of which it's nice to be able to view at once (see "Dimensional Flip," p. DM 31). And that's essentially the idea: the three dimensions we'll look at at any one time will be a particular view of a larger whole.

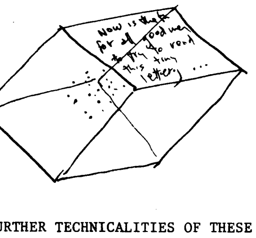
Now, the basic form of storage will be one of those Nelson-structures that drives computer people batty. Specifically, the basic data structure will be clusters of points.

Puns sometimes reflect a higher reality. Now it turns out that this structure in fact reflects a great Folk Truth: written discourse does in fact consist of "points" which you intend to get across. That we here intend to have them rotate as dots upon a screen reflects this structure.

Writing is, in fact, a projection from the intended "points" to a finished exposition which embraces them. Now, this is very like the view of language held in modern linguistics, namely, that a finished sentence is a "surface structure" constructed out of basic sentence kernels chewed up by certain transformations. Well, I am just pointing out here that writing is a surface structure of "points" which have been embedded and spliced in a structure of transitions, accordance-notes and so forth (see p. DM 43)*

The general idea of the Th3 system, then, is that the user may view the "points" he wishes to make, variously upon the 3D viewing surface. Successive drafts, then, will all be projections, geometrically, from this interior structure of points.

Finally, the unifying idea that gives the system simplicity is this: all views will be on faces of a cube.



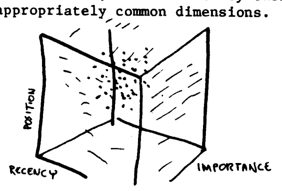
(FURTHER TECHNICALITIES OF THESE 'POINTS': Each point may have a value (numerical parameter) in any of a number of dimensions (which number may itself change). Such values may be null, as distinct from zero, showing that the point has no position on that particular scale.

Associated with each point may be one or more pieces and scraps or written material. Such scraps may be just phrases or single words. (Indeed, such scraps may be associated not just with a point, but with several specific values of a point.) Each scrap may also contain keywords.

Discrete relations between points may also be defined. There may be a variety of types of relation, which either exist between two points of don't.)

The crucial point here is that it's unified to the user: every version appears on a side of a box; and a typeset version is simply a magnified two-dimensional view in which the two dimensions are "position in overall text" (vertical) and "position on line" (horizontal).

Each side of a box may have a different view projected to it. This means that as many as three views of a specific cluster may be seen at once. However, for consistency these must have appropriately common dimensions.



By rotation and zooming the user may focus on the original pieces, and work with them, writing and revising.

Moreover, by using a combination of zoom and hardware clipping (as available on this equipment), the user may restrict his work to a specific range of material on particular dimensions.

GALAXY AND BOX

There are basically two views of what you are working with: the Galaxy and the Box. They appear in various manifestations, allowing you to study discrete relations and structures in the material; various "dimensions" of the material; alternate versions and drafts to be made from the material; and the complex collateration (see under "Thinkertoys") of different structures.

In what follows we will discuss the screen functions but not the control structures, which have not firmed up particularly.

1. GALAXY VIEWS.

The points are seen as a cloud of dots on the screen. If no view coordinates are supplied, the dots will be randomly positioned.

A. "Star Trek" effect.
Under a user's zoom control, the dots fly apart as if he is hurtling through space.

B. MAGNIFICATION. The user may "magnify" the dots, making each show its keywords, further text, and on up to the full Piece.

C. ROTATION. The 3D structure of the dots in space may be seen by the user at any time through short rotations.

D. Any relations that exist among the Points, insofar as they have been logged into the system, may be displayed among the points.

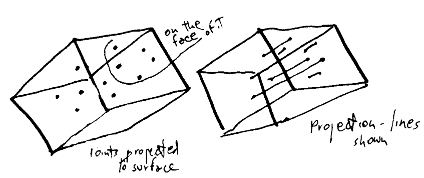
E. The user may sort the points by moving them with a lightpen.

F. The user may write within the individual pieces and splice them together, combining lightpen and keyboard operations.

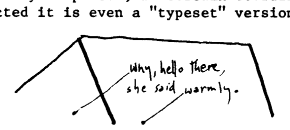
2. BOX VIEWS

In the Galaxy Views, the individual Points simply swarm about with no definable position. "Box Views" allow you to order the points on any dimensions that have meaning to you, in an arbitrary coordinate-space.

The box is more than a mere measurement-frame. On request the user may see the points projected on a specific face of the box (orthographically); and on request he may also see projection lines between a box-face and its corresponding point in the point cluster.



"Magnifying," as before, will create a view of the text: but in the box mode of viewing, the text appears on the side of the box. That is, the inner view will project to the outside, yielding a draft. Naturally, this is the current assembly of your pieces; if certain coordinates are selected it is even a "typeset" version.

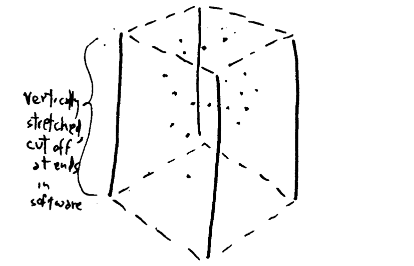


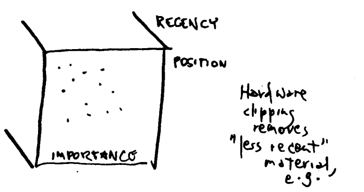
(Note: Vector General hardware does not allow character rotation; only keyword and headline rotation is possible, through software character generation. Thus text pieces on the side of a box show certain freaky movements if the side is not viewed square-on.)

*At the 1971 Spring Joint Computer Conference, I think it was, I was heckled by a linguist who accused me of being "unimaginative," insisting further that writing is merely an extension of speech and thus "merely" the application of further transformations; and he claimed further that what the user therefore needs is an input language to specify these transformations. This view, while interesting, is wrong.

A but/indeed control language might be interesting, however.

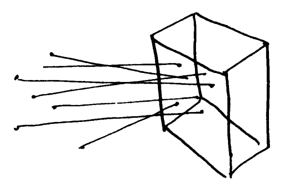
[Appended by the however-operation, a postfix "but." See "Writing," p. DM 43.]





COLLATERAL GALAXIES AND BOXES.

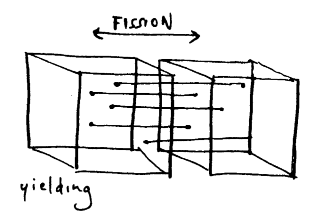
Viewing of collateral structures works through the same mechanism. Galaxies and boxes may be collaterated:

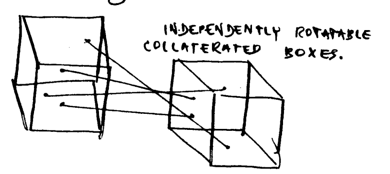


COMPLICATED NOTE: The extension of these mechanisms to pictorial graphics in two and three dimensions is straightforward, and to conceptual substructures (such as may exist) behind these graphics. The same goes for collateration and annotation of multidimensional cluster materials, e.g. in sociology: the system would allow, for instance, the viewing, annotation and collateration of sociometric clusterings.)

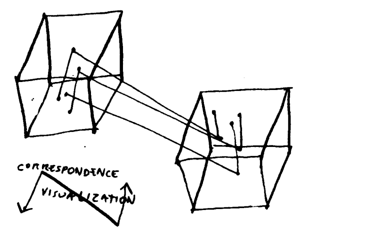
BOX FISSION. (The Beauty Part.)

For paired views of projections from the same cluster which do not share a common coordinate, a marvelous trick is possible: BOX FISSION. Starting with one box containing a galaxy, we pull it apart, making two boxes and two galaxies whose Points are linked.





Now both boxes can be rotated independently, and any view considered; equivalence-linkages may now be viewed between any two views. (The eye must, however, turn two corners.)



(It is interesting to note that the links in Box Fission are handled automatically, to an extent, by the hardware.)

WELCOME TO THE FUTURE. HUH?

This has summarized the development of some ideas for the viewing and manipulation of complex stuff. I offer this design, insofar as I have been able to present it here, as an example of fantic design (see p. DM 50). There is no logical necessity to it; it corresponds to the traditional structure of no technical system; it arises from no intrinsic or traditional data structures used for computer representation of these things.

But none of these considerations is to the point. This design has a certain stark logical simplicity; it extends itself plausibly from its basic outlook (or starting ideas, if you can isolate them) into a tool for truly intricate cross-consideration, without adding unnecessary and hard-to-remember "technicalities." At least that's how I think of it.

Obviously the aesthetics of it are important to the designer. But a more final criterion of its goodness-- its usefulness--may depend on the same parsimony and organizational clarity.

XANADU(TM)

(pronounced Zanna-Doo)

KUBLA KHAN: OR, A VISION IN A DREAM.

A FRAGMENT.

In the summer of the year 1797, the Author, then in ill health, had retired to a lonely farm house between Porlock and Linton, on the Exmoor confines of Somerset and Devonshire. In consequence of a slight indisposition, an anodyne had been prescribed, from the effect of which he fell asleep in his chair at the moment that he was reading the following sentence, or words of the same substance, in "Purchase's Pilgrimage:" "Here the Khan Kubla commanded a palace to be built, and a stately garden thereunto · and thus ten miles of fertile ground were inclosed with a wall. The Author continued for about three hours in a profound sleep, at least of the external senses, during which time he has the most vivid confidence, that he could not have composed less than from two to three hundred lines; if that indeed can be called composition in which all the images rose up before him as things, with a parallel production of the correspondent expressions, without any sensation or consciousness of effort. On awaking he appeared to himself have a distinct recollection of the whole, and taking his pen, ink, and paper, instantly and eagerly wrote down the lines that are here preserved. At this moment he was unfortunately called out by a person on business from Porlock, and detained by him above an hour, and on his return to the room, found, to his no small surprise and mortification, that though he still retained some vague and dim recollection of the general purport of the vision, yet, with the exception of some eight or ten scattered lines and images, all the rest had passed away like the images on the surface of a stream into which a stone had been cast, but alas! without the after restoration of the latter:

Then all the charm
Is broken—all that phantom-world so fair,
Vanishes and a thousand circlets spread,
And each mis-shape the other, Stay awhile,
Poor youth! who scarcely dar'st lift up thine eyes—
The stream will soon renew its smoothness, soon
The visions will return! And lo! he stays,
And soon the fragments dim of lovely forms
Come trembling back, unite, and now once more
The pool becomes a mirror.

As a contrast to this vision, I have annexed a fragment of a very different character, describing with equal fidelity the dream of pain and disease.—1816.

KUBLA KHAN.

In Xanadu did Kubla Khan
A stately pleasure-dome decree:
Where Alph, the sacred river, ran
Through caverns measureless to man
Down to a sunless sea.
So twice five miles of fertile ground
With walls and towers were girdled round:
And there were gardens bright with sinuous rills
Where blossomed many an incense-bearing tree:
And here were forests ancient as the hills,
Enfolding sunny spots of greenery.
But oh! that deep romantic chasm which slanted
Down the green hill athwart a cedarn cover!
A savage place! as holy and enchanted
As e'er beneath a waning moon was haunted
By woman wailing for her demon-lover!
And from this chasm, with ceaseless turmoil seething,
As if this earth in fast thick p ants were breathing,
A mighty fountain momently was forced:
Amid whose swift half-intermitted burst
Huge fragments vaulted like rebounding hail,
Or chaffy grain beneath the thresher's flail:
And 'mid these dancing rocks at once and ever
It flung up momently the sacred river.
Five miles meandering with a mazy motion
Through wood and dale the sacred river ran,
Then reached the caverns measureless to man,
And sank in tumult to a lifeless ocean:
And 'mid this tumult Kubla heard from far
Ancestral voices prophesying war!
The shadow of the dome of pleasure
Floated mid way on the waves;
Where was heard the mingled measure
From the fountain and the caves.
It was a miracle of rare device,
A sunny pleasure-dome with caves of ice!
A damsel with a dulcimer
In a vision once I saw:
It was an Abyssinian maid,
And on her dulcimer she played,
Singing of Mount Abora.
Could I revive within me
Her symphony and song,
To such a deep delight 'twould win me
That with music loud and long,
I would build that dome in air,
That sunny dome! those caves of ice!
And all who heard should see them there,
And all should cry, Beware! Beware!
His flashing eyes, his floating hair!
Weave a circle round him thrice,
And close your eyes with holy dread,
For he on honey-dew hath fed,
And drunk the milk of Paradise.

"Is that the river that runs down to the sea?"

James Stewart
in
"The FBI Story."

Everything is Deeply Intertwingled.

Patent work on Xanadu is in progress.

Xanadu, friend, is my dream.

The name comes from the poem (nearby); Coleridge's little story of the artistic trance (and the Person from Porlock) make it an appropriate name for the Pleasure Dome of the creative writer. The Citizen Kane connotations, and any other connotations you may find in the poem, are side benefits.

I have been working on Xanadu, under this and other names, for fourteen years now.

Originally it was going to be a super system for handling text by computer (see p. 12 and 13). But it grew: as I realized, level by level, how deep the problem was.

And the concept of what it was to be kept changing, as I saw more and more clearly that it had to be on a minicomputer for the home. (You can have one in your office too, if you want, but that's not what it's about.)

Now the idea is this:

To give you a screen in your home from which you can see into the world's hypertext libraries.

(The fact that the world doesn't have any hypertext libraries-- yet-- is a minor point.)

To give you a screen system that will offer high-performance computer graphics and text services at a price anyone can afford. To allow you to send and receive written messages at the Engelbart level (see p.DM46). To allow you to explore diagrams (see p. DM19 and p. DM51). To eliminate the absurd distinction between "teacher" and "pupil."

To make you a part of a new electronic literature and art, where you can get all your questions answered and nobody will put you down.

* * *

Originally Xanadu was programmed around the Parallel Textface (see p. DM53). But as the requirements of the Parallel Textface were better and better understood, Xanadu became a more general underlying system for all forms of interactive graphic environments. Its data structure has Virtual Blocklessness and is thus well related to the smooth motions needed by screen users. Thus in its final form, now being debugged, it will support not only the Parallel Textface (see p. DM53), the Walking Net (see p. DM51), Stretchtext (see p. DM19), Zoom Maps (see p. DM19) and so on, but indeed any data structure that needs to combine complex linkages with fast access and rapid changes. Because the data structure is recursively extensible, it will permit hypertext (see p. DM44) of any depth and complexity, and the collateral linkage (see p. DM52) of any objects of contemplation.

Xanadu is under private development and should be available, if the economy holds, in 1976. Regrettably, first prices will not be at the $3000 level necessary for the true Home System. Exact equipment for the production version has not been selected. A number of microprocessors (see p. 44) are in serious contention, notably the Lockheed SUE, but there's something to be said for a regular mini. The PDP-11 is of interest (see p. 42); (so especially is its Cal Data lookalike-- unless DEC would like to build us a PDP-11X with seven modes of indirect display addressing. Are you reading this, Ken Olsen?) And here's a laugh: a company called IBM may in fact make a suitable computer, except that they call it the "3740 Work Station." So for those customers who want IBM equipment, maintenance and prices, with Xanadu software, it's a definite possibility.

So, fans, that about wraps it up. I'll be interested in hearing from people who want this system; many hardheaded business people have told me nobody will. Prove 'em wrong, America!

Of course, if hyper-media aren't the greatest thing since the printing press, this whole project falls flat on its face. But it is hard for me to conceive that they will not be.

Xanadu:

BRASS TAX

WHAT IT IS: the heart of the Xanadu system, now being debugged, consists of a highly integrated program for use on minicomputers ("software"-- see p. 36) or microprocessors ("firmware"-- see p. 44). It is an operating system with two programs: a highly generalized data management system for handling extremely complex data in huge files, and a generalized display system, married to the other, for handling branching animation and retrieval and canned display programs. These ordain retrievals by the data system. The Parallel Textface (see p. DM53) and the Walking Net (see p. DM51) are two such canned programs.

These internal systems are intended to be sold with consoles of various types, as illustrated nearby, for stand-alone turnkey use (see p. 13). Xanadu is self-networking: two on the phone make a network, and more can join.

LANGUAGES: Xanadu programs will not be made available in any higher languages, mainly because of their proprietary character, but also because the display routines (and some of the retrieval routines) must be programmed in machine language.

The system has its own under-level language, XAP (Xanadu Assembly Program). While two higher-level display languages, DINGO (Display Lingo) and X^{ult} (the ultimate?) are contemplated, these will not ordinarily be accessible to the user. The purpose of Xanadu is to furnish the user with uncomputerish good-guy systems for specific purposes, not a chance to do his own programming.

Important features of the data system are huge addressability (in the trillions of elements) and Virtual Blocklessness. For advantages of this latter, see Zoom Map, p. DM19.

COMPATIBILITY: because of its highly compacted and unconventional structure, it is not compatible with other operating systems (including time-sharing). Anyway, to put it on a larger machine is like having your Mazda driven around in a truck. Because it uses a line-drawing display (see p. DM 22-3) and therefore draws individual arbitrary lines on the screen repeatedly, it is not compatible with television either-- unless you point a TV camera at it, or the equivalent. Sorry.

STANDARDIZATION. Taking a lesson from the integrated work of various people whose work has been described in this book, we see that if you want a thing done right, you have to do it yourself. (Great Ideas of Western Man: one of a series.) My good friend Calvin Mooers with his TRAC Language (see pp. 18-21) has discovered that trademark is one way to nail this as a right.

Several levels of standardization are important with Xanadu. One, all Xanadu systems must be able to work with all Xanadu files (except for possible variations in screen performance and size of local memory). Now, there are those who would not be concerned for this sort of universality, and who might even try to make sure systems were incompatible, so that you had to buy accessories and conversion kits up and down the line. That is one of the things that must be avoided: "partial" compatibility, subject to expensive options and conditions, a well-known technique in the field.

By stabilizing the "Xanadu" trademark, I hope to prevent such shenanigans. Thus every accredited Xanadu system will offer full compatibility with the data structure, and either full performance or substitutes as necessitated by the hardware. The "Xanadu" trademark can thus in principle be made available to manufacturers abiding by all design features of the system.

Second, all Xanadu systems should be able to work with outside systems either through or off the net, if they conform to the unusual data rules required by the unusual design of the system. This assures that Xanadu systems will be compatible with any other popular networks. It also assures others who want to offer Xanadu-class services to system owners (through, e.g., conventional time-sharing) that if they adhere to the rules (see "Canons," p.DM58) they can play the game on a certified basis.

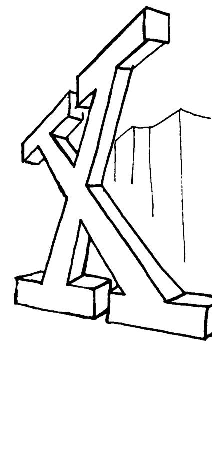
AVAILABILITY. It is hoped that Xanadu will be available in 1975 for at least one machine (guess which). As a program it will be available only in absolute form, without source or comments.

AHEM. There is a lot to talk about, but a lot of time can be wasted talking. It is suggested that thoughtful computer firms, interested in some form of participation, study this book carefully-- at least enough so no one's time need be wasted.

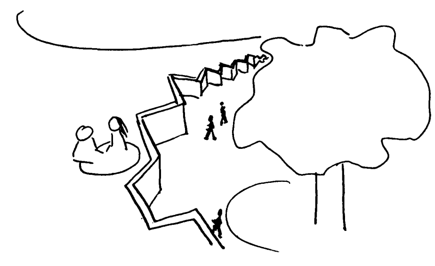
BIBLIOGRAPHY

"Nelson's the Name, and What He Proposes Could Outdo Engelbart." Electronics, 24 Nov 69, 97.

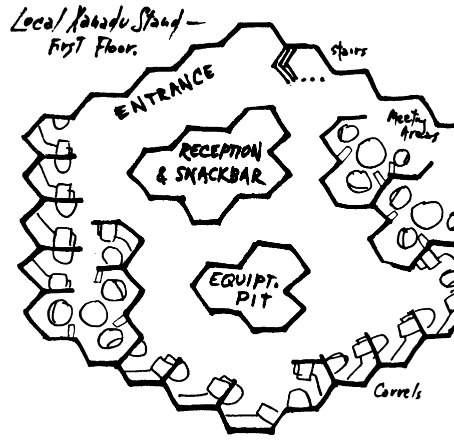
A recent report by Arthur D. Little, a Boston firm that makes its money by seeming to be omniscient, commented on the considerable market potential for on-line data supply systems. The report cost $400 or $4000, I forget which. Big-time interests are aprowl.

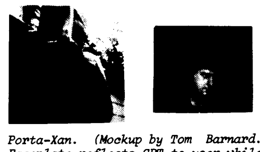


The Golden X's welcome the mind-hungry traveller.

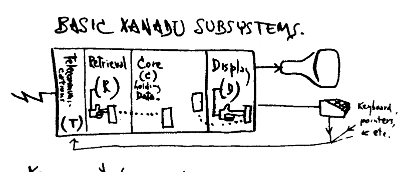


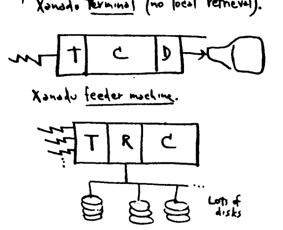
View from the snack balcony of a large Xanadu installation, overlooking the internal greenery. Hexagonal architecture permits physical expansion without interruption of services. (The mollusks have been telling us something about expansion.)





Porta-Xan. (Mockup by Tom Barnard.) Faceplate reflects CRT to user while he's abroad in the world. One-hand typewriting and pointing device frees the other. Can be built with available ruggedized components.





THE AUTHOR ANSWERS THE QUESTIONS HE IS MOST FREQUENTLY ASKED.

Q. If you publish your ideas like this, aren't you afraid someone will steal them?

A. No.
(The Law of Intellectual Property is about the strongest backing the individual has in this society.)
Besides which, there is here no revelation of the Xanadu Sneakrets.

Q. Won't some big company sweep your Xanadu under if they imitate it?

A. Let 'em. If they come up with a system having equivalent scope, which seems unlikely (see Canons, p.DM58), I might even feel I had achieved enough. But in the meantime, like the tortoise, and like DEC, I am going to continue to try to do it right.

Q. Aren't you afraid that writing a flippant book will keep people from taking you seriously?

A. I do not want to be taken seriously in some quarters until it's too late.

I have heard rumors that someone else in the field calls a computer product "Xanadu."
I tend to doubt this; and even if they did, my usage goes back to 1966.

I would like to thank (in chronological order) Elliot Klugman, Nat ("Kubla") Kuhn, Glenn Babecki, Cal Daniels and John V.E. Ridgway for the considerable time and involvement they gave to the Xanadu program design sessions; thanks also to various others who sat in from time to time. For the final selection of algorithms, however, no one is to blame but me.

I am grateful to the good offices of Swarthmore College for the use of their equipment in the continuing efforts to debug the Xanadu programs.

the XANADU NETWORK

First of all, bear in mind that Xanadu is a unified system for complex data management and display. This basically means that the same system (without the displays) can serve as a feeder machine for the data network itself.

So far, so good. That means that we can have a minicomputer network handling the entire structure: sending out library materials to users on call, and storing any materials they want saved. This saves all kinds of hassles with big computers and big-computer-style programming.

But who will pay for it? To build the kind of capacity we're talking about-- all those disks, all those minicomputers in a network-- won't it take immense amounts of capital? How, people ask me, will any American company ever back such a Utopian scheme?

Aha.

One method of financing has proven itself in the postwar suburban era, this time of drive-ins and hamburger stands.

Franchising.

What I propose, then, is the Mom-and Pop Xanadu Shop. Or, more properly, the Xanadu stand. "Mom and Pop" are the owners of the individual stand. But the customers can be families, too.

From far away the children see the tall golden X's. "Oh, Daddy, can't we stop? I want to play Spacewar," says little Johnny. Big Sis adds, "You know, I have to check something for my paper on Roman politics." And Mom says, "Say, that would be a good place for lunch."

So they turn in past the sign that says "OVER 2 BILLION SCREEN HOURS," and pull into the lot. They park the car, and Dad shows the clerk his Xanadu credit card, and the kids run to screens. Dad and Mom wait for a big horizontal CRT, though, because there are some memories they'd like to share together...

Sis's paper, of course, goes to her teacher through his Xanadu console.

THE PLAN. IS IT AS CRAZY AS IT SEEMS?

Deep inside, the public wants it, but people who think of computers in clichés can't comprehend it. This means "the public" must somehow create it.

One way to go is to start a new corporation, register it with the SEC and try to raise a lot of money by selling stock publicly. Unfortunately there are all kinds of obstacles for that. ("Reg A" is about as far as it will go.)

Through the miracle of franchising, now, a lot of the difficulties of conventional backing can be bypassed. The franchisee has to put up the money for the computers, the scopes, the adorable purple enamel building, the johns and so on; as a Xanadu franchisee he gets the whole turnkey system and certain responsibilities in the OVERALL XANADU NETWORK-- of which he is a member. He is assigned permanent storage of certain classes of materials, on call from elsewhere in the net. (Naturally, everything is stored in more than one place).

The Xanadu subscriber, of course, gets what he requests at the screen as quickly as possible-- or on priority if he wants to pay for it-- and may store his own files, including linkages among other materials and marginal notations to other things that can be called. (See collateral structures, p.DM52; these can automatically bring forth anything they're linked to. (See "Nelson's Canons," p.DM58.)) A user's historical record will be stored to whatever degree he desires, but not (if he chooses) in ways that can be identified with him.

Home users need only dial a local phone number-- their nearest Xanadu stand-- to connect with the entire Xanadu network. (The cost of using something stored on the network has nothing to do with where it is stored.)

(Special high-capacity lines need not be installed between storage stations, as appropriate digital transmission services are becoming available commercially.)

Various security techniques prevent others from reading a subscriber's files, even if they sign on falsely; the Dartmouth technique of scrambling on non-stored keywords is a good one.

The Xanadu stand also has private rooms with multiple screens, which can be rented for parties, business meetings, design sessions, briefings, legal consultations, lectures, seances, musicales, and so on.

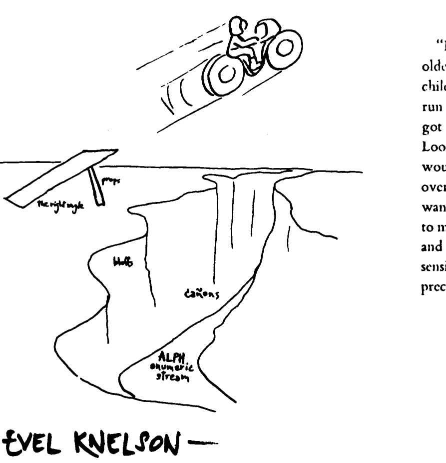
The choice locations for the Xanadu stands are somewhat different from hamburger spots. But that's probably not anything to go into here.

Within the Xanadu network, then, people may read, write, send messages, study and play.

****

XANADOODLES.™

fourteen years on a project and you get a little punchy.



EVEL KNELSON — WILL HE MAKE IT?

Thanks a lot, Sam Coleridge,
for those two symbols.

Xanadu.
And the Albatross.

"Listen," Mr. Wonka said, "I'm an old man. I'm much older than you think. I can't go on forever. I've got no children of my own, no family at all. So who is going to run the factory when I get too old to do it myself? *Someone's* got to keep it going—if only for the sake of the Oompa-Loompas. Mind you, there are thousands of clever men who would give anything for the chance to come in and take over from me, but I don't *want* that sort of person. I don't want a grown-up person at all. A grownup won't listen to me; he won't learn. He will try to do things his own way and not mine. So I have to have a child. I want a good sensible loving child, one to whom I can tell all my most precious candy-making secrets—while I am still alive."

Roald Dahl, Charlie and the Chocolate Factory, p.157.

I am sorry I have not been able to reply to all those who have written to me saying they wish they could work for The Nelson Organization at even a low salary.

So do I, my friends, so do I.

[As heard at the National Joint Computer Conference, 1973.]

How are we going to sell the Home Computer?
Well if you want to sell computers, let me tell you what to do:
You've got to talk to the housewives, and the children, too;
No one wants to program, they want something they can view...

It's got to offer fun, and it's got to offer truth;
It's got to give you something that'll lift you from the booth;
It's got to be uplifting to the Lady from Duluth.
You've got to have a vision; you've got to have an angle;
You should maybe sing a jingle (in a way that doesn't jangle);
It's got to have a tingle, in a way their minds can't tangle--

So continuing under our guidance inertial,
Let's have the XANADU SINGING COMMERCIAL.

[strings] It's got everything to give.
It'll get you where you live.

Realms of mind that you may roam:
[chimes] Grasp them all within your home.

The greatest things you've ever seen
[brass flourish] Dance your wishes on the screen.

All the things that man has known
[bass boutant] Comin' on the telephone--

Poems, books and pictures too
[TUBULAR BELLS!] COMIN' ON THE XANADU --

XAN-A-DU, OO--
[kettledrums]
THE-- WORLD-- OF-- YOUUUU!

Is Xanadu worth waiting for?

That depends, doesn't it,
on the value of the hand-bush
differential bird utility ratio.

fox box

quick brown

CRAZY LEICA FOX

WHAT NELSON IS REALLY SAYING

Told so that anybody can understand it without a Ph.D.

and maybe some with.

From "Barnum-Tronics"
(citation p. DM 2.)

1) Knowledge, understanding and freedom can all be advanced by the promotion and deployment of computer display consoles (with the right programs behind them).

2) Computer presentational media, coming soon, will not be technically determined but rather will be new realms for human artistry. This point of view radically affects how we design man-machine systems of any kind, especially those for information retrieval, teaching, and general writing and reading. Some practitioners see such systems as narrowly technical, with the computer hoisting up little pieces of writing on some "scientific" basis and showing them to you one grunt at a time. A Metrecal banquet. I disagree. The systems should be opulent.

3) The problem in presentational systems of any kind is to make things look good, feel right, and come across clearly. The things that matter are the feel of the system, the user's state of mind, his possible confusion, boredom or enthusiasm, the problems of communicating *concepts*, and the very nature of concepts and their interconnection. There will never be a "science" of presentation, except as it relates to these things.

4) Not the nature of machines, but the nature of *ideas*, is what matters. It is incredibly hard to develop, organize and transmit ideas, and it always will be. But at least in the future we won't be booby-trapped by the nature of paper. We can design magic paper.

It is time to start using computers to hold information for the mind much as books have held this information in the past. Now information for the mind is very different from "information for the computer" as we have thought of it, hacked up and compressed into blocks. Instead we can stretch the computer.

I am proposing a curious kind of subversion. "Let us design," I say; and when people see the systems, everybody will want one. All I want to do is put Renaissance humanism in a multidimensional responsive console. And I am trying to work out the forms of writing of the future. Hypertexts.

Hypertexts: new forms of writing, appearing on computer screens, that will branch or perform at the reader's command. A hypertext is a non-sequential piece of writing; only the computer display makes it practical. Somewhere between a book, a TV show and a penny arcade, the hypertext can be a vast tapestry of information, all in plain English (spiced with a few magic tricks on the screen), which the reader may attack and play for the things he wants, branching and jumping on the screen, using simple controls as if he were driving a car. There can be specialized subparts for specialized interests, instant availability of relevancies in all directions, footnotes that are books themselves. Hypertexts will be so much better than ordinary writing that the printed word will wither away. *Real writing by people,* make no mistake, not data banks, robot summaries or other clank. A person is writing to other people, just as before, but on magical paper he can cut up and tie in knots and fly around on.

I believe in calling a spade a spade -- not a personalized earth-moving equipment module; and a multi-dimensional spade, by gum, a hyperspade-- not a personalized earth-moving equipment module with augmented dirt access, retrieval and display capability under individulaized control.

I want a world where we can read the world's literature from screens rather than personally searching out the physical books. A world without routine paperwork, because all copying operations take place automatically and formalized transactions occur through formalized ceremonies at consoles. A world where we can learn, study, create, and share our creations without having privately to schlepp and physically safeguard them. There is a familiar, all-embracing motto, the jingle we all know from the day school lets out, which I take quite seriously: "No more pencils; no more books; no more teachers' dirty looks." The Fantic Age.

From "Computopia and Cybercrud."
(Citation nearby.)

MINIFESTO

My work is concerned principally with the theory and execution of systems useful to the mind and the creative imagination. This has polemical and practical aspects: I claim that the precepts of designing systems that touch people's minds, or contents to be shown in them, are simple and universal: making things look good, feel right and come across clearly. I claim that to design systems that involve both machines and people's minds is art first, technology second, and in no way a derivative specialty off in some branch of computer science.

However, presentational systems will certainly involve computers from now on.

Since hundreds of such systems are now being built, many of them all wrong, we must teach designers (and certain others) the basics of computers, and give them some good examples to emulate (such as Sutherland's Sketchpad, Bitzer's PLATO, and, I hope, some of my own designs).

Further, the popular superstitions about computers must be fought-- the myths that they are mechanistic, scientific, objective or independent of human intent and contemplative involvement.

NELSON'S CANONS

A Bill of Information Rights

It is essential to state these firmly and publicly, because you are going to see a lot of systems in the near future that purport to be the last-word cat's-pajama systems to bring you "all the information you need, anytime, anywhere." Unless you have thought about it you may be snowed by systems which are inherently and deeply limiting. Here are some of the things which I think we will all want. (The salesman for the other system will say they are impossible, or "We don't know how to do that yet," the standard putdown. But these things are possible, if we design them in from the bottom up; and there are many different valid approaches which could bring these things into being.)

These are rules, derived from common sense and uncommon concern, about what people can and should have in general screen systems, systems to read from.

1. EASY AND ARBITRARY FRONT ENDS.

The "front end" of a system-- that is, the program that creates the presentations for the user and interacts with him-- must be clear and simple for people to use and understand.

THE TEN-MINUTE RULE. Any system which cannot be well taught to a layman in ten minutes, by a tutor in the presence of a responding setup, is too complicated. This may sound far too stringent; I think not. Rich and powerful systems may be given front ends which are nonetheless ridiculously clear; this is a design problem of the foremost importance.

TEXT MUST MOVE, that is, slide on the screen when the user steps forward or backward within the text he is reading. The alternative, to clear the screen and lay out a new presentation, is baffling to the eye and thoroughly disorienting, even with practice.

Many computer people do not yet understand the necessity of this. The problem is that if the screen is cleared, and something new then appears on it, there is no visual way to tell where the new thing came from: sequence and structure become baffling. Having it slide on the screen allows you to understand where you've been and where you're going; a feeling you also get from turning pages of a book. (Some close substitutes may be possible on some types of screen.)

On front ends supplied for normal users, there must be no explicit computer languages requiring input control strings, no visible esoteric symbols. Graphical control structures having clarity and safety, or very clear task-oriented keyboards, are among the prime alternatives.

All operations must be fail-safe.

Arbitrary front ends must be attachable: since we are talking about reading from text, or text-and-picture complexes, stored on a large data system, the presentational front end must be separable from the data services provided further down in the system, so the user may attach his own front-end system, having his own style of operation and his own private conveniences for roving, editing and other forms of work or play at the screen.

2. SMOOTH AND RAPID DATA ACCESS.

The system must be built to make possible fast and arbitrary access to a potentially huge data base, allowing extremely large files (at least into the billions of characters). However, the system should be contrived to allow you to read forward, back or across links without substantial hesitation. Such access must be implicit, not requiring knowledge of where things are physically stored or what the internal file names may happen to be. File divisions must be invisible to the user in all his roving operations (FREEDOM OF ROVING): boundaries must be invisible in the final presentations, and the user must not need to know about them.

3. RICH DATA FACILITIES.

Arbitrary linkages must be possible between portions of text, or text and pictures; annotation of anything must be provided for; collateration (see p. DM 50) should be a standard facility, between any pair of well-defined objects; PLACEMARK facilities must be allowed to drop anchor at, or in, anything. These features imply private annotations to publicly-accessible materials as a standard automatic service mode.

The AI people don't understand,
the IR people don't understand,
the CAI people don't understand,
and for God's sake don't tell IBM.

I believe that an introduction to any subject can be humorous, occasionally profound, exciting, vivid, and appealing even to experts on their separate levels.

Perhaps someday I can prove it.

4. RICH DATA SERVICES BASED ON THESE STRUCTURES.

The user must be allowed multiple rovers (movable placemarks at points of current activity); making possible, especially, multiple windows (to the location of each rover) with displays of collateral links.

The system should also have provision for high-level mooting (~~see p.~~) and the automatic keeping of historical trails.

Then, a complex of certain very necessary and very powerful facilities based on these things, viz.:

A. ANTHOLOGICAL FREEDOM: the user must be able to combine easily anything he finds into an "anthology," a rovable collection of these materials having the structure he wants. The linkage information for such anthologies must be separately transportable and passable between users.

B. STEP-OUT WINDOWING: from a place in such an anthology, the user must be able to step out of the anthology and into the previous context of the material. For instance, if he has just read a quotation, he should be able to have the present anthological context dissolve around the quotation (while it stays on the screen), and the original context reappear around it. The need of this in scholarship should be obvious.

C. DISANTHOLOGICAL FREEDOM: the user must be able to step out of an anthology in such a way and not return if he chooses. (This has important implications for what must really be happening in the file structure.)

Earlier versions of public documents must be retained, as users will have linked to them.

However, where possible, linkages must also be able to survive revisions of one or both objects.

5. "FREEDOM FROM SPYING AND SABOTAGE."

The assumption must be made at the outset of a wicked and malevolent governmental authority. If such a situation does not develop, well and good; if it does, the system will have a few minimal safeguards built in.

FREEDOM FROM BEING MONITORED. The use of pseudonyms and dummy accounts by individuals, as well as the omission of certain record-keeping by the system program, are necessary here. File retention under dummy accounts is also required.

Because of the danger of file sabotage, and the private at-home retention by individuals of files that also exist on public systems, it is necessary to have FIDUCIAL SYSTEMS FOR TELLING WHICH VERSION IS AUTHENTIC. The doctoring of on-line documents, the rewriting of history-- cf. both Winston Smith's continuous revision of the encyclopedia in Nineteen Eighty-Four and H.L. Hunt's forging of historical telegrams for "The White House"-- is a constant danger. Thus our systems must have a number of complex provisions for verification of falsification, especially the creation of multilevel fiducials (parity systems), and their storage in a variety of places. These fiducials must be localizable and separate to small parts of files.

7. COPYRIGHT.

Copyright must of course be retained, but a universal flexible rule has to be worked out, permitting material to be transmitted and copied under specific circumstances for the payment of a royalty fee, surcharged on top of your other expenses in using the system.

For any individual section of material, such royalty should have a maximum: i.e., "by now you've bought it."

Varying royalty rates, however, should be the arbitrary choice of the copyright holder; except that royalties should not vary sharply locally within a tissue of material. On public screens, moving between areas of different royalty cost must be sharply marked.

BIBLIOGRAPHY

Theodor H. Nelson, "Computopia and Cybercrud." in Roger Levien (ed.), Computers in Instruction (Rand Corporation, 1971).

Theodor H. Nelson, "A Conceptual Framework for Man-Machine Everything." Proc. NCC 73.

"Rascal am I? Take that!"
Errol Flynn
in "Swords of Valor" (?)

FLIP OUT.

> I have had a most rare vision. I have had a dream, past the wit of man to say what dream it was: man is but an ass, if he go about to expound this dream. Methought I was—there is no man can tell what. Methought I was,—and methought I had,—but man is but a patched fool, if he will offer to say what methought I had. The eye of man hath not heard, the ear of man hath not seen, man's hand is not able to taste, his tongue to conceive, nor his heart to report, what my dream was.
>
> *Bottom the Weaver*

Now you see why I brought you here. This Gem-maniacal book has, obviously, been created as a crossroad of several cross purposes: to furnish a needed, grabby layman's introduction to two vast but rather inaccessible realms; to present a coherent, if contentious, point of view, and unroll a particular sort of apocalyptic vision after preparing the vocabulary for it; to make bright friends and informed supporters for my outlook and projects; to get home to some of my friends the fact that what I am doing is at bottom not technical; and finally, if nothing else, to set forth some principles about the way things should be, which others will have to answer if they propose to do less. Thus, overall, this book is a message in a Klein bottle, waiting to see who's thirsty.

I suppose it all started in college. Swarthmore left me with an exaggerated notion of the extent to which ideas are valued in the academic world; it took two graduate schools to clear this up. After that, as far as I was concerned, Ph.D. stood for Poophead. But I still cared about ideas, and the deep necessity of finding their true structure and organization. From writing I knew the grueling difficulty of trying to make ideas get in order. I believed in the pure, white light of inspiration and the power of the naive but clever mind to figure out anything, if not obstructed but dumb dogmas and obtuse mental schemata fostered by the educational system.

When I finally got the idea of what computers were about, sometime in 1960, I took endless walks at night trying to hash these things out and see where they led. The text systems came clear to me, at least in their beginnings, in a few weeks; the realization that 3D halftone was possible came to me as a shock the following spring, I believe as I was walking across Radcliffe Common. Since then trying to build these systems for creation and the true ordering of intricate thought has been my driving dream.

My own life among these dream machines has been a nightmare, thoroughly unpleasant, and if people are right in telling me that nobody wants systems like the ones I am designing, I'll get the heck out of this and be a disk jockey or a toy salesman or something.

I first got into this as a writer; all I wanted was a decent writing system that would run on a computer. Little did I realize the immensity of what that entailed, or that for some reason my work and approach would engender indignation and anger wherever I went. There is a fiction that everybody in these fields is doing something fundamentally scientific and technical, and this fiction is usually upheld in carefully enacted mutual playlets. Trying to cut through that and say, "Let's build a home for mankind that will at last be shaped to fit man's mind," does not seem to generate immediate warmth and welcome.

But I'm glad for the friends I've made in this field, and of course there have been a lot of laughs. (I'd really have hated to miss being in this field, just for the thrilling madness of it all.) All in all my adventures have been a sort of participatory journalism, which I'd like to write up properly some time. Some highlights:

The days of madness in '68, trying to start an honest corporation to do all this stuff, and suffering endless lunches with Wall Street hangers-on who were looking for a vehicle to take public. They wanted another chicken-franchise type company, though, and certainly not ideas.

Being briefed by four different corporations, most of them major, on the fantastic powers their interactive-movie system was going to have. One of these briefings was in the board room of a famous skyscraper. And now, only one of those systems is left-- Kodak's.

Then there was the courtly gentleman who was going to be my Noah Dietrich, my Colonel Parker. He assured me that through his business connections all was going to go marvelously, and then later intimated that as a special favor he was going to put me in touch with other universes and the flying saucer people. I just didn't have time for other universes.

Then there was the suppression of my first book (this is my second). You might say it was a misunderstanding, at least on my part. My boss's understanding was evidently that the advancement of my ideas would be detrimental to his. If it had been a question of free speech in Yugoslavia it might have been different. Well, it takes a long time to get a book together, but here we go again.

Then there was the time I was called in as a consultant on a vast federal system, never mind what. Numerous computer programs were to be coordinated by a hypertext system they had created and they wanted to know if they'd designed it right. It took months to find out from the programmers exactly what the system was, so I ended up writing the manual; after which I explained what was wrong with the project and the whole hypertext system was scrapped. And my job with it. I never quite got the swing of consulting.

Flying coast-to-coast with the president of a large corporation, he and I planned the whole Xanadu budget for the following year at something like half a million dollars. Two years later, reduced in circumstances and driving a yellow cab in New York, the miserable vehicle breaks down in front of those same corporate headquarters. And the reason I had that bad taxi was that I was out of favor with the taxi dispatcher, on account of having been absent the previous week-- I had had to fly to California to give a banquet address at the Rand Corporation.

Then there were my adventures with the CIA.

I was sitting in my office at Vassar, sagely advising a student, when the phone rang and the caller identified himself as John W. Kuipers, head of computer research at the CIA. He told me I had been noticed as a new bright young man in the field, and would I like to work for them?

Now, there is something about being a cynic and a romantic. (They go together: the cynic deflates ideas, the romantic falls in love with them.) It is not impossible for the cynical romantic to surmise that because everything he has seen personally turned out to be so lousy, that the true hope may lie at the heart of the vortex, just where everybody thinks is impossible. Also the Kennedy aftermath, when sophisticated people had learned to laugh at simple idealism as a facade for the real wheel-and-dealing, slap-and-tickle, may have had something to do with it; anyway, I was enchanted. Thus began the Kuipers Caper.

YES, THERE IS A McLEAN, VIRGINIA

I was given a handler named Bob, a jolly fellow, who kept assuring me that much money was just around the corner. I was regaled with success stories of other people in the computer field who really, undercover Worked for Them. (They weren't doing anything very exciting.) I got to show my slides in the CIA office building in Arlington, and to see there very fancy display equipment behind shielded (!) double-doors in a shielded (!!) computer room-- shielded to keep any planted bugs from transmitting out the contents of the computers' working registers. I even got to visit the main CIA "campus" in McLean, Virginia, where the sign says Agricultural Research Station. It is an incredible feeling to walk across that big eagle in the terrazzo, and to be given the visitor's badge that says "United States Government" all in wiggly lines.

They told me that they would be glad to set me up in business as a hypertext company, but I would have to have a corporation, because that was the way they always did things. And so it came to pass that The Nelson Organization, Inc. was founded at the express request of the United States Central Intelligence Agency. I wouldn't have had it any other way. If life can't be pleasant it can at least be surrealistic.

... BUT NO SANTA CLAUS

I was encouraged to write proposals for them, and write proposals I did. (I happened to finish typing the first one during a lightning storm, and lightning crashed just as I was signing the page; I felt like Faust.) I explained how hypertext might have prevented the Bay of Pigs. After due consideration, I did not say what hypertexts might have done for the Warren Report. Numerous jolly phone calls assured me that my first $25,000 was just around the corper.

The break came when Bob called me and asked me to rewrite a proposal one more time. He had circulated it, he said, among various people "at the shop," who he reminded me were holders of advanced degrees, and it had been remarked that they found my proposal meaningless: "Every place you say 'hypertext' you could just as well put 'gobbledygook' instead; you'll have to clear that up a little."

That did it. They couldn't read either. Who turns out to be in charge of computer stuff in the heart of the CIA, the inner sanctum, the nest of vipers, but the same old poopy Ph.Ds. I decided to resuscitate my virtue.

As far as I know, there is still not a Decent Writing System anywhere in the world, although several things now come close. It seems a shame that grown men and women have to rustle around in piles of paper, like squirrels looking for acorns, in search of the phrases and ideas they themselves have generated. The decent writing system, as I see it, will actually be much more: it will help us create better things in a fraction of the time, but also keep track of everything in better and more subtle ways than we ever could before. But nobody sees this-- I suppose it's only writers and editors that know they're trying to "keep track of ideas"-- and I have been unable to get this across to anybody. (The professional writers, of course, won't talk to me either.)

So here I am after fourteen years with exactly two systems to show for it: the main one, Xanadu, the text-and-animated-picture network system, and Fantasm (I shouldn't have spent the time but it was a labor of love), the simulated-photography system. Actually, I don't have either of them to show, it's all just flowcharts, but it turns out that if I work on either of them with university equipment, my work of fourteen years gets confiscated. So much for that; the outside expedients for debugging continue.

And, to lighten the burden, I've finally given up on trying to reach professionals, who evidently need a thick gravy of technicalism to make the obvious palatable; with this bookity I am taking my case to The People. It is there, anyway, out in Consumerdom, that the real action is going to occur. So the important thing is for everybody to know what's really possible, and what they could have. That is why I have shot off my big canons (and this epistol).

To me, you see, this is really a holy crusade, whereas I know guys to whom it's just a living. It's no less than a question of freedom in our time. The cases of Solzhenitsyn and Ellsberg remind us that freedom is still not what it should be, anywhere. Computer display and storage can bring us a whole new literature, the uniting and the apotheosis of the old and the new; but there are many who would not necessarily want to see this come about. Deep and widespread computer systems would be tempting to two dangerous parties, "organized crime" and the Executive branch of the Federal government (assuming there is still a difference between the two). If we are to have the freedoms of information we deserve as a free people, the safeguards have to be built in at the bottom, now. And the opulence which is possible must be made clear to everyone before we settle on an inferior system-- as we did with television.

Some people have called my ideas and systems "Orwellian." This is annoying in two ways. In the first place it suggests the nightmare of Orwell's book Nineteen Eighty-Four, which obviously I want no part of. (But hey, do you remember what that world of 1984 was actually like? The cryptic wars against unseen enemies that kept shifting? The government spying? The use of language to twist and manipulate? To paraphrase Huey Long: "Of course we'll have 1984 in America. Only we'll call it 1972.")

The second reason the term "Orwellian" is offensive is that it somehow reduces the life of Orwell, the man, to the world of "1984." This is a shallow and shabby thing to do to a man who spent his life unmasking oppressiveness in human institutions everywhere.

In the larger sense, then-- in homage to that simple, honest, angry man, who cared about nothing more than human freedom-- I would be proud indeed if my systems could be called Orwellian.

That reminds me. Nowhere in the book have I defined the phrase "computer lib." By Computer Lib I mean simply: making people freer through computers. That's all.

Fantically-- or fanatically--
Yours for a better world,
Before we have to settle for Any--

Ted Nelson