THE CARE AND FEEDING OF YOUR HOME COMPUTER by Bill Etra

SOL - THE INSIDE STORY by Lee Felsenstein

ALTAIR AND THE ART OF MOTORCYCLE SHOP MAINTENANCE

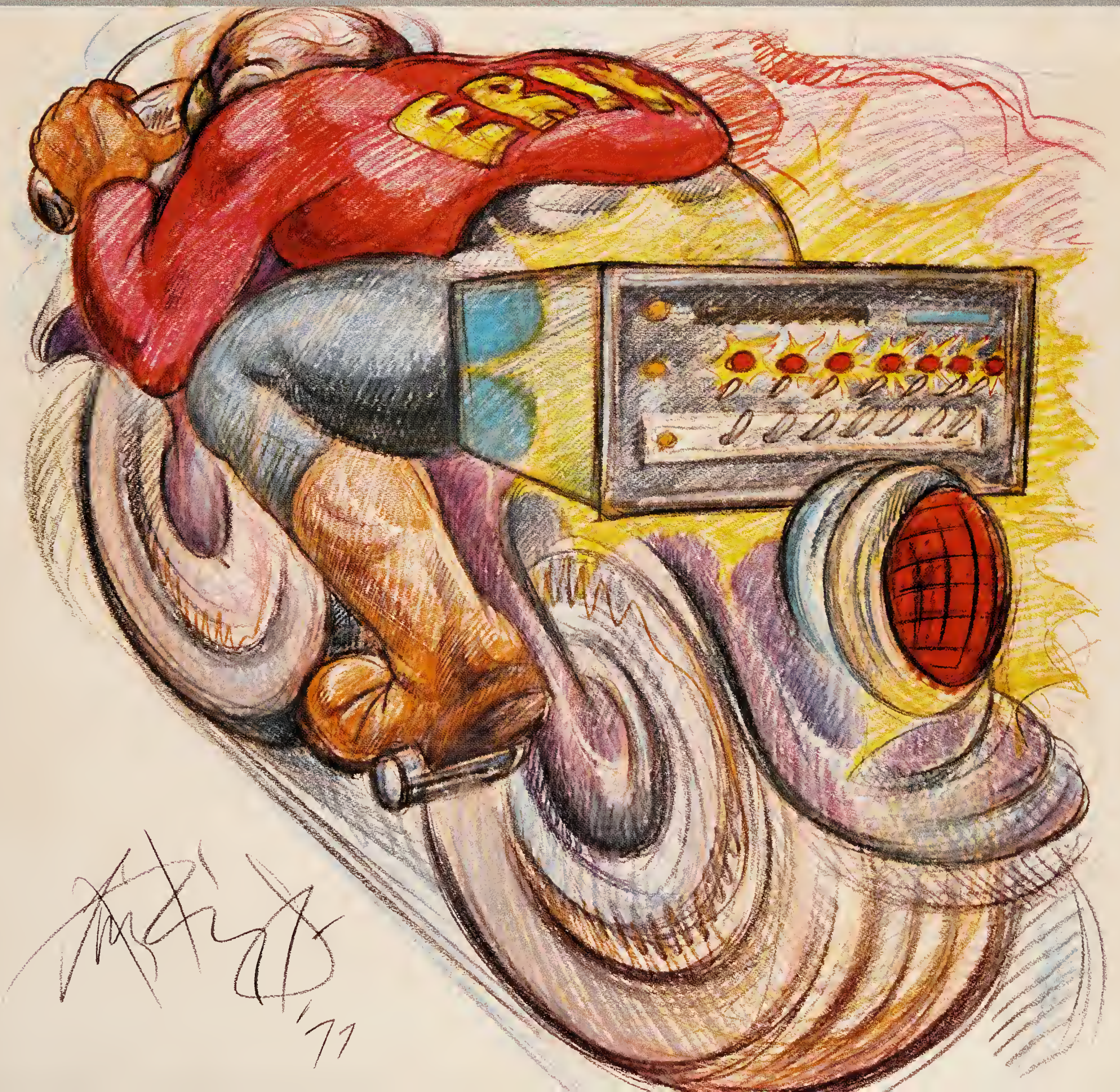
TOOLS by Joseph Weizenbaum

ROM

COMPUTER APPLICATIONS FOR LIVING

DIGITAL FOAM
The Sexiest
Peripheral
Ever

Volume 1, Number 1 $2.00



/ability.

It Comes Naturally With The Altair™ 8800b

The Altair 8800b from MITS: the second generation design of the microcomputer that started it all. The mainframe that has the abilities everyone is demanding from microcomputers today:

Expand-ability:

The Altair 8800b power supply and one-piece, 18-slot motherboard allow efficient and easy expandability for memory and I/O options. All Altair PC boards are designed to give you maximum capability/lowest power usage possible per board. This means that for each slot used you get more features and require less power, than with any of the "off-brand" Altair-bus-compatible boards.

Whether you buy an entire system up front or choose to expand gradually, it's easy to get the configuration you need with the complete family of Altair peripheral equipment, including floppy disk, line printer, audio cassette record interface, A/D converter, PROM programmer, serial and parallel I/O boards, choice of four different memory boards and many others.

Reli-ability:

The unique design features of the Altair 8800b, which have set the standard for the microcomputer industry, make it the most reliable unit of its kind. The Altair 100-pin bus, the now-standard design used by many imitators, has been "standard" all along at MITS. The unique Front Panel Interface Board on the Altair 8800b isolates and filters front panel noise before it can be transmitted to the bus. The all-new CPU board utilizes the 8080A microprocessor, Intel 8224 clock generator and 8216 bus drivers.

Flex-ability:

Meeting the diversified demands of an ever-increasing microprocessor market requires flexibility: not just hardware flexibility but software flexibility as well. MITS software, including the innovative Altair BASIC language, allows the full potential of the Altair 8800b computer to be realized.

8K ALTAIR BASIC has facilities for variable length strings with LEFT$, RIGHT$, and MID$ functions, a concatenation operator, and VAL AND STR$ functions to convert between strings and numbers.

Extended ALTAIR BASIC allows integer, single and double precision variables, automatic line numbering and renumbering, user-defined string functions, PRINT USING for formatted output and a powerful EDIT command for editing program files during or after entry. Extended statements and commands include IF . . . THEN . . . ELSE, LIST and DELETE program lines, SWAP variables and Trace On and Off for debugging.

Disk ALTAIR BASIC has all the features of Extended BASIC with the additional capability to maintain sequential and random access disk files. Utilities are provided for formatting disks and printing directories.

In all versions of ALTAIR BASIC you get the ease and efficiency of BASIC for the solution of real world problems.

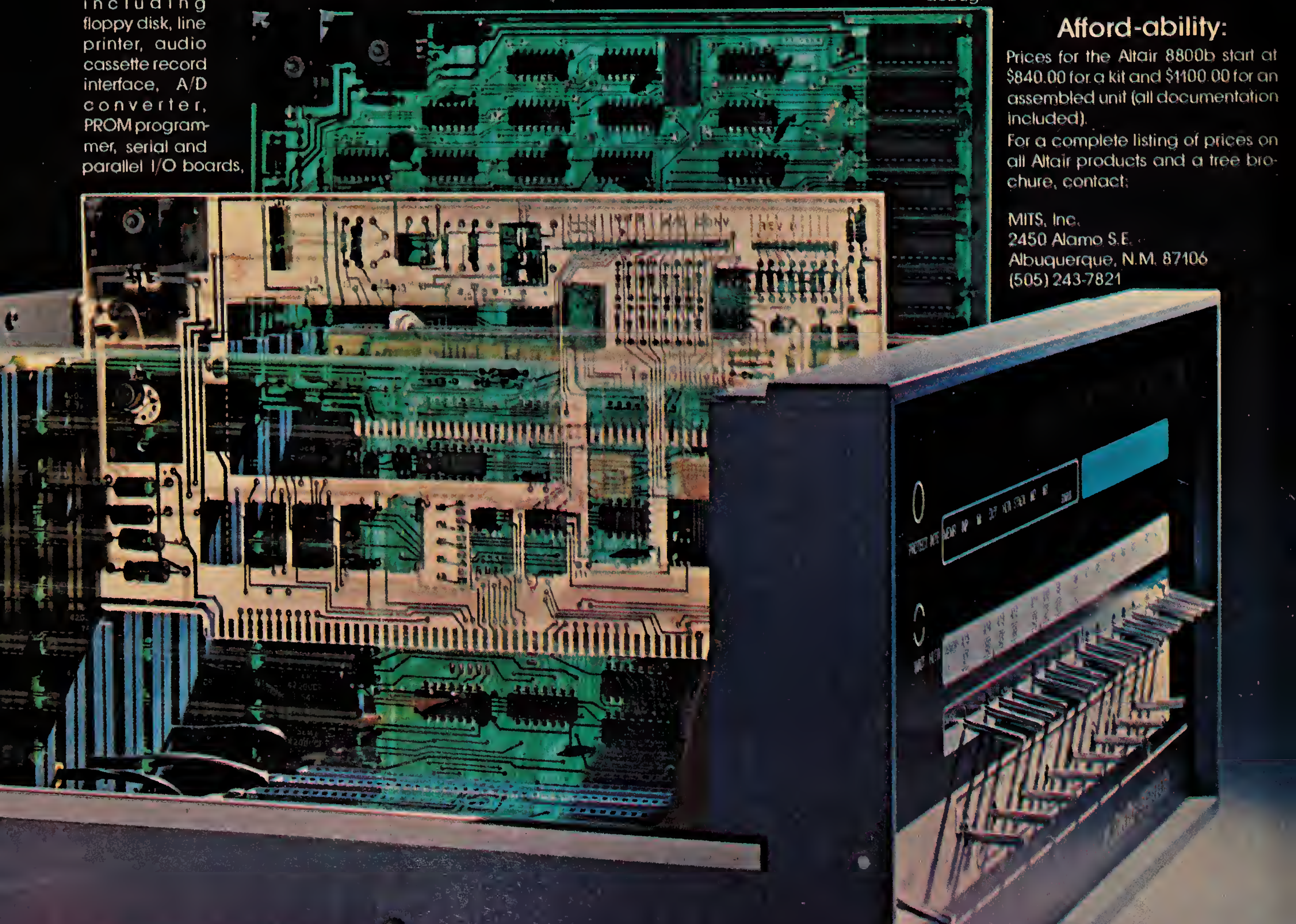
Package II, an assembly language development system for the Altair 8800b, includes system monitor, text editor, assembler and debug.

Afford-ability:

Prices for the Altair 8800b start at $840.00 for a kit and $1100.00 for an assembled unit (all documentation included).

For a complete listing of prices on all Altair products and a free brochure, contact:

MITS, Inc.
2450 Alamo S.E.
Albuquerque, N.M. 87106
(505) 243-7821



NOTE: Altair is a trademark of MITS, Inc.

Volume I, Number 1

Contents

July 1977

FEATURES

COLUMNS

DEPARTMENTS

ROM is published monthly by Rom Publications Corporation, Route 97, Hampton, CT 06247 (Tel. 203-455-9591). ∇ Application to mail at second-class postage rate pending at Hampton, CT 06247. ∇ Membership in Audit Bureau of Circulation pending.

Missionary Position

by Ted Nelson

The myth is: Computers are oppressive, and this is good.

The myth is everywhere.

We see it in the computer systems that laymen collide with as victims: as students, as welfare clients, as hospital patients. We see it in the computer systems that people collide with as consumers: charge accounts, billing systems, credit-denial systems, account-shutdown procedures. We see it in the computer systems that employees have to learn to use without understanding.

Computer programmers have been fooled by it, and have in turn fooled others. Computer salesmen have been fooled by it, and have in turn fooled others. Computer manufacturers have been fooled by it, and have in turn fooled others. Laymen have been fooled by it, and have no way to turn. So they hate computers.

The myth, most simply expressed, is:

Computers are oppressive,
and this is good.

You have to have a number. Things have to go in on cards. "We can't find that out, it's on the computer." The computer "made a mistake." Too bad for you. "It can't be changed, it's on the computer." "Sorry, your name can't be over twelve letters." "You couldn't understand that, it's computer code." "It has to be set up exactly this way for the machine. Those are the only categories." You can't understand it, it's the computer's fault, it has to be that way, there is no recourse.

Progress means regimentation. Complication is good. To use computers means everything has to be changed into numbers. When you computerize a company, its systems of work have to be completely thrown out. A man in an expensive suit who uses baffling phrases must know a lot. There is only one computer manufacturer. Nobody could sell a lot of computers unless they were the best possible. If a computer product is complex and incomprehensible, it represents the latest and the best. If a thing is "very technical," it must be right. Computer people always know what they're doing. And progress has to hurt.

It's all one big damned lie.

Intimidation, mystification, regimentation—the everyday kit with which all too many computer people, and some computer companies, have gotten their way: the litany, the "God's Will," of the computer priesthood.

But every computer system was set up by somebody. All those restrictions and nuisances, however excused, come from the conscious choices made by people, somewhere along the line.

(I remember vividly a meeting with the guy who was nominally in charge of security for a big-computer installation. A colleague and I wanted to attach a graphic system to the big computer, because we didn't understand the game. I will never forget the malevolent grin with which this fellow refused our every request. He had an answer to everything, neatly couched in systems terminology, but the grin and the meaning were entirely clear: you *will* not attach a graphic system to my computer. The game was to prevent the growth of interactive, potenially independent systems. And prevent it he and the rest of them did.)

And now all this is changing.

Little computers, with easy-to-use programs, are cropping up all over. And many of the people working on these systems have a real commitment to the opposite idea: computers should be easy to understand and easy to use.

The impact of these developments on the business world is going to be formidable, profound, and revolutionary. Easy systems for accounting, typing, filing, scheduling, and personal data basing are appearing quickly. First to grab are the innovative small businesses, but larger firms are following this lead.

And a paradoxical situation is emerging. Those companies with big computers, big computer staffs, complicated procedures, and everything nailed down are looking on in horror as their competitors—the competitors who earlier lagged in catching up with "progress"—begin to run circles around them. For the age of "one person, one computer" is almost here—the age of one-by-one transactions by people who know what they're doing.

The structure of the computer world evolved not from real considerations but from marketing tricks. Big computers and batch processing, once necessary, became a way of locking out minicomputer manufacturers and interactive systems. But it will become plain for all to see: most usage of computers is best done on interactive minis and should have been all along. Large data bases need large *disks*, not large computers. Interactive input and query systems can eliminate forms, eliminate paperwork, cut down on the red tape we all hate. Small computers for typing can not only make typing accurate and virtually instantaneous, they can automatically file in multiple directions simultaneously, so that, for example, a letter automatically connects with all the files on which it bears. Interactive systems mean that people can understand what they are doing rather than being brutalized into dead-end clerical peonage.

Presently, armies of programmers are still employed doing complicated things that someone thinks are wanted. But when the smoke clears, a great deal of this is going to turn out to

The impact of these developments on the business world is going to be formidable, profound, and revolutionary.

have been make-work: ad hoc, ungeneralized, and unstructured programming elaborately interfaced to horrid operating systems. The general problem is not *more* programs but *better* programs. Particular and temporary programs are going to be replaced by general and simple ones. I believe that in the future we will find a few simple programs doing most of what's wanted in business. And a new generation of businessmen will see that computers can be easy and accessible.

Reliability will increase: instead of the system being down, individual *units* will go down; a guy will just borrow a computer from the next desk.

We are on the brink of the home-computer age. By 1980 there will be personal computers, I believe, in some ten million American homes.

A host of services for the hobby and home user will appear, evolving from the computer store. Right now, a computer store is a place where you go to buy a computer kit; you're lucky if there's an assembled unit in stock. Tomorrow, the computer store will be an expanded service emporium as well, where you pick up your printout (like film from the corner drugstore, except you tweedled it in over the phone), where you take lessons, rent terminals, copy program cassettes, time-share, say hello to your friends. The terminal cluster at tomorrow's computer store may be like the potbellied stove in the general store of a generation ago—a place where you swap gossip and whittle, even if the whittling is done on a data structure rather than a physical piece of wood. When they get their act together, many of these stores will expand in franchised chains like McDonald's and 7-11.

What is not generally recognized yet is that there will be considerable overlap between home and business computer systems: my estimate is that they will have about 70 percent of their programs in common. All comers need text handling, retrieval, scheduling, financial planning, bookkeeping. Home users, however, do not need order processing and inventory; businesses do not need games.

The new developments in computer usage will cause convulsive changes, not merely among users, but within the computer community itself.

Now the programming community contain many Good Germans—doing what they're told and not wondering about it—as well as a few Nazis, who enjoy oppression and know full well what they are doing. But as people at large begin to find out how basically simple computers are, how easy and how incredibly useful in everyday life, it's not necessarily handshakes all around. I think it likely that there will be a lot of anger and a lot of hard questions asked. And the people who made computers oppressive, as well as artifically and intentionally complicated, have a lot of explaining to do.

This may lead to a period of agonizing reappraisal and collective guilt not unlike what followed World War II, when the Good Germans had to walk through concentration camps and confront what they had been contributing to. Some computer people themselves will come to feel that they have been living a fraud. Combine this with sudden unemployment across the field as fewer programmers are needed, and we are going to see suicides, weird religious movements, and perhaps strange political developments among the mortified castoffs.

The good goals are still attainable. On our way to a happier world, a better world, a more knowing world through computers, certain unfortunate circumstances have arisen. We can find our way out of them. There will be painful dislocations, but we can get there—to a world where the messy crud is taken care of automatically and information comes to us where, when, and how we want it. A world with a lot more knowledge spread around in it—and a lot more fun.

And it may be that access to information, *real* access under people's total control, may yet make this a better world—may reverse the tides of apathy and illiteracy that rise daily.

Great changes are in store.

A spade is a spade.

The emperor has no clothes.

The true frontier is not technical complication. It is simplicity and clarity.

The human oppression and degradation of the first computer era are coming to an end. The new age of computing will not build on the past, but repudiate it. We—peoplekind—could have used the last ten years. But let us see what we can do in the time that remains. ▼

"Harry dear, Leslie's computer wants to know if yours can come out and play Star Trek."

Korkie

SOFTWARE: THE GENIE IN THE BOTTLE by Tom Pittman

THE MEMORY PROBLEM by Ted Nelson

MEMORY, MEMORY, HOW MUCH MEMORY? by Stan Veit

APLomania by Eben F. Ostby

ROM

COMPUTER APPLICATIONS FOR LIVING

BINARY CLOCKS IN 0011/0100 TIME

Volume 1, Number 2
August 1977 / $2.00

Powerful in computing muscle, yet small in physical size, the Altair™680b offers many special features at an affordable price. Based on the 6800 microprocessor, the 680b comes with 1K of static RAM, Serial I/O port, PROM monitor and provisions for 1K of PROM as standard components. It's good thinking, when you're interested in making a modest investment on a highly reliable computer, to consider the Altair 680b.

Our PROM monitor eliminates the necessity for toggling front panel switches to load bootstraps or manipulate memory contents. Only a terminal and programming language are required for complete system operation. With Altair System software—Altair 680 BASIC, assembler and text editor—you may begin problem solving immediately with ease.

By adding the 680b-MB Expander card, many options are currently available:

*16K Static Memory Board—Increase your system memory with 16K bytes of fast access (215 ns), low power (5 watts per board) static RAM. 680 BASIC and assembler/text editor are included free with purchase.

*Process Control Interface—A PC card that uses optically isolated inputs and relay outputs that transmit sensory information to and control signals from the computer. A diverse world of control applications is opened up with the Altair 680b-PCI.

*Universal Input/Output Board—If your I/O needs exceed the serial port already on the main board, augment your I/O channels with the 680b-UI/O. By implementing the optional serial port and two parallel ports, you can simultaneously interface to four terminals.

*New Addition—Kansas City Audio Cassette Interface—Use the 680b-KCACR to interface your Altair 680b with an audio cassette recorder for inexpensive mass storage of programming languages, programs and data.

Available in either full front panel or turnkey models, the Altair 680b presents many computing capabilities at a low cost—without skimping on performance. See it today at your local Altair Computer Center or contact the factory for further details.

mits

Good Thinking.

2450 Alamo S.E. Albuquerque, New Mexico 87106
dealer inquiries invited.

Volume I, Number 2

Contents

August 1977

FEATURES

COLUMNS

DEPARTMENTS

ROM is published monthly by ROM Publications Corporation, Route 97, Hampton, CT 06247 (Tel. 203-455-9591). Application to mail at second-class postage rate pending at Hampton, CT 06247. Membership in Audit Bureau of Circulation pending.

THE MEMORY PROBLEM

by Theodor Nelson

People have a funny way of talking about the memory problem, as if it had a single simple solution that would be along any day now. "Soon there will be a disk—a new strange device—a better language—or a chip—that will solve The Memory Problem."

It'll never happen.

Basically, the memory problem is that there isn't enough fast memory, directly addressable, on your computer. (Or anybody else's.) Core memory, or whatever they're calling it these days, isn't quite big enough for the get-it-done-ers, isn't nearly big enough for the dreamers. This was true at the beginning and is now and will be later. This is true for your Altair and true for the PDP-10 down the block and true for the Illiac at Burpleson AFB. The memory problem is that there is never, *ever* enough. (In this it strongly resembles The Money Problem, as well as The Sex, Quiche, Time, and Energy Problems.)

Everybody has his own myth about the solution to the memory problem, if any.

Many solutions have been proposed. Obviously we want a generalized solution, meaning one that will work in the next state.

Some say the solution is more hard work. This has the advantage of always being possible and never requiring more equipment. It leads people to filling every spare bit and loose corner. It leads to programs which can never be modified. It leads to job security and bad programs.

A very popular interim solution may be stated more briefly. *More.* Especially, *more core.* This works temporarily for the get-it-done-ers who realistically only needed a little more. It is of no use for the dreamers. But soon even the former group will allow themselves to dream again, a little, and it's time for more core again. (The most parsimonious phrasing of the memory problem may be this: you never *can* get enough main memory. The faster you enlarge your computer, the sooner you need a bigger one.)

The hardware dinks—those who (at the upper level) think of software only as a stopgap, or who (at the lower level) think most programs fit on one page, and besides, what they like about home computers is the soldering, think—have always thought, always will think—that there will soon be some piece of equipment—some new kind of tape, or chip memory circuit, or radical new computer architecture—that will take care of the memory problem.

The classic such solution was to have been *associative memory:* special circuitry that would cause all data items with certain traits or connections to stand up and be counted. Supposedly this would take care of everything. Well, now there's a machine like that on the market—the Goodyear Staran—and it costs a great deal of money for a minicomputer-sized unit. Presumably it's a fine machine—but you need what we call a "well-tailored, special application" to justify it.

Now the Texas Instrument bubble chip has come along, and we hear the same sorts of things. 92K bits! For a hundred dollars! (Soon.) And fast, supposedly, though with new complications. Soon it will hook on the Altair and brethren. Is this the solution?

No.

A floppy disk, the disk itself, costs under ten dollars, and holds a quarter of a million characters that you may want permanently. If you put things permanently in the bubble chip, it will cost you a hundred. There's the problem.

Because programs ordinarily take place in addressable main core, the natural programming assumption is, "Get everything into addressable core." But there is the mistake. Main core can never be big enough. The distinction between core, disk, and tape is perpetual. There will *always* be core, disk, and tape. Not literally; I mean rather that there will always be some kind of fast, expensive internal memory, some kind of slower, cheaper mass memory, and some kind of *much* slower, cheaper mass memory.

This means that programs must always be *tiered:* some things must go fast and go on always, staying in core. Some things must come and go, from and to the slower regions. The management of fetching and storing is a fundamental aspect of computer programming. Doing this sort of thing right, or failing to do it right, constitutes much, perhaps most, of what goes on under the name of programming: getting and putting, backing up, updating, swapping, and overlaying, as well as the more advanced tribulations of access methods, cache and page fault optimization, allocation of overflow areas, and so on. Computer techniques are, in large measure, ways of handling this tiered memory: what's on slow-and-cheap memory is waiting for its chance to get into small-and-fast.

Professional programmers in the 1950s were excited to get into 8K of core. Now the day has come when 64K on your Imsai doesn't seem like quite enough.

Part of the problem is how much more we've come to expect. We want to chain our programs. We want to mix languages. We want to bring several data structures in at once. Now it's not enough to crunch a few numbers and print a few lines of text; we want to bring in TV camera images and put out whole orchestras via D-to-As. These things call for memory, memory, memory.

One very generalized solution to the memory problem is somehow to make all memory, fast and slow, behave as though it were one: set up a huge time-sharing complex wherein we do not notice any distinction between core and other forms of memory. Vast tracts of empty core stretch before the programmer, like the endless prairie that greeted the pioneers of the 1840s. (This is regrettably called "vitual memory," though it is in fact *virtual huge core.*) Of course this works only on time-sharing computers, where swapping is the name of the game and we might as well swap an individual's core by the acre while we're at it.

Such systems work; some work well. The thing is that they only work on a vast scale, and at big-company prices. So for those of us who need personal computing, they're no solution at all. (But at least it's awfully generalized: they've had operating systems of this type running each other as subprograms several deep. For some, recursion is its own reward.)

Another generalized solution is the back-end processor: a complete subsystem, attachable to your regular computer, that hands over data from within a grand information web as requested, shouldering all the file-management problems. That way the main computer just asks for a certain piece of something within the grand information web, just as if it were in some handy part of core, and lo! it appears. Such was Bachman's Integrated Data Store on the old General Electric computers; such are the minicomputer-based information systems now being backed up to 370s. I think it likely that such back-end storage and retrieval will soon appear as a telephone service for personal computing, necessarily at a far lower cost than conventional time-sharing. Such, I think, is also the solution to the hypertext problem: you can only put huge libraries on shared computer networks if you delegate the file management to very specialized retrieval subsystems.

But still there is no general solution to the memory problem-and there will be none. We will just get more core and more mass storage, and hope for cleaner and cleaner ways to interrelate them.

If hardware does not provide a "solution" to the memory problem, at least it keeps getting better and cheaper. Little stuff for the personal market is all over. Ten years ago I don't think there were any disks that cost under fifteen grand; now a dual floppy is around $2500.

The emerging personal computer market calls for a lot of similar redesign: for years they've been designing mass magnetic memories only to be used in big installations. Tapes, for instance. Computer tape drives were originally designed only for big-computer styles of application: you could only add things at the end, not replace in the middle. Then came the LINCtape, which was ahead of its time in 1962. It was developed at Lincoln Laboratory for the LINC personal computer, and it's still around; the one in my livingroom works very nicely. (Computer Operations makes it now; Digital Equipment's version is called a DEC tape.)

Same for disks: they were overdesigned. The old expensive disks all went thirty revolutions-per-second or faster; much later it was noticed that disks didn't have to be rigid; five revolutions-per-second was adequate for many purposes, thank you.

Same for the magnetic card memories. The RACE and CRAM of the sixties, for instance, had boxes of long vinyl strips the size of your arm, with magnetic recording on them. Fingers would select (by notches, as on a Linotype) the right strip; then the strip would shoot around a drum, divulge (or revise) its contents, and drop back into the box.

But what of the lowly Hollerith-size mag card, as on some typing devices? Now there's a recording medium, at about one dollar a card and 64K bits, that's about right for dinky computers—how about it?

Vinyl-iron technology is by no means exhausted. How about a megafloppy? You could drive a free-flying read/write head with compressed air and zip it around a sheet of any size. Perhaps card-table size would be about right; you could roll up the sheet to change contents.

There are openings for other weird things that exist but never quite have gotten around. For instance, sprocketed 16mm mag tape is used in the movie industry and on IBM's Mag Tape Selectric—does anybody have drives for that?

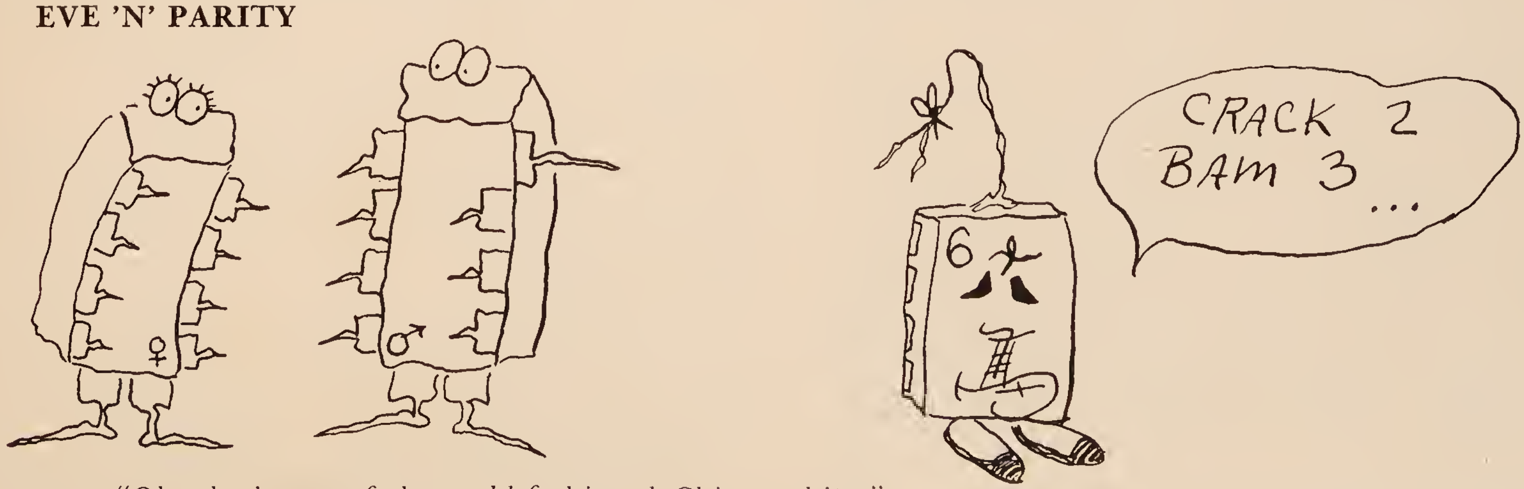
Personally, I'm still hankering after the floppy disk juke box somebody was offering last year, but the literature I sent for never came—are they still funct? (Then there was the "rotor memory" somebody advertised in *Datamation* in the early sixties. It appeared to be a polyethylene mixing-bowl covered with magnetic oxide. The literature on *that* never came either.)

Oh, well. Gadgets come and go; there are two sizes of floppies, two sizes of Philips cassettes, two sizes of 3M cartridge. But beyond this natter, there are definite trends.

Some things change and some don't. Soon you can have the whole book, *War and Peace,* in core; in a decade you will be able to have the whole *movie* of *War and Peace* in core, a million dots per frame. But by then you'll want more. Today's largest PDP-20 will seem small for a home computer not very long from now.

We may expect to see, heh heh, some dramatic changes in store. But the relation between levels of store is not likely to change much. ▼

EVE 'N' PARITY



"Oh, that's one of those old-fashioned Chinese chips."

Cartoon by Korkie

BIOFEEDFORWARD by Bill Etra

COMPUTER WRESTLING: THE PROGRAM OF CHAMPIONS

XEROXES & OTHER HARD COPY OFF YOUR CRT

A PAYROLL PROGRAM FOR YOUR SMALL BUSINESS

ROM

COMPUTER APPLICATIONS FOR LIVING

CHARGED COUPLES

Volume I, Number 3
September 1977/$2.00

Volume I, Number 3 September 1977

Contents

FEATURES

COLUMNS

DEPARTMENTS

ROM is published monthly by ROM Publications Corporation, Route 97, Hampton, CT 06247 (Tel. 203-455-9591). Application to mail at second-class postage rate pending at Hampton, CT 06247. Membership in Audit Bureau of Circulation pending.

Missionary Position

UPSTAIRS DOWNSTAIRS

by Theodor Nelson

Once a year, everybody in the computer field goes to some weird place, and they exchange brochures and say peculiar things in crowded rooms, and that is called the National Computer Conference.

This year it was in Dallas, and 35,000 people came, and lo there was much literature exchanged and beer swug and hot dogs et. And finally the national organizations that hold this shindig got around to recognizing that yes, there is something called Personal Computing. Portia Isaacson, who chaired the NCC this year (and also has a computer store in Dallas), made them include personal computing in the National Computer Conference.

The computer profession at last had to acknowledge the computer non-profession. Square computing met amateur computing head-on, and nothing happened.

This was my fourteenth National Computer Conference (counting the Joint Computer Conferences that came before), but for the last couple of years the action has been so heavy in personal computing that I'd almost lost the feel of what it's like in square computing, back where a mainframe is still a million bucks, or ten thousand. Here the two were side by side, making for a funny experience: you could pop back and forth between the two worlds, rather like an actor or a spy, experiencing the sudden feel of each repeatedly.

The main commercial exhibits were upstairs.

The personal exhibits were downstairs.

And it was just like in the British TV soap opera.

Upstairs they are haughty and pompous, and have no idea of how life is lived below. Downstairs they know what really goes on upstairs, and what closets the skeletons are in.

Upstairs they think the old ways will go on forever.

Downstairs they weep at the human consequences of the foolishness above, and pick up the pieces.

The big news upstairs, I think, was laser printers, and the "Pray for IBM" buttons that *Computerworld* was giving out. (Will anybody sell me one?) The rest was terminals and ever-improving, unexciting whatnot.

But the news downstairs will affect millions of people, which laser printers won't.

The big news downstairs was still the same as the big news in California in April—the Commodore Pet computer for six hundred bucks, including screen and keyboard and ROM BASIC. Except now they had production prototypes on the floor as well as the wooden model from California.

It was the biggest news on the grapevine in March; it was the biggest news at the West Coast Computer Faire in April—especially because it worked, though its rounded "futuristic" housing was plainly hand-shaped of wood—and it's still the biggest news, because now the cabinets seem to be production plastic and there's an order blank. Your $595 check must accompany coupon. It looks real.

I think they'll sell millions.

Commodore's strategy is very clever. They've acquired MOS Technology, makers of the 6500 chips and the KIM. The 6500 is inside the Pet. But the Pet Computer they sell for $595 is just the beginning, with its 4K of open memory, screen and keyboard and IEEE bus and ROM BASIC. Now for the bad news. The next 4K is another $200.

More software will come in ROM blocks, presumably blisterpacked, to be sold like Hot Wheels and Barbie Dolls. Commodore folks say they'll be delighted to publish software—as ROM blocks.

If the $50-a-K price for memory is any indication, though, the software is not gonna come cheap. In other words, they think they've got it boxed in.

So does Fairchild, with the Channel F. Channel F, to the public, is a red hot ziggety Video Game that hooks to your TV and costs $150. With a few games built in, it also takes a four-track tape cartridge that loads in *other* video games. At twenty bucks a cartridge (list).

That's the Channel F as the public sees it and as Fairchild presents it. As you and I see it, though, it's actually a dinky computer: inside is actually the Fairchild F-8 computer chip, several K of memory, a Dazzler-type bit-map display generator, and of course your video modulator that goes in through the antenna.

Wellsir! As I said in my after-dinner harangue at the Faire, there's a swell opportunity there for someone who wants to break open the Fairchild and help it be all that it is.

Just that has happened. Someone's broken into Fairchild's little Video Game and made out of it the computer it really ached to be.

Fairchild had talked as if you couldn't, but certainly didn't take any precautions: you scarcely have to do more than plug something into a socket that's already there to get a functioning F-8.

One booth at the NCC, Downstairs, was a father-and-son team who took the cover off the Channel F and found nice places to put a UART, making the device right off the bat into either a computer or a computer-driven graphics display. On top of which, a computer store says it will shortly offer a conversion kit.

This is really fun. Here is Fairchild, high and mighty, thinking it's got a lock on the box and only *its* chosen uses can be made of the product. Off comes the cover and ha ha ha, look what we can make it do, against Fairchild's wishes. But of course the last laugh will be Fairchild's, since to convert the box you've got to buy it. Even if they didn't see in the first place where home computers are going, there will be many hands, and conversion kits, to show them the way. (Certainly somebody else will be first to offer the box that *writes* tape cartridges for the Channel F.)

Which brings us back to Commodore. The Pet comes with its cover on, and the warranty will presumably say hands off. But how long will it be before the Pet has been opened, explored, adapted and kitted? How long till you can add S-100 bus, music and the rest?

Les Solomon, editor of *Popular Electronics,* gives it a couple of months. He expects a black market in Pet schematics before Christmas.

If the Commodore is good—and their stuff tends to be (I bought a printing calculator from them eight years ago, and it's still the best)—then this unit will be like the Volkswagen: a low-cost workhorse for many, chopped and channeled California style for many others.

Every year there has been *the* personal machine. In Year One of personal computing, 1975, it was of course the Altair. In Year Two it was the Imsai. This year it's the Sol. What's next, the Pet?

Hard to say. I don't think there will be "the" single *au courant* machine from now on, because the market is going to go so many ways.

The LSI-11, if indeed Heath is putting one out, will certainly be the top machine. But there will also be the S-100 world continuing, and below that the funny little dingbat programmables, and on a new branch of the tree the Pet-style machines.

(By next year the emphasis will have gone over to software anyhow.)

Some people from straight computing are tuning in to all this. Quite a few did come down to the personal computing exhibition. But a lot of people didn't.

Quite a few people, too, came to the personal computing sessions, but rather few—say, a few thousand, compared to the overall 35K that came to the NCC. Those that came to the personal sessions seemed an interesting menagerie: there were those already in the field; the curious, who might get into it; and others who had obviously been *sent.* Worried-looking, confused, usually middle-aged, these guys took the most notes; it seemed to me that their function was to report back to somebody what personal computing was about. But from the faces, and the style of note-taking, it was clear that the big picture wasn't getting across: those who were sent to report back won't be able to put it into words that their bosses can understand.

I went to a very nice buffet breakfast for the speakers. Two other guys were at my table, and they couldn't *imagine* what personal computers could possibly be for. They worked in large-scale simulation: the computer for them is something at the office that they work on, like a drill press.

In the text system session, I heard people tell how they use computers to get a lot of stuff printed out. I asked what ideas they had for personal writing systems. One panelist said he couldn't imagine anyone putting up funds for such a thing. Obviously he hadn't been downstairs.

In a session on small business systems, panelists told us that a small business system will cost about $40,000 for the hardware and $40,000 for the software. From the floor I asked for a show of hands as to who was working on software for the *cheap* machines: at least twenty-five hands went up.

From all these things I came away with this impression: the straight computing world still doesn't even know about its bastard child, its demon seed. I guess they think "personal computing" means merely the reclaiming of scrap machines, or doing odd programming tricks with pocket calculators. Something akin, in scope and profundity, to soap carving or toothpick model-building.

In a way that's true. People do do things like that, weird dead-end tricks, and in that frame of mind. And there are those who just like to solder.

But there's a much grander vision, and I think most of us in personal computing share it. It's the idea that your computer, yours, your truly own, will make your life better and simpler, easier and deeper and richer, both by taking care of your old problems and by providing you with new and wonderful things to do, alone or with those you love.

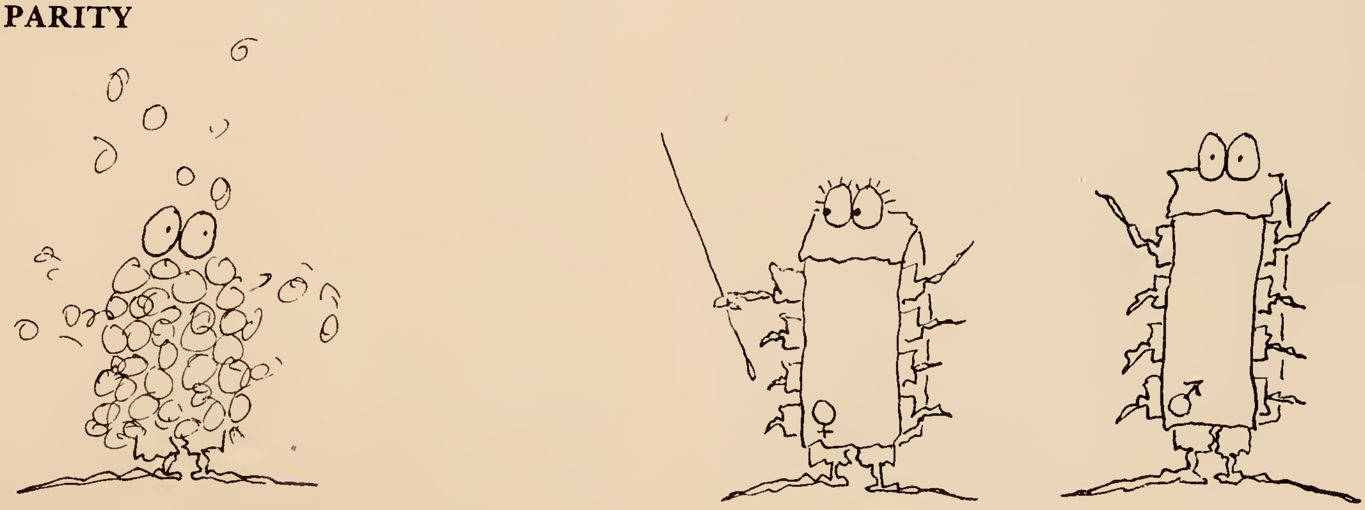
The computer will simply be a personal super-secretary and nerve center and visualization tool, for finances, writing, planning, idea-keeping. Taking the drudgery out of all these things and making everything simple. And an interactive movie-box that takes you to important worlds you must understand.

Words don't seem to help explain this to most people, though. There seems to be a certain necessary vision. Either it is all quite obvious to you or you will never be able to figure it out. Maybe in a while, when a good panoramic personal software system exists, it will be easy for anybody to get the idea, but it isn't yet. Meanwhile a lot of us just continue advancing slowly on the dream, on hands and knees.

What can you use a computer for in the home? Tell the squares you can pound nails with it. And hang in.

It may be that the distance between old-style computing and personal computing is narrowing, that they are drawing closer together. But it looks to me like the Titanic and the iceberg. ▼

EVE 'N' PARITY



"I'm going to puncture that bubble memory's ego bit by bit."

Cartoon by Korkie

GUARD AGAINST CRIB DEATH WITH YOUR MICRO

THE WONDERFUL DREAMS OF DR. K

THE COMPULSIVE PROGRAMMER by Joseph Weizenbaum

SCOTT JOPLIN ON YOUR SCI-FI HI-FI

ROM

COMPUTER APPLICATIONS FOR LIVING

KILOBYTE CARD
Memory for Pennies

Volume I, Number 4
October 1977/$2.00

Volume I, Number 4

Contents

October 1977

FEATURES

COLUMNS

DEPARTMENTS

ROM is published monthly by ROM Publications Corporation, Route 97, Hampton, CT 06247 (Tel. 203-455-9591). Domestic subscriptions are $15 for one year, $28 for two years, and $39 for three years. Canada and Mexico $17 for one year, $30 for two years, and $41 for three years. For European and South American subscriptions, please add $12 per year additional postage. For all other continents, please add $24 per year additional postage. Application to mail at second-class postage rate pending at Hampton, CT 06247. Membership in Audit Bureau of Circulation pending.

Missionary Position

ACCESS STRUCTURE AND LIFE

by Theodor Nelson

Starting with a simple technical distinction, which I've tried to generalize a little, I've come up with an interesting model of what we may call "access structure." It seems to have ramifications for a lot of matters: design of systems, storage, objects. And indeed, much of life is concerned with access structure.

This originally came to mind in the early sixties when I was thinking about mass storage. Obviously, there is a big distinction between information that's in core memory, where it can be gotten at instantly, and information that's on disk, requiring a fetch and consequent delay.

As I thought about this general distinction between near things and far things, it seemed to me applicable as well to everyday life. Mine, anyhow. People who don't understand how I look at access structure are dangerous to let into places I live. This form of analysis may or may not be useful in business or industry—they already have some inkling of it—but it may help us think about homes (you know, boxes for living), and the artifacts that people have in homes. I think homes are as a rule unspeakably irrational.

Suppose something is in your hand, a knife or a pencil or a pair of pliers, poised and ready for use. You've got it, it's ready, there is no impediment. Let us define this condition as *zero-order access* to this object.

Next, suppose the object is on the table and you have to pick it up. Nothing is physically in the way, but you do have to take the step. This is *first-order access.* One operation will bring the object to zero-order access; the space between your hand and the object is an impediment of sorts to be overcome by the intervening step.

But suppose now that some blocking object is in the way, perhaps in front of the object you want, perhaps just a sheet of paper covering it. This condition constitutes an impediment holding you back. The blocking object must be pushed aside, or the sheet of paper picked up, to render the object seizable. So now we say the object is at *second-order access,* because an operation stands between you and its first-order access.

Clearly the terminology may be extended indefinitely, to the tenth-and *nth*-order access. When we consider the way the world is laid out, and the different things we want to use, we begin to see compound access structures all over. (I'm not distinguishing here between sizes or shadings of accessibility. An intervening step is one access step, no matter how large or small. Obviously the model can be adjusted where this becomes inappropriate.)

This model is intended, among other things, to clarify and enrich the basic concept of "in the way": something which is "in the way" may be characterized for its exact effects on access structure.

There is a slight terminological problem here. Since this model of access structure is developed by induction on the basis of further and further removes from instant readiness, it makes sense to start at zero. Unfortunately, there is a clash between the phrase "high accessibility" and my term "low-order access"—which mean about the same thing. So access, as defined by access *order,* is the inverse of the more common term "accessibility."

Access may be serial or parallel. Serial or segregated access to a collection of things means we cannot get at them simultaneously: we can only get access to them one at a time but not jointly. Examples are conveyor belts and lazy susans.

Parallel or joint access means we can get at different items from a collection simultaneously and together, as from a bookshelf. In other words, several things are simultaneously at low-order access levels.

Many familiar objects are concerned with access structure. These we may call *access machines.* The most fundamental are counters and shelves.

To understand this, first think of putting things on the lawn or in a field. The ground is a two-dimensional place to put things, the elemental storage space.

If you've ever seen a lawn sale, you know how little it takes to fill up a lawn, especially if lanes are left to walk through.

The first thing we can do is raise the lawn up (except for the access lanes), so we don't have to squat down to pick things up—improving access by one order. Thus we have *counters.*

But more room is needed. The next step, conceptually, is to create further holding-surfaces or different levels. Rather than platform over the whole lawn, of course, we'll keep the access lanes between the holding-surfaces. The result is called *shelving.*

We spoke of the ground as the elemental storage space. Well, shelves permit several layers of ground to put things on, all in the same two-dimensional area. An access step is involved in reaching up or crouching down, but the packing density is much greater.

To help think about other access machines, first imagine yourself sitting on the ground with your things in a circle around you. Then only about ten small objects, laid around you, can be at first-order access. And that's pretty much it. But with pegboards and toolchests and swinging trays, a craftsman such as a cabinet-maker or a dentist can have hundreds of items at first and second-order access.

This is the glory of simple access machines. Access machinery includes shelves, lazy susans, pegboards, grabbing devices. (Drawers are merely enclosed shelves, as are cabinets. The extra access step is justified either for safety or prettiness.)

A word here against deep shelving. Shallow shelves permit first-order access; this is the spicerack principle. Nothing is blocked, you can see it all in parallel. But as soon as you start putting things *behind* each other, access deteriorates in several ways. To get at the things in back you'll have to either reach around the first stuff with diffi-

Illustrated by the author

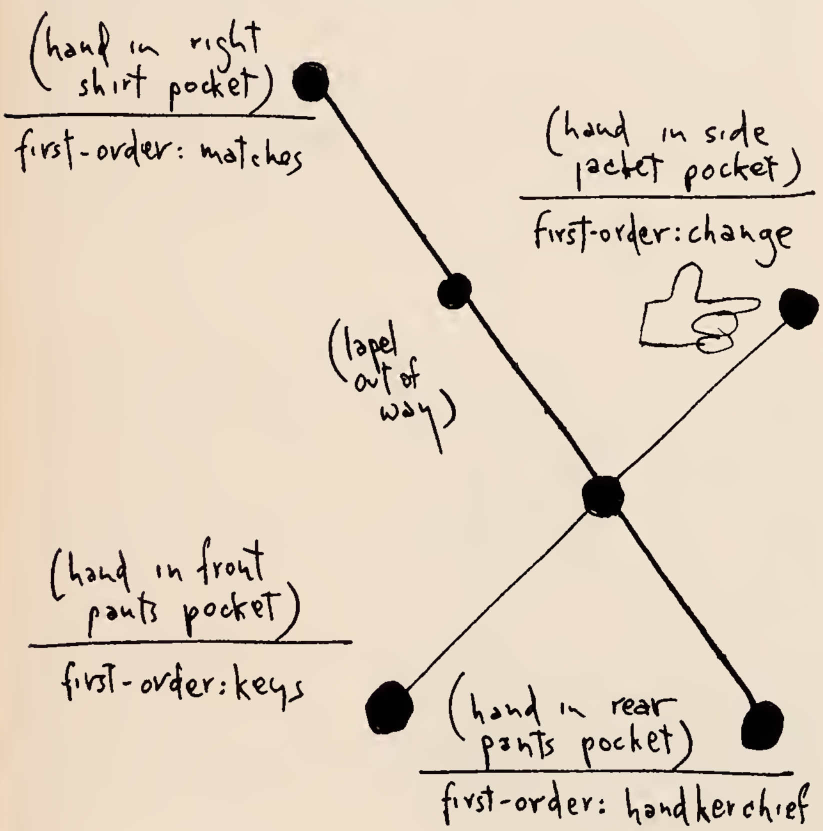
culty, which counts as a lost access step, or remove and buffer the front stuff somewhere else, like the floor, which makes at least two access steps (if you put the stuff back).

Worse, deep shelves and deep closets are not rational storage space, they are places to lose things. More on that later.

Because we do not usually think explicitly about these things, we do not have the access machines we ought to. At the turn of the century you could get modular mobile shelving—the glass-front sectional bookcases—but not now. Overhead hooking arrays, swinging-shelf arrays, all sorts of other access devices ought to be available commercially—but aren't.

Access analysis can be graphed; we can create access graphs of many objects and situations. Typically such a graph consists of points, lines between them, and a Reacher—person, hand or machine—which moves along the graph.

For example, we may graph the access structure of a man's right-hand pockets, and the Reacher, which is his right hand. The result is a star:



If the man's right hand is in the right jacket pocket, his change is at first-order access (since it must still be *grasped*), and the handkerchief in his pants pocket is at third-order access.

If the hand comes out, the change is now at second-order access and so is the handkerchief.

The map delineates states of a Reacher that moves between tasks: now it is closer to one thing, now to another. That is typical access structure, where a step toward one node tends to be a step away from another.

Access machines all have analyzable structures, some graphable. For instance, the access graph of a row of shelves is a two-dimensional grid. The access graph of a lazy susan is a circle which the Reacher touches at one point. A pair of pantographic grabbing tongs lowers the access order of all objects in its reach, since you no longer have to get out of your chair to glom onto them.

All the world's an access structure.

We may regard the world as consisting of process areas, buffers, storage, access lanes, and vistas. (I may have left out something, but never mind.)

Let's run through these.

A process area is where you (or some tool) have access to a thing to work on it.

A buffer is, of course, a place where you put things temporarily. Things do happen in buffer areas, though: dishes dry, dough rises, ideas jell, prisoners become old and discouraged.

Storage is where you put things non-temporarily. Storage is like buffering, but for a longer period. This may be as precise as we can get.

An access lane is a place that is kept clear (of either processes or buffering) in order to permit accesses through it.

A vista is a place that does not hold all the shelving it possibly could, or buffers or workspaces, for aesthetic reasons.

There are other places, but they do not concern us here. Now let's try on some insights:

- A chair is a body buffer; often it grants access to other access areas.
- A table is a counter with leg-space for closer access from a body buffer.
- Stairs and hallways and elevator shafts are access lanes.
- A theater is where people sit in body buffers offering visual access to a process area. (Some people prefer body buffers next to the access lane.)
- A factory is a place where workers have joint access to tools, materials, and products being created.
- A library is a place where books are stored and accessed. Sometimes they have privileged access areas (closed shelves.) There are also process areas where books are read and notes taken.
- An office is a place where papers are stored and accessed. New paper is generated from them at desks, which are, of course, just tables with drawers.

These principles apply as well to the private home. The kitchen is a prime example, as it employs both storage and workspace in an interpenetrating system.

The stove-top and oven are process areas. So is the sink. The dish drainer is a buffer, but one where the process of drying takes place. Kitchen tables and counters may be treated as both workspace and buffer, at the discretion of the user. (And at the initiative of guests. The sink, too, is often used as a buffer, but with very discouraging results.)

The living room, in many houses, is a collection of comfortable body buffers which may also be process areas (e.g. for conversation), a table which tends to hold magazines and drinks (storage and buffering) or be used for other purposes (e.g. a work surface for kit-building). Plus whatever vista pleases the residents, often with decorations symbolic of lifestyle.

The bathroom has process areas for washing and lounging, shallow shelves (cunningly hidden behind a mirror) for parallel access to body equipment and supplies, and, of course, an output buffer for organic material.

The bed is a process area for various bodily processes, though the access structure of what you can reach once you're in it will vary considerably.

Houses ought to have loading docks. Somehow, in the architectural image of the American home, there is still the idea of a little self-sufficient cottage. Yet, in fact, much of our waking lives is given over to the getting, loading, and unloading of objects. This, and not the much less frequent welcoming of guests, should be the focus of the major portal.

(I once lived in a house whose entrance to the basement was in the rear, and whose parking space was three-quarters of the way around, or 150 steps. Pine trees obstructed the short way. A ten-carton loading sequence, then, involved 3000 steps, or about a mile of superfluous trudgery.)

A large part of comedy has dealt with access structure, much of it around the home: Fibber McGee's closet, folding-bed jokes, inter-blocking and cross-prevention, the perils of trying to do several things at once.

Buffering—putting things aside or in piles—is a great problem in human life. Packing and unpacking, handling papers, handling anything. What shall we do with this thing? Put it on the back porch until—

Buffering has various causes, legitimate and less so.

The simplest buffering is when the things we buffer have well-defined destinations, but we don't want to make a separate trip around the house for each. (As when we unpack.) So we toss things in a pile and then walk the pile around. Here buffering is to save time.

Sometimes we buffer because the objects require treatment: bills to be sorted (paid if you're lucky), letters to be answered. And there may not be time to do so. Time-buffering again.

Unfortunately, this is also related to procrastination, which we, uh... won't get to right now.

Sometimes we don't know what we will (or can) do with something. A part may be lacking, or a plan, or the money, or the will. So we buffer out of uncertainty.

Sometimes we buffer for an overview. Getting ready for a trip, for example. "Do we have it all?" can be better answered if we've put everything for the trip in the foyer. Or, "Do we have enough?" when a picnic is imminent and nobody remembers what food is going, so we get it all together where we can see it.

And to sort is to buffer. Every destination pile when you sort is a buffer. (Often you can't have as many piles as the categories you mean to sort into, because you can't toss things far or accurately enough, or the room is too small. So you have to sort into piles which in turn must themselves be sorted. Levels of buffering.)

Project-involved people, who bring work home or have hobbies, need far more buffer space than people who don't. Indeed, projects tend to fill more and more space, making the home areas less and less satisfactory vistawise.

Of special interest is use of the bed as a buffer area. Project-involved people sometimes use the bed as the buffer space of last resort, in effect ransoming it to force completion or clearing of important work in progress—packing tomorrow's briefcase, say, or going through the mail. This represents a commitment that can only be shirked by pushing all the stuff onto the floor, or ignominiously sleeping in it.

There is a fundamental rift between shipshape people and buffering people. The shipshape principle, one approach to access structure, is that everything should be put back in a canonical storage space, in canonical condition, when it is not at zero-order (or first-order) access. This makes a lot of sense on boats, and some people use it as an organizing principle in their lives. But shipshapers are totally incompatible with project people, or any other bufferers.

We've spoken here as if you knew where everything is. How likely is that?

If you have enough shelf (drawer) space, you can store by category. That's great: just like a supermarket, aisle six for soup and canned meats. But the trouble is that you don't usually know at the outset how much there is going to be in each category, so you may end up with a lot of empty space and a lot of crowded space, and have to rearrange just to balance space.

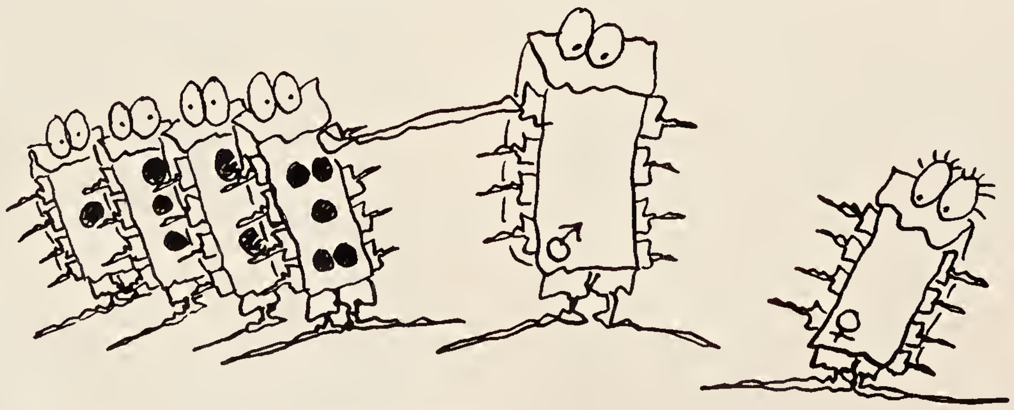
Another problem is that categories get a little sluffy. Consider the supermarket. You have probably never seen frozen worm patties or canned armadillo meat, but you probably know where they'd be if they were there. On the other hand, where do they keep *matches?* Stores do not seem to agree.

This is the problem of *conceptual clarity* of access structure. If you are going to find what you want, access structure has to have conceptual clarity.

It's great to have an eidetic memory, in which case you can just put anything anywhere. But if you don't remember perfectly where you put things, and especially if you are really absent-minded, there has got to be an *idea* of the right place to put each thing. And this idea has to be one you can remember, so you'll know where to look when you want things again.

Combine this with buffering and piling. Piles and buffer

EVE 'N' PARITY



"Watch this, Eve.... Domino Theory."

Cartoon by Korkie

spaces develop a conceptual unity; often you create a pile that has a certain meaning you can't put into words. But you know where it is and what's sticking out of it reminds you of *what* it is. (The visitor to my house is enjoined *never to combine two piles.* Each pile has a conceptual unity, but two randomly combined piles no longer have any meaning at all.)

Moving a pile, or "making it neat," thus has deleterious side effects of the worst kind. As does "straightening up," in the aunt-Harriet sense of rendering all straight lines orthogonal in consonance with the house's principal coordinate system. This destroys information about what is in each pile, which had been so easily seen from what had been sticking out diagonally.

(It is said that Hefner's round-bed editorial chamber is cleaned, first, by *mapping* the position of each object or pile of papers, then vacuuming and making the bed, then returning everything to where it was. Is this what Hefner really needs Polaroids for?)

"Straightening up" has a much more horrifying sense: *putting things in unknown places just so they don't show.* This is a capital crime in my abode.

And this is what leads to true mess. True mess is not how things look, it's a loused-up condition of access structure—exactly the opposite of what many neatness bugs think. Mess is a condition of information loss with respect to access structure.

When you lose information about where things are, it is nearly the same as losing the objects themselves. This leads to an important distinction. *Clutter* is merely what looks unappealing, and is of no concern in this analysis; mess is the presence of objects or collections *whose access structure is unknown.*

(This is another thing wrong with deep shelves and large closets: you don't see what's behind. Information *about* access is lost.)

Now we come to the Great Question.

Wouldn't it be nice to have everything always accessible? Zero-order access of more than one thing is impossible unless you're ambidextrous, or a one-man band. Even first-order access is usually impossible for more than ten things. So the question is rarely what is "accessible," but rather, what things are at what order of access.

If you had an immense array of shelving—say, a factory building two hundred feet square, with row on row of shelves, grafted onto your present house or apartment—it would make storage quite easy, wouldn't it. But look at the cost. Access costs and costs and costs.

Now, many people are trying to legislate us into keeping less. Architects who offer us cubes to live in, designers (like the authors of *Nomadic Furniture*), artists who have endorsed the throwaway principal (like Warhol and Les Levine). They tell us to keep nothing. So did Thoreau.

Well, that's their bag. I speak for access, for insights to help you keep things conveniently and get at what you keep so you can use it. It's not that we keep too much. It's that we haven't decent access equipment. Meaning, among other things, attics and hallways with good shallow shelves. ▼

PROJECT PROMETHEUS by Lee Felsenstein

HOW DOES YOUR CRYSTAL GROW by Sandra Faye

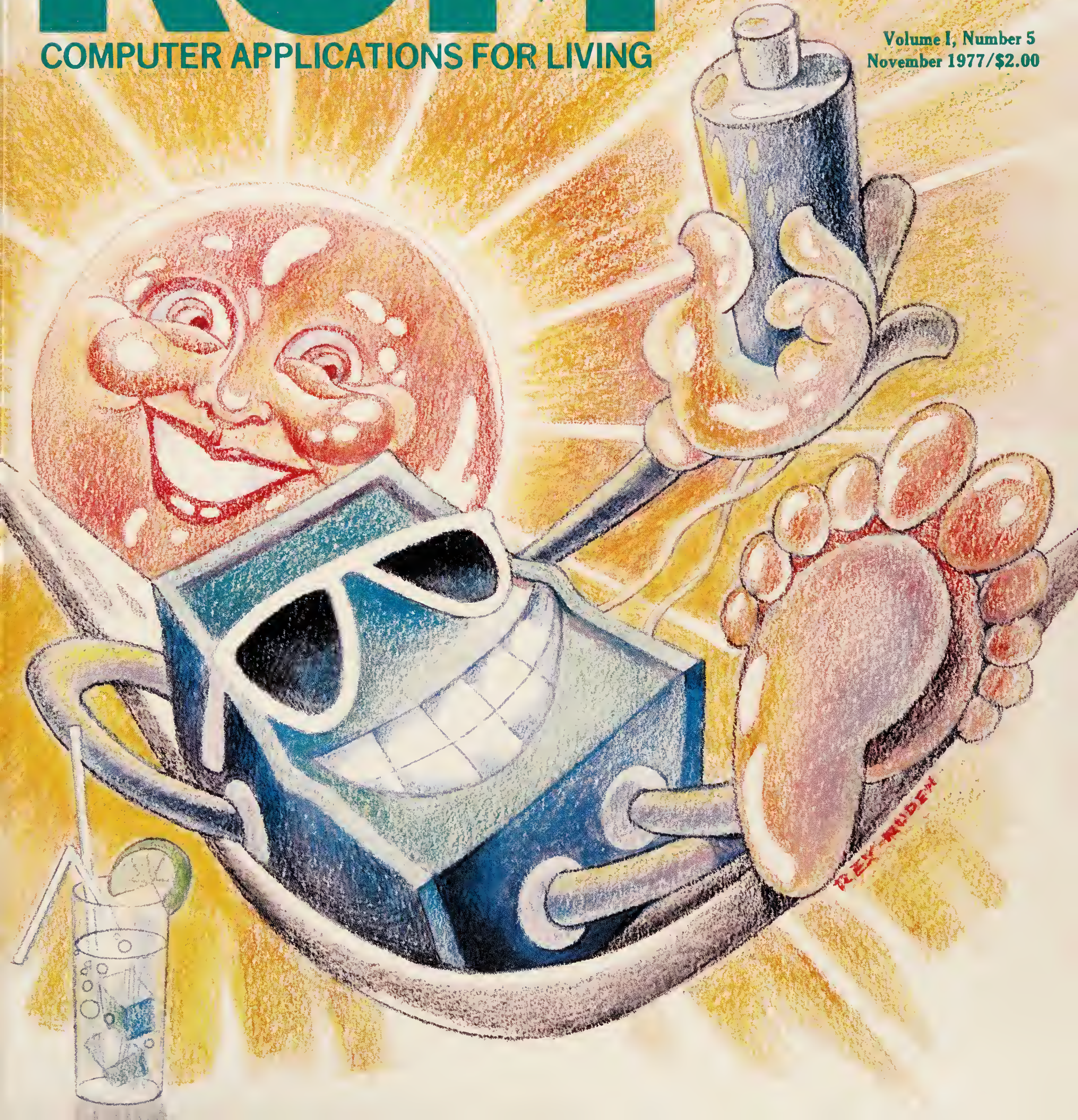
MAKE ME MORE MUSIC, MAESTRO MICRO by Dorothy Siegel

ROM

COMPUTER APPLICATIONS FOR LIVING

THE WORDSLINGER
2200 Characters
Per Second

Volume I, Number 5
November 1977/$2.00



The Small Computer

Twenty-five years ago a computer as powerful as the new Processor Technology SOL-20/8 priced out at a cool million.

Now for only $1350 in kit form or $1850 fully assembled and tested you can have your own small computer with perhaps even more power. It comes in a package about the size of a typewriter. And there's nothing like it on the market today. Not from IBM, Burroughs, DEC, HP or anybody else!

It fills a new role

If you're an engineer, scientist or businessman, the Sol-20 can help you solve many or all of your design problems, help you quantify research, and handle the books too. For not much more than the price of a good calculator, you can have high level computer power.

Use it in the office, lab, plant or home

Sol-20 is a smart terminal for distributed processing. Sol-20 is a stand alone computer for data collection, handling and analysis. Sol-20 is a text editor. In fact, Sol-20 is the key element of a full fledged computer system including hardware, software and peripheral gear. It's a computer system with a keyboard, extra memory, I/O interfaces, factory backup, service notes, users group.

It's a computer you can take home after hours to play or create sophisticated games, do your personal books and taxes, and a whole host of other tasks.

Those of you who are familiar with small computers will recognize what an advance the Sol-20 is.

Sol-20 offers all these features as standard:

8080 microprocessor — 1024 character video display circuitry — control PROM memory — 9216 words of static low-power RAM — 2048 words of preprogrammed PROM — built-in cassette interface capable of controlling two recorders at 1200 bits per second — both parallel and serial standardized interface connectors — a complete power supply including ultra quiet fan — a beautiful case with solid walnut sides — software which includes a preprogrammed PROM personality module and a data cassette with BASIC-5 language plus two sophisticated computer video games — the ability to work with all S-100 bus products.

Full expansion capability

Tailor the Sol-20 system to your applications with our complete line of peripheral products. These include the video monitor, audio cassette and digital tape systems, dual floppy disc system, expansion memories, and interfaces.

Write for our new 22 page catalog. Get all the details.

Processor Technology, Box O, 6200 Hollis St., Emeryville, CA 94608. (415) 652-8080.

Processor Technology

Volume I, Number 5

Contents

November 1977

ROM is published monthly by ROM Publications Corporation, Route 97, Hampton, CT 06247 (Tel. 203-455-9591). Domestic subscriptions are $15 for one year, $28 for two years, and $39 for three years. Canada and Mexico $17 for one year, $30 for two years, and $41 for three years. For European and South American subscriptions, please add $12 per year additional postage. For all other continents, please add $24 per year additional postage. Application to mail at second-class postage rate pending at Hampton, CT 06247. Membership in Audit Bureau of Circulation pending.

Missionary Position

KIDS, PARENTS, AND COMPUTERS

by Theodor Nelson

The millenium is continually being postponed.

Even though we've got the hardware, even though our industry can turn out computers enough for everybody in the next five years—at twenty-five dollars for a complete PDP-10 by 1982, say—the software picture is much more discouraging. The true personal software is coming all too slowly.

A lot of people seem to think the future will be like the past, and are writing programs for the little machines rather like the programs that exist already for the big ones.

This is rather depressing. The real computer revolution will come when people begin to realize that all computer applications should be interactive, using high-power text and graphic systems, as almost no computer programs have been before. But the kind of high-power, easy-to-use, interactive software that will make Everyman's Computer a reality is still on the drawing boards. And not very many drawing boards, I fear. (Except see Alan Kay's piece in the September 1977 *Scientific American.*)

This means that the true home computer market, which I thought was almost here, may be delayed a couple of years. Computers, for the next few years, will not be for everyone, because the programs to make the computer a personal information center for the layman will not be ready. So most laymen will continue to be cowed, apprehensive, or blissfully ignorant of what computers can do for them. The personal home computer market might even be stuck growing as slowly as some turkeys have predicted—say, thirty percent a year.

So what are the next markets?

The present markets are: first, the hobbyists who like to solder and change boards all the time, and who get off on circuits, and, second, venturesome small businesses, who think that machines of the present caliber can suit their needs.

The next market is kids.

It's well started, of course. I've seen father and son, both aglow, walking out of the computer store with their big IMSAI kit. It's really inspiring—especially if you've told them what they're in for, so they're going into it with their eyes open.

The era of kits-only is ending. You won't have to solder any more. Bit-level technicality, too, is about to become optional, as higher-level languages for hobbyists become more widely known (like BASIC) or more powerful and elegant (like TRAC Language or SMALLTALK).

There is still the veil of unnecessary technicality; but that is about to evaporate in the sunshine of the Sol and the APPLE 2, and the PET and the Radio Shack computers.

The only barriers now will be fear and novelty—enough to stop adults; but not kids. They do it all the time.

And anyway, kids are mentally front-loaded. They adapt to the here and now. You tell them it's 1977; they prepare to live in 1977. You show them a computer and they want to understand it, so you tell them about it, and they say, "Oh, *I* see," and they do. Whereas most adults believe they can't understand computers, and presto! they can't.

Now, of course, most kids don't have two thousand dollars for a computer, or the five hundred it'll take next year. But most kids have an adult connection, and if you consider how many kiddie motorcycles are sold, there's going to be an awfully big market for kiddy computers.

Some laymen are shocked that kids can quickly learn about computers. That's because they thought that the basic concepts of the computer field rested on mathematics, or electronics, or some other mystery that required prerequisite study. Naturally, you and I know better.

It is axiomatic that high school kids make the best programmers. There is no effort too great for them, and they have no intrusive expectations of reward based on adult salary experience. (This is important to those of us who are trying to carry out ideals rather than make money. While trying to implement dreams on shoestrings, you gravitate toward labor that appreciates ideals and deferred payment. So if computers for you are a way to create a new world, look in on the kids' computer clubs.)

For a variety of reasons—especially trying to implement great dreams and not caring for the stodginess of adults—I've gotten to know a rather large number of computer-involved kids. In over a dozen cases I have also known their respective parents.

These acquaintances and friendships, growing from accidental beginnings, have become ever more interesting as I've had a chance to watch the kids develop. Small persons who once, with squeaky voices, tried to explain to me such intricacies as the neutral-implied call test in TRAC Language, are now upperclasspersons in the colleges of their choice. Most of the high school computer kids I've known are now either in, or beyond, college.

It has been interesting (aye, heartwarming) to watch these kids grow and develop; and indeed to appreciate how much of it has been due to their parents' interest and concern. This parental concern has meant, at least for some of the families I've known, not merely encouragement but a lot of involvement and facilitation. Parents of computer-involved kids—let's call them computer parents—are perhaps more unusual than the kids themselves.

The computer parents I have known tend to be loving and deeply concerned with their kids. Also very understanding, liberal about their kids' behavior, and supportive. They host computer clubs and have everybody to dinner. They chauffer the kids and pay for trips to conferences. They install additional phones and pay for the calls. They are very hospitable to visiting firemen (like the author). This despite the fact that few of the computer parents have computer backgrounds.

At least two of the computer mothers I know are solicitous beyond all bounds of what mothers ordinarily do in our culture: handling procurement for their kids, wheedling and persuading outsiders, serving as secretaries and chauffeuses, running errands, guarding equipment from potential borrowers. It is not customary in American society for parents to be quite so deeply involved with their children's hobbies and/or careers.

One of these mothers complains—laughingly—of having been gotten out of bed at three A.M. to dance with her son when his program worked. (But I think this gives insight into why the program worked: because the son worked. And the son worked partly because he had some one to please.)

An interesting fact is that most of my younger friends in computers have been Jewish, or formerly Jewish. I'm not really sure what the connection is, but I think it has to do with the deep Jewish tradition of parental concern for children, and for their welfare.

This love and trust of children is a strong factor in encouraging whatever the kids want to do. Naturally, the strong Jewish traditions of intellect and career preparation also fire the parents' concern for their kids' computer involvement. But I think the sense of warmth and nurture are most basic. ("Eat! Eat! *Then* show me your program.")

Parents of course get involved with their kids' interests in traditional Protestant America, too; but (speaking as an alienated WASP) I see patterns there that create troubles. In Protestant America parents tend to be less intensely interested, more distant, concerned with results (but not helping with the means). They tend to leave their kids' education to the schools and tend to take a sink-or-swim attitude to their kids' careers. Where the Jewish pattern is one of earnest facilitation, the WASP pattern is stern watchfulness. Leave the kids alone, maybe they'll make good.

Worse, there can also be an element of unconscious competition between parents and their children in any family, and I see this frequently. (The classical joke of the father taking over the gift electric train is no joke.) This element of parents' unconscious competition can have strongly deleterious effects on kids' intellectual achievement. If a kid is repeatedly ridiculed or put down for knowing something, or if his ideas are sneered at, then knowledge and ideas will not be pleasant for him. Independent intellectual avenues to achievement or recognition will be cut off, and all that will be left will be sports, making trouble, or (for a few) homework.

(I think this may be one of the reasons for the popularity of science-fiction as a genre. It boosts interest in technical matters and sweeping ideas, but not in such a way as to encourage parental putdowns by engendering sassy remarks at the dinner table, or radical politics.)

Naturally these tendentious remarks are intended as cultural insights, not exact generalizations. It is fine to nurture kids' interests, whoever does it.

An interesting new type is also beginning to emerge: the *second-generation* computer kid. Like children of actors or microbiologists, they have a natural start in their parents' field from atmospheric exposure at home.

I was startled at the home of one such family to hear, "Hey, Mom! Guess what we're hooking to the 1130!"

At another level, I recently saw the thirteen-year-old son of a famous professor demonstrating to his father a graphics program he had written in LISP. Balls bounced around the screen in a complex simulation. As the father offered encouragement, the boy's face fell.

"Uh-oh," he said, "I forgot to give the universe a friction factor."

Computer professionals' feelings about their children and about the field intertwine in extraordinary ways. For instance, half-a-dozen years ago I was working with an artificial-intelligence person whom I considered somewhat of a fraud. (Artificial intelligence is the branch of the computer field that says, "We don't know how we'll do it, but any day now we'll have a program that's smarter than you.")

Don't ask me why people go for this point of view. But I was startled by a conversation with this guy over his newborn son. He said he would soon be installing a computer terminal in the infant's crib. "What will the child do with it?" someone asked in amazement.

"We don't *know,*" he replied.

At that point it seemed to me that perhaps the man was sincere after all. Perhaps the artificial-intelligence religion, for all its putdowns of everyday knowledge, is basically a form of pragmatic open-mindedness.

As computer kids grow up, I've often heard them say the involvement was only temporary, that they were going to get out of the field, but in most cases they haven't. Indeed, three years ago I reported in *Computer Lib* that my friends of the R.E.S.I.S.T.O.R.S. computer club, of Princeton, New Jersey, planned to go on to other things.

All are now in college or beyond, and that has not happened.

One got away into pure mathematics, one into theater, one into chemical engineering. The rest are in computers.

One tried liberal arts and came back. One tried history and came back. One tried economics and came back. One tried sociology and came back.

Partly, perhaps, because the money is too easy. Partly because it's one area that's not on the decline. But I think actually the reason is simpler.

Everything else is too dull.

So what I'm saying is this: kids will be the main market for computers till 1979, or whenever the good total home software packages appear.

A lot of kids may be disappointed at first that personal computers don't look like Artoo Detoo.

But soon they will, you see.

Why shouldn't your personal computer have wheels? And tweedle? My guess is that the first computer kit with wheels should certainly appear soon. (Actually, the tweedle-part was nicely worked out by Marvin Minsky and Ed Fredkin at M.I.T.'s Artificial Intelligence Laboratory. The Muse, a digital tune generator in a handsome brushed-metal-and-hardwood cabinet, came out a couple years ago. It was not successful commercially, but Fredkin may have a few left that he will part with at $300. An IC version of the Muse on your wheeled computer, and you've got Artoo Detoo.)

When the computer catches on, I think it will be essentially for kids; and even without the general-purpose personal software I mentioned earlier, there'll be a staggering demand. The demand for home computers will hit the millions by mid or late 1978; production capacity won't catch up till at least late '79.

Every kid needs a computer.

Every kid should have a computer.

(These statements are obvious; they should be obvious, at any rate, to readers of this magazine.)

A new children's crusade is not impossible. ▼

"Program your payroll, mister?"

COMPUTERS CHALLENGE AMERICA'S CUP

MICRO DIET: BETTER HEALTH THROUGH ELECTRONICS

COMPUTER MODELS IN PSYCHOLOGY

A BEGINNER'S GUIDE TO PERIPHERALS

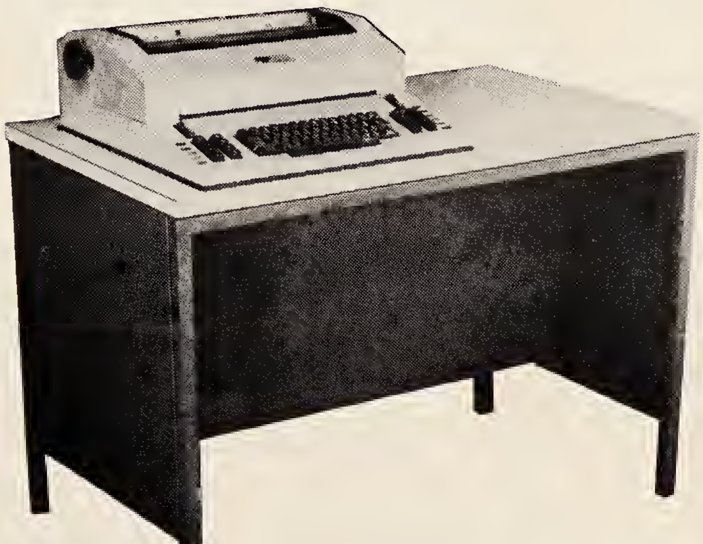
ROM

COMPUTER APPLICATIONS FOR LIVING

STAR WARS
Centerfold
R2-D2

Volume I, Number 6
December 1977/$2.00

SELECTRIC TERMINAL



- 14.8 cps print speed
- 13 inch line length
- 10 or 12 characters/inch
- 6 or 8 lines/inch
- IBM 2740-1, 2 line control
- IBM 2740 keyboard/printer

This deal can't be beat. Not only do you get a quality terminal for a fraction of the cost of anything else on the market, but you also get tyewriter quality. Can be interfaced to a mini- or microcomputer or used with IBM computers. These terminals are worth the price of a typewriter alone Our warranty is limited to replacing any defective parts for a period of 90 days following receipt of equipment. Interchangeable print element, of course. Includes attractive desk. Optional documentation package $25.00.

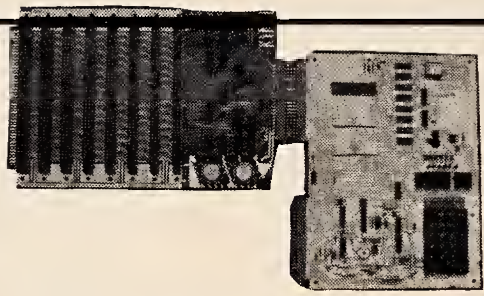
SUPER SALE

JUST **$675**

COMPUTER DEALERS ASSOCIATION

CALL [313] 994-3200 TO RESERVE

KIM MEETS *S-100*



Item	Kit Price	Assembled Price
Kimsi Backplane/S-100 Adapter 8 slots for S-100 available (requires +8V, +16V & -16V unregulated power supply)	$125	$165
KIM to Kimsi Connector Set Includes recorder cables	8	11
KIM-1 incl. documentation Send for more information	N/A	245
S-100 Edge Connector Kimsi accepts 8; 1 is supplied	5	7
Cassette Recorder for KIM-1 Hundreds sold for KIM use	N/A	55
4K Imsai RAM Board Bare minimum to run Tiny BASIC	139	239
8K Seals RAM Board Enough space for programs below	N/A	269
Poly Video Terminal Interface 16 x 64 Connects to a Monitor & Keyboard	210	280
Matrox ALT-2562** Video Graphics Board displayed 256 x 256 array	N/A	395
Itty Bitty Tiny BASIC Uses 2.5K; kit a paper tape, assembled a cassette	5	10
Focal [a DEC trademark] Includes floating point	N/A	50
6502 Assembler & Editor This is NOT the MOS version	N/A	40

Escape the single source blues. The Kimsi is the easy way to expand your KIM-1 system with readily available S-100 boards. Forethought Products has interfaced the 6502 to the S-100 bus by decoding the top 4K for IN and OUT instructions. Full capacity interfacing allows complete control of the 6502 and DMA from the S-100 bus.

SPECIAL! 10% off S-100 Boards when ordered with Kimsi.

Name ______________________

Address ______________________

City State Zip ______________________

Charge my BAC/VISA ____ M/C ____ Interbank # ______

Card # ______________ Exp. Date ______

Signature ______________________

☐ Please send me info on the new Commodore PET 2001 computer

Total for goods checked above

+4% Michigan Residents

+4% Shipping and Handling

+$1 if order's under $20

Total Amount

Limited 90-day Warranty

Return non-working merchandise within 90 days for replacement or refund. Kits are warranted to be complete with working components. All items subject to prior sale.

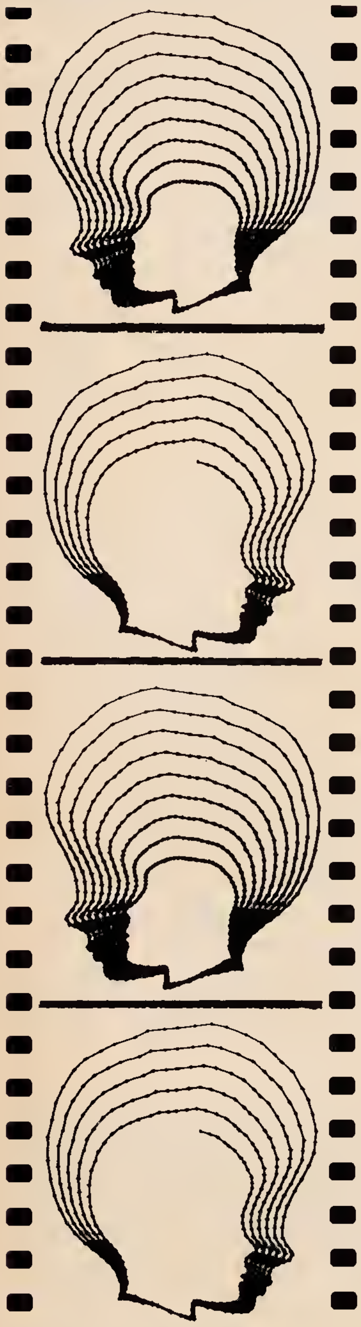
Newman Computer Exchange/CompuMart

1250 North Main Street Department **R1** P.O. Box 8610 Ann Arbor, Michigan 48107

(313)994-4445

Contents

Volume I, Number 6
December 1977



ROM is published monthly by ROM Publications Corporation, Route 97, Hampton, CT 06247 (Tel. 203-455-9591). Domestic subscriptions are $15 for one year, $28 for two years, and $39 for three years. Canada and Mexico $17 for one year, $30 for two years, and $41 for three years. For European and South American subscriptions, please add $12 per year additional postage. For all other continents, please add $24 per year additional postage. Application to mail at second-class postage rate pending at Hampton, CT 06247. Membership in Audit Bureau of Circulation pending.

Missionary Position

PERSONAL COMPUTING 1982

by Theodor Nelson

Well, 1982 certainly is an exciting year for personal computing. And Atlantic City in August, with gambling, is the same as it was but more so, if you can stand it.

America is computer crazy. Women of fashion wear silver pants suits full of twinkling lights and cunningly placed switches. Tully Peschke has become America's top fashion model, edging out Margaux and Farrah to model the new Hot Solder line of cosmetics, perfumes, and "component" jewelry.

There has been some social upheaval, but thanks to several new social-service agencies—especially, those offering rehabilitation for COBOL programmers—things have settled down.

Well, there sure have been a lot of changes in the industry.

The new ten-megabyte memory board from MITS has struck many as a disappointment. Considering the cost—a hundred dollars—they could have offered a lot more. However, a spokesman for their subsidiary, PERTEC, says, "This is just the beginning."

Processor Technocracy—a firm merging the old Processor Tech with its recent acquisition, Univac—has conservative plans. Their new desktop computer, SOLOMEN, only offers two megabytes of sixty-four-bit memory. "But we believe we stand for quality," says corporate vice-chancellor Steve Dompier.

Crothers Memorial Corporation (formerly Cromemco) has brought out its HyperDazzler: one million points square, 128 colors, with as many local CPUs controlling subsegments of the picture. They admit that widespread use in schools will have to wait a year or two.

Ohio Scientific has announced a new computer containing one of every processor chip, together with a compiler optimized to use them all in parallel, even when not needed.

IMSAI has just announced a radical new computer design called the Hypercube. How to program it is a mystery, but it promises many operations per second. Observers are puzzled by the press releases, which seem to have earlier dates crossed out.

The increasing number of IBM 370s coming onto the hobby market has raised the question, "What is the best way to hook up a 370 for hobby use?" Ignoring the obvious unprintable answers, several interfaces are available. Intel, despite its add-on product line for the 370, has staunchly refused to interface the 370 to the S-100 bus, which they call illegitimate (not the precise term), despite the three thousand interesting peripherals now available. The interface has, however, been supplied by Itel, not to be confused with Intel.

Another S-100 adapter for the 370, however, simply uses the 370 as a peripheral, and many users say this works best.

People's Computer Company has changed its name to Transamerica Peoplecomp, partly because of it being recently acquired, and partly to avoid its being confused with People's Computer *Commune* (formerly IBM Federal Systems Division). Transpeep, as they like to be called, has just introduced a new magazine: *Mrs. Tillie Mapes' Newsletter of Flying Elephants, Funny Green Snakes and Muffins,* which, of course, deals with hardware maintenance of S-100 products.

A second edition of *Computer Lib* is rumored to be in the works.

International Honey Control (a firm merged from IBM's Advanced Systems Development Division, Honeywell, and Control Data) has announced that it will go after the business market for large systems, whatever that may be. Meanwhile, Interplanetary Godco—a consortium effort of Intel, IBM Components Division, and Godbout Electronics—is tooling up for the production of HLIC (Humungously Large Integrated Circuits) in satellite orbit. (Speaking of mergers and acquisitions, DEC rejected a tender offer from Technical Design Labs. They rejected it even when the offer was made less tenderly. "We stand by our growth rate," says doughty Ken Olsen, who was actually standing by a stone wall as he made the remark.)

The scene in software remains in flux. Languages once thought brilliant, like SMALLTALK, FORTH, and PASCAL, have come and gone. But they have left their mark. Most versions of BASIC now have procedures, if-then-else, user-definable data types, and dynamically modifiable arrays of arrays of arrays of variable length. But people are increasingly attracted to the new languages, like R2D2, HYPERWUMPUS, GLITZ, PEEKABOO, and SNERD.

The autodialling accessories have created a new scourge in the society, nuisance telephone calls made by computers. These are used alike by politicians, advertisers, and callers who play tapes of heavy breathing. (In one strange case, the answering party hears *really* heavy breathing—followed by a strange voice saying, "This is a dragon!")

A happy event for the nation has been the surreptitious cancellation of the National Debt by government programmers. When Donn B. Parker vowed he would find the culprits, he found himself appointed Ambassador to Liechtenstein.

Well, the conference has been a smashing success. Though competing with Wayne Green's "Computer Machismo" conference in Acapulco, the Convention has still managed to draw a hundred thousand people. Incidentally, Green's new magazine, *Wire Rap,* has had some success in interesting everyday computer owners—housewives, secretaries, schoolchildren—in learning about hardware assembly and principles.

Yes, computer hobbyism has come a long way. Unfortunately, this Convention was, as usual, closed to those having prebuilt computers—approximately forty million by this time. And in the next few months that number is expected to double, with Kellogg's inclusion of a computer in every cereal box. (They expect to get you on the coupons for peripherals.) ▼

SYNTHETIC SKIN FOR YOUR ROBOT AND HOW TO MAKE IT

THE CODE THAT CAN'T BE CRACKED

TLC—THE VISUAL PROGRAMMING LANGUAGE

MICROCOMPUTERS HELP THE DEAF-BLIND

ROM

COMPUTER APPLICATIONS FOR LIVING

VIDEO-SKETCH
Micro Graphics
Program

Volume I, Number 7
January 1978/$2.00

Contents

Volume I, Number 7
January 1978

ROM
COMPUTER APPLICATIONS FOR LIVING

FEATURES

DEPARTMENTS

COLUMNS

ROM is published monthly by ROM Publications Corporation, Route 97, Hampton, CT 06247 (Tel. 203-455-9591). Domestic subscriptions are $15 for one year, $28 for two years, and $39 for three years. Canada and Mexico $17 for one year, $30 for two years, and $41 for three years. For European and South American subscriptions, please add $12 per year additional postage. For all other continents, please add $24 per year additional postage. Application to mail at second-class postage rate pending at Hampton, CT 06247. Membership in Audit Bureau of Circulation pending.

Missionary Position

THE ART OF THE COMPUTER SCREEN

by Theodor Nelson

The interactive computer screen will be mankind's new home.

The sooner we understand it, the better.

Your Interactive Graphic Screen

You can get graphical screens for your personal computer already. Most of the prebuilts are being offered with graphic screens; you can put them on the others as accessories.

People want them for games. People want them for practical uses. And people want them for sheer excitement. With this equipment—and suitable programs—you can make your own cartoons, your own interactive pictures, your own complete console for living.

But so far the programming to be seen on hobbyist screens has been rudimentary and difficult. (For instance, every time you see a Video Dazzler, you generally see Steve Dompier's same little picture of a champagne bottle pouring.) There are few interactive animations for these systems, as yet.

Just what are we talking about?

The Commodore PET offers a screen with text and certain picture capabilities. Short line segments, vertical and horizontal, can be combined into pictures or animations. Patterns of dots may also be put on the screen, but in certain very restricted arrangements.

The Radio Shack computer allows a certain pictorial capacity with little squares, 48 (vertical) by 128 (horizontal). Separate TV required.

The Merlin video board for S-100 machines allows graphics of 96 by 128 squares. Separate TV required.

The Video Dazzler from Cromemco offers color graphics of 64 by 64 squares in eight colors. This is also an S-100 system. The Super-Dazzler, still in the works, promises much higher resolution, but we don't know when. Separate TV required.

The Levine Board (available from the Itty Bitty Machine Co., Evanston, Illinois) is an S-100 board offering 256 x 192 squares of graphic animation. Unlike the Dazzler, it does not slow the computer down. Separate TV required.

The Compucolor machine, a prebuilt with color video included, offers graphics in color, 192 x 160 boxes. This has certain peculiarities, restricting the display to only two colors within small regions. But the machine is inexpensive at $3000, considering all it does.

These are only a few of the many fabulous pieces of equipment *now* on the amateur-computer market, offering different kinds of interactive pictorial capability. We won't even get into the programming problem. But we will talk about what it's for.

More and better will be available soon. The thing to do now is understand what you can *do* with the screens, understand what they portend, and prepare.

What's Coming

Perhaps what will matter most in the coming decade will be the design of interactive systems for people to use in their everyday lives. These will resemble nothing so much as video games; but they will be video games about real life and video games for the mind. Tomorrow's desk, tomorrow's automobile dashboard, tomorrow's control panel—all these will use the computer screen as a magic viewer and magic wand; a gateway to what we want to see or do.

How hard it is to write about this in a column! If you saw it in front of you you'd understand it immediately—the smallest child would. Five years from now you'll see it everywhere. But right now, at this instant, the brink of the new world, I have to fumble with words.

Earlier we saw how easily a computer can be made to behave interactively. The general principle is this: something appears on the screen, typed by the computer; you type something back (take your time); the machine replies at once with something new.

The Most Important Computer Program Ever Written

All the computer-screen systems of tomorrow were foreshadowed by one astonishing program created by an isolated genius in the early sixties.

A stern, thoughtful young man named Ivan Sutherland, then a graduate student at MIT, was given permission to use the special graphics computer at Lincoln Laboratory.

Lincoln Laboratory is a stern, thoughtful complex on the outside of Boston where they do electronics research associated with warfare. The special graphics computer was the TX-2, built especially for experimentation with pictures on computer screens. What did this have to do with war research? Only that the military finds out about new developments first, and so that is where computer screens got their first boost.

Ivan Sutherland, in any case, showed a rare vision in what he chose to do with the TX-2 computer—and how he did it.

He created a system that allowed you to draw on the screen. For this reason he called his program SKETCHPAD.

The SKETCHPAD program allowed you to draw on the computer screen as you might on paper—but with remarkable new capabilities.

You could draw a picture on the screen with the lightpen—and then file the picture away in the computer's memory. You could, indeed, save numerous pictures in this way.

You could then combine the pictures, pulling out copies from memory and putting them amongst one another.

For example, you could make a picture of a rabbit and a picture of a rocket, and then put little rabbits all over a large rocket. Or, little rockets all over a large rabbit.

The screen on which the picture appeared did not necessarily show all the details; the important thing was that the

details were *in* the computer; when you magnified a picture sufficiently, they would come into view.

You could magnify and shrink the picture to a spectacular degree. You could fill a rocket picture with rabbit pictures, then shrink that until all that was visible was a tiny rodent; then you could make copies of *that,* and dot them all over a large copy of the rabbit picture. So when you expanded the big rabbit till only a small part showed (so it would be the size of a house, if the screen were large enough), then the foot-long rockets on the screen would each have rabbits the size of a dime.

Finally, if you changed the master picture—say, by putting a third ear on the big rabbit—all the copies would change correspondingly.

The drawing operation in SKETCHPAD was very special. The user would point with the lightpen at a starting-point on the screen, and draw a line from that starting-point to any other position. A line would extend from that position to the tip of the lightpen, and when the lightpen moved, so would the line, stretching like a rubberband from its starting-point. This was called a "rubberband line;" it allowed the user to try out different positions without erasing.

Then, when the user wanted to join two lines, there was a way of *attaching* them: two lines that were attached remained attached, even when the user decided to move one of them.

One of the most important aspects of SKETCHPAD was this: working on a screen, you could try out things you couldn't try out as a draftsman on paper. You were concerning yourself with an abstract version of the drafting problem: you didn't have to sharpen any pencils, or prepare a sheet to draw on, or use a T-square or an eraser. All these functions were built into the program in ways that you could use through the flick of a switch or the pointing of the lightpen. And the drawing itself existed in an abstracted version, that could be freely changed around with no loss of detail.

Thus SKETCHPAD let you try things out before deciding. Instead of making you position a line in one specific way, it was set up to allow you to try a number of different positions and arrangements, with the ease of moving cut-outs around on a table.

It allowed room for human vagueness and judgment. Instead of forcing the user to divide things into sharp categories, or requiring the data to be precise from the beginning—all those stiff restrictions people say "the computer requires"—it let you slide things around to your heart's content. You could rearrange till you got what you wanted, no matter for what reason you wanted it.

There had been lightpens and graphical computer screens before, used in the military. But SKETCHPAD was historic in its simplicity—a simplicity, it must be added, that had been deliberately crafted by a cunning intellect—and its lack of involvement with any particular field. Indeed, it lacked any complications normally tangled with what people actually do. It was, in short, an innocent program, showing how easy human work could be if a computer were set up to be really helpful.

As described here, this may not seem very useful, and that has been part of the problem. SKETCHPAD was a very imaginative, novel program, in which Sutherland invented a lot of new techniques; and it takes imaginative people to see its meaning.

Admittedly the rabbits and rockets are a frivolous example, suited only to a science-fiction convention at Easter. But many other applications are obvious: this would do so much for blueprints, or electronic diagrams, or all the other areas where large and precise drafting is needed. Not that drawings of rabbits, or even drawings of transistors, mean the millennium; but that a new way of working and seeing was possible.

The techniques of the computer screen are general and applicable to *every*thing—but only if you can adapt your mind to thinking in terms of computer screens.

It should be obvious that you can use the techniques of computer screens to do bookkeeping, writing, design, architecture; to plan how to move your furniture, to catalog your goldfish. Whatever your field, whatever the kind of data, you can use the computer screen to store, retrieve, choose, draw, rearrange, correct, adjust; to see instantly the results of an idea, and change the idea accordingly; to enact your work, and see it whole, rather than guess at its consequences and work with little pieces.

This is, of course, completely the opposite of "the computer" that so many people think of: cold-blooded, demanding, and requiring everything people tell it to be set up in difficult codes.

The Failure To See

In the fifteen years since SKETCHPAD, no initiatives worth discussing have been taken by the computer industry to bring us closer to a world of computer screens for everyone. It was not in IBM's economic interest to make computers easy to use, but to sell complication and make it sound necessary. The computer companies, mostly following like goslings after IBM, have simply brought out smaller computers and cheaper terminals. (Screens have finally appeared, but merely because it has become cheaper to put out a terminal with a screen than a terminal that prints; but most screens show no pictures.)

The brainlessness of the ordinary computer companies has now become plain, however; for personal computing has arrived with a bang, and with it the certainty, for all to recognize, of a computer-screen future.

Most people have not seen SKETCHPAD, or the movies of it, and nobody was motivated to tell them. Even many people in the computer field, technically-minded and preoccupied with their own areas, have failed to see the revolutionary implications of these developments. Indeed, many see computer graphics as worthless frivolity, rather than what it is: the beginning of a new world.

In the meantime, the hundreds of young people who have seen what would soon be possible with computer screens have retreated to the universities, or elsewhere, to wait out the situation.

And of course the public has hardly heard of it at all.

Of course most people are not yet prepared to think in terms of computer screens. There is some wrench, some about-face required, much like that of learning to live with the printing press, or the telephone. But for many it will only take five minutes of real interaction to see what's coming, and start thinking about what *they* want.

Some Important Screen Systems

The computer screen is something new on earth. That few people have seen how to use them, or seen how immense will be their impact on society, should perhaps be forgiven. People didn't know what they had on their hands when movies were first invented, either. (I've discussed this in "Getting It Out of Our System," in *Critique of Information Retrieval,* edited by Schechter and published by Thompson Books, 1968.)

But a few dazzling examples have begun to show us how computer screens should be used.

SKETCHPAD showed us what could be done at the screen with pictures. Another system, NLS, has shown what can be done with text.

Douglas Engelbart's "NLS" system, created at Stanford Research Institute, allows a user to read from screens and write on screens, instantly pulling to the screen whatever he wants from large quantities of stored text—or putting new things away.

The many users of Engelbart's system can share the writings that are stored in it, and even make marginal comments on each other's work—all stored electronically.

The only drawback of NLS—aside from its presently high cost—is that it is not for beginning users. To learn its use takes ten days, not ten minutes. The kind of performance it offers is terrific; later systems of this kind will have to be simpler for most people to use. But Englebart has shown the way.

The third spectacular example is Alan Kay's "Dynabook" at Xerox Palo Alto Research Center. The Dynabook is simply a small computer with screen, keyboard, and SMALLTALK language. But the dazzling screen manipulations—pictures, animations, fancy text—are exciting to everyone.

A fourth example is PLATO. The PLATO system, created by Donald Bitzer at the University of Illinois (and now being sold by Control Data Corporation), allows a thousand users, all over the country, to have highly interactive computing and graphics on super-looking graphic screens. (See "PLATO Makes Learning Mickey Mouse" by Elisabeth R. Lyman in *ROM,* September, 1977.)

PLATO costs far too much, and is in its present form a dead end, since it uses an expensive central computer instead of little private computers, like Dynabook; but it remains the most publicly visible system for the human use of interactive computers.

The Anatomy of the Computer Screen

The computer screen is something new on earth, and so we are just discovering—and inventing—its nature.

What to use it *for* is obvious: everything. But how to design overall systems is another question. It can be very hard to do well.

However, the different things people have been putting on the screen can be described and categorized, together with their uses so far.

A *cursor* is a movable marker on the screen. When you control a cursor, it serves to tell the computer program what you are pointing at. When the program controls a cursor, it is a way of showing you what you should be looking at, or where the next thing you type will appear. (The Latin root of "cursor" means runner, and the cursor does indeed run around the screen for you.)

A *menu* is a list on the screen of things the computer is ready to do for you; and if you point at one of the items on the menu, the computer then does it. If there is a dot of light to point at, that is called a *lightbutton.* If the menu is composed of symbols or pictures to point at, it is a *symbol menu.*

A *menuplex* is the complex of menus a user may weave through.

Often a screen will be divided into sections having different functions or activities going on. These are called *panels* or *windows.* A place set aside with no borders is simply an *area.*

If advice appears as to what you may do next, it is called a *prompt.* If an area is set aside for prompting, it is the *prompt area.*

Some systems expect you to type whole commands in, and leave an empty line for the purpose at the top or bottom. This is the *command line.*

Sometimes a symbol on the screen will indicate what is going on; when something else begins, it changes to another symbol. This is a *ding-dong.* (If a cursor changes shape depending on what's happening, this is a *ding-dong cursor.*)

Pop-ins are symbols that appear out of nowhere under certain conditions.

A *peekaboo* is something that appears on the screen if you touch a smaller symbol (the *doorbell*).

These names, of course, give no flavor as to what you can do with them.

Just for an example, let's invent a console for a musician: someone who gives live performances, and plays a piano-type keyboard. Let's call him Irving. We'll call the system SAM, or System for Augmented Music.

Very well: a piano-like keyboard, for input.

The keyboard connects to a central small computer, which actually generates the sounds. Probably there are several computer chips; one to handle all the timing and switching and screen-work, several more to create the tones. (Making tones by computer chip is now becoming cheaper and simpler than having a whole music synthesizer, which has to be wired up specially.)

There are loudspeakers: let's be generous and say eight.

And there is the screen, just above the keyboard. A lightpen dangles before it, ready to be pointed more specifically. Irving will press a footswitch when he wants to tell the computer to act on what he is pointing at.

Irving the musician sits down at his instrument. On the screen, in the main panel, is a menu of voices he may want to play in, like organ-stops. Besides the usual names, like FLUTE and DIAPASON, he also has voices called BAUTANT, TWEEDLE, GRUNDOON, and SNAZ—voices he created through the screen.

With lightpen Irving now selects the name of the voice he wants to play in, BAUTANT. That name now appears on a top reminder line, saying that this is the voice he is playing in.

But more: at the bottom of the screen appear some pop-ins, a miniature map of the loudspeakers. Aiming his lightpen between the speakers on the map, he tells the machine where he wants the sound to appear to be coming from: in this case, the center of the room.

And he plays for awhile.

Now he decides to change the sound. Pausing for a moment, he touches a doorbell next to the word BAUTANT in the main panel. A diagram of the sound appears; swiftly he modifies that diagram. He lets it go, releasing his foot on the pedal; the diagram disappears, but he is playing now in the newly modified sound.

(Note that this part of the facility actually exists in Alan Kay's office at Xerox.)

Now suppose Irving wants to play an orchestral piece with himself (like Mike Oldfield's "Tubular Bells").

Basically it works like this.

As Irving plays on the keyboard, SAM "notes" the timing and pressure of each key-pressing. The timing is noted to the thousandth of a second, the touch about as subtly. Thus an accurate recording is made of what keys were pressed when; this is recorded by the computer as a list of symbols.

This list can be used to replay music just as if it were coming in live. Irving merely touches a lightbutton labeled, "Play It Again, SAM."

And as the computer replays each voice, Irving adds yet another "instrument" to the swelling orchestra—chosen from the voices listed on the screen.

Naturally, each of these instrumental contributions can be modified later if he doesn't like it.

Note that this is not exactly a canned recording. Each of the separate instrumental contributions can be left out, and Irving can replace it with a live performance.

This is something like having many synchronized tape recordings: except that each one can be modified, changed in its sound, or changed in its apparent location—all through the screen.

This is just an example. We could design panels, menus, symbols in great detail, but there's no point right now. These machine functions were just chosen off the cuff; any other things you might want a machine to do can be handled as easily. (But note that a number of computer musicians are building systems for themselves that are rather like this one—including Carl Helmers, the editor of *BYTE* magazine.)

Today, screen-facilities like these are so expensive and esoteric as to be available only to our air traffic controllers, utility companies, and war-control centers. But as the costs go down (and the programming becomes easier), we will have graphical computer consoles for everything.

Consoles for writing, for making music, for communications switchboards, for executives making telephone calls; consoles for artists (that's right), moviemakers, newsmen; for darkroom work, pottery, origami, woodcarving.

Basically they will all have computer, keyboard, screen, disk memory. The interconnections to the outside world will vary, and hence the cost.

But they will use menus and panels and the other things we have mentioned. No systematic study has ever been made of the art of such layout, the menus and symbols and their relation to what you want to do. The closest book so far is James Martin's *The Design of Man-Machine Dialogues,* which treats this study as a form of engineering, not an art.

Views

If something is in a computer system, there must be a good way to view it on a computer screen. There may, indeed, be some new and special way.

Since programs can be created to zip through stored data and analyze it in various ways, someone who is concerned with a particular form of data naturally has an interest in creating viewing-programs specially suited to those concerns.

For instance, text.

Someone interested in text naturally wants to run it forward and back on the screen, meaning up and down, at great speed; to be able to see all the headings, and from the list of headings to jump to the text beneath any one of them, just by pointing.

(Sophisticated users will probably need text systems with a much more elaborate structure, however; see *Computer Lib.*)

If you are interested in such things as census data—complicated boxes of numbers—the computer can be programmed to analyze it into all kinds of statistical breakdowns: numerical tables highlighting various aspects.

But wait! Why be satisfied with numerical tables? The graphical screen can be easily programmed to give you bar charts, pie diagrams, diagrams in proportional shades of grey. Or even new kinds of diagrams that can be rotated in multiple dimensions, presenting to the eye things you could never see before.

Then consider maps.

When the computer stores maps, it can store them in new forms. Through the screen you can magnify the map from the entire nation down to an individual street, if the information is there; no, down to the fine print on a chewing-gum wrapper in the gutter, if *that* information is there.

Map data is two-dimensional. But the computer can also hold information allowing it to present three-dimensional scenes.

Some screen-systems show a three-dimensional object as a system of lines—as in *Star Wars,* where the map of the Death Star, in three dimensions, is brought to the good guys just in time by Artoo Detoo. The three-dimensional line-drawn map in the briefing was in fact created on just such a system, on *our* planet.

Such three-dimensional mapping will become of increasing importance, especially in architecture, research, and teaching.

But once you have three-dimensional data—that is, information precisely describing the coordinates of spatial objects—it need not be viewed as lines only. Certain very expensive viewing-systems permit you to see it as a *colored photograph,* showing exactly how such scenes or objects would appear to a living viewer. And this offers the advantage that you need not build the object physically to visualize it, or view it, or photograph it. You need only create the data structure that represents it in the computer system.

NASA has used this approach very successfully, to make "photographs" of what certain complex space equipment would look like if they built it. This way both Congressmen and engineers can be sure they're talking about the same thing.

Soon, it will be possible to do trick visual effects like the big ones of *Star Wars*—great rockets, planets, monsters, scenery, what have you—without having either models or made-up actors. It will only be necessary to create a computer representation of the desired stuff, and the computer will make the movie or the visual insert, frame by frame.

Finally, one clever engineer thinks he can put this all in your home or school. The big fancy systems for fake photography, the kind you'd use for *Star Wars,* cost a great deal of money, like a million dollars. But Ron Swallow of HUMRRO, a research organization in Alexandria, Virginia, believes he can put it all in a box with a color TV. So instead of your home computer screen merely showing *regular* interactive graphics (and two-dimensional pictures), you can travel through whole worlds—cities and canyons and planets and playgrounds—that look almost *real.* He says the terminal will cost $5000 in a couple of years.

All these different kinds of views will become important. And all will increasingly appear, and become familiar, in different panels of our control screens.

The Frontier: Clarity

Many people seem to think that bigger and better complications mean progress in computers.

They are totally wrong.

Beyond the Computer Screen

Anything you want to do with information can be done at a screen; soon it can probably be done better there.

For instance, if your screen is connected to a good text system and sufficient memory, you can certainly do better writing there than is possible with a typewriter. (Unfortunately, there are as yet few good text systems—but there will be more soon.)

Outside Control Diagrams

Yes, for handling information the computer screen is tops. But it has a more portentous capability still.

You will recall that computers can be hooked up to any other machine that can be controlled electronically. Thus a computer program can control a gas pump, a rotisserie, an oil well.

But in turn, *you,* at a computer screen, can direct the computer to take action in the outside world, making it turn on an eggbeater, or a drawbridge, or a stereo. By adjusting a picture to what you want.

A diagram that controls events—in the computer itself, or in the outside world—is a *control diagram.* If the diagram controls things outside the computer, it is an *outside control diagram.*

Control diagrams can be used, as we have seen, to control the operation of your computer itself. Whatever you want to do with a computer can ultimately be done most easily with control diagrams. But control diagrams are a powerful way to work with the outside world as well.

A practical application of outside control diagrams: there are now oil refineries where nobody goes around turning valves by hand any more, when the petroleum is supposed to take a new route.

Instead, an operator studies a map of the refinery on the screen. Selecting an area of the refinery where he wants to reset a valve, he touches that part of the screen with his lightpen; that area expands to fill the screen. He keeps expanding the map, and more details come into place, until he sees the valve he wants—the magnification is now sufficient to show it. With the lightpen he touches the valve's symbol, and a changing number shows the changing percent of flow.

Satisfied with that one, he changes a dozen more; all in less than a minute.

It's all going to be that way.

There will be setups run by control diagrams for editing movies, for running factories, for opening and shutting down public buildings, for lighting cities.

(You could probably drive your car with a lightpen on a control diagram—but your state Department of Transportation might not think it was safe.)

You should note one difficulty with controlling objects in the world by computer: *it's expensive.* The centralized hookup between the outside and the computer is the hard part, especially if it has to be reliable. The computer itself, and even the program for it, is negligible in cost by comparison.

Clarity and the Design of Objects

Let us briefly digress from the subject of computers, and talk in general about machines that are sold for human use.

Industry persists in turning out badly-thought-out objects that nobody can understand.

The technical things that consumers buy, like tape recorders, have always been badly designed. Designers have come out with a chaotic variety of confusing objects, differing widely. Most tape recorders are difficult to use, some ridiculously difficult. Yet tape recorders only do a few simple things; it's their bad design that makes them complicated.

Recent laws have made it mandatory for all contracts involving consumers to be written in simple English. What we need is a corresponding rule for the design of objects and systems for consumers. Just as the criterion for consumer contracts is that they must be readable by the average high school graduate, a corresponding rule for things sold to consumers ought to be that they have to be understandable in less than ten minutes of instruction. This ten-minute rule should be tatooed on everyone who designs consumer products.

Many engineers and technicians have claimed that this can't be done. Balderdash! It is merely difficult. Moreover, it takes intense dedication to clarity, and repeated revision and rethinking. You have to try over and over until a thing gets simple enough, just as you have to try over and over to make writing clear, and just as you have to rearrange over and over to edit a movie just right.

Another reason that technicians do not like the ten-minute rule is that it deemphasizes what they like to do, and minimizes their achievements in their favorite area of operations. Technical people like to think about technical things; that is why they are technical people. (One engineer has confided in me that he is never really happy unless he is *feeling those chips with his fingers.* This is a very poignant admission.) They think that designing a tape recorder, or a computer program for people to use, is a technical matter. It isn't.

Designing an object to be simple and clear takes at least twice as long as the usual way. It requires concentration at the outset on *how a clear and simple system would work,* followed by the steps required to make it come out that way—steps which are often much harder and more complex than the ordinary ones. It also requires relentless pursuit of that simplicity even when obstacles appear which would seem to stand in the way of that simplicity.

Much has to be reconsidered, of course, when it turns out that the simple-and-clear design is not feasible in its premeditated form; after many changes and reconsiderations, it is the brave designer who wins simplicity and clarity out of the tangle of different pressures.

This is not a column about tape recorders; suffice it to say that I have only seen one tape recorder I considered well designed. This was the Sony TC-50. It is no longer available. People think they want a lot of buttons. ▼

ARTIFICIAL INTELLIGENCE by Joseph Weizenbaum

FLOWGRAMS—A NEW PROGRAMMING TOOL by Tony Karp

TRANSLATE! by Gordon Morrison

UP AND RUNNING AT THE ELECTIONS

ROM

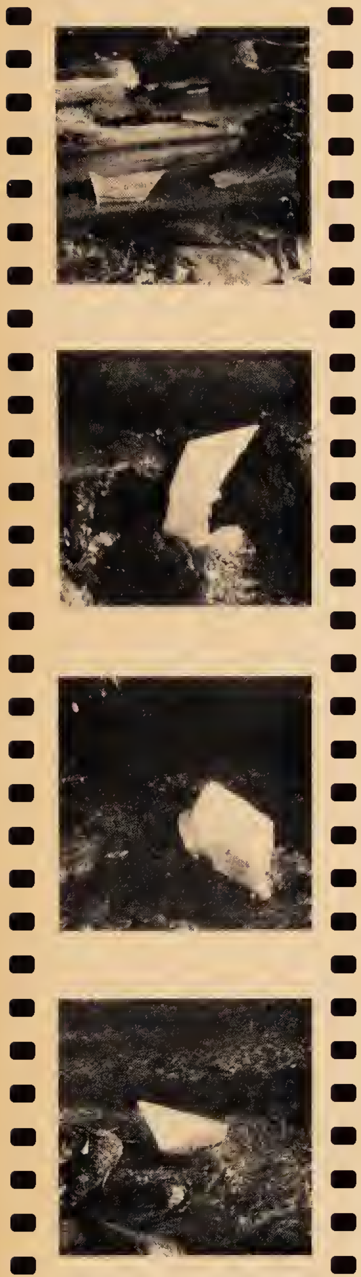
COMPUTER APPLICATIONS FOR LIVING

THE MAILING PROGRAM

Volume I, Number 8
February 1978/$2.00

Contents

Volume I, Number 8
February 1978



FEATURES

DEPARTMENTS

COLUMNS

ROM is published monthly by ROM Publications Corporation, Route 97, Hampton, CT 06247 (Tel. 203-455-9591). Domestic subscriptions are $15 for one year, $28 for two years, and $39 for three years. Canada and Mexico $17 for one year, $30 for two years, and $41 for three years. For European and South American subscriptions, please add $12 per year additional postage. For all other continents, please add $24 per year additional postage.

Missionary Position

MORE ART OF THE COMPUTER SCREEN

by Theodor Nelson

Fantics of Design

Actually, what is needed is an entire different idea of design. The design of something today should be principally the psychological design of how its controls will appear and feel.

The central issue is really one of *fantics,* which is the art and technology of showing things to people, and making things clear. This field stretches across quite an expanse: at one corner is *writing,* which is trying to put ideas and feelings across to people through words; at another corner is movie-making, which is trying to put ideas (and feelings) across through pictures. Creating diagrams, giving directions, setting up exhibits, putting up signs, setting up stores and campgrounds—all these are activities where some kind of ideas and understandings have to be communicated to people, either explicitly or by suggestion.

> *Buyers have bought what had the most shiny buttons without considering whether they could ever learn how to use the thing.*

The fantic problem, then, is getting something across, either by what you say or what you suggest.

But to make it easier, the thing to be communicated had better be simple and uncluttered. And so the fantics of *design* is how to design things whose function *can* be easily communicated.

In this new era, simplicity and clarity and ease of use will become important as never before. These matters have always *been* important, but buyers haven't known it; they have bought what had the most shiny buttons, rather than considering whether they would eventually be able to learn how to use the thing.

But that ends now.

The buyers who have supported Kodak, those purchasers of Instamatic and Box Brownie cameras, have had good reasons for their choice: simplicity and clarity. And these principles must be served if people at large are to be served by computers.

This study is often lumped under a heading called "human factors" or "human engineering." This would be all right, except that it seems well-focused when it isn't. Most people who do "human factors" measure shinbones and reaction times and the like. A new name helps to focus attention on what is really important.

Clarity in Programs for People

People are going to buy computer programs the way they buy cameras, for their personal use.

But computer programs are not shiny, and they do not have buttons and knobs. Either they come in a cassette, in which case they all look alike, or they come in some other inscrutable form that has nothing to do with their purpose.

So their outward appearance does not matter at all; they have to do what's wanted in a comprehensible way. Each impediment to understanding will stand in the way of sale. They *have to be simple and clear.*

There is a natural limit on the complication of physical equipment. It is hard for even the worst designer to introduce more confusions than there are buttons on the box. And if purchasers do not understand a mechanical system at all, they usually do not purchase it, so that presents a final limitation.

But the number of complications in a computer program can be infinite. There is no limit. Till now, computer programs of unspeakable complication were purchased in the corporate world. But individuals won't get taken—at least not to that degree.

There is a myth that things which are simple and clear are not powerful. This is ridiculous: combining simplicity and power is the problem that now confronts us.

If you and the computer are merely typewriting, the effect can be exciting enough. But there are many ways that the computer can make pictures, and that these pictures can respond to our actions. This will come into our lives for recreation, for business, for literature, for art, for every possible thing we do.

Here on the screen you can see a schedule for all the television shows you might like to watch this week. You point at the ones you think you'd like to see, and the system automatically makes note of them. Later, it will turn on the TV automatically at the times planned.

Here is a cartoon character—say, Irving the Elephant. He is looking out at you on your computer screen and his talk balloon says, "Well, what should I do next? Try to catch the crook, go after the girl, or go swimming?" You now point at the words on the screen which represent your decision, and the cartoon character directly begins whatever you commanded him.

Here on the screen is the beginning of a book: say, *Alice in Wonderland.* By pointing, you may choose to see it in a plain type-face or in fancy typography with flowing illuminations; with the original Tenniell illustrations, or with animated cartoons illustrating the story; or with annotations (like those assembled by Martin Gardner for *The Annotated Alice*).

Simply point?

Every computer needs a ROM -so do you!

PLEASE
PLACE
STAMP
HERE

ROM
COMPUTER APPLICATIONS FOR LIVING

ROM Publications Corp.
Route 97
Hampton, CT 06247

...u abreast of the latest microcomputer ..., and office. Written by professionals ...icrocomputing in a lively, readable, ..., ROM is fun. ROM is instructive. ... wanted in a computer magazine.

...nissing without your monthly ROM!

..., Andrew Singer, Bill Etra, and A.I. ...elligence, The Future, The Human ... Plus practical software, listings, ...erals, interfaces, games.... And

Wonderful Dreams of Dr. K by Hesh Wiener ☐ *The Kilobyte Card: Memories for Pennies* by Thorn Veblen ☐ *The Unlikely Birth of a Computer Artist* by Richard Helmick ☐ *Scott Joplin on Your Sci-Fi Hi-Fi* by Dorothy Siegel ☐ *Building a Basic Music Board* by Eben F. Ostby ☐ *The Compulsive Programmer* by Joseph Weizenbaum ☐ *The Very Best Defense (a short story)* by Laurence M. Janifer ☐ *Chart Up and Flow Right* by Eben F. Ostby ☐ *Computer Wrestling: The Program of Champions* by Lee Felsenstein ☐ *Forget Me, Forget Me Not* by Avery Johnson ☐ *PLATO Makes Learning Mickey Mouse* by Elisabeth R. Lyman ☐ *Charged Couples* by Sandra Faye Carroll ☐ *Xeroxes and Other Hard Copy off Your CRT* by Bill Etra ☐ *The Kit and I, Part Two: or Power to the Computer* by Richard W. Langer ☐ *How Computers Work* by Joseph Weizenbaum ☐ *Personally Yours from IBM* by Eben F. Ostby ☐ *A Payroll Program for Your Small Business* by Robert G. Forbes ☐ *Memories Are Made of This* by Lee Felsenstein ☐ *Memory, Memory, How Much Memory?* by Stan Veit ☐ *Software—The Genie in the Bottle* by Tom Pittman ☐ *Your Computer or Your Wife* by Susan Gilpatrick ☐

ROM Publications Corp.
Route 97, Box R
Hampton, CT 06247

Name ______________________

Address ______________________

City ________ **State** ________ **Zip** ________

U.S.A.: ☐One year $15 ☐Two years $28 ☐Three years $39
Canada & Mexico: Please add $2/yr. additional postage
Europe & South America: Please add $12/yr. additional postage
All other continents: Please add $24/yr. additional postage
☐Check/money order encl. ☐Master Charge ☐BankAmericard
Exp. date ________ Card# ________
Please allow 4-6 weeks for delivery.

What is this magic?

The answer is that when the computer is set up to show things on its screen, its program can also sense *what you point at* on the screen.

So that in any situation where you have a choice to make, the simplest way to make that choice is simply to have the alternatives listed on the screen, and point at what you prefer.

Now what do we mean by "point"? Well, it depends on the setup. Some computer screens actually allow you to use your finger, poking the screen at the position you choose. But such setups tend to be unreliable. So most systems that allow pointing give you some sort of a tool—going by names like lightpen, joystick, or mouse. We won't try to explain the differences here; when you get to use one, you'll see how to use it in seconds anyway.

Most people who do "human factors" measure shinbones and reaction times and the like.

The computer's response to a choice is called *branching*. Branching illustrations and cartoons, allowing people to explore new works of art and writing, offer limitless possibilities which have scarcely been explored at all.

Your graphic screen can be used for home study of the freest kind. Suppose on the screen you see a picture of the interior of the human body, with different internal organs labeled. Suppose now you can point at the one you would like to know more about—the heart, or spleen, or pancreas—and get an explanation of it with animated pictures.

Or say you are lost in a store. But here, next to the escalator, is your friendly computer screen! You point at the department you want to go to, and a map appears, with a little cartoon character—say, Howard the Duck—waddling along a dotted line that takes you to the department you want.

So far, interactive screen systems, with pictures and branching, are only to be seen in the video games. But tomorrow, as businessmen and consumers begin to realize the possibilities, they will be everywhere, for doing everything.

Computer screens on the kitchen table. Computer screens by the bedside. Computer screens on the office desk. Computer screens on the school desk. Computer screens on automobile dashboards, Coke machines, ticket dispensers. (In a few years, it will be cheaper and simpler to put the directions for a new machine on a computer screen, built onto the side of the machine, than to print them on paper.)

Each activity, each form of work, will of course require programming to make the wonderful new environments. And this will require a new kind of artist: someone with vision first, and the technical understanding to carry the vision through.

There is a myth that things which are simple and clear are not powerful.

Virtuality: The World beyond the Screen

Everybody knows what "reality" is supposed to mean, though we may disagree over what's out there. But "reality" generally means the nuts and bolts, the solid metal and dirt, as distinct from ideas and feelings.

Very well. An important opposite of "real" is *virtual*. It means *as-if*. ("In essence or effect, although not formally or actually."—Oxford Universal Dictionary.)

Virtuality, then, is the *seeming* of a thing, the ideas and impressions and feelings you get from it. The virtuality of a magic show is the illusion of doves from a hat, or a sawn woman walking away intact. The virtuality of a Cadillac is a cushy drive and the sense of luxury; the virtuality of a movie is the world it puts you in.

The important thing about computer screen systems is their virtuality. You do not care, as a user, whether they work by transistors or by rice pudding. You do not care whether the screen is on the end of a big vacuum tube or a sandwich-panel of neon.

You care *what it's like to use it*. That is the virtuality.

The virtual structure of a screen-system is what structure it seems to have.

For instance, the virtual structure of SKETCHPAD.

SKETCHPAD was a facility where you could make master drawings, and combine copies of these drawings of any size. If you changed the master drawing, all the copies would change correspondingly.

The virtual structure of SKETCHPAD, then, was of a space which could be stretched to any degree on the screen, and instances of pictures that could be copied.

This is the virtuality of SKETCHPAD, or at least that part of SKETCHPAD we have described here. The kind of computer it ran on was not significant to its virtuality, nor was any other feature of the computing hardware. Its virtuality was what you could do to the stored pictures through the screen, and how it felt.

Designing Screen Systems:

Virtuality First, Technicalities Second

When a technically-minded person creates an interactive computer system, he generally decides first *how he wants it to work internally,* then considers its external details one by one, as if they were superficial and cosmetic.

This is entirely wrong.

The way to design a screen-system is like making a movie: decide first what is to be its virtual structure, what are to be its basic concepts and performance on the screen, then devote technical effort to making that come out right. Only this way will it have the conceptual clarity, or the feel, that you wanted to put into it.

For instance, suppose you are programming a simple system to let users write and revise their text on a screen. (A "text editor" or "word processor.")

This is a common sort of program, and most programs of this type are abominable.

The way a typical programmer goes about it is this.

First he decides how the text should be stored, based on what's easiest for him.

Then he figures out a convenient way for the program to make insertions and deletions within this stored text.

Then he figures out some way for the user to command the system to insert or delete; based again, usually, on his convenience.

Then he puts it all together and gets it working, and there it is—another lousy text editor.

(What about rearrangements within the text? "That can't be done," he says. Meaning that he didn't feel obliged to make the program deal with it.)

The *right* way, of course, is to think; what sort of structures of text might a user want, and what kinds of operations upon them? With a lot of thought, you might very well decide that plain sequential text is inadequate. What virtuality then?

(For more on this subject, see *Computer Lib.*)

It's much like making a movie. The movie-maker begins by deciding what story to tell, and what overall quality the finished film should have. Whether it should take place in ancient Rome, in a haunted house, or on another planet; whether it should be cheerful or sinister.

Then plans are made for how to make the film: what scenery to build, what to film on far-away locations, and so on. When it appears that one method is too expensive for a given effect or scene, another is chosen.

The same goes for interactive computer screen systems.

Now the design of screen-systems has many complications. Some tricks on the screen can be done easily by computer programming, others cannot. Some things can be done rapidly by the computer, along with other programs, on the side; other things will take all the computer's time.

The design of screen-systems requires that you know what effects you want, and take whatever steps you must to get them. When something doesn't work, you see what else is possible; but always, like the movie-maker, press toward that structure and quality you want to achieve.

Virtuality consists of both the conceptual structure of a thing, and its feel. The conceptual structure must, for the most part, be planned out first; the feel can usually be fine-tuned later. But all aspects of virtuality should be considered from the beginning of the design process.

Control Structure and Virtuality

The controls by which we operate something have a virtuality, a seeming: they feel a certain way, and they make the things you are operating on seem a certain way.

For instance, the control structure of a car—with automatic shift—is basically very simple. You go or you stop; and you turn left and right when needed. Forward, sideways, sometimes backward; and that's it.

But the controls of the car do not reflect this. You have one pedal to go, another to stop; and only gradually, as a driver's skill grows, do starting and stopping become in the mind a unified continuum.

In other words, the separateness of the controls promotes a conceptual separateness which is only gradually unified in the mind.

It's the same with all machinery. The skilled user is someone who has, in his mind, learned to see all the separate pushings and turnings of the equipment as parts of overall movements and intentions.

It follows that we can make people skillful much faster, and less likely to make mistakes, if we think out controls to be conceptually unified in the first place.

That is the design of control virtuality.

Example: the pilot of today's aircraft does not directly control the individual flaps and throttles by hand. These separate actions are taken by an "avionic" computer. The pilot's controls are designed as *mentally unified* operations; the signals from these controls are translated by the computer into the separate flappings and turns.

When the computer is set up to show things on its screen, its programs can also sense what you point at on the screen.

(The pilot of the F-111 can simply set his plane to follow the terrain at any given height—with a bumpy or smooth ride.)

This kind of conceptual control design is what we will be doing for tomorrow's screen computer systems. Control of all other machinery and information will be handled, not with buttons and levers anymore, or the way laymen think of "computers," with numerical codes; but with clear, elegant pictures and diagrams and texts on computer screens.

How Computers Should Always Be Used

The true meaning of interactive screen-systems is that people can do things easily and without confusion. Sitting at screens at home or the office, they will type—or point—and the system will respond clearly, with a clear virtuality.

Such systems should always involve *text;* to explain what to do, ask questions, present information.

Such systems should always involve *pictures,* helping to clarify and visualize, adding fun.

Such systems should always be *highly interactive,* so that each time you press a key the system responds at once.

If these standards are correct, it follows—paradoxically—that *IBM computers can never be used as computers should be used.* This is because IBM computers cannot ordinarily be set up to respond to each stroke of the user at the keyboard. Whatever key or keys the user wants to press must be followed by another nuisance pressing, either the key called "Carriage Return" or "Attention." This makes highly interactive systems totally unworkable on IBM computers. (Xerox's Dynabook system and the PLATO system now being sold by CDC both have avoided this, and will respond to any key.)

The Computer Media

Consider the different machines and instruments which have caught on in the consumer market in the last century. The biggest have been the automobile, the telephone, movies, radio, and television. Computer media will have an impact comparable to these big five.

The new computer media, however, will uniquely combine elements of all of these: the visual entertainment of reading and of television; a personal environment comparable to the automobile; and the personal intercommunication of the television. The computer media on tomorrow's screens will include text and visual material, animation, and branching alternatives.

Text describes or explains or narrates. Pictures illustrate, provide a mental framework, give atmosphere, decorate. Making the pictures move—animation—lends understanding, emphasis, a sense of action, heightened involvement for the user. Or it tells a story.

But we have not yet worked these things out.

In the book, movie, radio, and television, forms have gradually been discovered for organizing and segmenting the material, for orienting the user's thought, for creating continuity within a work, and for keeping up consumer interest between works. The same thing will happen for the computer media. We will be discovering and inventing new presentational materials for some time to come. We will be discovering the viable forms, structures, organizations, continuities, segmentations of the new computer media.

Besides plain programs and games, we will soon see works for reading and visual exploration, adventures and stories for the screen that are like movies, cartoons and comic strips; and interactive diagram-wonderlands. All of these, of course, will branch, allowing the user to make a variety of choices as he goes along.

But the real precedents for what is going to happen with computer media are not to be found in the nineteenth or twentieth centuries. The real precedents are the printing press, and before that the spread of writing itself.

Inventing Computer Media

Many separate screen techniques have been invented by the pioneers of computer graphics, especially Sutherland, Knowlton, Baecker, Kay, and a handful of others. (See *Computer Lib* for more details.)

But these separate techniques are not the same as a structured system of media.

This can only be explained, right now, by analogy.

When movies began, they did not have closeups, or any editing at all. They did not have pans, dollies, cross-cutting, mattes, double exposures or zooms. These evolved.

When radio began, they did not have station breaks, theme songs, announcer transitions, musical bridges, or time slots. These *evolved.* As did commercials, jingles, and other mixed blessings.

When television began, they did not have the above elements, or the voice-over, holding visuals, visual transitions or anchormen. These *evolved.*

In all these media, too, the units of presentation—the shows and programs—developed gradually as a genre.

In other words, there is strong reason to suppose that the computer screen media will evolve in similar ways to a system of structures, units and transitions.

However, because interactive computers include pictures, sound, and (implicitly) all the other media we have just mentioned, we may expect a higher virtuality to appear that combines these other elements and weaves them together. ▼

BYTE
the small systems journal

THE ABSOLUTE-TIME CLOCK AND HOW TO BUILD IT
WHAT COMPUTERS STILL CAN'T DO
THE SUPER OTHELLO PROGRAM
"DADDY, IS IT THE PET?"

ROM

COMPUTER APPLICATIONS FOR LIVING

CLOSE ENCOUNTERS
The Mothership
Centerfold

Volume I, Number 9
March-April/$2.00

Contents

Volume I, Number 9
March/April 1978

ROM
COMPUTER APPLICATIONS FOR LIVING

FEATURES

DEPARTMENTS

COLUMNS

ROM is published monthly by ROM Publications Corporation, Route 97, Hampton, CT 06247 (Tel. 203-455-9591). Domestic subscriptions are $15 for one year, $28 for two years, and $39 for three years. Canada and Mexico $17 for one year, $30 for two years, and $41 for three years. For European and South American subscriptions, please add $12 per year additional postage. For all other continents, please add $24 per year additional postage. Application to mail at second-class rate pending at Hampton, CT 06247. Membership in Audit Bureau of Circulation pending.

Missionary Position

OUT ON THE LANGUAGE LIMB

by Theodor Nelson

Computer Languages

The original computer languages were simplified ways of writing down the undermost small instructions of the computer, one by one. This is still very respectable. But when such details do not matter in their particulars, we use programming systems that take care of such things for us, letting us think clearly about the instructions that really need our attention. These systems are often called "higher" languages, or just computer languages.

Surprise! There are thousands of different computer languages.

There are at least two dozen important computer languages; the experienced programmer generally knows between three and seven.

The Higher Languages

A computer is, as we have seen, a device for following a plan. This plan can be expressed in any number of ways, provided that the computer is properly set up to recognize and carry out the steps of the plan. Computer languages are simply these different ways of expressing plans. And there is no single standardized way.

The different computer languages arise from the profusion of things computers can do. Computers can do so many things—pictures and music and printing and sorting, not to mention numerical applications—and the more you think about it, the more different possible things you may want the computer to do.

There are many kinds of things people want done with computers, and many styles for doing them. Indeed, little astonishes the newcomer as much as the complete blankness of the computer, the fact that it really can be made to do anything whatever that its electronics will allow.

But different people have different things in mind. Since the very beginnings, many have used the computer for rapid numerical calculation. Others use the computer principally for business accounting and for storing records of business transactions. Yet others see the computer as an extremely deft motion-picture toy.

All these people are right; no one is wrong. But with these different emphases, and the natural variation of human mentality, many different styles of programming, and local rules of operation for programmers to follow, have come into being. By and by, using the computer for a given range of problems, and in a certain style, gives rise to a new programming language. A computer language does not jump out of the air. It is designed by someone to be a useful way of telling the computer what he or she wants.

Each of the higher computer languages allows you, as a rule, to program some particular range of problems, and in a particular style. In part this is because each language handles a lot of details for you automatically. Today's larger programs call in dozens or thousands of littler programs which have themselves been perfected—little programs for putting things in alphabetical order, typing a character on a terminal, moving a picture on a screen, and thousands of other functions. These are called *subprograms* and are of various types. While you do not want to have to create each of these subprograms, you want to be able to use them. So you need a shorthand method of telling the computer to carry out these little programs, and of tying them together. And such a shorthand method is a computer language.

Beginners are startled to learn what a lot of different computing languages there are, and what little agreement about their merit there is among experts. Indeed, laymen commonly ask "How do you say it in computer language?" and this has no general answer at all—because there are so many.

Just as the blind men misconstrue the elephant, and just as different computer users see the computer differently, different computer users likewise prefer different languages, because the different languages are tied to people's different ways of seeing and areas of concern.

People get very uptight about computing languages; the subject is as touchy as religion, if not more so. If you insult a man's favorite computer language, you cease to be his friend.

Indeed, there is no more emotional issue in the computer field than that of computer languages. While physical violence rarely occurs, the levels of emotional commitment and rage to be seen when computer people discuss computer languages is truly awesome. Many hobbyists who have only learned BASIC tend to go through this stage. Since all they have seen are programs in BASIC, all they can imagine is programs in BASIC, and thus they naturally think computers can have no uses *except* those which are easily programmed in BASIC. And indeed they get indignant, just like regular computer people, to hear anyone say they might be missing something.

The most important subject for the computer beginner is not electronics or mathematics; it is a subject that did not in any way exist thirty years ago. It is the subject of computer languages.

The Main Computer Languages

While this is not the place to get into computer languages deeply, let's at least do a rundown of some main areas. Because there are thousands of computer languages, there are also many different ways of categorizing them, . . . and the categorization we will make here is a simple categorization. (It might startle some professionals.)

Traditional Languages

In lumping together the following as "traditional" languages, I am taking a few liberties. . . . Traditional languages require the programmer to figure out ahead of time the exact division of memory to be used for each piece of information that needs to be stored or operated upon. One way or another, the programmer sets places aside for each kind or piece of information that will be needed. (This is one of the main pitfalls of the traditional languages, as it reduces their flexibility.)

FORTRAN

Because the first use of the computers was for arithmetical and formula computations, it was natural that a computer language should be developed which simplified the programming of algebraic formulas. This language was called FORTRAN, supposedly standing for "formula translation." Because it was the first, it became standard. Once it was a milestone; now it is a millstone. People learn it first because it is standard. It was originally designed for mathematical applications; but it is, in most cases, far inferior for these purposes to APL (described later). But still they go on teaching it in the universities.

People get very uptight about computing languages; the subject is as touchy as religion.

COBOL

Spurred particularly by the efforts of Grace Hopper at the Department of Defense, a language was devised for business application, called COBOL (Common Business-Oriented Language). It has certain strengths, but is very inflexible compared to the Lambda languages (described later). COBOL programmers are the coolies of the computer field.

ALGOL

In Europe, mathematicians and scientists who became disturbed at the inflexibility of FORTRAN created a language capable of expressing (and thus programming) much more elaborate and subtle types of procedures. The resulting language, ALGOL, is widely used in other countries, and is standard even in this country as a way of writing down computing procedures so that other programmers can use and understand them. This is because it has no extraneous features, as does FORTRAN.

PL/I

The language PL/I (Programming Language I) was developed as an IBM product. Roughly speaking, it is a combination of FORTRAN, COBOL, and ALGOL all together, preserving the complications of each and the distinct philosophy of none. Many companies with IBM computers use it, however.

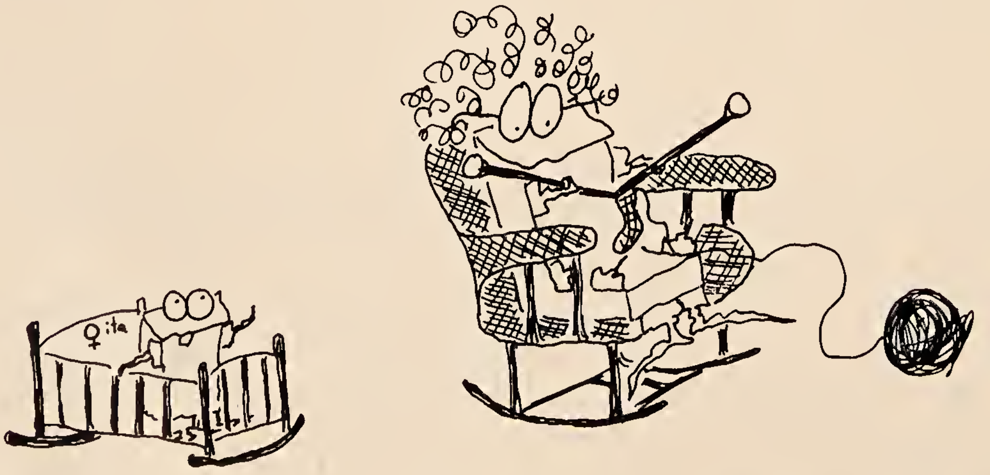
BASIC

A group of determined young men at Dartmouth College, in the early 1960s, created a computer system for everybody there to use, acting on the determination to make computers easy. For this they created a new programming language called BASIC, which was the simplest of all languages to learn at the outset. Since that time, BASIC has become the standard language of hobby and amateur computing, and indeed has caught on throughout the world for many other purposes.

"Basic" is not a description, it's a name. Essentially, BASIC is a simplified FORTRAN. The BASIC language, then, is not (as you might think) language somehow intrinsic to computers, but a language which was created to make programming quick and easy.

The fact that BASIC is easy to use does not mean it is efficient, and there are a lot of things that simply cannot be done in BASIC. Truly complex programs can be created in BASIC only with the greatest difficulty. However, the new computers being set up for home use all come with BASIC, and so its use is growing dramatically even while its limitations are felt ever more painfully by those concerned with creating really versatile and complex programs.

EVE'N'PARITY



"Knit 0001, purl 0010."

Cartoon by Korkie

By common consent, the amateur world is deeply committed to BASIC; but there is no exact standard of what BASIC is, and so there is plenty of room for improvement. One possible hope is that the best elements of LOGO could be slyly introduced to BASIC, until BASIC comes more and more to have some of the power of LOGO. (One sort of superBASIC, called GRASS, may become available soon for amateur machines.)

The Lambda Languages

The second category of computer languages will be, in the opinion of the author, the important ones for tomorrow. They offer a power, and, in some cases, a simplicity that has not been widely seen as yet. The Lambda languages are called that because they are based, somewhere deep down, on something called the Lambda Calculus. But you don't have to know about that.

This mysterious thing, the Lambda Calculus, is simply a systematic way of tying things together; of taking the results of one operation and making them the starting point of another operation. The Lambda languages, accordingly, are extremely versatile, as the results of any operation can be used as the beginning of any new operation. Thus, they have few of the restrictions that are so common in the other languages. Space need not be exactly prearranged, as in the traditional languages.

The Lambda languages were first used in obscure research laboratories, especially those where many delightful odd people work on what is referred to as artificial intelligence. The original Lambda language is called LISP, and it is so intricate and obscure to most computer people that its practitioners have come to be seen as strange eccentrics— a priesthood within the priesthood. Yet there was a reason for this strange computer language, and all of its frightening parentheses: anything which can be done in any other computer language can be done in LISP, while things that can be done in LISP cannot be done in any other computer language.

People versed in FORTRAN and COBOL were alarmed by LISP, because it contained hundreds of parentheses. The parenthesis is the most common character in LISP. This annoys and offends those who don't understand it, because they naturally think anything can be programmed in FORTRAN and COBOL, which is not true.

But LISP ordinarily only runs on big machines (although a group at MIT is endeavoring to build a LISP machine small enough to be a personal computer).

There are, however, other languages which have all the power of LISP, and yet have certain other advantages. An important one of these is LOGO. Created by Papert, Feurzeig, and others, LOGO is as simple to use as BASIC, but far more powerful. It may well become available for hobbyist computer machines in the near future.

A group at MIT, doing research in LOGO as a tool for teaching programming to children, asserts that in two weeks of instruction, children who were taught LOGO could program circles around children of the same age being taught BASIC for comparison.

But LOGO has so far been a washout for political reasons.

Picture the situation if you will. Some extremely bright and visibly eccentric people, who have very little respect for computer programming as it is ordinarily done, have been saying that computer programming should be taught to very young children in a way that most computer programmers don't understand. They have asserted that this scheme will make the children better programmers than the professionals; and they have sought funds to carry on this teaching in schools where nobody knows what a computer is at all.

Such is the computer field.

Another Lambda language which may become important is TRAC language, invented by Calvin N. Mooers, the same man who brought you the phrase "information retrieval." (Mooers may sue me if I neglect to mention that TRAC is the trade mark and service mark of Rockford Research, Inc., 140½ Mt. Auburn St., Cambridge, MA 02138. He does make things difficult for those who try to use it without his permission.)

Once FORTRAN was a milestone; now it's a millstone.

TRAC language will run on a much smaller computer—one authorized version of TRAC language runs in only 8K spaces of the main hobby computer. TRAC language is like LISP in that it uses many parentheses. Computer people who have been turned off to LISP—and that seems to be a lot of people—see the parentheses in TRAC and say, "Forget it." People who only know BASIC often have the same reaction.

But TRAC has certain special qualifications which ideally suit it for the very small computers that are now becoming so very widespread. It does not need large amounts of memory, and it has important features for highly interactive systems. The ability to control user input, so that if a user types the letter *F,* he instantly sees, say, a picture of fish instead of the letter *F,* is an extremely important feature for user-level systems of the future.

The last Lambda language we will mention here is probably the most exciting. It is called SMALLTALK and was devised by Alan Kay and his associates at the Xerox Palo Alto Research Center. It's written up with neat pictures in the September 1977 issue of the *Scientific American,* pages 231-244.

This language was created around Kay's notion of a personal computer, which he calls a "Dynabook." (Apparently the term *Dynabook* simply means a computer that you can program with the SMALLTALK language.) But Kay and his associates have proceeded on the correct assumption that it would be possible within a few years to build a computer the size of a book that will run on batteries, have an elaborate graphics screen, and sell for $400.

People were alarmed by LISP because it contained hundreds of parentheses.

This prediction, which seemed outrageous to some people only a few years ago, now seems firmly possible for the year 1980. Whether the management of Xerox, deeply entrenched in a paper-oriented way of thinking, will understand this development and bring it to market, remains to be seen. SMALLTALK, anyway, is a Lambda language with numerous exciting features. The parentheses are few, not the tangle of LISP. Instead, some commands of the language consist of smiling faces and pointing hands, among the other symbols and phrases.

Secondly, the language is set up for the use of a finely detailed computer screen, of some half-million dots, on which the programmer may typewrite in numerous typefaces. SMALLTALK may produce dazzling animations on the screen, interacting with the user. (In another amazing form of interaction, Kay hooks SMALLTALK up to an organ keyboard coming out of loudspeakers through the computer. At the same time, the SMALLTALK program shows the notes on the screen transcribed from his pressings of the keys.)

SMALLTALK programs are sectioned into a number of parts, called "processes," which are independent entities with a special kind of autonomy. Processes cannot interfere with each other, and thus a program may be debugged, or corrected, by sections.

But numerous copies of a process may exist. SMALLTALK programs, amazingly, are much more "like real life" than most computer programs. For instance, if you write a program to simulate traffic, you have one copy of the "car" process for each car on your highway.

If you've done ordinary programming, you know how odd that seems to most programmers. Yet it has an intuitive simplicity. Thus SMALLTALK may turn out to be both the most powerful computing language and the ideal language for beginners. (Let's hope the Xerox management gets moving on it.)

Other Languages, Especially APL

There are many other languages; some have very specific ranges of purpose, others are "general purpose" but reveal a certain slant and certain special aptitudes. Foremost among these other languages is APL, or "A Programming Language," devised by Kenneth Iverson. Iverson is a fiery and upright figure, with the dignity and self-certitude of a Raymond Massey, or a religious leader.

Iverson claims that his language was always intended as a way of writing things down, especially for mathematicians and scientists, and feigns surprise that it turned out to be "a good way to drive a computer." For Iverson's notation is a powerful and elegant system of expressing mathematical meaning. Having detected, as a young mathematician, that the notations of science and mathematics are really quite chaotic and irregular, he began writing them out in a form which adhered to certain basic rules. Working all this out, he gradually put together a notational system of computer generality.

No attempt will be made to give examples here. (See "APLomania" by Eben Ostby in the August 1977 *ROM.*) But Iverson's language has become one of the most influential forces in the world of scientific computing. APL is a work of art, not unlike a beautiful set of surgical tools, or a set of matched gems.

Iverson's language permits the expression of mathematical concepts from across the whole of science and statistics, thousands of different ideas and functions each resolved to a crisp and concise expression in this new, common form.

The language requires learning new symbols, but a few hours of time spent with an interactive terminal and a good tutor make one able to do astonishing things.

It is interesting to note that APL has come into use almost entirely on a word-of-mouth basis. An ever-growing fraternity of scientists (and, more recently, business users) have discovered its power for a vast assemblage of purposes.

The original APL program was created within IBM, not as a planned product, but as a private project at the initiative of Iverson and his friends. But the language then caught on with IBM, becoming addictive to its users, and became a part of the IBM product line by popular demand from the outside. It is now affecting the rest of IBM's product line, as both scientific and business users work with it more and more.

APL is now available for personal computers, especially the 8080. (Prices vary from $10 to $650 for different versions.) One version sells for as little as $10; but the version from Microsoft, a very respectable programming firm, is expected to sell for about $650.

For many purposes, APL is slow and inefficient—for interactive graphics and music especially. But then again, David Steinbrook, a doughty young composer, is using it as a music machine anyway, and maybe he's onto something.

IBM sells a small computer that runs APL. This is one of IBM's best products. However, because of its cost ($5000 to $15,000), we will not consider it here as being within the range of personal computers.

Other Non-Standard Languages

There are fifty or a hundred languages that ought to be mentioned. But you can see there is no room for that here. The different languages embody different ways of thinking, different styles, different purposes. Many are variations of ALGOL. (If you want to immerse yourself in the great range of them, Jean Sammet's monumental book on programming languages is surprisingly readable.)

Lambda languages were first used in obscure research laboratories where delightful odd people worked on AI.

Suffice it to say that if you get serious about computer programming, you can make computer languages your never-ending study. Or if you go to do research at the Gazerkis Institute of Tough Science (if there is such a place), you will probably become a fan of their language and see no other. ▼