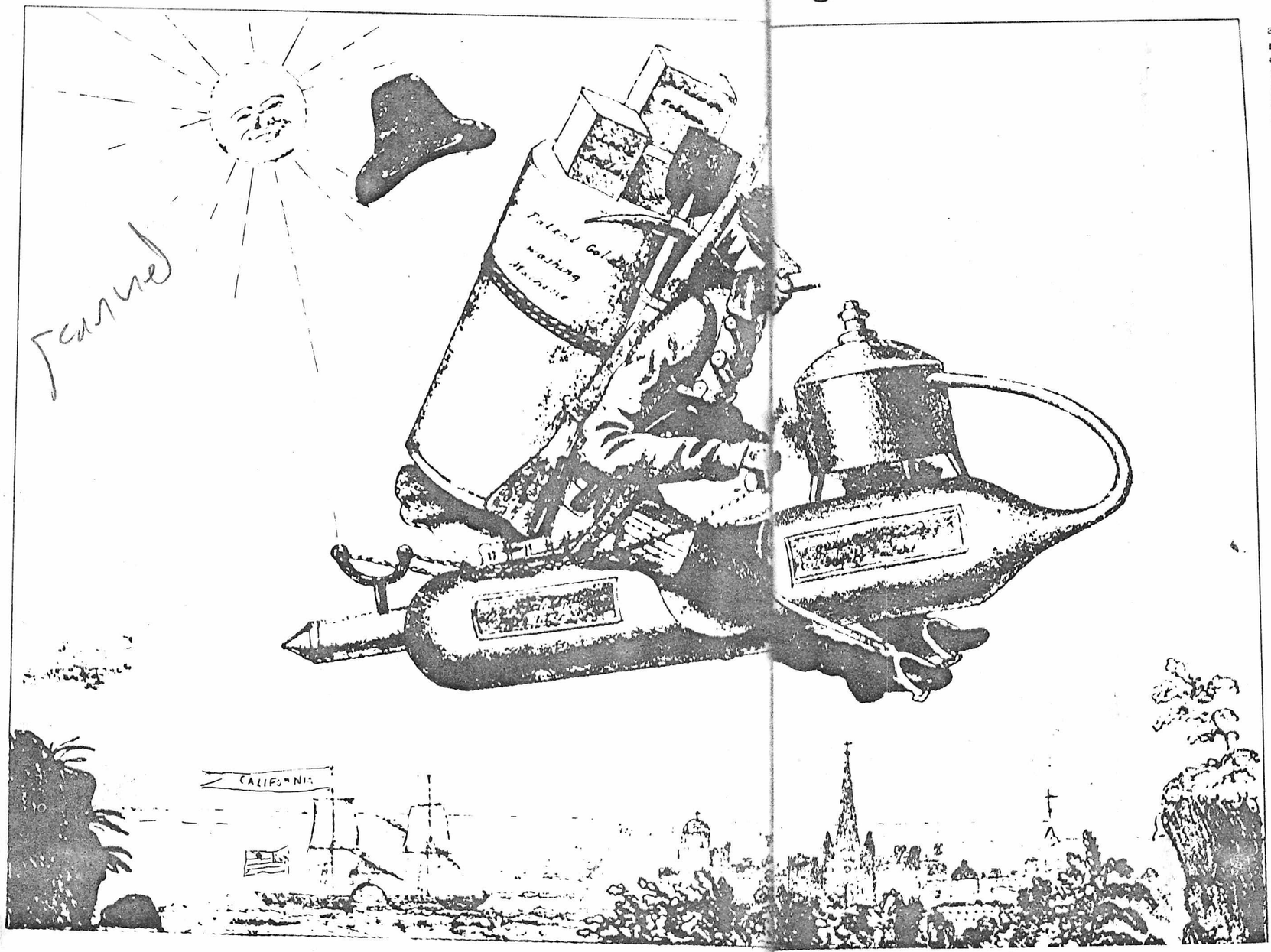
Interactive Systems and Design of Virtuality

Part Two

Ted Nelson



In Part I, we considered some nice examples of highly responsive systems. The reality of their implementation details is comparatively unimportant. What *is* important is the design of the conceptual structure and feel of a system; we call this its "virtuality" as distinct from the (unimportant) reality.

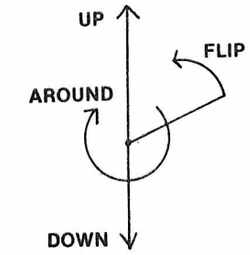
In this concluding section we consider some more design examples, and endeavor to find the right principles on which to base the design of interactive systems in general.

A COMPLETE SYSTEM

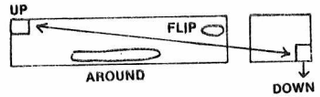
In one design, the Funny-Face Softree™ system, I have endeavored to show that one simple, overarching control structure can be used for a *complete* personal computer system—including word processor, scheduling system, graphics package, bookkeeping package, typesetting and layout programs, etc. (I do not wish to imply, of course, that this is the only way to organize such an integrated system; merely that this one interests me.)

There are four basic controls. These are the *only* controls. They may be understood quickly in a brief demonstration, but in fact the further ramifications of their interaction may become clear gradually.

The controls we call *up, down, around* and *flip.*



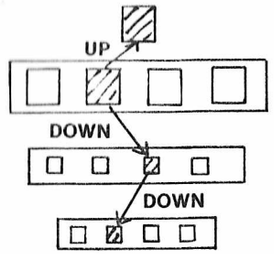
I would marry these to the Radio Shack keyboard as follows:



Up and *down* are the easiest. The user

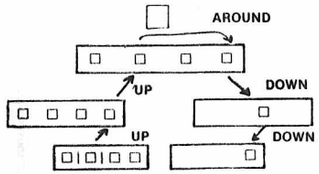
Virtuality, cont'd...

is at all times on a tree of functions. Each node is a particular activity or waystation on the tree. *Up* of course takes you to the node above you on the tree. And on this tree, *down* is always specified at any given moment as one of the specific alternatives below.

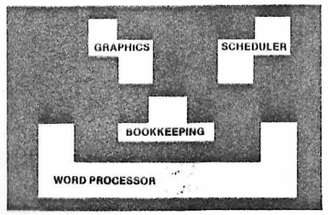


We may call this a *latching tree*. From your current node you may go down or up. If you go up, you get to the top; if you go down, you follow the path of already latched, or chosen, selections.

How do you change the selection of the node which is *down*? You do this by pressing *around*, which selects in turn each of the different alternatives below. (I call such a circular succession of choices a *ringstep*.) Thus to go between any two places on the tree only a few particular steps are required: something like *up, up, around, down, down*.



How do you see where you are and make the choices? Now comes the really unusual part. Each menu is a *jack-o-lantern face*.

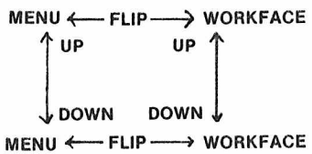


You go up and down a tree of menus. Each face has one of its features (left or right eye, nose or mouth) *flashing slightly*. This is the current selection below.

Now a frequent complaint about menus is that you have to take time to read them. In this system that is only true at first; because *every menu has a different facial expression*. So that as you become familiar with the different menus, their scowls and grins tell you where you are at once, and you can make your choices faster and faster.

At the very bottom level of the tree are particular activities; *down* there commands the events themselves.

There are also working faces, however, corresponding to every menu, on which materials may be viewed, scrolled, etc. This working face is the "other side" of the menu. You get to the workface, or back to its menu, by *flip*.



That's essentially all there is to it. What you have seen is what the beginner sees. I have left out showing how the different parts combine, so that, for example, the graphics tablet used with the scheduler produces animation, or the scheduler used with the word processor permits a magazine layout.

I would point out certain other features, however. One is that there are very few steps between paired activities, and the user going repeatedly back and forth between them gets into a rhythm. Faster methods would be in reality less simple.

Another aspect is the system's uniformity of replicative structure. You can go anywhere with confidence that the structure will hold. (It does become quite irregular, however, at the bottom or execution level.)

Some people tell me they'd rather have an input-string command language. That's a mater of taste. Other critics say this system lacks generality, which misses the point. It is simple, easy to learn, and integrated. You cannot get lost. And the funny faces are good for a laugh.

THE XANADU ™ HYPERTEXT ENVIRONMENT

The Xanadu ™ hypertext system, toward which I and colleagues have worked for some twenty years now, is intended as a super document library and annotation system, among other things. We may also think of it as a new form of storage and publication.

The Xanadu system is planned as a network of storage computers in McDonald's-like franchised stands around the country. By dialing into your local Xanadu stand, you may get anything on the whole network – to which your local stand is tied by high-speed lines. You must access the system from a fairly powerful terminal – that is, a computer, for reasons which will become clear later.

While most of the Xanadu work has gone into problems of its implementation – especially algorithmic design and analysis – the system's emerging virtuality has acquired an extremely interesting character, which I will now describe.

Everything stored in the Xanadu system we call a *document*. A piece of text, a picture, a movie (someday), a lonesome marginal note – each of these is a document.

Any document you want comes when you ask for it, if you are entitled to it. A document is private or public – that is, published. Any user may call up any public document instantly, as well as his own private documents or any other private documents he has permission to use.

LINKS AND WINDOWS

Links may be put anywhere in any document. Links, like footnotes or marginal comments, permit a user to jump to related material at any time – and come back from that other material when he likes.

Free-form, non-sequential writing of any kind – what we call collectively "hypertext" – is made possible by these links. But the virtuality of general hypertext would take a book in itself.

An important type of link is the window. A window may be thought of as a "hole" in one document through which shows a part of another document.

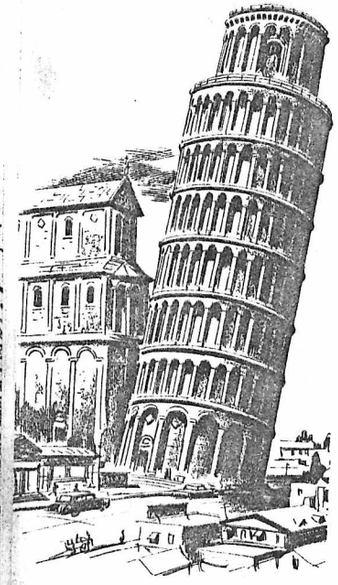
CHANGES AND VERSIONS

Not only may an author store a document in its present form; he may, if he chooses, write or rework the document on the system, with the changes themselves stored. The Xanadu system does this at a uniquely low incremental cost, since our data structure and algorithms essentially assemble parts of a given version as they are needed – without ever bothering to assemble the full consecutive structure, unless it is asked for.

Thus the user has access, if the materials are saved and open to him, to a reconstruction of any previous version of a document at any previous moment he cares to specify.

Not merely consecutive historical changes, but alternative versions, may be generated at any time. Thus a document may be "rewritten" for different types of readers, and these different versions stored at low overhead.

The user may ask to see any given piece of text (or other information) in any version or at any previous time.



A link made to a certain part of one version of a document may be automatically followed through to the same material in any other version of that document or in its previous incarnations or in other public documents that windows it.

We believe that this "versioning" facility, of linkage across backtrack and alternative versions, solves a central problem of text systems — that of cross-referencing any parts still being worked on; a problem which is chopped at and nibbled at everywhere but is often dealt with in ineffectual ways.

FREE LINKING BY ANYBODY

You may create a document that links to any other documents, if they are public (he who publishes must agree to this in advance).

You may create, in your document, windows to anybody else's public documents. (Since they get the royalty when their part shows, they should be pleased.)

This is how we handle marginal notes. If you create a marginal note, it is automatically put in a new companion document, your document, which is permanently linked to the document you have annotated.

(This "companion document" idea also frees you to *alter and rewrite any public document any way you like*—since the alteration is in a private file of your own that points to the intact original.)

COPYRIGHT

"What of the copyright problem?" you ask. Our solution is simple: as you use the system, you are continuously paying small increments of royalties to copyright owners. These are modest amounts, the same for all users: for instance, if we can supply the service for two dollars an hour at 30 characters per second, the fixed royalty runoff will probably be about five cents an hour. This is divided among the copyright holders in proportion to how much you used from each—sliced very finely.

What keeps people from making copies? Nothing, since terminals are under the control of individual users; but since everything is still stored on the system and available instantly, the cost and inconvenience of making and filing private copies will be often seen as superfluous.

OVERVIEW: THE XANADU SYSTEM AS A VIRTUALITY

The above description specifies a general and powerful facility for business, literature, correspondence and digital storage of all kinds.

As such it represents a cohesive and unified virtuality which has been thought about and reworked for years. Its appearance of simplicity and obviousness is the distinctive quality of a carefully wrought design: *There are hundreds of other ways to do these things*, as experienced computer people well know; yet making the parts hold together clearly, complement each other, and *make sense*, takes a very great deal of work.

XANADU FRONT ENDS

Of the functions described above, only a few are actually handled by the Xanadu service network: *put this away* and *give me that in such-and-such a version* are really all that the Xanadu back-end machines do.

The rest has to be done, actually, in your personal computer. Marginal notes, for instance, require making a companion document out of your marginal notes, for instance, and declaring it and putting it away in the network. Most users will also want to keep track of how they have been jumping among various documents and activities. These necessary functions belong in your own computer.

Thus the "full" Xanadu system, as we recommend it be used, entails a cooperating program in your personal machine that acts in these ways.

Thus full Xanadu service has two parts. The "back end" is the proposed Xanadu network; esentially all it does is store and fetch by versions and links.

But a high-powered terminal is needed by the user, to show the documents sent by the back end, to present the possible actions the user may take, and to translate these choices into the proper fetch-and-store instructions for the back end.

This is of course the "front end." There are many possible ways to visualize and control the Xanadu functions—even before graphics or music are stored on the system—and we welcome imaginative front-end programs of any design, even if marketed independently. The Xanadu project will, however, offer guidelines for front-end design.

If you choose to use the back-end network in some other way, that is your privilege as a customer; but in order to encourage what we see as desirable modes of operation, we will be offering various trademarks to software vendors who wish to create cooperating front-end programs.

Given the overall virtuality of the Xanadu system, there are countless possible ways to summon, visualize and control its operations on screen. All of these are valid and welcome. To give some ideas of the possible varieties, I will discuss two very different Xanadu front ends.

(Since these are highlights of the two front ends, no attempt will be made to show all the functions, reconcile their different emphases, or intercompare them.)

THE XANATREK™ FRONT END

The standard Apple computer, laudable as may be its general qualities and capabilities, has a few conspicuous limitations. One is its text screen, only forty characters wide.

However, an Apple strength is fast-action low-res graphics. Two pages of hardware memory are dedicated to either text *or* low-res graphics. We will proceed to use this fact.

The Xanatrek front end has been designed for fast and exciting use of the Xanadu facilities, as well as for invigorating use of its low-res graphics.

The system was, quite frankly, inspired by *Star Wars*, and shows how far you can go in playful and analogous use of graphics.

One of the things a Xanadu user must be able to do instantaneously is ask exactly what he is looking at—that is, having jumped to something or wandered by degrees from his original activity, he

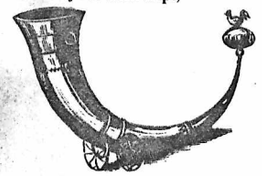
eeds an instant and valid explanation of what he is looking at.

Very well. While reading anything from the Apple screen (from page one of memory) the user may instantly demand a map of what he is seeing. This is continually available and up-to-date in low-res color, on page two. But aha, you say, how can you read it, since the color display disables the character generator? The answer is that the various patches of identifying text are indeed visible on this map as patches of seemingly random color, but since the Apple allows *one text window* on a low-res screen, successive boppings of a particular control will step the various text labels into a readable panel.

The most amusing visualization in the Xanatrek front end has to do with seeing the major features of a document such as chapter breaks and seeing links as well from a companion or other document.

This brings out the "Star Wars" styling. What the user sees looks like a huge passing spaceship or perhaps a packing crate, in the vault of night.

One of its visible sides shows the major parts of the text itself, as streaks of color. The other visible side shows the entrance points provided in your companion document. (You may select any of these places for your next trip.)



Other ships that pass in the night are documents linked to this one. Want to see the links? Hit a button, and animated squares *fire from one ship to the other*, with that p'tew p'tew sound we *Star Wars* fans have grown to know and love as the sound of a laser weapon in a vacuum.

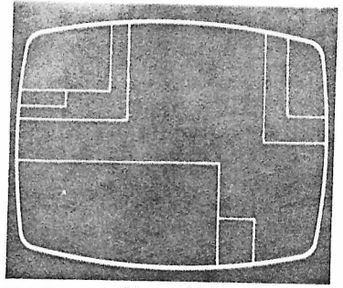
THE CORNERCOPIA™ FRONT END

This Xanadu front end has been thought out for implementation on an actor language on a small Sorcerer computer.

There are many approaches to the design of screen panels. One approach, generally associated with Xerox PARC, strews panels diagonally on the screen. The approach that follows is intended to be a little easier.

Five to ten screen panels are accesible at a given time. They come out of corners of the screen.

Each panel keeps one free corner anchored in a particular corner of the screen. Its opposite corner remains always visible but may be moved by the



user to any position which does not obscure any other panel's free corner. As with PARC panels, any "behind" panel may be instantly brought to the front without moving its borders.

Each panel is labelled with a one-line title (at the top of the lower panels or the bottom of the upper ones).

The text may show and scroll in any panel; naturally dependent or "parallel" text (a standard Xanadu statement) may scroll in any other visible panel with links to the independent text shown by scrolling symbols on the panel borders.

Perhaps this environment seems not to show enough. Very well: some panels themselves represent *other* such environments; when brought to the fore they swell up to become other multipanelled views.

OOF

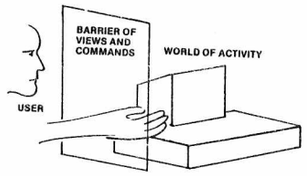
The "office of the future" will consist basically of cabinets for incoming correspondence, printers for outgoing correspondence, and in between, screens, screens, screens.

In this highly competitive area, harried programming managers everywhere are under pressure to work out what *happens* on the screens. But what do they, or *anybody*, know about it? It's *not a technical problem!* It's merely delegated like one.

The problem has nothing to do with technicalities; all of these are squared away. The problem is in the design of virtuality. But I know of few designers at present competent and imaginative enough to make those screens come alive and make working at them a joy. Which is the *real* problem.

WORLD AND VIEWS

An interactive virtuality is essentially a



world created by a programmer or designer. This world has a certain structure which may be easy to understand or hard. This World is visible through different *views* allowed by the designer.

The World is what you're really thinking about; the view is the temporary way you're looking at it.

The distinction between World and View is crucial. The World is what the user is supposed to be acting on and thinking about; the View is all he really gets. (Controls are in a way part of the view.) If Views are good, the World comes to seem real, natural, at hand, under control. Poor Views (or worse, a hard-to envision world) create confusion and poor usability.

The system should have easy-to visualize states and conditions, and, preferably, some kind of spatial orientation that readily becomes a map in the user's mind.

The designer should begin by thinking about visualizing the *World*, not the Views, and let the Views come later.

(Yet designers are always getting seduced by particular views and treating them as the world itself.)

The principles of the World are the central, integral, virtuality; how you see it is secondary. It is important to acknowledge the cruciality of World design, and consequently the importance of the principles you develop for it.

The designer creates a simplification or stylization of the original world. There must always be some reduction or stylization; the important thing is that these reductions or stylizations not detract from the principal things you need to understand and control.

In transposing an old activity, the question is what to retain in the world and what to dismiss as part of the view. (For instance, Text Pages—divisions of text—are part of the View, not the World.)

Anything can be shown, any buttons or sticks or whatever, with any preentational machinery. People are always asking for bigger screens—but actually to ask for a bigger screen is usually a copout. Ask for higher performance. Faster flips and flaps and scrolls and panel pop-ins. Fast action and seething cues. Leave several things on the screen at once, to remind you of what you've been doing, what you might be doing, what else there is to do, and any other current options.

The operations in the user enviroment should feel more and more like operations on the world. As stated above, controls are in a sense part of a view. Anything can be controlled by almost anything, buttons or sticks or keyboards.

An interactive system should have very few controls and these few should have far-reaching and powerful uses.

Marry the available controls and the

desired functions. Menus should be used, rather than input languages or the fictitious "natural language dialogue;" or better, yet, control diagrams.

Actions should be easily reversible and their consequences immediately recognizable so the user can back out of a mistake without being punished. (Compare this with the word-processor horror stories you hear all the time.)

Most important, the overall *principles* you choose for a system should be sweeping and have few or no exceptions. In order to clarify these issues we must consider the issues of both soft principle and soft clarity.

THE PHILOSOPHY OF SOFT PRINCIPLE

The following discussion has to do with the design of principles, which is in fact the essential issue.

SOME FAMILIAR IDEAS SOFTENED AND RECONSIDERED

	Hard	*Soft (or mixed)*
IMPLICATIONS, RAMIFICATION	Hard Implication, Consequences	Possibility, Tendency, Expectation, Connotation
PARADOX	Contradiction	Irony, Oxymoron
COMPLICATION	Obstruction, Interference, Contervailing Principle, Something in the Way; Amendment, Modification	Things to be Clarified, Resolved Worked Out

We frequently consider something and ask ourselves: *What are the implications of this?* And one of the nice things about science and technology is that the implications tend to be clear and exact.

In many cases, though, implications tend to be less certain. Implications don't follow clearly from premises. Those who want clear-cut answers become edgy or annoyed. The main tradition of Western thought has been to try to find the exact implications of every idea. (Ideas which don't seem to have exact implications, as well as people who *prefer* unclear situations, cluster in the humanities or "fuzzy studies.")

But some things are by their nature unclear in implication. These include both cluster-concepts ("Democracy," "Womanhood") and design ideas ("Let's see, maybe it could fold back onto itself somehow").

By tradition we often tend to talk of such ideas as improperly formulated or incomplete. I would like to put it another way and call a principle whose implications are inexact, a *soft* principle.

This throws things in another light. Rather than suppose the soft principle is just "not finished yet," let us consider it instead *another logical category*—somehow analogous to the conventional hard principle, but not subject to deduction.

If you can't deduce, how can it be logical? The answer may be that we've been looking at the wrong features of logic and have missed the analogy. There are in a sense soft equivalents to implication, contradiction, and other logical configurations. (See table.) I hope to develop these ideas more broadly at a later time.

What good is this analysis? At the very least it is suggestive. If a principle is *by nature* soft then we can understand it on its own terms rather than insisting that it hasn't properly hardened "yet."

Or take soft design ideas. A given idea could be worked out into hard form in numerous different ways. Some you may like better than others, and a variety may be valid.

Now take *several* soft design ideas, all at once. How do their ramifications fit together? The answer is *indeterminate*, since the ramifications of each could take many forms. But if you are aware of this, then you can search carefully for the *combinations* of possible workings-out, their variety and their interactions.

The "inspired" design of something final and precise comes, I believe, from sifting many such co-implications of possible hardenings of the ideas.

And the important guideline is: *don't rush it*. Don't take shortcuts. Don't assume that decisively pinning down one aspect of a design will speed things up; it's like nailing your left shoe to the floor.

If we think of design as the search of many possibilites, "soft design" is that which is sensitive to unexpected simplifications, conveyances and harmonies.

In short, *don't be too sure of what you're looking for, and be ready to appre[ciate] the ramifications of surprises.*

Principles in Practice

Eventually, the soft design principles we have tried out lovingly must be hardened into specific hard forms of computer operation. What should the principles be like? Again tradition may militate against recognizing the best design decisions.

The general principles of a system, *once chosen*, should be consistent, but "consistent" according to looser criteria than the designer may be used to. In particular, a design principle may be psychologically clear for people to work with, easily visualized or imagined, yet not reducible to any customary formalism.

Indeed, "consistency" here takes on a strong psychological flavor: a thing is consistent if users think it is consistent and use it consistently—even if we don't like it, like the double negative in Spanish. (We may call this naive consistency or soft clarity.)*

Thus the final chosen principles need not be "logical" in the rigid sense of conforming to somebody's predefined notion of how things should behave. But working out in soft form, we study their fittings-together in great detail.

The designer should eliminate any background notion that the user must be like him. All too many designers reward the user for being like himself, the designer, or punish the user for being different or thinking differently. The objective is to be of service, not to clone yourself.

TECHNICAL TRADITIONS VS. SOFT DESIGNS

The design of virtuality is essentially the design of operating principle. The design of principle, in turn, has to do with the generation and modification and inter-sculpturing of soft principles.

The biggest design problem though, is that the designer tends to freeze too quickly on a particular set of rules and arrangements. Technically-oriented people tend to seize one or two principles and hang onto them through thick and thin, not perceiving when it is time to rework their ideas.

I have learned through bitter experience, indeed, that only a small proportion of technical people are even capable of *listening* to this viewpoint. The soft design of virtuality seems to be totally alien to technical training.

* Mark Miller, who worked on the original JOT system, considers it a consistent virtuality, even though it "corresponds to no known paradigm of program structure."

Virtuality, cont'd...

Those who design interactive systems tend to be technically trained, and technical training generally promotes the background assumption that what you are working on is given and well-defined.

Training in the arts and creative fields, on the other hand, promotes the ideas that a design (or piece of writing or a movie) is fluid, may take many forms, and will be reworked over and over until it reaches a final state that may be wholly unlike its earlier stages. I believe this latter outlook is far more appropriate for the design of interactive systems.

CONCLUSION

Interactive system design is a field in itself, utterly unlike what is taught in any computer science department I know of. If I have not proved this point, I hope the designs and ideas presented here will at least provoke some unease.

(This is no claim that these designs are righter than any others; but rather that these designs are a unified package that feels right and is therefore of interest. They represent local peaks in design space, in the sense that small changes would, I think, detract from their unity and clarity.)

These designs represent hundreds of hours of work, but the difficulties of the decisions and the rough edges don't show. (That's part of good design and art).

The art of designing things in general is very little understood. People think that something is well-designed if it is *sleek, stylistically unified*, and *if its controls look as much alike as possible*. (An example is the "designer" audio equipment from Bang and Olufsen, show at the Museum of Modern Art and copied everywhere, where every control resembles every other control.)

This approach is wrongheaded beyond belief. (I think stereo equipment is poorly designed, and B&O the worst of them.) You do not want controls that look alike. You want controls that look and feel *different*. If you have a big round knob for the volume control, you should have a square knob, or a slider, for the tuning. There should not be a row of similar buttons for different functions, but a row of *different* buttons—or better, not in rows, but some other arrangement contrastively arrayed. Do you need glasses to read what it says above the knobs? Lousy. Can you tell at a glance one control from another? Good. CAN YOU WORK IT IN THE DARK? Terrific.

As a rough guide, *good design is inversely proportional to the probability of a user making a mistake*. And this criteria carries over to interactive computer systems.

To make a system easy to use is extremely difficult and time consuming, in the same way that it takes more work to write a short article than a long one.

You should not "design the system" first, and then put on a "friendly front end", (although this is what must be done in many cases), any more than you should first shoot a movie and decide what it is to be about (although this occasionally works).

An interactive system should become second nature, and become second nature quickly. This is essential for many reasons. One is that we will have to move among many different interactive systems in the future, and there will be no time to savor and adapt to the local complications of each. They will have to spring clearly and straighforwardly at the mind and hand.

Moreover, interactive systems will be used intensely for hours, often by tired, high-strung, frantic people, who are trying to get a job done in a hurry, and who are thinking only of the world they are trying to operate in—not the intervening complications. It is up to us as designers to create fast, safe, elegant systems of view and operation without snags, dangers or complications.

The system designer, or movie director—let's call him *you* —must have a full understanding of what things are easy to do, what things are not, and what is hopelessly impossible. You then make a collection of all the ideas and visualizations (and scraps and parts) you would like to put together in your system. Then your rework them and rework them, and rework them.

THINK OUT THE WORLD

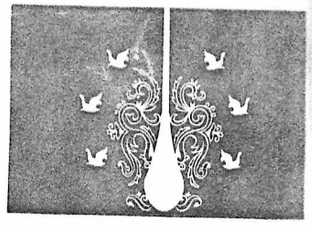
—Its many views and aspects; it s *real* nature (unlike what has been thought of as its nature);

IMAGINE ALL THE CONTROLS AND PRESENTATIONS YOU'D LIKE TO HAVE,
REDUCE THE CONTROLS
AND PRESENTATIONS
TO AN ADEQUATE,
POWERFUL,
EASY-TO-UNDERSTAND SET:

MARRY THEM TO THE AVAILABLE SCREENS, KEYBOARDS AND POINTING TOOLS.

ABOVE ALL, DESIGN THE FULLEST SYSTEM FIRST—THEN CUT IT DOWN, IF YOU HAVE TO. YOU MAY FIND YOU DON'T HAVE TO.

That this is nowhere taught is much worse than regrettable. Because unfortunately the salaried programmer has, in effect, a license to inflict on innocent users anything he likes under the pretense of technical necessity or on the basis of some off-the cuff (or cufflink consultant's) assessment of "user needs."



I regard the decisions involved in designs like those as intricate and interdependent as moves in chess. This kind of design needs a respect and even reverence for the far-flung ramifications of tiny decisions, and the staggering complexity of making things simple.

I hope I have given a sense of this style of design.

I hope, too, that the reader will see it as an art form—somewhere between movies, diagramatics, the design of machinery, the design of games, and the building of philosophical systems.

When done well, it is done with simplicity, consistency, conceptual clarity and vividness. This is not "technical" work in any usual sense. I consider it a form of design and a form of art.

I believe that interactive design is, more than anything else, what the computer field is really about. I find it monstrous and appalling that these general principles are so little understood; that despite all the pompous "computer science curricula," nobody teaches these anywhere; and that innocent customers who want an easy-to-use system—really, is it too much to ask?—are too often led by consultants and tekkies down a primrose path to endless horrors of complication and unnecessary claptrap.

How you feel about all this depends on what you think computers are all about and where the world should be going.

If you want to show off to your family and friends—or financial backers—as a macho master of complicated technicalities, then you don't *want* things to be easily comprehensible. (In that case you should be reading certain other personal computer magazines.

But if you believe that somewhere beyond all the technicalities lies some kind of hope for a better future and a smarter mankind, rich in ideas and knowledge and dreams—as well as gadgets—then the question is how to front-end the gadgets so that they bring us knowledge, and ideas, and dreams, without the technicalities being in the way. □