
Apple Manuals


Apple][Pascal


Macintosh


SwyftCard


Canon Cat

Jef Raskin Information

HUMAN INTERFACE DESIGN:
JEF RASKIN INTERVIEW

DDJ / MAY 1986 / PAGES: 5

SOURCE: MR. HELGE HORCH / MAY 1997

DOCUMENT NO. 52

COMPILED BY
DAVID T. CRAIG
736 EDGEWATER, WICHITA, KS 67230 USA
E-MAIL: 71533.606@compuserve.com


Apple Manuals


Apple][Pascal


Macintosh


SwyftCard


Canon Cat

"RaskinDoc052_0.PICT" 207 KB 2000-02-22 dpi: 400h x 400v pix: 2928h x 3953v

ARTICLES

Human Interface Design: *Jef Raskin Interview*

by *DDJ* Editors

Jef Raskin's involvement with personal computer user-interface design is hardly recent. In early issues of DDJ, *he inveighed against needlessly complex systems. At Apple Computer, he wrote many of the company's product manuals. Later he was a member of the original Macintosh design team. Today he is the founder of Information Appliance of Menlo Park, California, which currently offers a product called SwyftCard for the Apple IIe and IIc.* DDJ *will review SwyftCard next month.*

Raskin cares tremendously about user-interface design, on computers and elsewhere. During our discussion he said, "As a person I'm easily frustrated. I'm very annoyed at the little things that most people put up with." He got up and walked out of the door, shutting it behind him, and immediately opened it and reentered the room. "What a nuisance! If I had two bags of groceries and a locked door! The door is a very, very bad design! Someone will have to improve it. But we're so accustomed to it that we don't think about it."

If Raskin doesn't like doors in buildings, does that mean windows on computers are also unacceptable? In the following interview with DDJ, *he discusses a wide range of issues concerning user-interface design.*

DDJ: For how long have you been thinking along human-interface design lines?

Raskin: When I was a computer center director at UC San Diego, there was a major computer center with a huge machine. I built another center. It didn't have fluorescent lights. It used minicomputers—the very early Data General Nova—and it was time sharing. Instead of submitting things on punch cards in the big computer center, you could come over to this little computer center I had built, and there were 32 terminals, beanbag chairs, and incandescent lighting with Japanese globes. You had individual terminals to work on. Students loved it. Even people from the big center would come over to use it.

When I go to a conference and tell people why I hate a mouse I get applause.

We also had a language called FLOW, which had only six or seven commands and was very good for teaching programming. So working on better language design and the whole ergonomic question dates back to the 60s, when everyone else was looking at anything but that.

DDJ: Can you identify some of the important criteria in the design of user interfaces?

Raskin: In any user interface there's an almost unquantifiable factor of feel. And this is something I find so hard to explain. You can only achieve it by massive testing on human beings.

DDJ: Like with SwyftCard?

Raskin: SwyftCard is a little product for the Apple II, and we tested it on 1,500 people. There were focus group tests, one-on-one tests. There were small-group tests. There were tests in schools. We learned a lot. We made a lot of changes and did a lot of fine-tuning.

DDJ: But you started with something close to the product's current user interface. You had a preconception of what would work, didn't you?

Raskin: I've been concentrating on the question of how to make systems feel good for 18 years now. Sometimes that kind of concentration produces ingrown, moribund ideas, and sometimes it produces results. I think in this case it has produced results.

DDJ: Besides feel?

Raskin: There's habit, and here we're on better theoretical ground. A system should allow a person to form habits. There are two fundamental principles that help people form habits. One of them is well known in this industry, and that is making things modeless. It's just a fact of life that people can't keep track of modes. You make mode errors when you think you're doing one thing and you press a key, but because you're in the operating system and not the editor, the system does something else. I remember in the old UCSD Pascal system, I would sometimes press E for exiting, and in some other place, it meant execute—it would try to execute a letter to my mother or something. You can imagine the kinds of Pascal syntax errors you get from a letter to your mother!

DDJ: No modes. That sounds like an underlying rule.

Raskin: If a system is modeless, then you develop habits. If you want to do something, you just reach and do it that way—you don't get frustrated.

Modelessness has never been defined satisfactorily. It means that a given action has one and only one result. When you put the definition in that form, you can easily form a converse: To get a particular result, you want one and only one action. If that's the case then you don't have these branches. If you read Stewart Card's book *The Psychology of Human-Computer Interaction* (Hillsdale, NJ: Lawrence Erlbaum Associates), one of the better books in the field, you find a lot of the time you take isn't the doing time but the thinking time—working out a path. How am I going to get there from here? If you have only one way to do something, then you don't have to think about that. You can form habits even faster. If it's modeless your habits won't trip you up. In the industry we call this other property monotony, for lack of any other name. It means for a particular action, we try to have one and only one way to do it. If a system is modeless and monotonous, then you can form habits because whenever you want to do something you always do the same thing. It's like tying your shoe. Imagine if every Thursday your shoes exploded if you tied them in the usual way. This happens to us all the time with computers, and nobody thinks of complaining.

DDJ: So you have no modes, plain and simple?

Raskin: We actually did deliberately introduce one mode. We're not so doctrinaire that we won't do something when it does make sense. But our system is largely monotonous. There's generally only one way to do things. I remember reading a description of a system where the developers said, whenever they had a disagreement about how something should be done, they did it both ways. That's not a design decision.

DDJ: As in whether to use a mouse or command keys?

Raskin: Usually if you can't decide which way is better, it means you haven't found a good way to do it. On the Macintosh there's almost always a way of getting around the mouse by using the keyboard. This should have given the designers, after I left, the hint that maybe the mouse was not the right way to do it. If you're always looking for a way around it, you've obviously got some kind of problem.

DDJ: Documentation has always been a specialty of yours. Does it enter into your definition of the user interface?

Raskin: Our user manual has something interesting in it; aside from having schematics and the usual theory of operation, here's something I've never seen in any other manual that I've ever looked at—the user-interface theory of operation, or how we designed the user interface.

DDJ: You believe the user must understand the theory?

Raskin: No, I don't feel the user needs to know. I know, as a reader of manuals, I often wonder: Why do they do this? How did this come about? You can use the system without reading any of the theories of operation. You can learn the system by reading about 40 pages. Some people learn it from a reference card. We have only five commands. We could have had a manual that just ex-

plained the five commands. But this manual is part of our user interface.

Let's talk about manuals. We have a long, cross-referenced, handmade index. A lot of work went into this. Before we did this we typeset the manual for testing because we wanted people to feel they had a finished product. Then we got feedback not only on the product but also on the manual, so what people get as our first-edition manual is a tested manual. Manuals are an integral part of the product, not an afterthought. This whole product was designed from the very beginning with the manual in mind. In fact, the very first thing I did when I started the company was write my dream manual for this product. Then I built it and finally wrote a real manual.

DDJ: That brings to mind Apple's ad that boasts of the minute amount of documentation needed to learn the Mac compared to an IBM computer.

Raskin: Our first idea was to have a thin manual. We tried that—experts said, "Wonderful," but beginners said, "I don't know what I'm doing." If you say that an insert command undoes the latest block delete, people may have forgotten what a block delete is and *insert* is a funny word. We wrote the manual so we explained everything. If we had to explain something five times, we explained it five times. Don't get lazy in the manual!

DDJ: Beyond modelessness and monotony, what else do you have in mind as you develop a product?

Raskin: One of the most important concepts is that things you do frequently must be fast. Things that you do infrequently can be slower. People have critiqued our way of setting the widths of paragraphs, saying it seems a little baroque, but you hardly ever do it. One thing you do very often is move the cursor, and one of the big advances of this product is the cursor-moving mechanism. This was not theoretical. It hit me while driving through Marin with my wife. No amount of theoretical analysis will give you a better system without inspiration. I know of no way of automating that process.

DDJ: But you began with an antipathy toward the mouse?

Raskin: I happen to hate mice, and I have since 1973 when I first started using them at Xerox PARC. I always preferred joysticks. I did want a different cursor-moving mechanism on the Macintosh. That was designed as a graphic machine, and you do need a graphic cursor-moving device. SwyftCard is nongraphic and doesn't need a graphic device. *(At this point Raskin started to demonstrate the SwyftCard while he talked.)*

First of all, it does not require you to move your fingers from home row, and it uses your thumbs, which are under-utilized fingers. Even in touch-typing the right thumb is used for the space bar only; the left thumb does nothing. Strangely enough, the Dvorak keyboard, for all its supposed efficiencies, doesn't use the thumbs either.

To get anywhere in about 20 pages of text, you have to type an average of 3½ characters. Research has shown that with cursor-moving keys the average time is around ten seconds, and with the mouse it drops to around four seconds. Here it drops to around a second. So it's somewhere between two and four times faster than a mouse, and it's a heck of a lot less expensive to manufacture.

DDJ: Had you seen anything like this that helped you in the early stages of development?

Raskin: No, after I designed this I learned about the Find command in EMAX, which is also an incremental search. It has a few problems: First of all, you have to go into Find mode; and second, it ends up on the last character of the pattern, which is a big mistake. The other thing with EMAX is, when you leap somewhere and you start typing, you're still adding into the pattern. So it's modal, and it puts you on the last character. It's only one direction.

Let me make a typical error. I want to move the cursor to the word *good*, so I should press the left Leap key and type "good." I'll press the right

(continued from page 34)

Leap key and type. It found it anyway. The system does one thing that all systems should have done from day 1: If you tell it to search one way for something and it doesn't find it, it searches the other way in case you made a mistake. Most systems didn't do this because if you did find it then you've lost your place. In this system if you want to go back, you just bang on the keyboard. *(Raskin slams both hands on the keyboard, and the cursor returns to the point in the document at which his search began.)*

The other thing about this system that allows us to use this paradigm is that the longest search and display through all the text takes 300 miliseconds. Cursor motion is very important, and we've got it better than anyone. Speed is very important, and here on a stupid old 6502 running at 1 megahertz, we're moving the screen and doing things faster than you will see on any computer from any manufacturer at any price with anybody's software.

DDJ: On the Mac development team, was there an absence of theoretical drive?

Raskin: The original team was Bud Tribble and myself for software development. When Steve took over, Bud Tribble left. And I left. He then brought in a group of people who essentially implemented a standard operating system and just put a different appearance on the front of it. There was a strong theoretical drive initially; all the people who were doing that left. Our name for the word-processing program you get with the Macintosh is Macwait. If a little clock ever appears on a computer of mine, I'll shoot it. A computer is supposed to be fast.

DDJ: Could you talk about the cursor design on SwyftCard?

Raskin: One of the things we've done is we've paid a lot of attention to details people have taken for granted for years. The way I've gotten out of my ruts is by watching people try to learn to use our own systems. The cursor is in two parts. There is the blinking part, which we call the cursor, and the other part, which we call the highlight. The rule is, when you type a character, it will always appear where a blinking cursor is, and whatever character was underneath it gets moved out of its way. Always. It will never happen to the right or left of the cursor. The highlight is where the next character to be deleted will be deleted when you press the Del key. When you're typing it shows exactly what's happening. I don't know how many times I've made the mistake of mispositioning a cursor on a system that deletes to the left and inserts to the right, like on the Macintosh. That causes a lot of confusion.

We spent six months before we realized a cursor has to be in two parts. We played with cursors between the letters, on the letters, cursors that flickered back and forth—dozens of designs. You always want to use a left delete after you've been typing. If you move the cursor somewhere, you always think of words from the beginning of the words. You move there, and you always want to delete the other way after you've moved the cursor. We observed that to be a 99 percent phenomenon. So we have it automatically. Whenever you leap, the cursor knows to delete to the right, and whenever you're typing, it automatically backspaces. This is definitely modal, and once in a blue moon you will make a mode error. You will press Delete and take out the wrong character. But it's so much gain.

DDJ: So you've eliminated many commands normally found in an application, especially in a program with several applications.

Raskin: Throwing out commands is a very big thing. This system does word processing, information retrieval, telecommunications, and calculations, and it has five commands. Any other word processor has more than five.

DDJ: You've complained that the Mac ended up with a traditional operating system. How does your system differ?

Raskin: We threw out the whole concept of an operating system. By definition, an operating system is the program you have to fight with before you can fight with an application. For a single-user system on which you're not developing software, who needs an operating system? There is no operating system running underneath this. The editor runs right on the bedrock of the silicon. There are no menus. You know that people like broad, flat menus rather than deep menus. What is the broadest and flatest menu you can imagine? A few things labeled on the fronts of keys. Everything is available instantaneously and simultaneously. If I want to look up a telephone number, I don't have to get into Information Retrieval mode, I just use the Leap command. Leap is information retrieval. If I want to do a calculation, do I say "Calculation mode" or pull down the calculator? No, I simply type the equation and press the Calc button. That's the way it should always have been.

DDJ: The most basic concept to most interfaces is the separation between applications.

Raskin: In today's world! In tomorrow's world, interfaces like this will clobber the usual present kind, and in a few years many kinds of products will have this kind of interface.

DDJ: What's held people up from seeing this sort of integration?

Raskin: We already have integrated software, and that's integration by a menu. We call this homogenized. Why didn't I see this years ago? I have no idea.

DDJ: Can any application be homogenized?

Raskin: Yes. You name it, and I can homogenize it. The basic principle can be applied to all applications that I know of on computers. The work we've done here can be applied to any computer, small or large, and any application.

DDJ: Given the right hardware?

Raskin: Given the right software. Most people think the Apple II is the wrong hardware for any spiffy human interface. Part of the problem is this doesn't show up on television. All you see is your work. But isn't that what you want to see? Do you want to see the computer mechanism, or do you want to see your work? We don't waste any of the

screen with indicators or messages. It's all yours. On a 64K Apple II with our system, the user gets 40K. If you buy a 128K Mac and MacWrite, you get 20K out of 128K. Our entire code for this product is 14.5K. Nobody even thinks you can write an application, much less four or five applications, in under 16K bytes. Those numbers have not been heard for years. Go back to the early days of *DDJ*!

DDJ: But you must feel some frustration with this product on the Apple II?

Raskin: Sure. The keyboard layout isn't optimal. The Leap keys should be larger. They weren't designed for leaping use. There are many applications on other hardware that we simply don't do here. For one thing, we don't have the spreadsheet integrated because the Apple finally ran out of power at a certain point. But the hardware is not so important. And the amount of stuff you can do with an 8-bit processor and even 16K bytes of memory have not been exhausted by humanity and will never be.

People like Steve Jobs say that the way to get simplicity is greater complexity. Well, they don't put it quite that boldly. What they say is, if we have intelligent enough systems and big enough systems, we can make them very easy to use. I have this stupid idea of simplicity via simplicity—and for many applications it works. I'm not denigrating all of AI and that stuff. I think there's a lot to be gained from it but not for little things that people are going to use on an everyday basis.

I was a visiting scholar in the artificial intelligence lab at Stanford, and I had an office at PARC. I was never an employee at PARC.

DDJ: So you think the mouse created a barrier toward designing better user interfaces. What's better for graphics?

Raskin: My favorite graphic input device is a tablet. That's what I find easiest to use. It feels like a pencil. I spent years practicing, using pencils. I've thought the mouse was a mistake for a decade and a half now. I think it's being foisted on a lot of people. When I go to a conference and I tell people why I hate a mouse, I usually get applause.

DDJ: Specific applications create certain needs in interface design. What's an example of this?

Raskin: Most people who do a lot of heavy telecommunications end up having two computers because, if you want to receive messages at an arbitrary time, you have to leave your telecom package up. With your telecom package up, you can't do anything else. On this system, it's always in Telecom mode. If you send me a message, whether I'm working or not, I won't be interrupted. If I'm not there, it'll just accept it; if I am there, I can finish my thought and then read your message. So here's a place where modelessness buys you a whole computer.

DDJ: Have the main issues changed in terms of what a programmer wants and a user needs?

Raskin: I think so. There are some programmers I haven't hired here because they say, "Hey I'm not going to get a chance to hack at systems or Unix or anything like that here." The programmers we do have here have as their first priority making things work well with people. That means you can't have an interest in developing a new and better operating system. In a few years all computer-science departments will have user-interface courses; some do now. And that will become more and more recognized as legitimate and worthy. We have to have a stronger theory of human interfaces. Part of what I'm doing is developing such a theory in my spare time.

DDJ: What would you concentrate on in a user-interface class?

Raskin: First of all, you've got to get people to recognize when they're having trouble. People always say, "It's me," but it's the computer design that's so dumb. So first, you have to get programmers out of the cocky position of believing they know how to design stuff and to a humble position that if a person is having a problem, it's not the person who's dumb, it's a dumb design. That's the start. And have them learn statistics and experimental design. And then understanding how human beings learn and work—cognitive psychology and learning theory.

DDJ: Will different languages help in the creation of better interfaces?

Raskin: Definitely. We used Forth. That helped us here because it was small and runs like a bat out of hell. You don't have many compunctions about dropping down to assembly language if you really need speed. So for sheer performance, it was a good choice. For human-interface design, most of the languages like PROLOG and LISP or APL are totally inappropriate. I've been reading some articles about people writing simulators on which you can test human design interfaces. They say it's very slow, which invalidates it. Unless they can simulate the real speed, they're not getting any useful data. There's no language per se.

DDJ: What thoughts did you have along these lines while you were doing things for *DDJ*?

Raskin: Go back and read the very first article I ever published in 1976 in *DDJ*. Read Jim Warren's little comment about me at the bottom. *["Jef Raskin is well known for his heretical belief that people are more important than computers and that computer systems should be designed to alleviate human frailties, rather than have the human succumb to the needs of the machine."]* It's still true, word for word. I've been trying to fulfill that belief ever since.

DDJ