
Apple Manuals

Apple][Pascal

Macintosh

SwyftCard

Canon Cat

Jef Raskin Information

DOCUMENT TITLE

RASKIN INTERVIEW

"PROGRAMMERS AT WORK"

SOURCE

1989

DOCUMENT NO. 7

EX LIBRIS
DAVID T. CRAIG
736 EDGEWATER, WICHITA KS 67230 USA
E-MAIL: 71533.606@COMPUSERVE.COM


Apple Manuals

Apple][Pascal

Macintosh

SwyftCard

Canon Cat

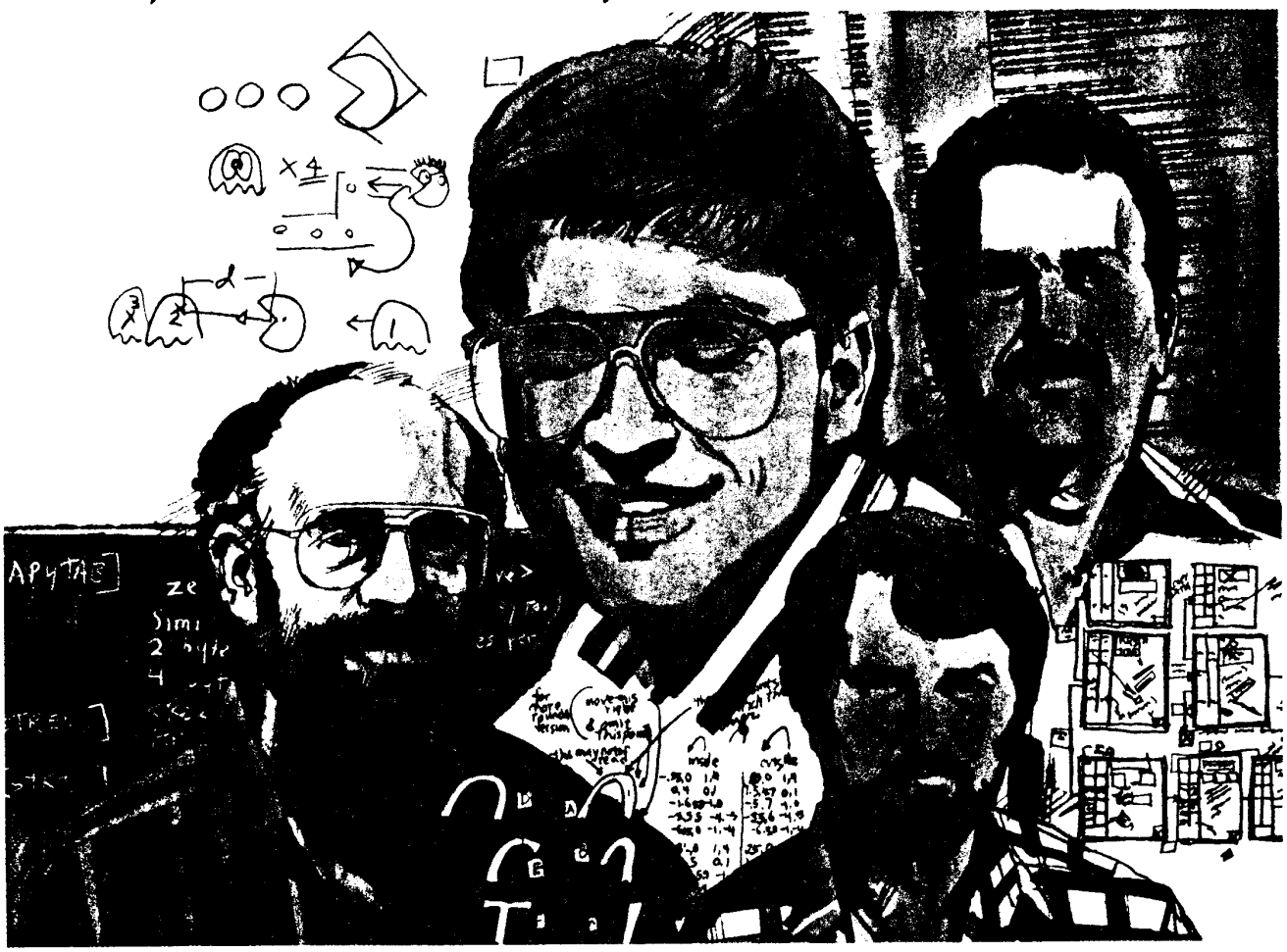
"RaskinDoc007_0.PICT" 141 KB 2000-02-21 dpi: 300h x 300v pix: 2220h x 2895v

20 pages

PROGRAMMERS AT WORK

JEF RASKIN INTERVIEW

Microsoft Press, 1989



Interviews with 19 Programmers Who Shaped the Computer Industry

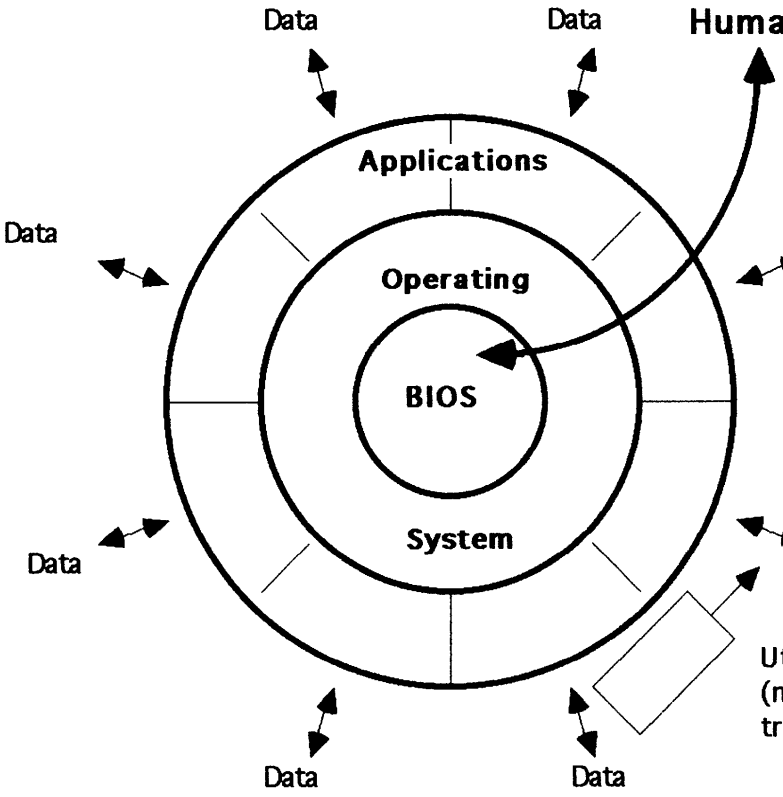
DAVID T. CRAIG

Susan Lammers

Sketch by Jef Raskin showing the concepts that led him to start Information Appliance Inc. in 1982

Source: Programmers at Work, Susan Lammers, Microsoft Press, 1989, p. 386

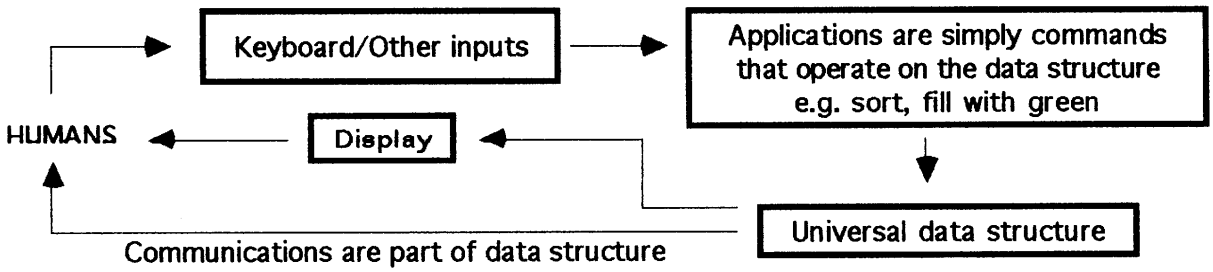
OLD:	One computer - many people	OLD:	Command languages
NOW:	One computer - one person	NOW:	Windows / icons / menus
FUTURE:	One person - many computers	FUTURE:	Direct action



User and data far apart

HARD TO USE

Old Model: Many layers, slow, clumsy, lots of code, many data types (e.g. Mac, IBM, Unix, etc.)



New Model: No O.S., fast, simple (e.g. Swyft, Cat)

EASY TO USE

Jef Raskin 1987
and copied 1989

"RaskinDoc007_002.PICT" 237 KB 2000-02-21 dpi: 400h x 400v pix: 2920h x 3728v

SYMMETRICAL AIRFOIL GENERATION PROGRAM. JEF RASKIN, 1984

While this program is nothing that experienced programmers will learn much about coding from, it demonstrates the power of being able to embed executeable code into the text produced by a word processor. What this does is to make the mechanics of documentation simple. Using a typical program editor for extended text is usually bothersome enough a chore so that our program comments become terse and more difficult to read. I have observed that when I document a program in this thorough fashion, it often runs on the second or third try, whereas when I do not document carefully, much more debugging is required.

Please note that this text you are reading is part of the program (or, equivalently, that the program is part of the text).

The program itself plots, on an Apple II, given Information Appliance's SwyftWare environment, a family of airfoils that I am developing for my aerobatic model airplanes. It does it by a transformation of a circle, much as the Joukowsky transformation does. This permits relatively easy calculation of the lifting characteristics of the airfoil, although a program to do that is not shown here.

The program begins by setting up graphics mode and a few constants. First, for graphics mode:

```
90 HGR
```

The constants include

```
100 pi = 3.1415926
```

and one that determines how much the airfoil will "cusp" or be concave toward the rear.

```
110 c = .81
```

Another important variable controls the thickness ratio of the airfoil.

```
120 f = .22
```

The program works by using the usual pair of formulae

x = r cos t y = r sin t

to generate a circle parametrically by letting t vary from 0 to 2 * pi in convenient steps. r is chosen to be .5 so that a circle of diameter 1 results.

```
130 FOR t = 0 TO 2 * pi STEP .04
140 x = .5 * cos(t)
150 y = .5 * sin(t)
```

The circle is centered on the origin, and we first move it to the right by .5 so that all the x-values are positive, and are, in fact, in the range (0,1).

```
160 x = x + .5
```

Apple IIe
SwyftCard example

To understand how this transforms a circle into an airfoil rounded at one end and pointed at the other, note that if we multiply each y-value by the corresponding x-value we will, near x=0, have very small y-values, yielding a point. But as we approach x=1 the curve approaches a circle.

```
200 y = y * x^c
```

The exponent of x changes x's effect. Many functions of x will do for this role.

Airfoils tend to be long and thin. Thus we want to squash the shape in the y-direction. The thickness ratio is thus applied to y. This could have been done in the previous line of code, but it is separated out for clarity.

```
210 y = y * f
```

We now have -f < y < f and 0 <= x <=1. This has to be translated to Apple screen coordinates where 0 <= x < 280. Again, these transformations could have been included in some of the earlier program statements, but for clarity the plotting is separated from the generation of the foils. The 250 is chosen as conveniently large for the screen. This will make x vary from 0 to 250

```
400 x = x * 250
```

and to keep things square, we apply the same scaling to y, remembering that y has already been decreased by the f factor. We also have to make y always positive. Obviously, we will go off-screen if f is too large, but we will assume smart users who read this documentation and behave accordingly rather than bother with too much error checking.

```
410 y = y * 250 + 100
```

Now to plot these points, and close the loop, and then go into an infinite loop so that the computer will not mess up the screen with messages.

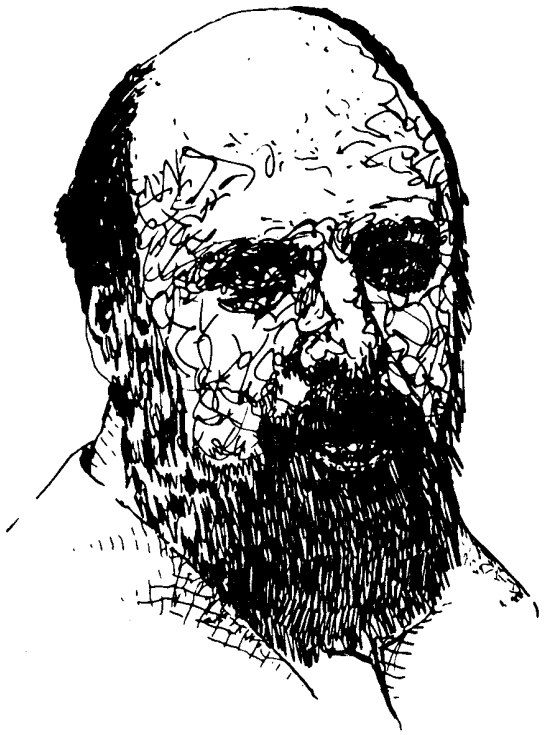
```
500 HPLOT x,y
510 NEXT t
520 GOTO 520
```

All that's left is to type and use the CALC command on "run". You will have to modify the program to run on systems other than SwyftWare-equipped Apple II's.

See p. 233+ for SwyftCard details

226

This program demonstrates how Raskin embeds executable code into text that is produced by a word processor.



Jef Raskin

APPLE COMPUTER'S

Macintosh project creator, Jef Raskin, is a man of many talents. He is also a past conductor of the San Francisco Chamber Opera Company; a holder of patents in packaging design, aircraft structures, and electronics; an artist whose work has been exhibited in the New York Museum of Modern Art and the Los Angeles County Museum; and currently chief executive officer of Information Appliance Inc. He was born in New York City in 1943—as he puts it, about the time the digital computer was born.

At the State University of New York at Stony Brook, he studied mathematics, physics, philosophy, and music, while earning a number of scholarships and a National Science Foundation grant along the way. After five years of study, he graduated with a bachelor's degree in philosophy. He received a master's degree in computer science from Pennsylvania State University, and later became a professor of Visual Art at the University of California at San Diego (UCSD). He taught at UCSD for five years, and was also director of the Third College Computer Center.

He resigned from UCSD with the gesture of flying over the Chancellor's house in a hot air balloon, which, he says, is all anyone needs to know about his reasons for leaving. He then became a professional musician, doing both teaching and conducting. When the 8080 microprocessor was introduced, Raskin founded Bannister & Crun to exploit the new technology. The company found a profitable niche by writing manuals and software for Heath, Apple, National Semiconductor, and other companies.

In 1978, he became Apple Computer's thirty-first employee and manager of publications. He later became manager of advanced systems and formed the group that created the Macintosh. After leaving Apple in 1982, he taught at the Dansk Datamatik Institute in Denmark before returning to Silicon Valley to found Information Appliance Inc.

When told the title of this book, Jef Raskin was quick to point out that he wasn't a programmer per se, but rather a designer or a "metaprogrammer," and that his company is not involved with traditional software. Jef Raskin is a person who strives to distinguish himself from the crowd. He is an inventor with original ideas for new software and innovative ways to make microcomputers perform. He is not the person who sits down at the computer and writes the source code. This distinction between a programmer and a software or systems designer is becoming more and more prevalent as software becomes more complex, and as more people are trained in the techniques of translating specifications into code.

The office of Raskin's new startup company is on University Avenue in Palo Alto, the town that seems to have become the center of activity for many defectors from Apple Computer. The offices of Information Appliance were not like the elegant, ostentatious offices of many other software companies; they were, in fact, relatively small, somewhat dingy, and just plain ordinary. They could have been the offices of a small appliance company, and I presumed this was all part of Jef Raskin's microcomputer philosophy.

Raskin, with his graying beard, slightly balding head, and alert, darting eyes, had the look of a seasoned professor. He greeted me somewhat breathlessly, explaining he had been in a bicycle accident on the way to work, which made him late. He appeared in a somewhat unsettled state. He ushered me into his small office and proceeded to straighten things up a bit. Behind his desk to one

side was an Apple IIe computer. Against the other wall was a Yamaha keyboard synthesizer. Behind me was a ten-speed bicycle and a bicycle pump, and what appeared to be a small oriental shrine.

Raskin is a very animated, excited, and excitable fellow. At certain moments, I felt I was in the midst of an amusement park; one minute Raskin would give an impressive demo of his new product as if performing magic tricks; a little later he would break into a juggling act; next, he would swing around and play a song on his synthesizer; and then he would send a little model car whizzing around on his desk, all the while discussing the past, present, and future of microcomputers.

He has a vision of the role microcomputers should play: not one of glorious, mysterious, elusive machines, but one of invisible, easy-to-use, practical appliances.

INTERVIEWER: You're best known for having created Macintosh at Apple. What role did you play?

RASKIN: In 1979, Apple was working on the Lisa. Believe it or not, it started as a character-generator machine. I was manager of advanced systems at Apple, and I was really unhappy with the Lisa. It was very expensive, and I thought Apple was foolish to tackle DEC, Data General, and IBM in the minicomputer price range.

When I was a visiting scholar at the Stanford Artificial Intelligence Laboratory in the early seventies, I spent a lot of time at Xerox's Palo Alto Research Center. I thought the work Xerox PARC was doing with bit-mapped screens, generalized keyboards, and graphics was wonderful. So I lobbied hard and got Apple to change Lisa to a bit-map machine. I helped put Xerox and Apple together. At one time Xerox owned about 10 percent of Apple stock.

What I proposed was a computer that would be easy to use, mix text and graphics, and sell for about $1,000. Steve Jobs said that it was a crazy idea, that it would never sell, and we didn't want anything like it. He tried to shoot the project down.

So I kept out of Jobs' way and went to then-chairman Mike Markkula and talked over every detail of my idea. Fortunately, both Markkula and then-president Mike Scott told Jobs to leave me alone.

I hired the original people: Bud Tribble, Brian Howard, and Burrell

Smith. We went off to a different building and built prototypes of the Macintosh and its software, and got it up and running.

Later, after he took over, Jobs came up with the story about the Mac project being a "pirate operation." We weren't trying to keep the project away from Apple, as he later said; we had very good ties with the rest of Apple. We were trying to keep the project away from Jobs' meddling. For the first two years, Jobs wanted to kill the project because he didn't understand what it was really all about.

The original Macintosh was a careful, rational design. Eventually, everyone at Apple realized it was Apple's main hope for a product to follow Apple II. Then Jobs took over. He simply came in and said, "I'm taking over Macintosh hardware; you can have software and publications." He threw out the software design, made the Macintosh software compatible with the Lisa, and insisted on using the mouse. The machine became much larger, more complicated, and much more expensive. It now runs like a tub of molasses. Have you ever used MacWrite? We call it MacWait around here. And then a few months later Jobs said, "I'm taking over software; you can have publications." So I said, "You can have publications too," and left. That was in May of 1982. He and Markkula said, "Please don't leave. Give us another month and we'll make you an offer you can't refuse." So I gave Apple a month; they made me an offer, and I refused.

INTERVIEWER: But the machine was a success. Jobs must have contributed something.

RASKIN: He did do some very important things, especially in the early days of Apple, which made the difference between Apple being just another little computer company and the great company that it became. He had good ideas: putting the Apple II in a nice one-piece case, marketing it, going out and getting really good people, and seeing that it had good manuals.

INTERVIEWER: You seem rather sensitive on the issue of getting credit for your work.

RASKIN: It's not my intention to take all the credit for Macintosh; it was a team effort. If Jobs would only take credit for what he really did for the industry, that would be more than enough. But he also insists on taking credit away from everybody else for what they did, which I think is unfortunate.

I was very much amused by the recent *Newsweek* article where he said, "I have a few good designs in me still." He never had any designs. He has not

designed a single product. Woz (Steve Wozniak) designed the Apple II. Ken Rothmuller and others designed Lisa. My team and I designed the Macintosh. Wendell Sanders designed the Apple III. What did Jobs design? Nothing.

INTERVIEWER: Would you say your experience at Apple was a bad one?

RASKIN: No. The first few years, '76 to '80, were wonderful. I had a great time and have no regrets. In the early days, Jobs and Wozniak were a joy to work with. But then from late 1980 through '82, working at Apple became a real nightmare for me.

INTERVIEWER: When you left Apple, did you think you'd go back into business again?

RASKIN: I thought I'd never do that, or anything dumb, like work in Silicon Valley again. I was tired of working seven days a week.

"I didn't want a computer. I wanted something that worked like an appliance."

After leaving Apple, I went back to teaching, in Denmark. I'd just gotten married, and was still on my honeymoon when I had this idea. I realized that what I'd been trying to do all those years was wrong. I'd been trying to design a better computer and I didn't want a computer at all. I wanted something that worked like an appliance. And my idea seemed too good not to share.

INTERVIEWER: And this was the idea that inspired you to create a new company, Information Appliance. How does the word "appliance" reflect the products you are developing here?

RASKIN: Have you ever noticed that there are no Maytag user groups? Nobody needs a mutual support group to run a washing machine. You just put the clothes in, punch the button, and they get clean. To do information processing, I don't want hardware and software; what I really want is an appliance to do my tasks. And what tasks do I do? Surveys show that 85 percent of all personal computer users use word processing, so I need a word processor, a wonderful word processor—the best in the world. But I am sort of simpleminded. I can only remember ten or fifteen commands. That's why the system I use has only five. That way I can wake up at 3:00 a.m., get out of bed, go over to the computer, and just type out an idea.

INTERVIEWER: You mean you've tried to simplify a system?

RASKIN: Right. Let's consider what it would be like if a computer company had designed a toaster. You wake up and you want a piece of toast for

breakfast. The first thing you do is switch the toaster on. If it was designed by General Electric, you'd put the toast in and off you'd go, but no, this toaster was designed by a computer company. So what happens? First of all, it does a two-minute toaster check. Then you put in the system disk and boot the system. After that you put in your breakfast disk and then you type "Load TOASTED.CODE."

So, what happens next? Up comes the menu. It asks, "What kind of bread are you going to have?" If it is a California program, it'll say croissant, bagel, English muffin, whole wheat, and at the bottom, of course, white bread.

They're labeled A, B, C, D, E, so you hit C because you feel like a muffin this morning. Nothing happens because you forgot to hit return. You'd think the machine would be smart enough to respond to C, but you have to hit return anyway.

Do you think it does anything now?

INTERVIEWER: Well . . . ?

RASKIN: Of course not. It's designed by a computer company. It says "Are you sure?" Now you're ready to throw it through the wall. Are you mad yet? Haven't you been mad at computers for years? But because you spent a couple of thousand dollars on it, you put up with all this stupidity, and so does the rest of the world. Millions of people go through nonsense like this every time they use a computer.

So you type "Yes" and hit return, but get an error message because you were supposed to hit something else. You consult the manual, but it tells you nothing because the people who wrote it were describing a prototype system that has changed. Finally, you put the bread in slot two, indicate whether you want light, tan, or seriously burnt bread and the computer asks, "Do you want to save this breakfast so you don't have to redo this again?" So you type "Yes," and it tells you to place a disk in slot one, but you discover that you're out of formatted disks.

You call up the dealer and ask if there's any way to save this while formatting the disk without losing everything you've already done this morning? He says, "Yeah, just buy this $3,000 hard-disk system with MS-DOS 9.8 and that will solve all your problems. It comes with a manual and a handtruck." The handtruck is for moving the manual. And you're already late for work . . . . But that's the way things are. Our product says a lot about this kind of dilemma. Let's move over to my computer.

[We seat ourselves before an Apple IIe equipped with a SwyftCard and keyboard labels.]

Watch this. There is no disk in the drive, and I want to type a message, "Remember to bring home some milk." How do you like that? I turn it on and start typing. No need for commands, no insert, no getting to the editor, I can just start typing.

Now I want to print the message and put it in my pocket, so I can use it later. I press a single key, and it prints. Isn't that convenient?

INTERVIEWER: Can you make toast on this?

RASKIN: No, the crumbs get into the disk drive.

INTERVIEWER: What else can users do on this appliance?

RASKIN: We can do calculations easily. Before, whenever I was using the word processor and wanted to do a calculation, I'd get out my pocket calculator and have to use a separate calculating program, or get up SideKick; on the Mac, you call up the calculator and paste it into your document. We also have telecommunications capability.

INTERVIEWER: All in the same program?

RASKIN: Sure. There is no difference between all the applications. What's a word processor? You use it to generate text, move it around, change it if you make a mistake, and find things. What's a telecommunications package? You use it to generate text, or receive text generated by someone else. Instead of it coming in from a keyboard or out from a printer, it comes in or out over a telephone line. And what's a calculator? You use it to generate numbers, which are just text, and the answer should come back into your text. So, one day it dawned on me, if these applications do the same thing, why not have one little program that does them all?

INTERVIEWER: Well, what is this product you've developed to cover all of these features?

[Raskin holds up a simple card.]

RASKIN: It's called a SwyftCard. It's a board for the Apple IIe. Not much to it, eh? That's why we can sell it for $89.95. It's so friendly, so simple, that we've actually had people tell us that they sold their IBMs or Macintoshes and bought an Apple IIe with a SwyftCard after seeing this.

INTERVIEWER: Does this only work on the Apple IIe?

RASKIN: The SwyftCard works only on the Apple IIe now. We will soon have identical software in a product called SwyftDisk for the Apple IIc. The

See BYTE magazine (Oct 1985 p. 357) for SwyftCard ad.

SwyftDisk will probably be available in early March.

INTERVIEWER: Do you have plans to implement it on other computers?

RASKIN: One of the things about our company is that we never say anything until we have something already done. The first time you read about this product, you could order it. But let me show you some other advantages to this system .... [Raskin returns to the computer screen.] If you forget to save some text and you load in another file, normally you'd lose that text. This system is smart enough so that it never loses anything. But I'll do something even more outrageous. [Raskin gets up from his computer and places a disk on a giant magnet hanging on the wall.] This is now a totally unformatted blank disk. I'll apply this very powerful magnet to it to prove that it is blank. I'll insert it into the disk drive and type some text and use the DISK command [the disk whirs for four seconds]. Now I'll do something even worse; I'll switch the machine off. Still, I guarantee that when I switch the machine back on, the text will have been saved. [He turns the machine back on; in four seconds the text appears on screen.] See? The system saved the text on a totally unformatted blank disk in just four seconds. DOS can even format a disk in less than half a minute. Not only that, but when we turned it on, even the cursor was exactly where we left it.

"Have you ever noticed that there are no Maytag user groups? Nobody needs a mutual support group to run a washing machine."

INTERVIEWER: Okay, it's smart, but is it fast?

RASKIN: It will keep up with typing much faster than any human can type, and it automatically word wraps, formats, and paginates as you type. The cursor-moving technique is faster than the mouse. Not only is it inherently fast, but you will use it faster because you don't get tripped up by it.

INTERVIEWER: How did you get the speed?

RASKIN: Very, very clean design. This system neither drinks nor smokes. It exemplifies the kinds of things I'm working on these days.

INTERVIEWER: Most users are dazzled by the mouse; it has market acceptance. Why do you dislike it?

RASKIN: I hate mice. The mouse involves you in arm motions that slow you down. I didn't want it on the Macintosh, but Jobs insisted. In those days, what he said went, good idea or not.

INTERVIEWER: Your whole approach seems counter to the industry trend to make bigger computers to accommodate bigger programs...

RASKIN: Yes. Instead of saying bigger, bigger, we're saying better, better. When I told our investors about this project, I said, "We're going to have a word processor, information retrieval, and telecommunications package with only fifteen commands and 64 bytes of code." All the other companies were talking hundreds of commands and hundreds of bytes of code. The company (Information Appliance) was surprised that it came down to five commands. It's the only project I've ever been on that got simpler with time instead of bigger or more complicated.

We have a whole valley full of people talking UNIX versus MS-DOS. What do you need *any* of that for? Just throw it all out; get rid of all that nonsense. Maybe you need it for computer scientists, but for people who want to get something done, no. Do you need an operating system? No. We threw out that whole concept. Applications like VisiOn, Gem, and Windows are just cosmetic treatments on hidden operating systems, but we have no operating system beneath this. You know what happens when you apply heavy cosmetics to something? You get that heavy cosmetic look.

So here is a program that runs on an ancient Apple IIe, a one-megahertz processor, and from the user's point of view it runs faster than IBM, Macintosh, mainframes, SuperVax, or anything.

INTERVIEWER: Was your motivation on this project different from what motivated you to design the Macintosh?

RASKIN: Some of the motivation was the same, some of it different. With the Mac, I was trying to make the best computer I knew how to make. When I created Information Appliance, I was no longer trying to make computers. I just wanted to make the benefits of computer technology easily available to everyone.

That idea goes back to when I was an interdisciplinary professor at the University of California at San Diego in the sixties, teaching in the visual arts department. I was an artist and a musician as much as a computer scientist, and I loved teaching computer programming to people in the arts and humanities. I also taught computer programming, computer art, and computer movie making to people who normally didn't work with those fields: a class of Mother Superiors, or a class of third or fourth graders.

INTERVIEWER: Prior to Apple, you were very involved in academia. What

exactly was your background in those pre-Apple days?

RASKIN: Well, my sixth grade project was a digital computer that I built with relays and big switches. I studied mathematics, philosophy, music, and physics at SUNY (State University of New York), Stony Brook. At Penn State I began a Ph.D. in philosophy, but got a master's in computer science instead. At U.C. San Diego I began a Ph.D. in music, but became an art professor instead. I was a professor for five years, and then I became a visiting scholar at Stanford at their Artificial Intelligence Lab.

Next, I was a conductor of the San Francisco Chamber Opera Company, and taught music in San Francisco. Then along came the personal computer, and I thought, "Hey, computers interest me again." I bought one of the first Altair kits and built that. I thought the manuals were dreadful, so I formed a documentation company called Bannister & Crun, named after two characters out of an old English radio program called the Goon Show. I wrote manuals for Heath, National Semiconductor, and Apple. I started a model airplane company, too.

INTERVIEWER: Do you feel there is any parallel between music, the arts, and the computer?

RASKIN: Not a parallel. But I'm definitely trying to do things that will make people happy. One of the reasons I liked being a musician is that musicians have rarely done anything bad in the world. If you're a physicist, you might invent something that explodes. But musicians never do. Artists can create propaganda posters but with musicians, it's usually pretty neutral stuff. I have always been interested in doing something that would make lots of people happy.

People who use the SwyftCard won't lose files, will spend a lot less time working, and be less frustrated. They're going to be happier people than they would have been using something else. I'm trying to do genuine good in this world. Another fact: This system is usable by blind people because of the way the cursor is moved. Visually oriented systems like the Macintosh are totally unusable by blind people.

We currently have our systems at the Veteran's Administration, Western Rehabilitation Center, and the Sensory Aids Foundation, and they're reporting back to us, saying, "Hey, this works with blind people." So here I'm doing something that will enfranchise a large block of people.

INTERVIEWER: Do you still find time to pursue the arts?

RASKIN: I don't do visual arts, but I'm still a musician, though I don't perform very much any more. I don't have enough time to practice now that I'm running the company. Occasionally, I'm asked to play at a friend's wedding. That's how I worked my way through graduate school: playing in bars and nightclubs. Every Wednesday night at Penn State, I'd play the piano for the old-time movies. At home I have a nine-foot concert grand. I like to play Bach and Mozart.

INTERVIEWER: Such a heavy academic background must have made you a hot property at Apple . . . .

RASKIN: During Apple's first few years, I was the only person with a degree in computer science in the whole company. And I didn't tell people, because if they had known I had actually been a professor and a computer center director, they might not have let me in the company.

INTERVIEWER: Why?

RASKIN: Because there was an anti-academic bias in the early Apple days. I was able to come in and write good manuals because I said to Woz, "You're an expert in hardware," or to Randy, "You're an expert in software." Then I'd tell them, "Well, I don't know much of that stuff, I just know how to write, so I don't bother you, you don't bother me."

INTERVIEWER: How long did the SwyftCard take to develop?

RASKIN: The company's been here for about three years, but we are doing other things too, so it's hard to say how long. We didn't get into business to produce a board for the Apple IIe,* but it seemed like such a good idea that I would have felt very bad not to have released the product. I saw a lot of good products at Apple and Xerox pass from desktop to desktop, and never get to the market.

INTERVIEWER: Was it difficult to find the backing to start Information Appliance, or did the strength of a good idea and your fame from the Macintosh carry you through?

RASKIN: After I laid out the details of what the SwyftCard was going to do, I swore some friends from Apple and Xerox PARC to secrecy and showed them the specs. They took one look at it and said, "No way; it won't work; we've been trying to do that for decades." Then they all called me up next day and said, "Hey, you know, it looks like it'll work." So I decided to hire a few programmers, just with my own money, to see if I could implement my idea. I decided to implement it in FORTH, because FORTH is a rather compact

Randy = R. Wiggington (Apple II AppleSoft BASIC, Macintosh MacWrite)

* Doug McKenna (1993), financial backer of IAI, said other backers forced JR to make IIe card.

language and is inexpensive to implement. It's not my favorite language, but I thought it was suitable for this particular application. I always believe you should use the right tool for the job.

I went out and hired a FORTH programmer and a few other people, mostly personal friends of mine. Nobody from Apple. I didn't touch the company. I didn't want to get into any legal hassles, and Apple was nasty enough then that I worried about such things.

INTERVIEWER: Did you embark on a heavy marketing campaign to promote your product?

RASKIN: We didn't have much budget for marketing. We just took a little bit of money and ran a few ads to get the message out to those people who wanted it. The reaction we've gotten is that this is a great product, and sales are climbing much faster than we anticipated.

INTERVIEWER: Are you wary of aggressive marketing and hype? Do you think it has caused problems in the industry?

RASKIN: It sure has. I saw a full-page color ad in a magazine for an LCD screen for an Apple IIc and so I called the company. They told me, "We expect to have them ready in ten months." A full-page color ad! I think they're wasting their money.

INTERVIEWER: As a professor on campus, a reporter for magazines, a writer of manuals, and a designer of computers, you have carried a message—namely that computer benefits are accessible to the masses. Is Information Appliance simply another vehicle for your message?

RASKIN: If this company is at all successful, I'll again be reaching millions of people. What's going to get the message out is the product. If the company makes a lot of money, everybody will listen to my message. So, the way to carry a message in this world is, unfortunately, to make money.

If I wanted to tell the world that bit-mapped screens are great, no matter how many articles I wrote, nothing would happen. Xerox wrote dozens of articles, and who was listening? But get Macintosh out there, at a reasonable price, sell a few tens of thousands, and guess what? Everybody who buys one or reads about it discovers the whole idea of bit-mapped screens, graphics, and the fact that you don't need separate graphics and text mode, that letters are just another example of graphics, and that you can do fancy fonts without additional hardware.

It's absolutely ugly, but unfortunately quite true of the world today; the

more money you make, the more people tend to listen to you. If you're not quoted in *Fortune* or *Forbes* or the *Wall Street Journal*, then nobody listens. If you say something that makes a lot of money, whether what you said is true or not, people listen.

INTERVIEWER: Speaking as one who has seen both ends of the corporate spectrum, what do you think of the big company versus the small company?

RASKIN: You should have heard what Jobs thought about what he was going to make happen when Apple grew large. He forgot his ideals so fast. Woz kept his a heck of a lot better. There's a person at Apple who is one of the few people who made a bit of money and has not changed at all, and that's Brian Howard. He's a wonderful person. I don't know how much he's worth, but he still drives the same car I sold him for $75 ten years ago.

"Are you mad yet? Haven't you been mad at computers for years?"

INTERVIEWER: What do you think is the biggest problem your business faces?

RASKIN: How in the world do you sell something that's different? That's the biggest problem. The world's not quite ready to believe. It's like in the early days at Apple, they said, "What's it good for?" We couldn't give a really good answer so they assumed the machine wasn't going to sell. But I do know the way I plan to sell my product is by word of mouth. Some people will try it and say, "This product really gets my job done. It doesn't have fifteen fonts. I can't print it out in old gothic banners five feet long, but I sure got that article finished under the deadline." That's how I can sell it. Later, people will understand it.

This is like when I first started Macintosh, nobody, not even Jobs, knew what icons or bit-mapped screens were. Now they are in everybody's vocabulary in this industry. I've done it before and I'll do it again.

INTERVIEWER: What do you think about the current lull in the computer industry?

RASKIN: I think that's why we're going to make a lot of money. The reason for the downturn, in my opinion, is that everybody who's going to buy one of these really hard-to-use machines has done so, and they are only selling to people now who are pretty much forced into it. Forced into it by the fact that they have to work with it, or because someone else tells them they have to. Or because the Joneses down the street have one. Right now, there's a lot of room

for a product that is genuinely easy and fun. I think the market is saturated with complex machines and I think this company's going to open up a whole new marketplace.

After you've seen our system and go back and use your system, every time you use your computer you're going to say: "If I were working with an Information Appliance, I'd be done by now. I wouldn't be sitting here waiting, twiddling my thumbs, I wouldn't have lost that idea or that nice sentence I'd thought of." I'm trying to sell a system that makes people happier. You can get more work done and you'll spend less time being bothered if you use one of our products.

INTERVIEWER: Do you think the computer will eventually do other things besides word processing, spreadsheets, and things like that? What about all those promises people made about running your home?

RASKIN: I've never believed those very much, and I'll tell you why. First of all, there will be special-purpose systems which have microcomputers and do various tasks around the house. There already are, but you don't think about them as computers: they're appliances. When you use a fancy microwave oven, do you think about the fact that it actually has a microcomputer inside with a little bit of RAM and ROM, and it runs programs? Does anyone ever think about any of that? All you do is punch the time, put in the food, and get it warm for breakfast. That's a hidden microprocessor. We have lots of hidden ones. But are computers going to do that? No way. First of all, have you ever seen any two computer products that are really compatible? Do you think that the XYZ Company that's making automatic electric windows is going to make their product compatible with Company G's computers? You know what a mess it is trying to run other manufacturers' printers with your computer. It doesn't work; it won't happen.

INTERVIEWER: Besides your work in computers, do you have other creative projects?

RASKIN: All kinds. I'm a typical inventor. I've got projects coming out of my ears in all directions. For example, I have a company that makes radio-controlled model airplanes. I'm also working on a new design for a piano. It will probably be as successful as other piano innovations of the last hundred years: namely, a total failure. My new piano sounds lovely, but I don't know if it is ever going to sell.

I'm working on a numerically controlled milling machine that will sell

for under $5,000, so small shops can have them. Right now the only ones available cost $30,000 to $100,000. I've got one of mine working at home now.

INTERVIEWER: In looking through the manual for your product I noticed two things: First, it's very thick for such a simplicity-oriented product, and second, it's very comprehensive and easy to read. Does this book reflect any of your feelings on computer literature?

RASKIN: The whole attitude of this manual is different. The attitude is that the people who are going to use this manual are real human beings, just like me and you. We don't want to reference a hundred manuals to find an answer, and we also like to read English. So when I started this project, I set out to write the simplest, neatest, nicest manual possible.

Before we published this manual, we went to the trouble of writing and typesetting a first-draft manual, which we knew was not going out to the world, yet looked really good, so people didn't feel they were getting a preliminary manual. Why? To have people try it and learn from it. We wanted them to feel they were getting a real, finished system. We did not write the final manual until we had the actual product. When you run your own company, you have a chance to try to do things right. I've never been any place where they would let me write a manual first, throw it out, and then rewrite it. We also did all those silly little things that too few companies in the industry do, like giving credit to everybody who worked on the project.

There's a reason for the manual's size. Our idea is that if you have this manual, and also have other equipment, you won't have to look at the other manuals to get the answers. Our manual has most of the answers, not only for our equipment, but for almost everything else you're likely to hook up to. We think that's different from most manuals. If you want to know why the product is so simple, you can read the schematic, and the "Hardware Theory of Operations." No secrets. If you want to know why it's so fast, you can read the "Software Theory of Operations." There's also a "User Interface Theory of Operations." I've never seen that anywhere else. The manual has a long glossary, which can help teach you about the industry. Every term we use is defined. It's not just one or two pages, it's a real glossary. And there is a cross-referenced index that goes on for pages and pages.

INTERVIEWER: Do I detect a contempt for user-interface systems and manuals that don't use simple English commands and text?

RASKIN: I usually can't figure out what icons mean, but I can figure out

SwyftCard manual

David T. Craig has a copy of the pre-release SC manual (see Raskin doc #54)

what English words mean. It's easier for me to translate English into Spanish and French than it is to make up icons that can be understood by everyone everywhere in the world.

I've been working in this direction for a long time. It wasn't until I got away from Silicon Valley and let my head clear for six months that I suddenly saw I'd been simply barking up the wrong tree.

INTERVIEWER: You mean you had been swept into a whirlpool of complexity there?

"I learned a lot about the world in a Rolls Royce. After I left Apple, I sold the Rolls."

RASKIN: Icons, windows, mice, big operating systems, huge programs, integrated packages.... I would like to remind the world that just because two things are on the same menu doesn't mean they taste good together.

INTERVIEWER: Is that your message to Silicon Valley? To show the creators of complex systems that there is a simple alternative?

RASKIN: Yes.

INTERVIEWER: Would you say that this manual is the high point of your career as a technical writer?

RASKIN: I entered the computer industry as a freelance reporter for *Doctor Dobbs*, *Byte* magazine, and a now defunct magazine called the *Silicon Valley Gazette*.

I became manager of publications at Apple and wrote some manuals that were highly rated, but this manual is beyond anything I'd even come close to writing. But I didn't write it myself. The lead author is David Alzofon; I wrote less than half of it.

SC manual writer

The whole system is based on a few rules, a few psychological principles that make an easy-to-use system. If you read the "Theory of Operations" sections in the manual, you find out that the company has a theory of how human beings work, and this enabled us to design our product.

INTERVIEWER: What is the theory?

RASKIN: It's relatively simple. I asked myself the question: "What makes people form habits?" When I'm using a system, I'm happiest if I can keep my mind on what I'm trying to do, rather than on the system itself. The system should not intrude. When I'm tying my shoes, according to habit, I don't really think about it. I'm not putting my attention on tying my shoe. I wanted my system to be so simple that it would not distract users from their main

task; I wanted using my system to be natural, like a habit. So I asked, "What are those things that make for habit?"

Well, the psychology books I read usually say a habit is something that you do the same way repeatedly, and that you're likely to do the same way again. There are some immediate implications from this. Picture yourself driving your car. Let's say that every Thursday the gas pedal and the brake pedal are interchanged. It would drive you up the wall, or through it. You couldn't live with it. Yet our computers change things around all the time by using "modes." A system should be "modeless." The same user action should always have the same effect.

SwyftWare is essentially modeless. Most computer designers, for some reason, delight in providing many ways of doing something. If there are fifteen ways to do something, they think it gives you freedom. The fact is that most users don't use the majority of the commands on their word processors. There are a few that everyone always uses. And even though they've read the manual and know it might be a little more efficient to use a special technique, they don't bother. They use the same ones every time.

On Macintosh, there is a backup of keyboard commands for all mouse commands because nobody's going to use the stupid mouse all the time. Okay, but whenever there's more than one way, there's that moment of hesitation while you think about the manual. Even after you've developed the habit, sometimes you're doing some work and think, "Gee, isn't there a faster way?"

So not only should one action do only one thing, but there should be only one path to a particular goal. That way you always use it, and you never have to think about it. We call that "monotony." My system is not perfectly monotonous, but it is as close as I know how to make it. [Laughs] Maybe some day I'll learn how to make a system totally monotonous.

INTERVIEWER: What do you feel artificial-intelligence programs can contribute to society?

RASKIN: Artificial intelligence teaches us a lot about ourselves and about knowledge. Any reasonable artificial-intelligence program will not fit on a very inexpensive machine, at least not these days.

Real artificial intelligence is something like religion. People used to say that just above the sky were heaven and angels. Then you get a rocket ship out there, and now you know that's not true. So they change their tune. As soon as you accomplish something, it is no longer artificial intelligence.

At one point, it was thought that chess-playing programs encompassed artificial intelligence. When I was a graduate student, you could get a Ph.D. in artificial intelligence by learning to program chess. Now you can buy a chess player for $29.95 and nobody calls it artificial intelligence. It's just a little algorithm that plays chess.

First, there's a problem of definition. Then it gets more complicated. People say that programs should understand natural language, but our utterances are too inexact for a computer, or anybody, to figure out what is meant to be done; that's why we have programming languages. If anyone's ever worked from a spec prepared in English, they know that you can't write a program from it because it's not exact. So if human beings can't do it, there is almost no way we can expect to make a machine do that kind of thing. When you're dealing with so-called artificial-intelligence programs, the computers have got to learn a vocabulary. Let's say you have five commands and you want the machine to understand any possible English equivalent to them. But it won't understand any English equivalent: One person might say, "Get employee number," while an Englishman might say, "Would you be so kind as to locate the numerical designation for our employee." That's exactly the complaint AI people are trying to solve.

A lot of the promise of artificial intelligence is misunderstood. What artificial intelligence has already taught us about the nature of languages is wonderful. So, do I think artificial intelligence is worthwhile? Absolutely. Do I think it's going to turn out great products? A few. Do I think it's going to fulfill the promise that you read about in the popular press? Not at all. Will I be putting a lot of money into artificial intelligence? Nope.

INTERVIEWER: *Before, you equated power and prestige with money. Having tasted success, what changes have been forced on you?*

RASKIN: Let me tell you a funny story. I used to have an old orange truck that I bought in 1970. An International Harvester. It had been across the country a couple of times, held everything, and you could sleep in the back. Wonderful truck. And here I was an executive at Apple, taking people out to lunch in this old orange truck. But I was too busy to keep up with the latest things in cars. I hadn't been in a "toy store" for months. Eventually, little hints began to be dropped. "Why don't you get a good car?" Well, okay, I'm a responsible employee.

Everybody was driving around in these Porsche 928s or Mercedes

Benzes and I just never wanted a luxury car of any kind. But they really wanted me to have something that was respectable. So my brother and I put our heads together and he came up with the idea. He said, "You know for about the same price as a new Mercedes or Porsche, you could buy a used Rolls Royce in absolutely magnificent condition. That way nobody could ever complain that you don't have an appropriate car, and you wouldn't have done what they expect you to do." So I got a used Rolls Royce. It's funny. Some people in the company had never spoken to me before, but they were very happy to talk to me then.

Anyway, I discovered a lot about this little democracy we have. When I went into a gas station, all of a sudden they carefully wiped off the window and called me "sir." And I could park out in front of a restaurant in a no-parking area and they would say, "sir" and "thank you," and "We're glad you parked your car right in front of our marquee where everybody can see that a Rolls Royce customer comes to eat here." It was always fun to go to the drive-through at MacDonald's, too.

I learned a lot about the world in a Rolls Royce. I went to the airport and five guys came over and opened up the door and said, "Welcome to San Francisco International Airport, sir." That never happened when I drove anything else there. They expected a huge tip. It was also a lot of fun driving in downtown San Jose among the low riders. They respect heavy iron as much as I love their cars.

I could go right up to big plants like Kaiser, where they have all those guards, and just drive on through. I always wanted to take a tour inside but could never get permission. In the Rolls, I just drove up and said, "I have an appointment with Mr. Mumford." "Thank you very much. Drive on through."

After I left Apple, I sold the Rolls.

FINIS

Programmers Debate Languages, "Corporate" Programming

Some of the micro industry's best-known programmers gathered recently in San Jose, CA, to talk about how they do what they do. Although the panel—consisting of Bob Carr, Andy Hertzfeld, Scott Kim, Charles Simonyi, Jaron Lanier, John Page, Jef Raskin, Peter Roizen, and John Warnock—did not discuss projects they are currently working on, they did discuss the future of programming languages and how programmers can survive in a corporate environment.

Hertzfeld (developer of the Macintosh operating system) and Carr (chief scientist for Framework-maker Ashton-Tate) said they prefer programming in assembly language. Roizen (developer of T/Maker) and Lanier (former Atari video game programmer) work with Pascal. But Simonyi (from the Microsoft group that has produced Multiplan, Word, and Excel) generally spoke for the group when he said that C is probably the only general-purpose language for most software development. "At Microsoft, we program only in C," he said. "C now and C forever."

Raskin, one of the creators of the Macintosh, pointed out, however, that certain programming tasks often require certain languages. This view was seconded by Carr, who said that the software marketplace will probably become fragmented, making it more exciting than the "homogeneous software environment we've seen over the past couple of years." That environment, he said, will have specialized vertical programs that may be written in specialized languages, perhaps even macro languages that are part of applications programs themselves.

The programmers debated the virtues of single-person programming versus "corporate" programming (in which a team of programmers produces a product). Hertzfeld bluntly stated, and Roizen and Kim (author of the book *Inversions*) concurred, that the best programs are written by a single person. Page (of Software Publishing and PFS fame) disagreed, however, saying it just isn't realistic for a large company to expect an individual to produce enough code to get a big product out in a timely fashion. Carr agreed, saying that the hundreds of worker-years it takes to build a program such as dBASE or Framework makes it impossible for a company to afford the luxury of "one programmer, one program." Warnock said Adobe has found that a three-person programming team is the most effective. "Two people isn't enough and four is too many. That third person in the group adds balance," the developer of PostScript said.

The panelists also disagreed about the increasing tendency among programmers to use third-party subroutine libraries. Both Carr and Page said they feel that off-the-shelf routines can save programmers both time and money. Hertzfeld countered by asking, "How can you care about your program if you use someone else's code?" He continued, "I consider myself an artist. If I were another kind of artist, a writer for instance, would it be right for me to go out and buy a paragraph here, a chapter there, and include them in my book? Would it still be mine?" Kim added that it usually takes him longer to read and understand another person's code than it does to write it himself in the first place. All the programmer panelists, however, said that over the years they had put together their own subroutine libraries, which they regularly use.

BYTE — JUNE 1987 — P. 38