

2

## My PC

Interviews with Bill Atkinson, Paul Bradley, Bill Verplank,  
and Cordell Ratzlaff



Bill went home; in one night he developed the entire pull-down menu system! Everything! He hadn't just moved it to the top of the screen; he had the idea that as you scanned your mouse across the top, each menu would pop down, and they would ruffle as you went back and forth, and appear so that you could scan them all. . . . He'd thought up the whole thing in one night! I can't imagine what happened that night.

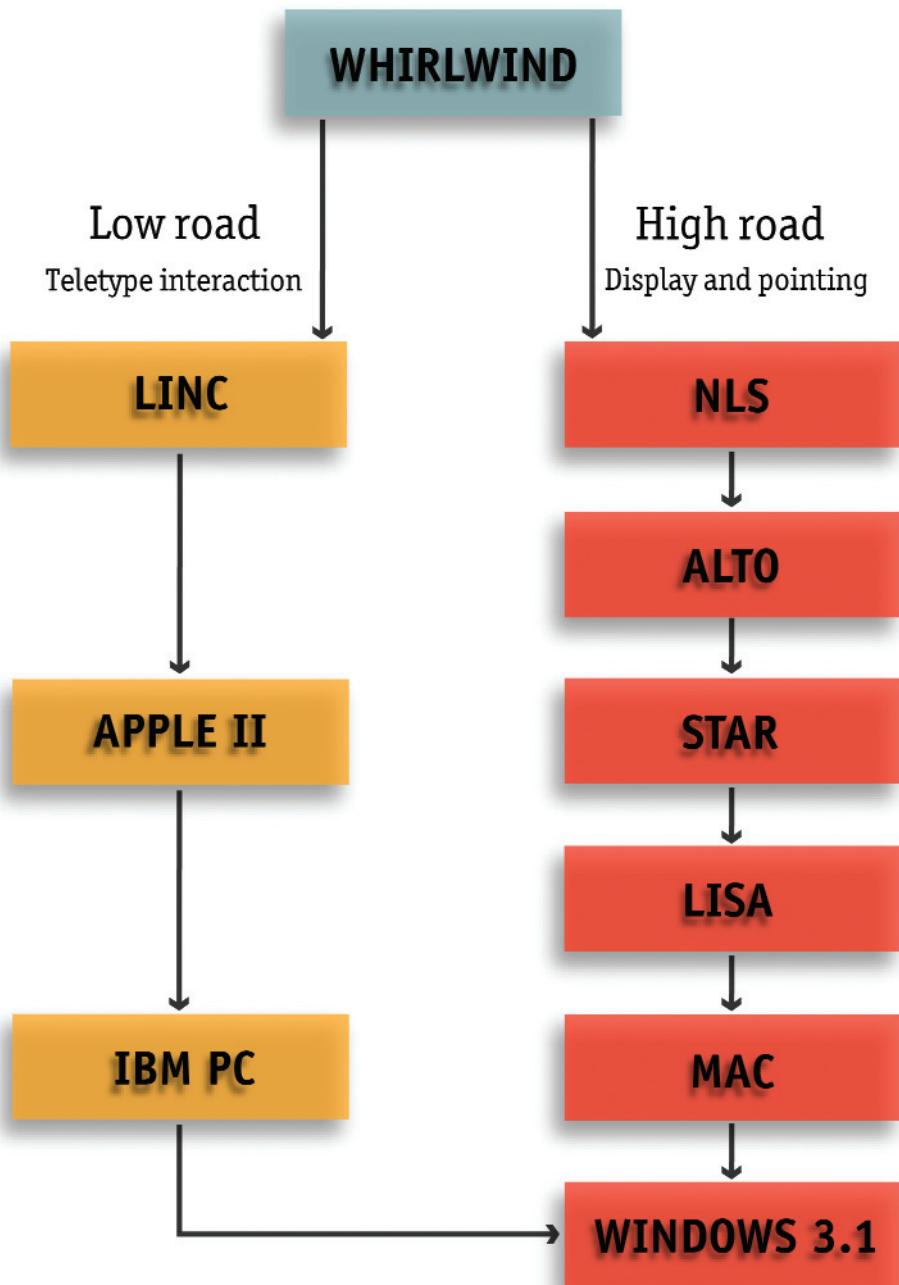
Larry Tesler, interviewed in 2003, talking about Bill Atkinson in 1982

- Apple Mac and Lisa, from 1984 brochures  
*Photos Courtesy of Apple*

THE DESKTOP AND mouse are the dominant designs for the personal computer. In chapter 1, we looked at the invention of the original concepts at Xerox PARC and the roots of the design thinking that defined the interactions. This chapter follows the development of the designs for the first versions that were affordable enough to be personal, with Apple Lisa and Mac, and then examines two significant steps forward with the first Microsoft mouse, and the desktop design for Mac OS X.

We look first at the parallel paths that led to personal computers. The low-cost road built on the typewriter as an interface to the computer, leading to the IBM PC, and remained dominant until the Macintosh arrived. The high-cost road started with the superior interface of the desktop and mouse but remained too expensive for commercial success until the cost of the components fell far enough to allow the breakthrough from Apple.

Bill Atkinson was the architect and designer of most of the precedent-setting software that made Apple so successful; he talks about his partnership with Larry Tesler, as together they invented many of the concepts that define the desktop. As a condition of



the Xerox investment in Apple, Steve Jobs negotiated the rights to use the technology in the Xerox mouse, but he found that the design was still very expensive and not yet reliable. The story of the development of the first affordable mice from Apple is told—and the controversy over the number of buttons that would make the interactions easiest to learn and use.

In 1987 Microsoft decided to try to set a new standard for the design of mice; Paul Bradley, the industrial designer of the first Microsoft mouse, describes how they achieved some significant advances. Bill Verplank worked with Paul on the human factors for the mouse; before that he had contributed to the interaction design for the Xerox Star and has fostered connections between screen design, human factors, and computer science. He explains his point of view about designing interactions with a combination of words and drawings.

The design of the desktop seemed to be going nowhere in the eighties, and it took the return of Steve Jobs to Apple to shake out a new approach. Cordell Ratzlaff was responsible for the design of the versions of the Macintosh operating system (OS), from Mac OS 8 to Mac OS X, and he tells the story of the first major change in a decade, with the design of Mac OS X.

## The Low Road or the High Road?

INTERACTIVE COMPUTING STARTED with Whirlwind, the supercomputer of its time, which was conceived in 1945 for antiaircraft tracking and by 1951 was working well enough to use. Whirlwind had two kinds of interactions, a Teletype interaction, and an interaction through a display and a pointing device. This led to two parallel paths; the first was to cost-reduce the Teletype interaction, a road that was initially attractive because it led to cheaper and more accessible machines. The second path pushed the limit of designing the human-computer interaction, developing the display and pointing device until the value of using a desktop and a mouse was proven, and it became the

dominant design. The low road and the high road stayed parallel for a while, with the IBM PC winning while price was the most important issue. As soon as the costs came down enough, the graphical user interface won out. By the time Windows 3.1 arrived, the victory of the high road was apparent.

The low road built on the Teletype interaction, following the path of least resistance. It was much less demanding for the technologists, as they could build directly on the history of the powerful computing machines that were already normal in the industry, from ENIAC to IBM equipment, and cost-reduce the hardware to offer lower prices and better performance to their customers. That led to time-sharing through a series of machines, and to the LINC computer in 1962, the first attempt to build a personal machine that was not time-shared. That eventually led to the Apple II (1977) and the IBM PC (1981). The Teletype interaction evolved to become a “Glass Teletype,” with alphanumeric characters on a monochrome screen. The human operator was thought of as a component in the system, and trained to do the bidding of the machine. This was the low-cost road, and the commercially dominant one during the eighties.

The Apple II was one of a flurry of hobbyist machines that were inexpensive enough to make computing accessible for the enthusiastic people who wanted to write their own programs and plug together the hardware themselves. The key to it being a personal machine was the price, making it available to people who did not think about it as a tool for work, paid for by the company or the university, but rather as a plaything for evening and weekend tinkering.

The IBM PC was a renegade product within IBM, put together at a skunkworks. The founders set up a separate shop at the IBM plant in Boca Raton, Florida, far away from the corporate procedures and politics. In order to get the new product out really fast, they bought most of their components from outside IBM, including the operating system. CP/M was their first choice, but the president was out hang gliding on the day that they visited the West Coast to talk to him, so they went to see Bill Gates instead, who didn't actually have an operating

system but knew where he could get one. He picked one up and then licensed it to IBM. The PC had neither a desktop nor a mouse, but it did have the right price to appeal to legions of buyers in the purchasing departments of big companies. It also had sound basic ergonomics, with a keyboard that was profiled to match the format of the Selectric Typewriter, and good tactile feedback. The screen had clear fonts that showed up strongly against the dark background. There was no proportional spacing, just a character-based interface with a blinking cursor—nothing fancy, but it would do the rudimentary jobs once you had learned how.

The path to the high road came from the mouse. The most obvious link between Whirlwind and the future graphical user interface (GUI) was a pointing device. Once you had one, you could use it for choosing commands from menus. This allowed you to move from recall to recognition, so instead of having to remember all of the commands on the computer in a particular instance, all you had to do was recognize the one you wanted.

The high-cost road combined the display and a pointing device, leading to the desktop and the mouse. At first this seemed very difficult to achieve and impossibly expensive, unless you could justify it with something as important as the defense of a nation. “Personal computer” had the same sound then that “personal nuclear reactor” would have now, because computers were things that filled rooms. To most people, the idea of being able have a computer like this for yourself seemed ridiculous, but then there was Moore’s law<sup>1</sup> to consider. Moore’s law said that the stuff you could get on a chip, and hence the power of the computer, was going to double every eighteen months. If you asked, “How many years before I can afford to have one myself?,” the law gave you an astonishing answer of ten years or so. That meant that you could build machines that were very expensive at the time with the capabilities that you wanted, and then watch the costs follow the Moore’s law slope right down. The personal computer arrived via the low road, but the value of the desktop and the mouse did not take long to dominate, as they offered better interactivity.

## Apple Lisa and Mac

AT XEROX HEADQUARTERS there was growing concern. A lot of money had been invested in Xerox PARC, and suddenly there were inexpensive computers on the market, including the Altair, Commodore PET, and Apple II. The arrival of VisiCalc and WordStar in 1979 gave people who wanted to try a spreadsheet or a word processor a reason to buy an Apple II. The Xerox management was starting to get a little nervous, saying: “They’re calling these things personal computers, but I thought that’s what we were doing. These things are \$800 and ours are \$10,000, and we’re worried.”

They put together a task force and sent some people out to PARC, but the prevailing opinion there was that the new machines were only toys. Xerox knew that they could never manufacture Altos cheaply, because of their union contracts and the way they’d built their machines, so they decided to invest in Apple. The idea was to start with an investment and perhaps buy Apple after a while, thus learning about cheaper manufacture methods, and possibly have Apple do the manufacturing of a commercial version of the Alto. It was a \$1 million investment.

Apple was heading toward a hot initial public offering (IPO), with everyone wanting to invest, so as a condition of the investment, Steve Jobs was able to negotiate access to some of the PARC technology, including the mouse. A lot of the people at PARC were very upset about giving rights to Apple, worrying about the cumbersome speed with which Xerox moved, but several demos were arranged, which are well preserved in books<sup>2</sup> and legend. The significance of the transfer of ideas from Xerox PARC to Apple has been greatly exaggerated, as Apple created new and different products starting largely from scratch, but several people left PARC to go to Apple, and took their experience and beliefs with them.

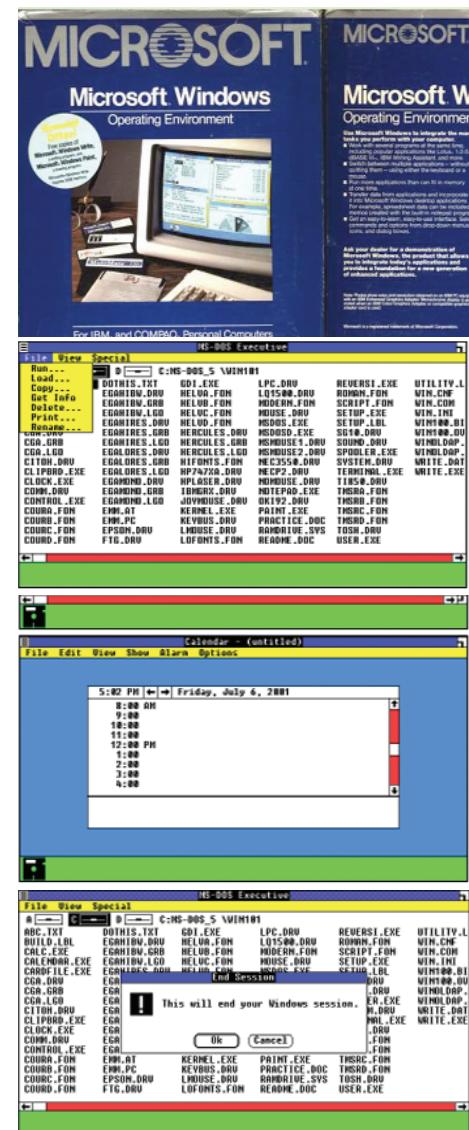
Both Lisa and Mac were very different from Star—full of new interactive features and dramatically less expensive to manufacture. Lisa was a wonderful accomplishment of interaction

design, crafted over a period of four years by a team of twenty people in applications software engineering and user interface design. Like the Star, it was document-based rather than application-based, a regression that was forced on the design of the Macintosh to achieve the reduction in cost. You saw the documents, and the applications were just the things that backed different document types, resulting in a more coherent system than if you had to load applications individually. Lisa was full of innovations, including the header bar and pull-down menu structure that is characteristic of the version of the desktop that we use today. An extraordinary collaboration between Larry Tesler and Bill Atkinson<sup>3</sup> was the source of this invention, which is recounted in the interviews later in this chapter. Lisa was a multitasking operating system, but it was too big to run on the Mac 128, so the Mac had to sacrifice many of the most valuable features. The Mac made up for this in design flair; the software was developed by a small team of only ten people, inspired by the design leadership of Bill Atkinson. It shipped only eight months later than Lisa. The price point was right, and it was Mac that made Apple successful, with Lisa failing commercially but providing the important interaction design precedents.

## Microsoft Windows

MICROSOFT WAS QUITE vulnerable at that time, as Apple had the superior design, but Apple was restrained by the dilemma of having very high premiums for the Mac hardware. If they had licensed their operating system for use on the PC, the PC hardware would have been a lot cheaper than the Apple hardware. For the very uncertain return of making its software available for the PC platform, Apple would have given up the well-defined high margins it had and probably would have been forced out of the hardware business.

Microsoft had DOS as its operating system and tried to do a version of Windows, but the screen was too crude, so Windows 1

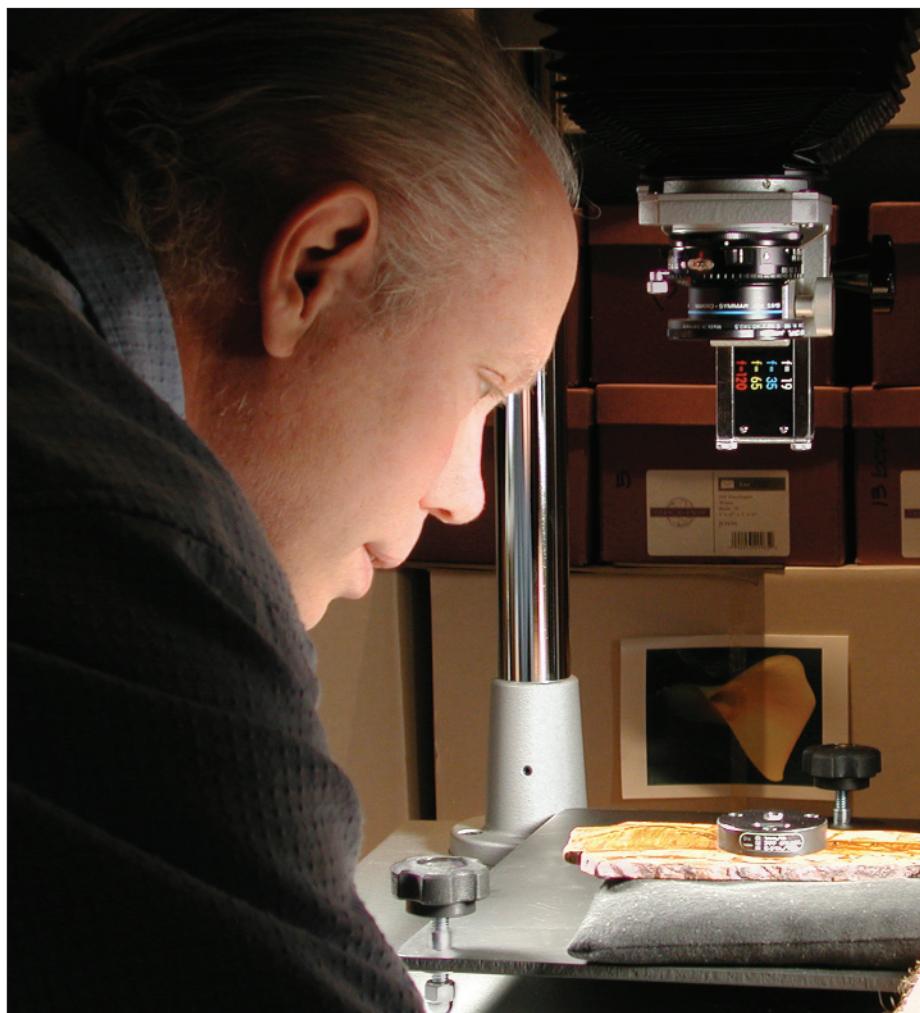


Original Windows packaging ■  
Windows 1.01—file list and menus ■  
Windows 1.01—Notepad text editor ■  
Windows 1.01—calendar current day ■  
Windows 1.01—goodbye via ALT-F4 ■

and Windows 2 were unsuccessful. Windows 3 was hardly any better. At last, Windows 3.1 was good enough, and was suddenly widely adopted. By the time Windows 95 was launched, Microsoft was in a very dominant position and the desktop and the mouse were the universally accepted paradigm for personal computers.

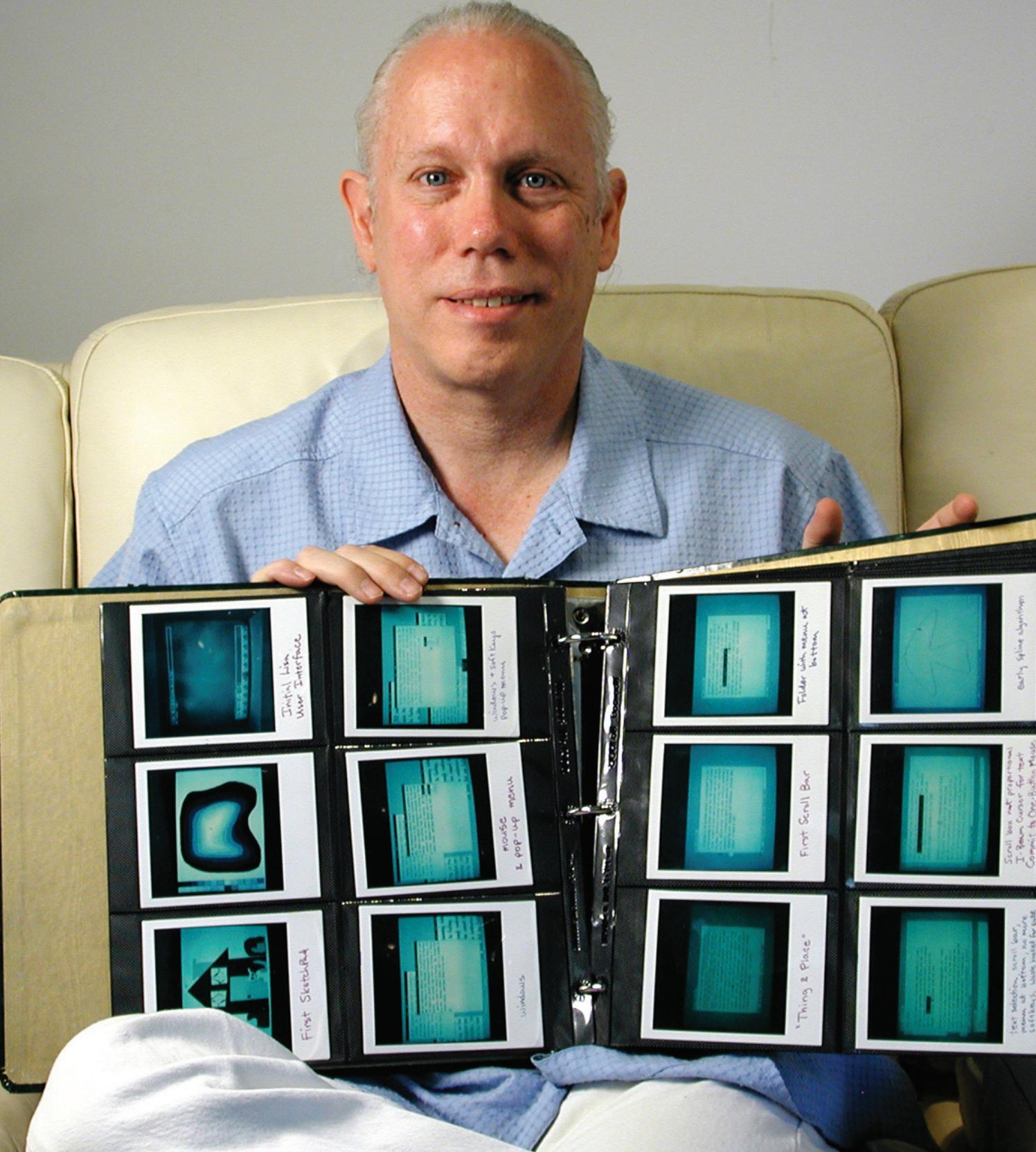
After Steve Jobs was pushed out, John Sculley took over the leadership of Apple. The company sued Microsoft over the use of the interface, and after long and expensive litigation the courts decided in favor of Microsoft. The judge's decision was based on a precedent about functional versus visual copying; he said that Windows did not look exactly like Mac. The litigation may have distracted Apple from pushing ahead with innovations and improvements that would have ensured their lead. Microsoft's domination was consolidated, and Apple nearly went under, just recovering when Steve Jobs returned. He pushed forward to develop Mac OS X and inspired his team of industrial designers to create captivating products.





Bill Atkinson

Since 1995, when he retired from General Magic, Bill Atkinson has been pursuing his passion for nature photography full-time. The basement of his home in the hills above Silicon Valley is full of sophisticated equipment for photography and color printing. He has published a book<sup>4</sup> of photographs of polished slices of rock that glow with the most amazingly vivid colors, in designs as rich as any Kandinsky. His love of perfection comes across in every detail of this work. He has researched and published color profiles for accurate printing; he has taught the art of printing to the most famous nature photographers in the world; and he has photographed the very best samples of rocks from the collections of gem and mineral specialists. In 1978 Bill was a graduate student in neuroscience at the University of Washington when he got a call from Apple, suggesting that he visit. Steve Jobs persuaded him to join, saying, "If you want to make a dent in the world, you have to come to the place where it's happening!" He was hired as the "Application Software Department," and went on to lead the development of the software for Lisa and Macintosh, and to design MacPaint and HyperCard. He was a leader and inspirational designer at Apple for twelve years, during which time Apple grew from thirty people to fifteen thousand. In 1990 he branched out with two friends to found General Magic, based on a notion called Telecards, "Little electronic postcards that would fly through the air and, like magic, land in a loved one's pocket." Bill Atkinson designed several of the interactions that define the desktop.



Initial Mac User Interface

First Sketches

Windows + Soft Keys  
Pop-up menus

Mouse  
E pop-up menus

Windows

Folder with menu at bottom

First Scroll Bar

"Thing 2 Place"

Early Spline algorithms

Scalable font proportional  
I-beam cursor for text  
Commit to One-Button Mouse

Fast selection, scroll bar,  
menu of buttons, no more  
stacking, better suited for fast

# Bill Atkinson

- Bill Atkinson holding his file of Polaroid photographs, which he used to document the interaction design of the Apple Lisa

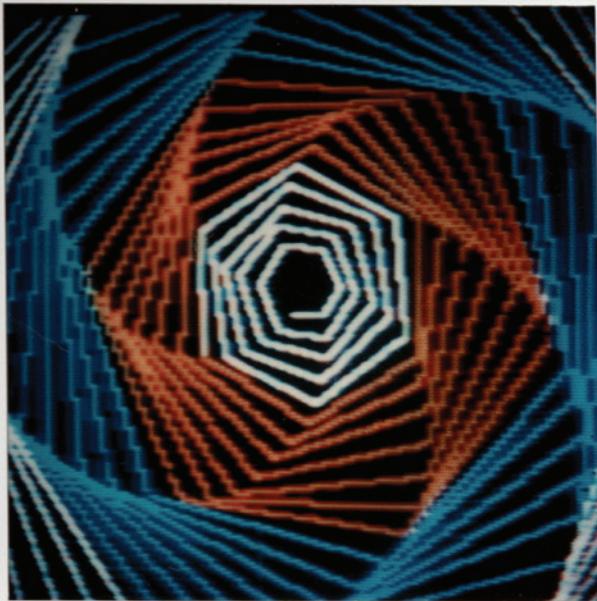
*Photo  
Author*

## Apple Lisa

As a boy, Bill was fascinated by chemistry. He combined different ingredients to make homemade gunpowder and rigorously tested for the optimum mix to power his skyrockets. This passion for research and experiment pushed him toward a career in neurochemistry. As he studied the human brain, the other side of his own brain was always engaged with taking beautiful photographs and making exquisite prints. Perhaps that dichotomy was a sign of his design talent, as he was habituated to both problem solving and aesthetic values.

His undergraduate studies in chemistry were at UC San Diego, and that's where he met Jef Raskin, whose book *The Humane Interface*<sup>5</sup> makes you think hard about the underlying concepts behind today's user interfaces. Bill describes Jef:

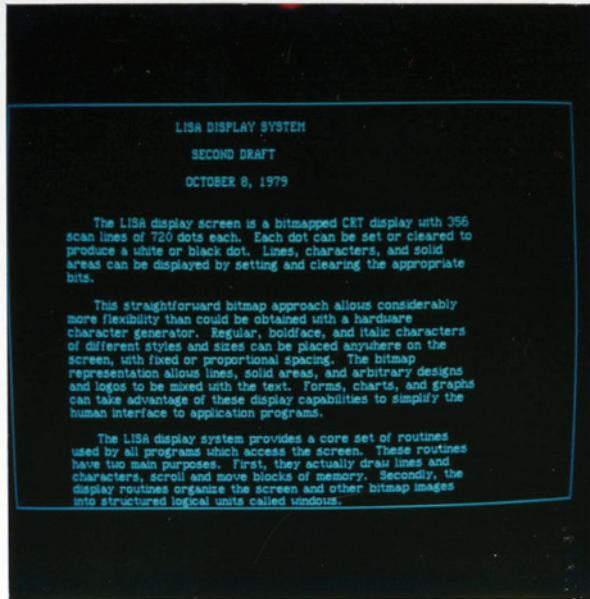
Jef Raskin was a very brilliant and very independent-thinking professor. I remember one time he got in trouble with the campus computing center because he used his computing budget to buy a minicomputer, terminals, and bean bags for his students. This was at



Apple II graphics  
for Pascal

```
*%&`0m+,./0123456789;:<>?  
@ABCDEFGHIJKLMNPQRSTUVWXYZ(\`)-  
'abcdefghijklmnopqrstuvwxyz(())"
```

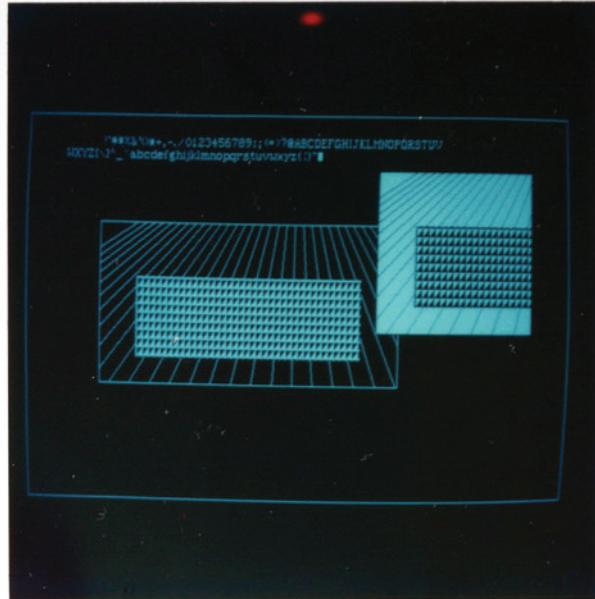
First Characters  
on Lisa



The LISA display system provides a core set of routines used by all programs which access the screen. These routines have two main purposes. First, they actually draw lines and characters, scroll and move blocks of memory. Secondly, the display routines organize the screen and other bitmap images into structured logical units called windows.

This straightforward bitmap approach allows considerably more flexibility than could be obtained with a hardware character generator. Regular, boldface, and italic characters of different styles and sizes can be placed anywhere on the screen, with fixed or proportional spacing. The bitmap representation allows lines, solid areas, and arbitrary designs and logos to be mixed with the text. Forms, charts, and graphs can take advantage of these display capabilities to simplify the human interface to application programs.

The LISA display screen is a bitmapped CRT display with 356 scan lines of 720 dots each. Each dot can be set or cleared to produce a white or black dot. Lines, characters, and solid areas can be displayed by setting and clearing the appropriate bits.



Hand-Built Font  
Proportional Spacing

Text, Lines, Patterns

a time when computer science students had to punch cards, submit a deck to the mainframe operators, and come back the next day to find out what mistakes they'd made. Jef was always thinking in terms of user interface and making things more humane for people. I credit him as giving me my interest in how to make things more humane. Later he was the one who invited me to Apple Computer and got me started there.

Bill picked up programming and computer science as a peripheral aspect of his studies, but his habitual thoroughness pushed him to master both the science and the black arts of writing code for software. Jef Raskin saw the latent design talent in his young student and thought of him when Apple needed some help in designing application software; in those early days at Apple, software was thought of as a necessary evil to sell the hardware products. When Bill got there, at first he brought together and cleaned up user-contributed software—little basic programs for calculation, games, and so on. The first major project that he did for Apple was porting over the Pascal system from UC San Diego to the Apple II.

His big opportunity came when he was given responsibility for leading the design of the graphics and user interface for Lisa, writing the QuickDraw graphics routines that all the applications used and designing the user interface. He talks about his process:

In the user interface design, a lot of it was by trial and error. We tried different things and found out what did and didn't work. A lot of it was empirically determined. I kept bringing new stuff in and saying, "What about this?" and Larry would set up tests so that different people could try it.

For example, if you have a scroll bar, which way should the arrows go, and where should they be? When you scroll toward the bottom of a document, the document moves up, so there's some reason to think of a down arrow, and some reason to think of an up arrow. A really good question to ask is, "What do people expect? When people see an arrow, which way do they think it will move?"

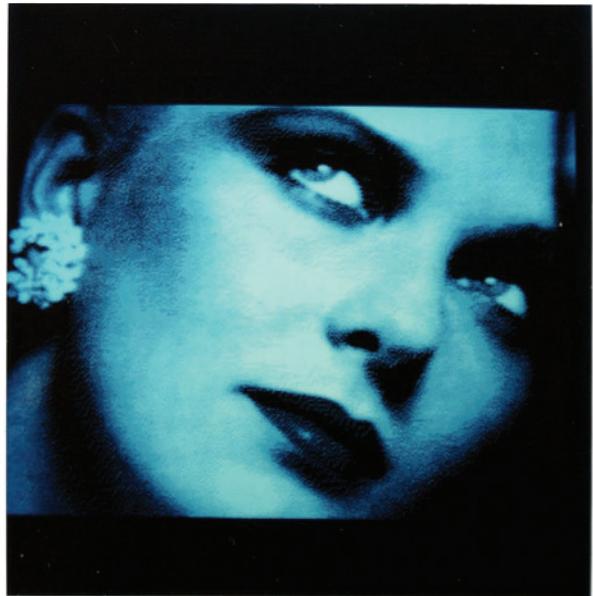
What I found mattered much more than whether the arrow went down or up was where the arrow was; if the arrow was at the top, they expected to see more of what was above, whereas if it was at the bottom, they expected to see more of what was below.

- Polaroids of Apple II graphics for Pascal and early Lisa development on dark background

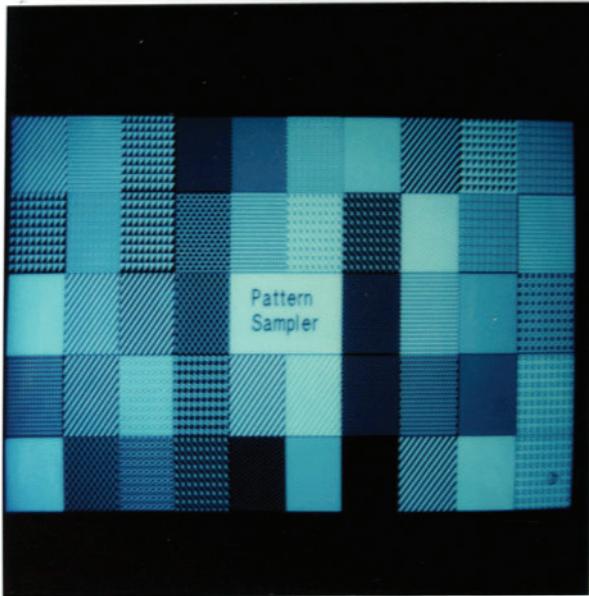
Photos  
Bill Atkinson



Three-D Graphics



Halftone Images



Patterns



25x80 + softkeys  
80x9 = 720 dots wide  
Scanline pointers

Bill also had to defend the value and importance of user interface issues against the pressures of cost and technology limitations. For example, the screen initially had a black background, with labels for soft-keys at the bottom of the screen. The black background was different from the printed version on white paper and could not obey the WYSIWYG (What You See Is What You Get) maxim, so Bill insisted in changing to a white background. The engineers complained like crazy, saying that it would draw more power and burn out the tube faster, but he said, “Get over it! Find a way around it.”

Lisa was designed for an office worker, whereas the Mac was aimed at a fourteen-year-old boy. The investment that Xerox had made in Apple and the agreement to use the mouse fueled the legend of Apple stealing from PARC. Bill Atkinson is irritated by this:

The people at Xerox had done some wonderful work on using a mouse and using windows, and that was work that presaged what we did at Apple. What we did was not just to replicate that, but to start anew with a fresh research project, asking the question, “What is the best user interface for this whole thing?” We tried a lot of things. We didn’t just take what they had done; in fact, many of the things that we did they didn’t have at all.

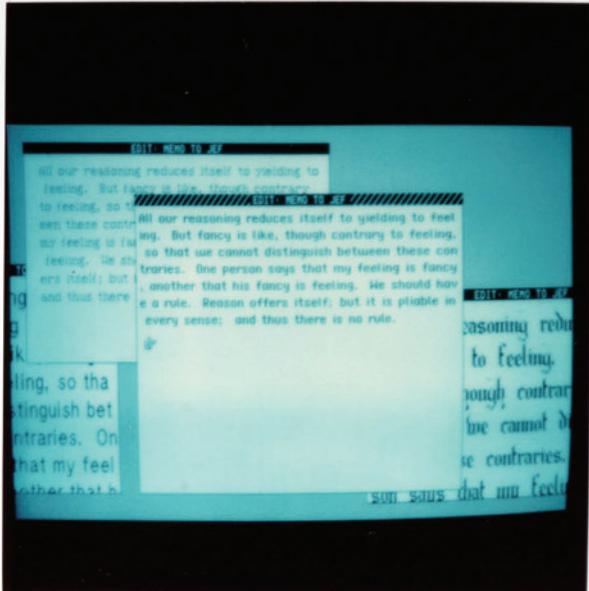
I actually got to go to Xerox PARC for one and a half hours; that’s the whole time I’ve ever been at Xerox PARC. There was a whole lot of original research done at Apple on the user interface design, and I was in the center of that. I wasn’t the only person doing the design, but I was sort of lead, and I kept throwing up ideas and seeing which ones worked and which ones didn’t. What we saw when we went there for that hour and a half was the Xerox Alto, their older system, and it was running Smalltalk. We didn’t get to see their Star, which I think was later.

Larry Tesler is more sanguine, as he had actually come over from PARC:

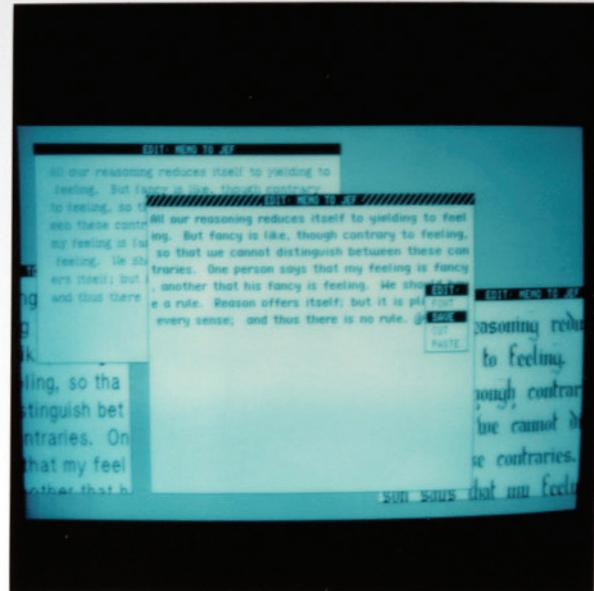
The people who had come from Xerox had a rule that we could not ever disclose what we had done at Xerox. When I got there, I asked them how this worked.

- Polaroids of early Lisa design concepts for graphics, soft-keys

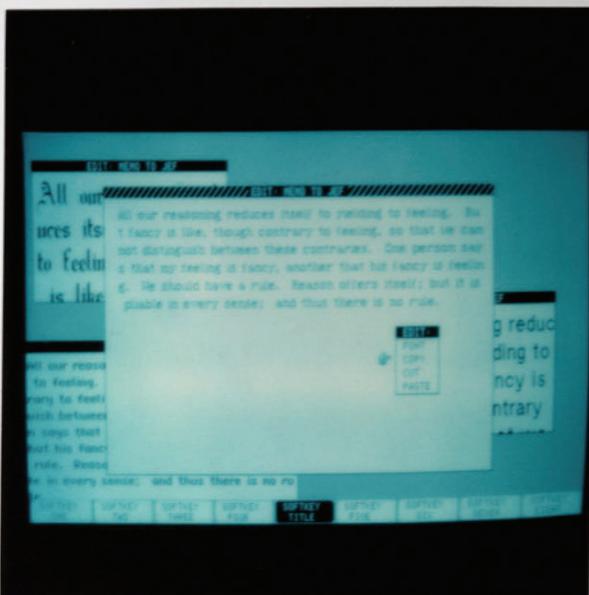
Photos  
Bill Atkinson



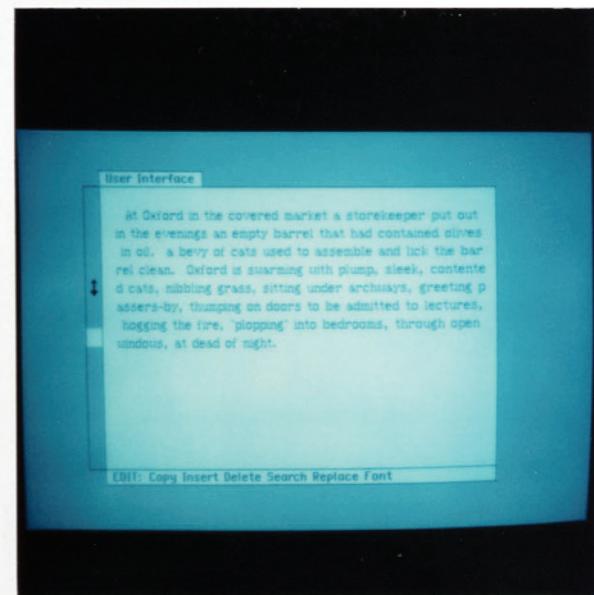
windows



mouse  
& pop-up menu



windows + soft keys  
Pop-up menu



First Scroll Bar

"We never tell them anything that's in a Xerox product that is secret," they said, "but we just let the discussion go, and encourage certain directions, and discourage other directions. We just help them not to spend a lot of time on the dead ends, and encourage them to explore the good stuff. What's funny is that they usually come up with a totally different way to do it that's actually better."

That's why the Lisa looks quite different from the Xerox Star. Very few people working on Lisa had ever seen the Xerox Star. I was one that had, but we were intent on having it not be a Star, and not letting the Star things come into it. It looks a little more like Smalltalk, which they had seen and were allowed to see. Most of Lisa was made up by these people. It was like letting a designer see a concept car through smoked glass, so he couldn't quite see it, and then having a discussion with him, and letting him design his own car.

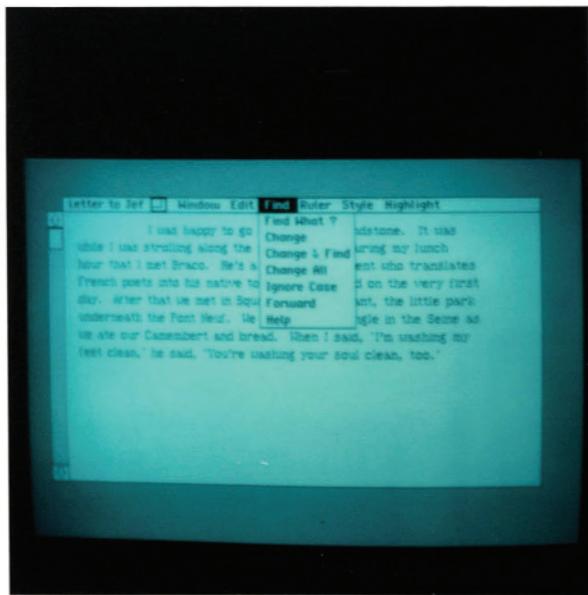
In some cases the PARC precedent spurred the Apple team to invent new designs that were not part of the Alto, as for example with the partly covered window. Bill Atkinson saw overlapping windows during his hour and a half at PARC, and he thought he saw text and graphics being refreshed in one of the windows that was partly covered, with only an L-shaped part of it visible. Because of this mistaken glimpse, Bill was convinced that it could be done, so when he was working on the QuickDraw graphics, he came up with a really innovative new way to solve the problem and developed a solution that was perhaps a hundred times faster than the previous state of the art. Later he was told by someone who used to work at Xerox:

No, no! We didn't have a clever way of doing that; when you wanted to draw into a window with another one on top of it, you had to completely redraw the one that was behind, and then the other one on top.

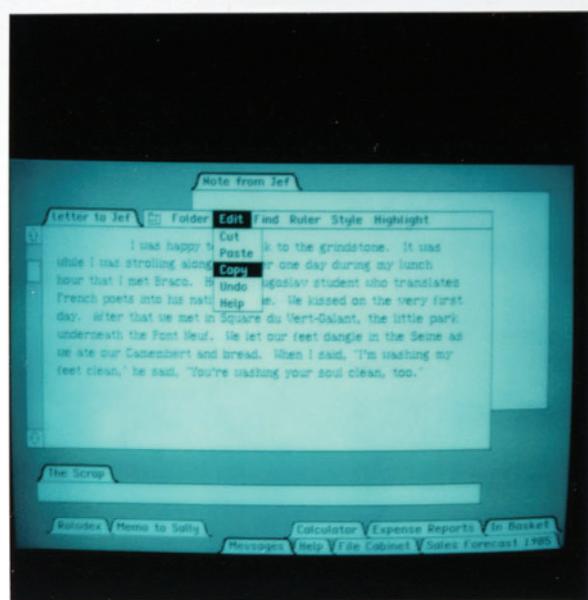
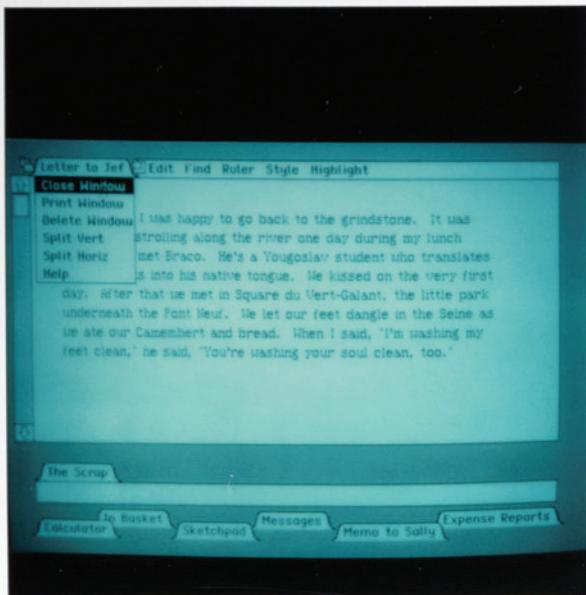
Sometimes believing that something is possible is empowering! Bill thought he had seen it done, so he found his own way to achieve it.

- Polaroids of development of overlapping window management, first scroll bar

Photos  
Bill Atkinson



pull down menus, grow icon next to title, separate horiz & vert scroll bars



rounded folder tabs, active folder hilited by bold. Double click on tab to open or close.



Menus moved to top.  
Grow icon in bottom right.  
Both scroll bars required

# Pull-Down Menus

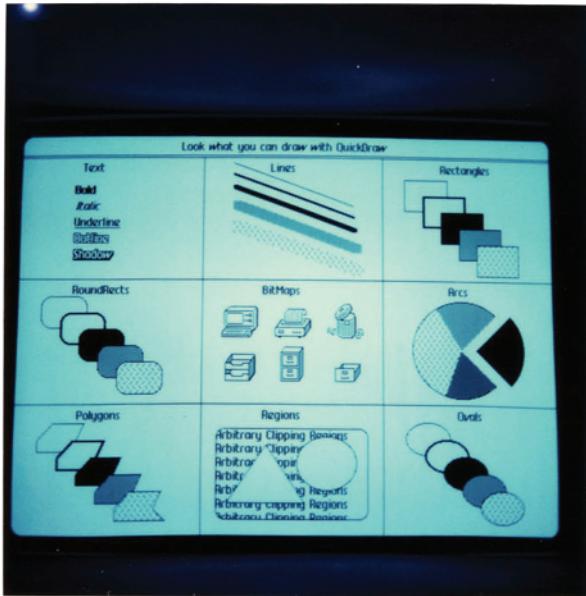
AS THE FIRST user interface specification for Lisa was being formulated, Bill and Larry formed a close partnership and developed a round-the-clock working relationship, similar to the way that Larry had worked with Tim Mott on Gypsy. Bill would work nights and Larry would work days. During the night Bill would make prototypes of user interface concepts, written in a robust enough code to support some form of testing. Then Larry would run user tests during the day. It was easy to find subjects, as many of the new Apple employees had never used a computer before. Larry would give Bill a report at the end of the day and tell him what he had learned from the tests. Then they would brainstorm and decide what to try next, so that Bill could go off and spend the night programming. In the morning he would bring in a new version and then go home to bed. They used this method for several intense weeks, until the specification was solid.

- Polaroids of development of designs for pull-down menus

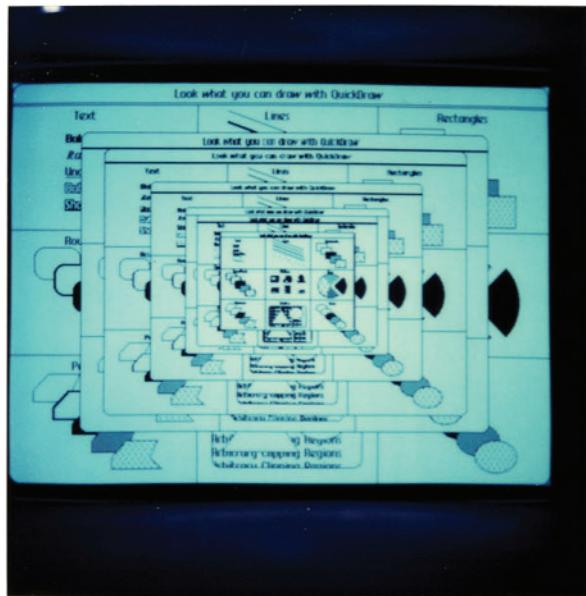
*Photos*  
Bill Atkinson

After that it became much more methodical. The team was getting bigger, so different members would implement code, and Larry would run usability tests whenever something was ready to try. He would write up the results and make recommendations. The group would argue about what to do and what not to do. Larry was always seen as the guy who was delaying the project because he would want to make changes, but he would defend the need to develop a system that was easy to use. Lisa ended up taking over three years to develop, from 1979 to 1983: the marketing people thought it was a six-month project that took three years, but the engineering team felt that it was not that long to develop a new user interface, a new operating system, as well as five unprecedented applications.

It was during the intense early collaboration that Bill and Larry designed the arrangement of pull-down menus across the top of the screen that is so familiar today. Initially there were a lot of people from HP working on the Lisa team, and they had already made many decisions about the design of the hardware. They had a bitmap display; at the bottom of the display, imitating an HP machine, was a row of buttons as rectangles, and at the top



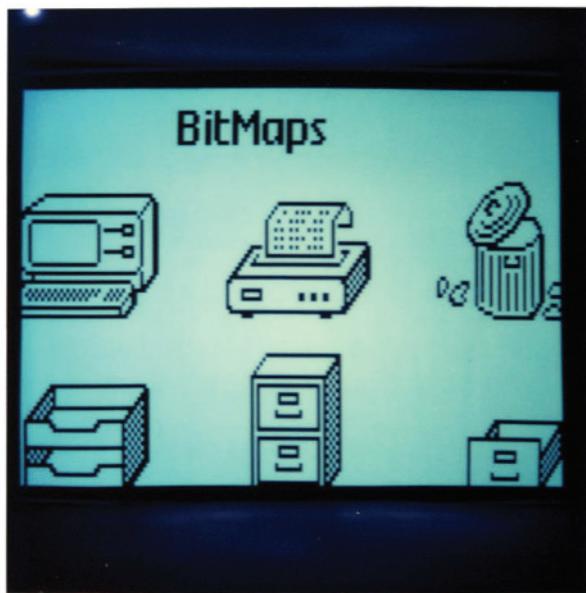
QuickDraw  
Sampler



Pictures and  
Scaling



Bitmap Stretching



of the keyboard there was a row of unlabeled physical soft-keys, with the labels on the screen above. If you pressed one of the keys, you had to look on the screen to see what the key meant at that moment. There was a hierarchy of menus: when you pressed the key, all the labels would refresh and change to the next level. Larry objected to this approach, as the only advantage it offered the user over the HP machine was the option to use the mouse to select, instead of using the soft-keys.

For the next iteration they tried putting a row of buttons along the bottom of every window, and a keyboard without the soft-keys, so you always had to use the mouse. Larry complained:

No, guys! People can't keep all these menu hierarchies in their heads.  
Everything needs to be apparent. If you want to do something, there  
should be a key on the keyboard or a menu choice on the screen that  
you can see right now!

Bill agreed but was not sure what to do instead. Larry suggested making a menu appear when you clicked one of the buttons along the bottom of the window, but the problem with that idea was that if your window was near the bottom of the screen there was no room for the menu to drop down.

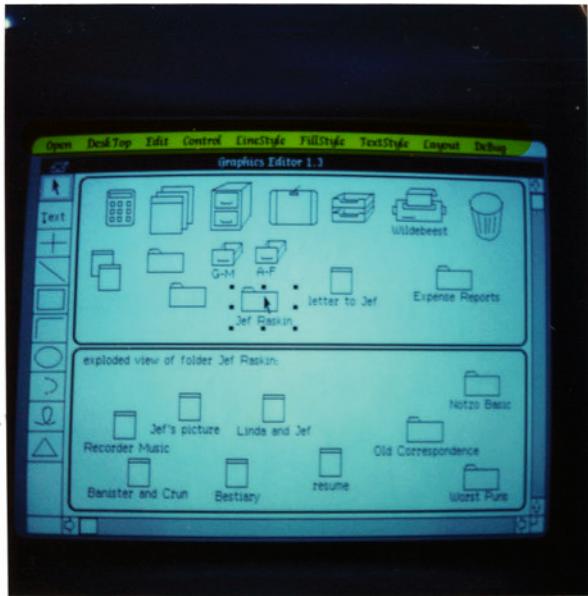
"Why don't we move it to the top of the window instead of the bottom?" Larry suggested.

Bill was willing to try anything, so he tried the top. Next they had a problem with narrow windows, where the menu titles would have to wrap around and have multiple rows of buttons, or go off to the side of the window, which looked really weird. The breakthrough came when they thought of putting the menus along the top of the screen, rather than in the windows. They both remember it as their own idea, that they suggested it to each other, and that they pushed it forward in spite of resistance from other members of the team. Larry describes the night when Bill converted the concept into a design:

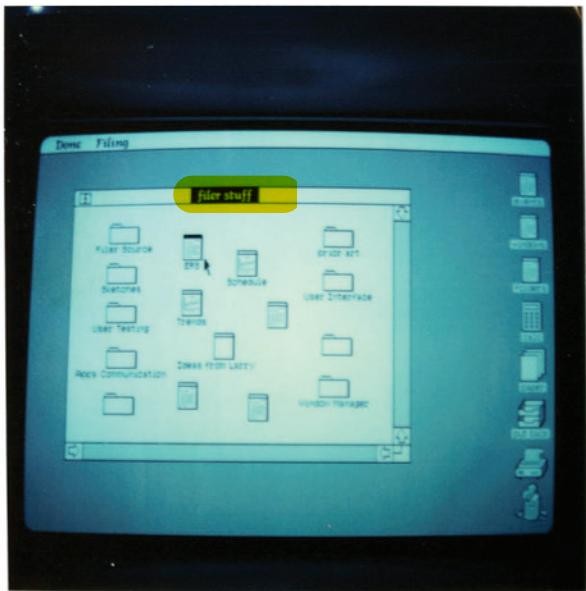
In one night he developed the entire pull-down menu system!  
Everything! He hadn't just moved it to the top of the screen; he had  
the idea that as you scanned your mouse across the top, each menu  
would pop down, and they would ruffle as you went back and forth,

- Polaroids of QuickDraw manipulations, bitmaps of icons

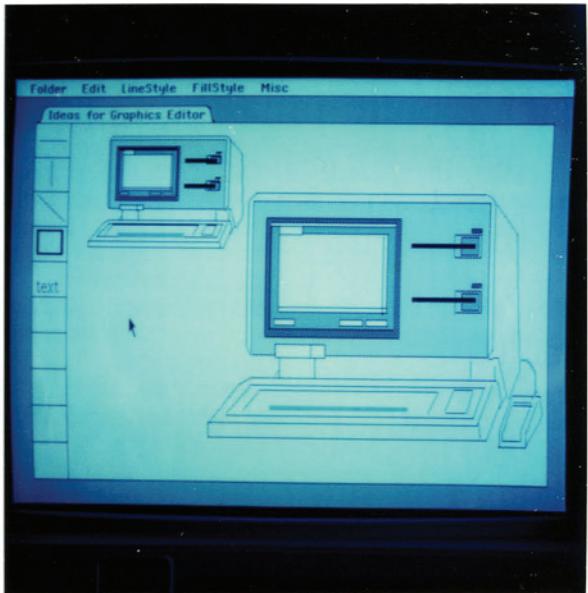
Photos  
Bill Atkinson



Office / Desk filer  
lower box is exploded view.



graphical diskette  
directory.



Ideas for Graphics Editor



and appear so that you could scan them all. When he came in the next day he went directly to Steve Jobs's office to show him first and then came to show us.

"Oh right!" I went, "This is what we want. We want everything to be apparent!"

He had come up with highlighting them as you moved your mouse down inside them, command shortcuts, and a way of showing which window was active and associated with the menu bar. He'd thought up the whole thing in one night! I can't imagine what happened that night.

One advantage of having the menus at the top is that you have the whole screen height to work with, and always the full width, regardless which window was using it. A kinesthetic feel of where that item can be found is soon learned subconsciously; you start reaching for it before you even think about it. Another advantage of the top edge of the screen is that the size of the target for the menu extends into an endless vertical column in virtual space; when you move your mouse upward, you need to be accurate in the side-to-side location, but you don't have to worry about the vertical position; it just has to be at the top or above the top of the screen. Surprisingly, the design of Microsoft Windows lost this advantage by putting an active header bar above the menus, forcing you to target the menus accurately in both planes.

- Polaroids of designs for file management, and graphic editor showing an illustration of Lisa

Photos  
Bill Atkinson

Bill recorded his prototypes by taking Polaroid photos of the screens, annotating them, and storing them in binders. You can see the development of new ideas very clearly as you look through these Polaroids.<sup>6</sup> For a time there was some confusion between folders and windows. They had a tab that looked like a folder on every window, but then they realized that a folder is a container that contains documents, rather than being the document. Another new feature of Lisa was the dialog box, which came down as an extension of the menu and offered check boxes and radio buttons to choose the parameters of a command. This was different from Xerox Star, which had a "property" key on the keyboard, so that you could select something and then press the key to see and edit the properties.



# Apple Mac

BILL ATKINSON BECAME “Mr. User Interface” at Apple. He designed the graphics for Lisa and wrote the window manager, event manager, and menu manager. Later these were ported to the Macintosh. Andy Herzfeld improved them and translated them into Assembly language, and they shipped in the first Mac. The design team for the Mac was deliberately kept small, with only ten people, in contrast to the hundred-strong group still perfecting Lisa.

Lisa was the product that allowed Apple to grow a new and unique interactive personality, but it was with the Mac that the character of the company fell into place; here the Apple brand was expressed as a coherent offering, integrating the physical design with the software. Bill remembers the difference in approach for designing the icons for the two products:

I remember an interesting incident on the icons for Lisa. You need a way to show whether there is something in the trash; if you say, “Empty the trash,” is there something to empty? The very first version of the trashcan that I wrote had little flies buzzing around it, but they got sanitized out. It was also more three-dimensional and that got sanitized out. I think some of the work in designing the Lisa user interface was a little bit hampered by who we thought it was for; we thought we were building it for an office worker, and we wanted to be cautious not to offend. When I was working on the Mac, we thought the person we were building it for was a fourteen-year-old boy, so that gave us more freedom to come more from the heart, and a little bit less from fear of offending. I think the Lisa got a little bit too sanitized and a little bit too bland.

Those of us on the Macintosh team were really excited about what we were doing. The result was that people saw a Mac and fell in love with it. Only secondarily did they think, “How can I justify buying this thing?” There was an emotional connection to the Mac that I think came from the heart and soul of the design team.

Steve Jobs was a masterful integrator of all aspects of the design that made people fall in love with the Mac. The graphical user interface was a key to making the computer a lot more

- Steve Jobs and Bill Atkinson, holding a Mac, in 1983

Photos  
Bill Atkinson

friendly, and the physical design had a perky cuteness that was very appealing. The famous 1984 Super Bowl ad by Ridley Scott had an enormous effect, and all aspects of the supporting materials, such as packaging and print matter, had panache. Steve also managed to develop a culture for the company that was evangelical in its fervor, so that people at Apple believed that they were teaching the world a better way.

Bill had enabled the graphical user interface by developing the underlying graphics primitives in QuickDraw, and his first application program was MacPaint. It started as a little test bed, to try out the graphics primitives, but as it grew he realized that he wanted to make something that would be both a tool and a toy, something that was fun to play with as well as a tool for more serious drawings. It also taught people how to use the new interface, with tool palettes, and the notion of point-and-click to draw something. Here is how Bill describes the process:

 I learned from MacPaint that in order to get a piece of software to be smooth, you must start over a number of times. You need to test it on a lot of different people and have them use the program. A lot of what I would do was just watch Susan Kare<sup>7</sup> using it.  
"What is it that you'd like to be able to do?" I would ask her,  
"What's the most frustrating thing about this?"

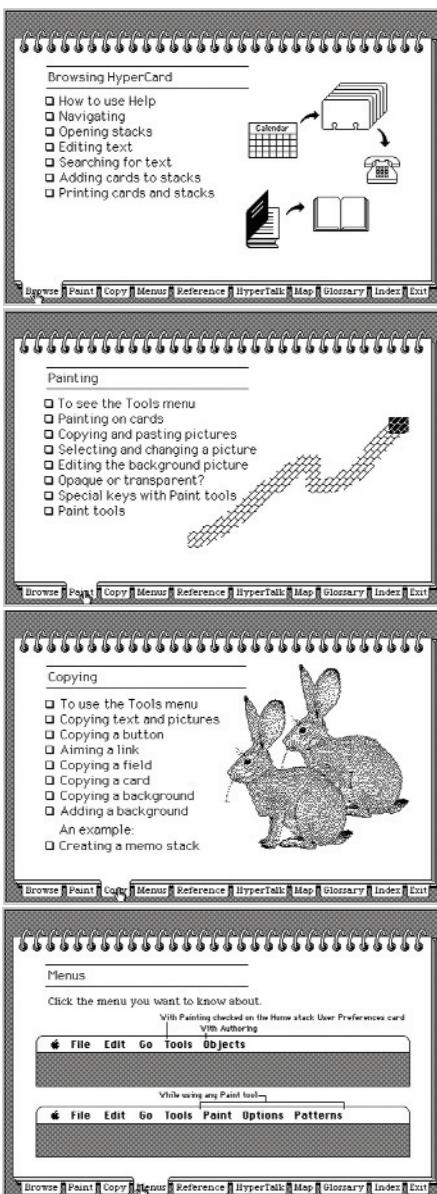
Then I would go back and see what I could do about that. I think that the more user testing a piece of software has, the smoother it can become. The process of software design really is one where you start with a vague notion of what you're trying to make, and that vague notion slowly congeals and gets better defined. As you work with it more, it gets to the point where it is something, but as you try it you realize, "You know, I've kind of missed the mark here. This is sort of what I want to do, but what I really want is more like that!"

For example, first you pit it and pat it, and you realize it's some kind of a table. Then you throw away all of the code and build a table from scratch, and you've got a clear, clean model. Then you start pitting it and patting it, and adding things that people want, and it gets a little lopsided and difficult, and you realize after a while, "You know, what I'm really building here is more like a cobbler's bench." That's when you have to put it aside and build a

**cobbler's bench deliberately, and craft it to be right for a cobbler's bench.** You iterate like that, testing, and then being willing to set aside and build from scratch again. So much software today doesn't get that luxury. Partly because the Macintosh hardware wasn't quite ready yet, I got that luxury. **Too many pieces of software today ship when their first prototype is built, and then it's much harder for them to evolve, because they have to keep everybody happy by keeping all of the features the way people have become familiar with.**

The original Mac only had 128 k-bytes of memory. Alan Kay referred to it as a "Ferrari with a one-pint gas tank." It was particularly challenging to design a painting program that would do a lot with a small amount of memory, as a copy of the whole screen had to be kept in a buffer to allow the undo command. Bill remembers the worst-case scenario, when you typed using one of the bigger fonts, you had an undo buffer and a selection going at the same time, leaving only 138 bytes free. He knew he could only succeed under those conditions by rigorous testing. Large parts of the Macintosh were rewritten in Assembly language for no other reason than compactness; and later they had to be rewritten again in a high-level language to be more maintainable.

# HyperCard



- HyperCard help stack—browse
- HyperCard help stack—paint
- HyperCard help stack—copy
- HyperCard help stack—menus

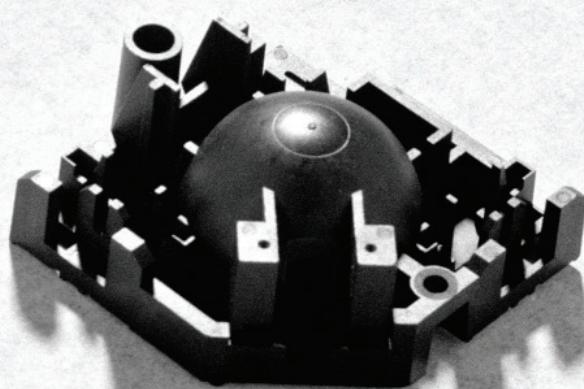
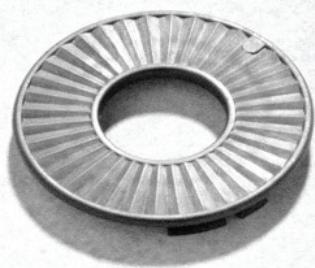
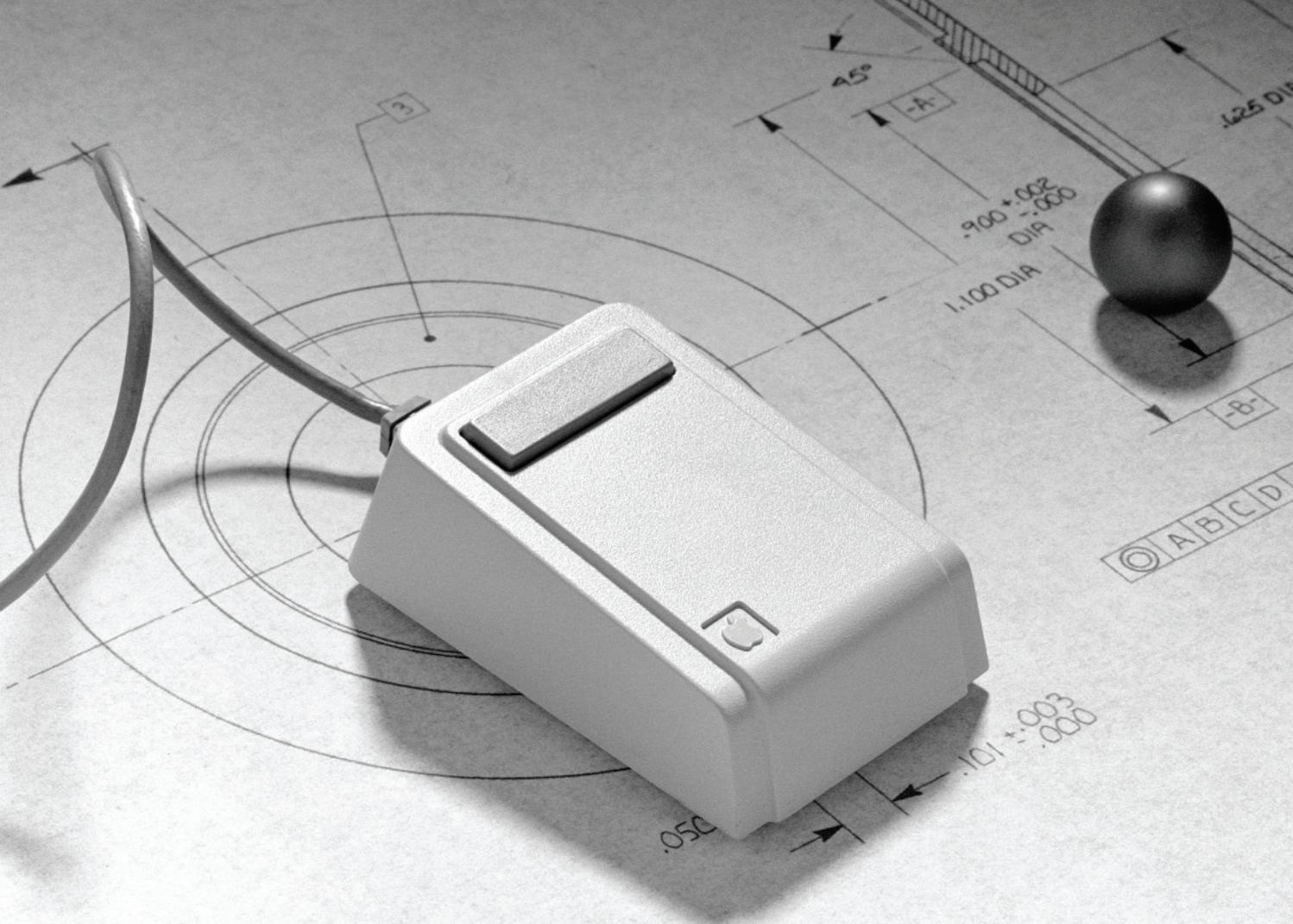
WITH MACPAINT AND MacWrite completed, Apple had graphics and text, but there was no tool to represent interactive behaviors. Bill was looking for a document format that could also support interaction—something that would respond to a user's prodding. He had a little Rolodex program that made him think of the metaphor of a stack of cards, and it occurred to him that if the cards could have graphics and text on them with links to other cards, an interactive format would be simple to use and to build. If you added a simple scripting language, you could do more than just move around the stack. HyperCard was born! Bill talks about the relationship to Web browsers and about the way HyperCard could be used for designing interactions:

I think of HyperCard as a software Erector Set. You didn't have to be able to fabricate—to machine and lathe all the parts; you just bolted them together. In many ways, HyperCard was the precursor to the first Web browser. It was like a Web browser that was chained to a hard disk, instead of connecting to the World Wide Web. Pages on the Web are like cards on a stack—they all have graphics, text, and a scripting language.

The medium changes as you go, just like someone making a bronze statue will start with pencil or charcoal sketches. They'll work quickly and make hundreds of sketches of different angles and positions and think about what it is they're trying to express. Then they'll switch to another medium and make a more deliberate, more exact painting or even technical plans of how the bronze statue is going to be supported. In software it's a little like that. The most important thing is to start with the user interface, so we use what I call "string and baling wire" prototypes. These are software programs that have no depth to them, which would easily crash if you did anything other than the prescribed course of actions, but with which you can feel what it would be like to use this program.

The process of going from one of these mockups to a rigorously crafted software application, that can withstand users banging on it and trying all sorts of weird things, is a big jump. It is a different medium. The most important thing with that first medium is to be able to try different ideas and iterate quickly. Prototyping

environments, like high-level authoring systems or Smalltalk, are very useful because you can put things together quickly. I found that when people made a HyperCard stack, they could put a prototype together in an afternoon and get it to do what they wanted. If they wanted to craft this result into a robust application that a lot of people would use, they could use the HyperCard as an example, and they could sit down with their C compiler and write the software in a different medium.



# Apple Mice

- Lisa mouse with components and engineering drawing

*Photo  
IDEO*

THE WHEELS OF Doug Engelbart's original mouse were inclined to skid and took considerable dexterity to master, so when the concept was adopted by researchers at Xerox PARC, they set about coming up with another approach to the mechanical design that would perform more consistently. The idea that successfully ousted the roller wheels was a steel ball that protruded through the bottom surface of the mouse and rolled on the surface of the desk. The movement of the ball was read by two brushes, set at right angles to each other, and mounted inside the body of the mouse. This design moved more smoothly and improved the ease of use, but it was still "An expensive species of Mouse which used lots of ball bearings, was susceptible to clogging from eraser fragments, cost around \$400, slipped on a Formica surface, and wore out after not too many hours of use (the brushes disappeared!)."<sup>8</sup> It was acceptable in a laboratory environment, to be used by researchers who were willing to learn its foibles, but would not be acceptable or affordable as a consumer product.

Apple gained rights to use the mouse designs as one of the conditions of the Xerox investment, so the Apple development

teams were working with the Star mouse, which had two buttons (rather than the Alto's three), while they looked around for alternative approaches that would be dramatically cheaper and more reliable. A design consulting company called Hovey-Kelley Design<sup>9</sup> was working closely with Apple on the physical design of Lisa. In the summer of 1980 Bill Dresselhaus, the industrial designer at Apple who was responsible for Lisa, suggested that Doug Dayton of Hovey-Kelley Design be given responsibility for designing the "appearance and mechanical package concepts" for a new mouse. Dean Hovey was more ambitious. He had been lobbying Apple for some time for the opportunity to be more deeply involved; he wanted to demonstrate a broader level of technical expertise in the design and development of new products and to take responsibility for the manufacturing. Apple turned him loose for the product development, in parallel with another consulting firm. Dean confirmed that the design specs<sup>10</sup> would be:

1. Resolution of 1/100 of an inch
2. Three control buttons to be located on the mouse
3. Will not require a special pad to roll on
4. Inexpensive to manufacture
5. Reliable and manufacturable

The design<sup>11</sup> that they created turned the mouse into a consumer product, which was manufacturable for around \$25. The mechanical design was used for both Lisa and Mac, with different external shapes.

The way in which the mouse was used for interactions with the software changed dramatically from the original specification by reducing the number of buttons from three to one. The three-button design had been inherited from Doug Engelbart's Augmentation Research Center (ARC) design and had stayed that way at PARC as the interactive approach from ARC filtered into the Alto. It was changed to a two-button mouse for Star.

Larry Tesler had always had a bias toward a single-button mouse, because of his passion for simplicity. When he joined the Lisa development team, he found that Tom Malloy was already



■ Original Lisa mouse  
■ Original Mac mouse  
■ Common mechanism

building word processing software that used the two buttons, and all the documents for the user interface referred to functions for the left button as well as the right button. Larry kept trying to promote his theory about pointing with one hand and commanding with the other, but most of the engineers were strongly for the two-button approach. Bill Atkinson was neutral, but Jef Raskin was interested in the one-button idea.

Larry and Bill ran some experiments and proved that the single button was acceptable. Then they went through the whole interface and found alternatives to using the second button, the most controversial of which was using the shift key to find the other end of a large selection. Larry wrote a single-page memo called “One button mouse,” advocating the approach, and Trip Hawkins, the head of product marketing, got really excited when he read the memo:

This is what we need to do! We need to make this for the average person and this is the kind of simplification we need to have. We're going to go with the one-button mouse!

Larry remembers:

Tom Malloy was very unhappy about it. We're good friends now, but we had many years of taunting each other about one- or two-button mice.

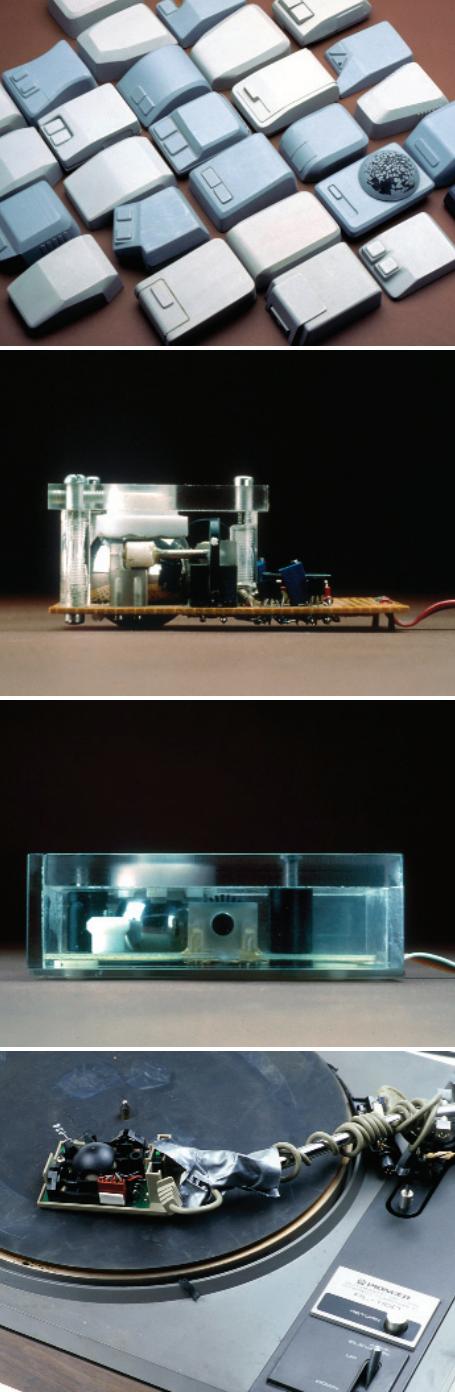
“You know,” he was always saying, “we should have put just one little tiny extra button on that mouse.”

I had to finally admit a few years ago that I now sometimes use a two-button mouse. My argument at the time was that our job was to take people who had never used a mouse before and convince them that it would be easy to use by going to the extreme of so easy that you couldn't make a mistake.

“Once they get really good at it,” I said, “then you can use a two-button mouse, but not now!”

Apple still does that; they ship a one-button mouse with the new machines, but implement the software so that you can change to two buttons when you are expert.

The third item in the spec, “Will not require a special pad to roll on,” caused the Hovey-Kelley team the most work. They



- Alternative designs for external shape
- Early working prototype
- Developed prototype
- Life test with record turntable

spent a lot of time researching and experimenting with materials and coatings for the mouse ball, in order to find a solution that would run smoothly on the normal surface materials of desks, and at the same time would operate the x and y encoders.

A reliable solution was a more complex requirement. They came up with a simple way of accessing the ball for cleaning and removing any dust on the rollers that operated the encoders. They devised a life-testing fixture for operating the ball and were not content until “after an effective three years of running in circles on Formica, the mouse has shown only a minor degradation in performance.”<sup>12</sup>

Reliability of the switch for the single button was another challenge. Larry Tesler was responsible for the functional performance of the mouse in the context of the interactions, and he warned that the characteristics of the switch needed to be just right, as if it was hard to push down it would resist and tire your finger, and if it were too easy you would click it by mistake. After a few iterations they found a switch that seemed good. Larry recalls a conversation with Bill Lapson, the Apple project manager:

“How many times will you be able to click it before it wears out?” I asked.

“Oh, these are the best; a hundred thousand times,” he replied.

“A hundred thousand? That’s not enough! It’s got to be millions.”

Bill couldn’t believe it; he thought that most of the time would be spent typing, and just once in a while you would need to point at something and click on it. Would that be once every five minutes, or once an hour?

“No, no!” I said, “Constantly, click, click, click, all the time!”

We grabbed a piece of paper and calculated the number of clicks, and it was two orders of magnitude more. The switch would have lasted two weeks. They had to find a better switch, which raised the cost, but it was essential.

The characteristics of the cord were also important for the interactions. If the cord were too stiff it would cause the mouse to move on its own, which was very disconcerting as the cursor would move across the screen; if it were too floppy, the mouse

would trip over the cord. The Hovey-Kelley team collected samples of all the available cords, and Larry Tesler experimented with them, and with the length.

The Apple mouse set a precedent for the design of a pointing device that worked well at the right price for personal computers. Alps developed comparable products, and Logitech entered the market with a series of pointing devices, but the precedent that Apple had set continued to be the design to study until Microsoft decided to develop a mouse in 1987. Paul Bradley tells the story of designing the Microsoft mouse in the interview that follows.



Paul Bradley

"The Microsoft mouse was my personal favorite design. Designers enjoy doing something that is well regarded not just by the design community, but also by the people that use those products. I can still remember the early reviews that came out about the Microsoft mouse. It won design awards, but if you looked into the business press and the trade press, it was winning all the awards there too. There was a certain pride in designing something that had a legacy and was really well accepted by all the audiences that it was intended for." Paul Bradley discovered his vocation while he was studying architecture at Ohio State, changing to the industrial design program there when he found out about it. He joined Mike Nuttall at Matrix Product Design in 1984 and continued with the organization when it became IDEO. After only three years of experience, Paul had the chance to work on the design of the Microsoft mouse, and since then he has designed more than ten other input devices, including mice, trackballs, joysticks and new types of devices for Logitech. He was recently invited to Logitech to celebrate the production of their five hundred millionth mouse, with an acknowledgment that he had made a significant contribution to this success. Paul has designed many different technology and consumer products for companies such as Dell, Intel, Samsung, and Nike, and he helps the other designers at IDEO with his design philosophy, ideas, tools, and advice. The Microsoft mouse was unique in that he had an opportunity to search for and set a new standard for the design of mice.



# Paul Bradley

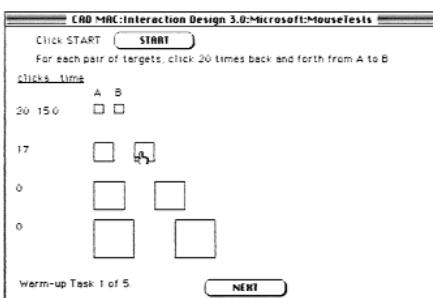
- Microsoft mouse with development models

*Photo*  
Rick English

## Microsoft Mouse

IN 1987 MICROSOFT approached Matrix Product Design with a brief to design a mouse that was going to be better than anything that preceded it. What an exciting challenge for a group of designers! Matrix specialized in industrial design at the time, so they turned to their usual partners to form an interdisciplinary team, ID TWO for interaction design and human factors research, and David Kelley Design for mechanical engineering design and to support the manufacturing. This partnership worked well and formed a precedent for the three companies to come together to form IDEO a few years later.

Mike Cooper, the program manager from Microsoft, proposed a schedule that seemed impossibly ambitious at the time, wanting to bring a product to market within seven months. He managed to bring it off in spite of the fact that a typical development and tooling cycle at that time was over a year, but the vendors were so eager to work with Microsoft that he was able to jump to the head of the queue and get the fastest service that they could offer.



- Mike Cooper and team—Paul Bradley RH
- Handhold example
- Testing setup
- Test for mouse designs—tapping task

Paul Bradley describes the initial exploration of possible design concepts:

The question was really, What could a mouse be? We were going out and talking to users about how they use the mouse, observing them, watching them. Looking at the nuances led us to certain beliefs, so we built prototypes to test the ideas. We thought that by building small mice that were symmetrical, a small round or square mouse, that you could capture the mouse in your fingertips like you capture a pen, and create a higher degree of accuracy. We were very surprised when we did testing with prototypes against other mice. Although these smaller mice felt better, they failed in 70 percent of the tests against the more traditional elongated mouse.

We discovered that having the mouse fully encompassed in your hand gave more control, although the sense was that your fingertips would give you more control. It was the resting posture of your arm and hand on the table that allowed you to perform more accurately and quickly.

## Tests

BILL VERPLANK,<sup>13</sup> WHO at that time was working with the author at ID TWO, devised tests. He worked out five tasks that would exercise the ways in which a mouse is used and wrote a program in HyperCard to compare time and errors for these tasks with different mouse designs. The tests were run on a Mac SE, with a range of experienced and naive users trying out the various prototypes and existing mice.

A tapping task measured the tradeoff between speed and accuracy for the most common mouse usage, that is, move and click, by asking the user to click twenty times back and forth on pairs of targets ranging in size.

Next they were asked to trace their way through a maze to reveal steering ability; this showed a dramatic advantage in moving the ball to the front of the mouse, so that it lay between thumb and finger, rather than in the conventional location further back.

A task to test precision asked the user to position the cursor exactly in the center of arrays of dots, and then to click; this revealed the importance of having the buttons on the top rather than on the side.

Writing the word “TAXABLES,” with one character in each of eight boxes, required repeated short strokes with the button down and then up; for this task a mouse is much better than a trackball, where the buttons are hard to hold while rolling the ball.

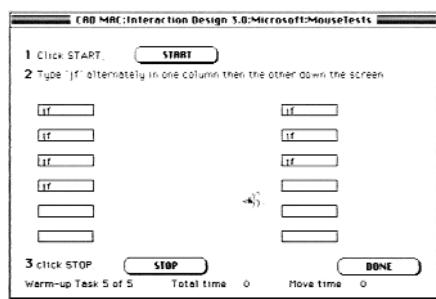
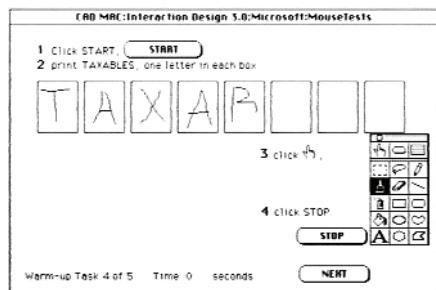
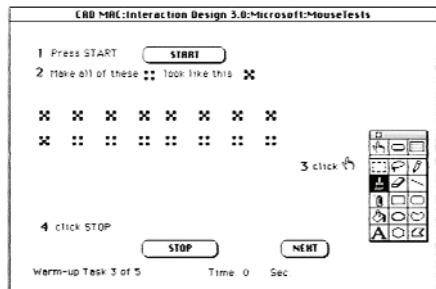
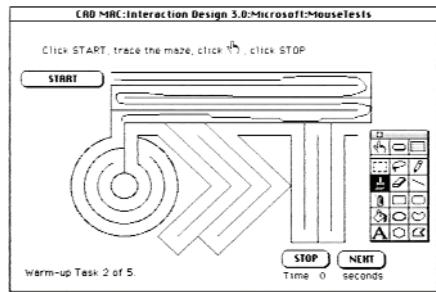
The final task required typing, then pointing, then typing, then pointing, again and again; it measured homing time as you move from keyboard to pointing device, showing that the mouse did just as well as devices with fixed positions.

These human factors trials allowed quick evaluations of a wide range of design concepts. The combination of user testing and rapid prototyping was a key to the success of the project.

## Industrial Design Development

PAUL BRADLEY REMEMBERS the many hours he spent making models, and how the inspiration for the shape came to him during the process:

We built about eighty foam models, quickly exploring different possibilities and directions. The core idea for the appearance of the final design actually came from a traditional rubber sanding block. We were in the shop, exploring shapes in foam models, working with this thing that requires a movement which is very similar to the movement of a mouse on the surface of a desk—the movement of sanding side to side or back and forth. That traditional rubber pad is a somewhat crude shape, but a very appropriate shape for what you’re doing. The symmetrical curvature from the front to the back of the mouse was inspired by those sanding pads. By adding a curvature in the side to side direction, and details of the form in other places, we were able to improve on the traditional ergonomics in the sanding pad.



- Test for mouse designs—maze task
- Test for mouse designs—precision task
- Test for mouse designs—writing task
- Test for mouse designs—homing task

Earlier mice were designed as extensions of the computer, borrowing their visual design language from desktop computers, the early PCs, and Macs. They were rectilinear and architectural, having little to do with the human form and human touch.

“Does this look right next to our keyboard?” was the question, or “Does this look right next to our CRT?”

The Microsoft mouse took a different approach: “This is about your hand. This is about the human being. This is the contact point between people and product.”

Paul was searching for a shape that looked like it belonged in the computer environment but was more connected to the person interacting with it. He developed a soft shape in pure white, and made it very shiny so it felt smooth to the touch, unlike the heavy textures that previous mice had used. He extended the buttons over the front and side edges, doubling their surface area, simplifying the appearance, and avoiding the risk of touching the surrounding frame by mistake.

Moving the ball forward caused the biggest single change to the internal geometry, as it was cheaper to have the ball in the back, leaving room in front of it for a PCB carrying the switches for the buttons. The observations of the motions of wrists and arms during the user testing, particularly for the maze task, made the designers think that positioning the ball between thumb and finger might offer better control. When they prototyped and tested the idea, the improvement was dramatic, and Microsoft was very excited; this was the kind of functional and usability improvement that they were looking for. They were happy to incur higher manufacturing costs to get something that would perform better. When they launched the product, they published the data from the tests, and soon all of the other manufacturers followed suit.

In most applications the left button is used 65 percent to 85 percent of the time. The testing process was recorded on videotape, and when the design team reviewed the tapes, they noticed that both right-handed and left-handed people tended to skew their grip asymmetrically to be centered over the left button. This made them decide to make the left button larger than



- Sanding block inspiration
- A softer shape to fit the hand
- Ball location moved forward

the right, but the people at Microsoft were worried that by designing an asymmetrical mouse, it would be more appropriate for a right handed person than a left handed person. Working prototypes were tested as soon as they were available, and the asymmetrical design won the day, but it was necessary to add a ridge between the buttons, so that people could feel the separation without looking down at the mouse.

The design was coming together, and it was time to commission manufacturing partners for the implementation. Paul Bradley describes the trips to Japan and long meetings in smoke-filled rooms:

We talked to a shortlist of four or five possible manufacturing partners, but it came down to Alps and Mitsumi. A typical meeting would involve Mike Cooper, the program manager from Microsoft, his assistant, Jim Yurchenco from David Kelley Design, and myself. I would tell them our design vision and what we expected.

"No, we can't do this," they would push back, and, "The schedule doesn't allow this!"

Then Jim would explain to them how they could do these things. It turned out to be a good process because Jim was extremely knowledgeable in tooling and manufacturing processes, and he very quickly gained their respect. They realized that either they would have to come up with better reasons for staying with their existing ways of doing things, or they would have to meet our requests.

Microsoft wanted a look and feel for this mouse that was different, so the idea of doing something white in a world of beige, and the idea of doing something high gloss in a world where everything was textured and matte was a way of differentiating themselves; in this case we demanded that we have a super high polish on every part, consistently across millions of parts.

Microsoft had also seen a lot of mice that had a year or two of wear on them, when the pad printing or screen printing of the Logo began to peel around the edges, so the brand was made to look sloppy on the device. That was completely unacceptable to them.

"We can't have Microsoft wearing off the surface of the mouse!," Mike Cooper said. "How can we avoid this?"

"You can avoid it with a double-shot molding," we replied; "you take a second color of plastic for the Microsoft logo and inject it into the actual mold."



Paul Bradley enjoying karaoke ■

He was very excited about that idea, but the manufacturer was not. After a week of negotiations, they came back.

"We can do this, but it's going to take two more months on the schedule."

Mike Cooper decided to build two sets of tooling, one just for the first two months of production. That was pad printed and coated with Urethane to give a high degree of durability. This actually turned out to be a problem on the white mice, because with exposure to sunlight the Urethane started to yellow. In parallel we built a more complex set of tools for the double-shot process, and when they were ready we replaced the initial tooling.

The feel of the mouse was just as crucial as the appearance in order to achieve a higher standard of interactivity. Central to the feel was the way the ball rolled as the mouse was moved across the work surface; it had to be smooth and delicate and connected to the movement of the cursor with consistent directness. The design team explored options for the surface and material of the ball, and discovered the performance improved with a heavier ball, so they went with a dense material that was heavier than steel. This gave the mouse itself a heft and weight, making it feel more accurate, and creating more traction on the surface of the table.

They also used Teflon pads on the underside. The whole lower shell was designed to be as clean as possible, so instead of four feet in the corners, Paul designed two wide pads that wrapped across the front and back. He used those Teflon feet to cover the screw holes as well as the label, so when you turned the device over it was clean and simple and had no mechanical details exposed.

The feel of the buttons was another challenging problem because of the larger surface area. With many of the mice at the time, when you pushed on the front edge of the button, there was a noticeable difference in the force needed to actuate it than if you pushed on the back edge. Paul tells how they worked hand-in-hand with the manufacturer to design a switch geometry that would bring the arm that attached the button as far back into the body as possible:

The further that you bring it back, the longer the lever, and therefore the actuation force is more even over the length of the button. We



■ Underside of mouse

had a sixty to eighty gram spec, so that anywhere you pushed on the button it had to be within that bracket. When the parts started rolling off the tools, Jim Yurchenco came in with his little force meter and started pushing on the buttons. If it didn't meet that spec, Jim said that they must go back and make changes to the design until it was just perfect.

Microsoft made a big splash for the product launch. They sold more than a million in the first year, and eleven million before it was replaced by another model. Initially it was packaged with Windows, and later as an independent product, but it was always retained as an exclusively Microsoft design, as they wanted to use it to enhance their own brand rather than offer it to Dell or another PC manufacturer. They went on to form an internal group for the development of physical products and began to offer peripherals to support new software features that they were developing. The success of the first mouse blossomed into a much broader offering. Paul remembers being invited to go to New York for the launch:

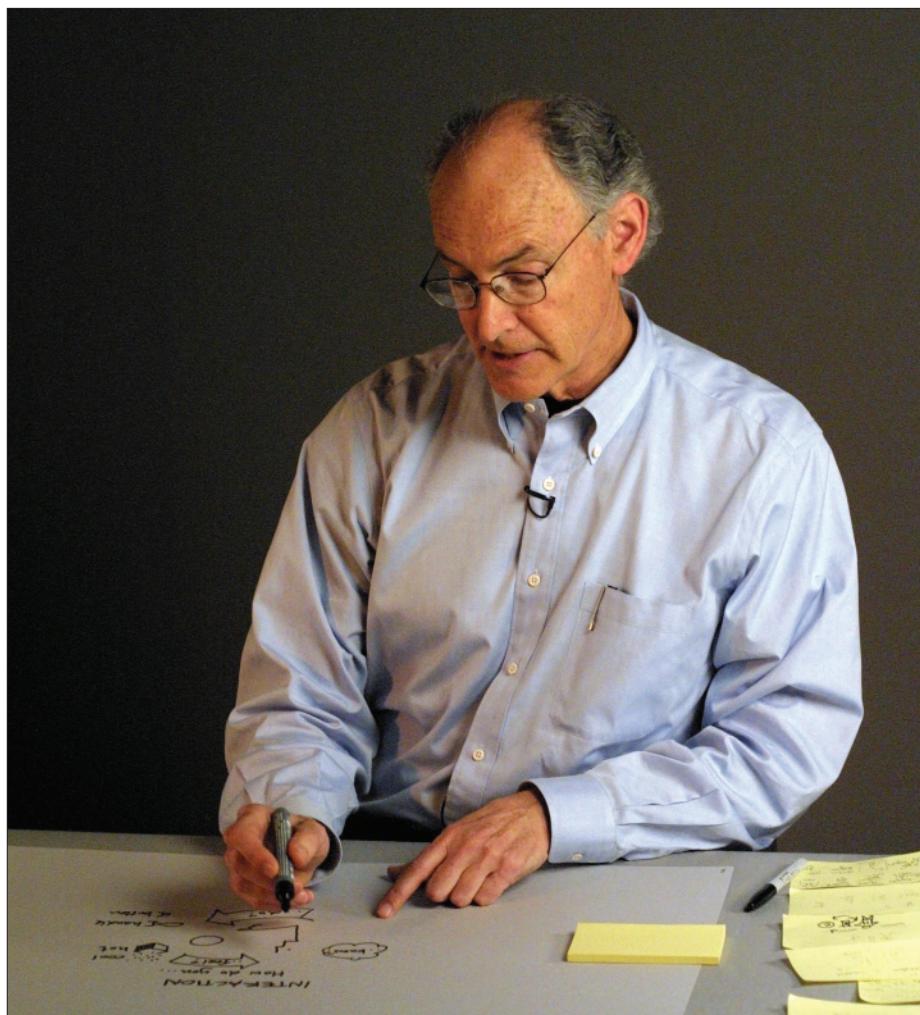
Bill Gates rented a hall next to the Museum of Modern Art in New York, and brought all the press there. Then he got up, took the mouse in his hands, and said, "Microsoft has designed the best mouse in the world!" He took us all to Broadway shows, lobbied the Museum of Modern Art pretty hard to put it in the museum right then and there, and published comprehensive promotional documents that really talked about ergonomics. He not only wanted to put out a message about the look of this device, but also that the interactivity was really important, and how they'd improved it.

First credit for setting a new standard in the design of mice goes to Mike Cooper for his determination to drive the program forward and to create a leadership position for Microsoft. It was also a precedent-setting collaboration between the three companies that came together to form IDEO a few years later. Paul Bradley, from Matrix Product Design, brought his passion for designing beautiful physical forms for products. Jim Yurchenco, from David Kelley Design, built on his experience designing the first Apple mouse and enforced the highest standards of

engineering and manufacturing quality. Bill Verplank, from the author's team at ID TWO, brought his knowledge of interaction design and human factors research. The partnership worked well.

Bill Verplank had joined ID TWO the year before, bringing with him a wealth of experience in designing interactions, both input devices and screen-based interfaces. Bill had already worked out how people think about computers and software, and he had developed a design process. In the interview that follows, he illustrates his ideas with drawings.



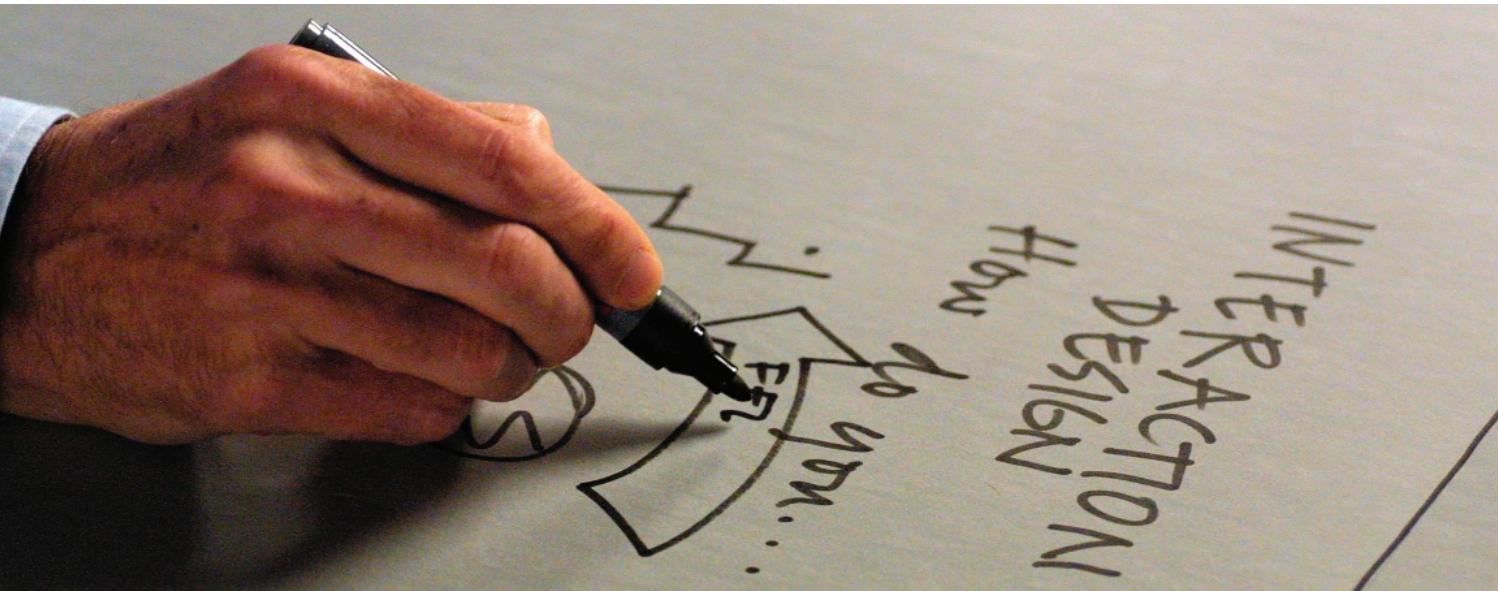
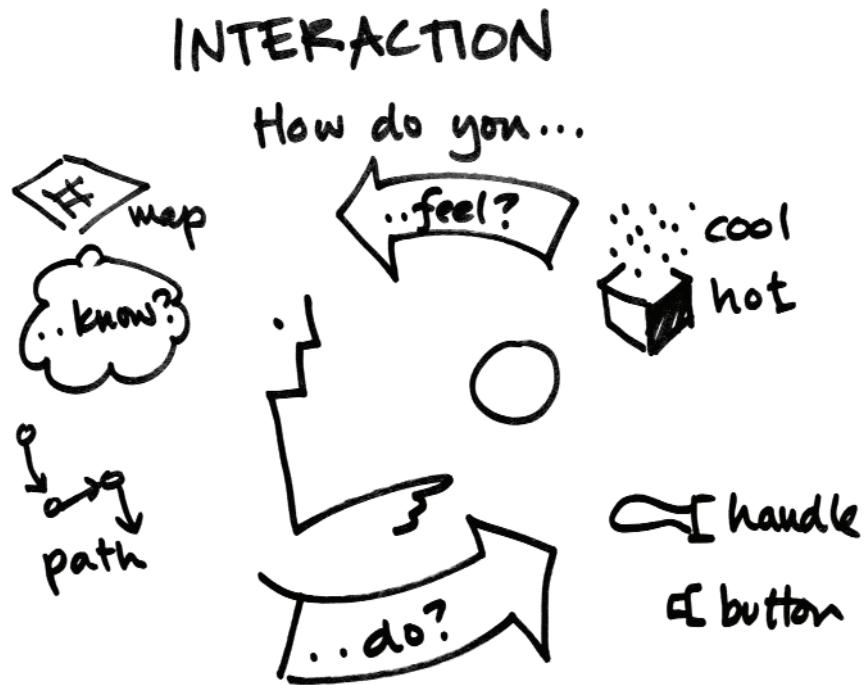


Bill Verplank

Bill Verplank has an amazing ability to draw at the same time as he talks. If you meet him and ask him a question about interaction design, you can sit at the nearest table or desk and be mesmerized by the fluency of his answer. His words are easy to understand, and as he talks he builds a beautiful diagram that reinforces what he is saying. You can take the drawing with you as a reminder and summary of his ideas about interaction design, which have evolved over many years. His PhD from MIT was in man-machine systems, applying information and control theory to measuring human operator workload in manual control tasks. At Xerox from 1978 to 1986 he participated in testing and refining the Xerox Star graphical user interface. From 1986 to 1992, he worked as a design consultant with the author to bring graphical user interfaces into the product design world. At Interval Research from 1992 to 2000 he directed Research & Design for Collaboration. He has helped to establish the Interaction Design Institute Ivrea and is now a visiting scholar in haptics in the Music Department at Stanford University. He summarizes interaction design by answering three questions about how you act, how you feel, and how you understand. He explains the context of the history and future of interaction design with paradigms that serve as patterns for the way people think about the subject. He describes the process of designing interactions with a concise diagram, and gives an example to illustrate it. He created the drawings that follow as he talked about his ideas during his interview.

## Bill Verplank

Bill says that the interaction designer has three questions to answer; they are all "How do you . . . ?" questions.



# How Do You . . . ?

## 1. "How do you do?"

**How do you affect the world?** A human, a person that we are designing for, does something, and we provide affordances. We either present handles that they can continuously control, or we give them buttons for discrete control, pressing the button and giving up control to the machine. When you are designing the way people act, there is a choice between handles and buttons. **You can grab hold of a handle and manipulate it, keeping control as you do it.**

**Alternatively you can push a button, or click on one, delegating control to the machine.**

## 2. "How do you feel?"

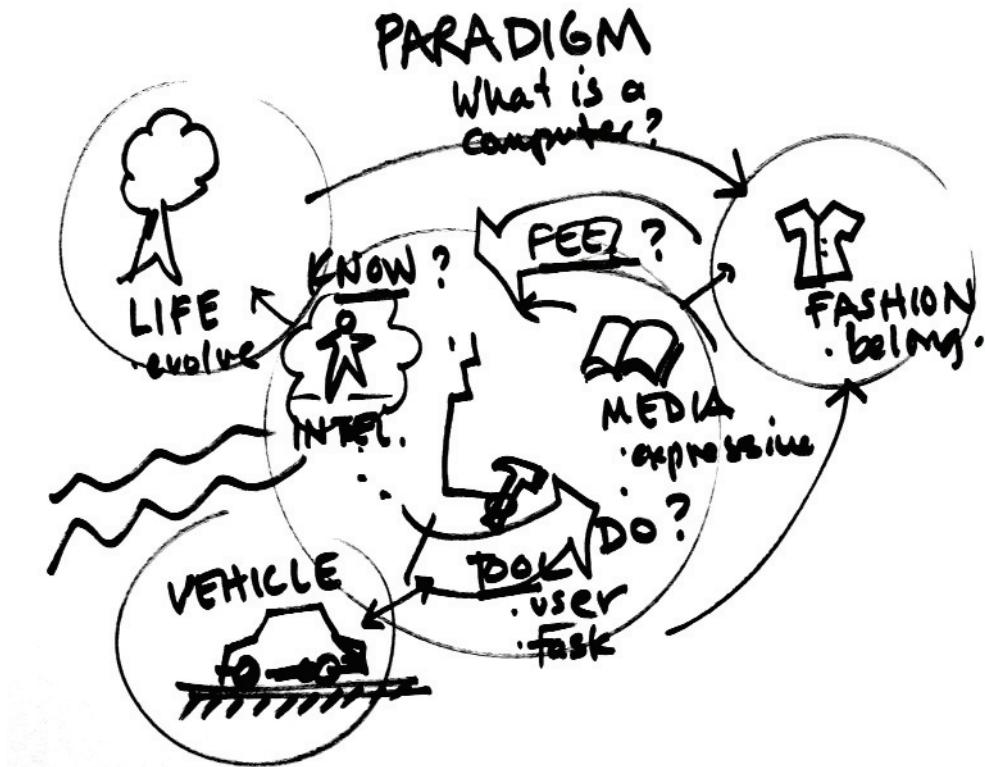
**How do you get feedback?** McLuhan made the distinction between what he called "fuzzy," or "cool," media, and "distinct," or "hot," media. **Early TV was a cool medium, with its fuzzy images. Cool media draw you in. A book with careful printing or a gravestone with carved lettering is hot, or immutable—you cannot touch or change it.** We design the way that the machine, or the system, gives feedback to the user, or the book looks to the user, or the sign communicates. That's where a lot of feelings come from; **a lot of our emotions about the world come from the sensory qualities of those media that we present things with.**

## 3. "How do you know?"

**As we design products with computers in them, it is very difficult for a user to know exactly what they are going to do.** A map gives the knowledge that you may need if you are designing complex systems. A path offers the kind of understanding that is more about skill and doing the right thing at the right moment. **It is the responsibility of the designer to help people understand what is happening by showing them a map or a path.** The map shows the user an overview of how everything works, and the path shows them what to do, what they need to know moment by moment.

# Interaction Design Paradigms

A PARADIGM is an example that serves as a pattern for the way people think about something. It is the set of questions that a particular community has decided are important. For interaction design there is often some confusion about what paradigm you are working with. The basic question is, What is a computer?



Intelligence

In the early days, designers thought of computers as people and tried to develop them to become smart, intelligent, and autonomous. The word "smart" is one that we associate with this paradigm, expecting the machine or product to be smart and to know how to do things for the person who uses it.

## Tool

Doug Englebart, the inventor of the computer mouse, thought of the computer as a tool. Styles of interaction changed from dialogs, where we talk to a computer and a computer will talk back to us, to direct manipulation, where we grab the tool and use it directly. The ideas of efficiency and empowerment are related to this tool metaphor.

## Media

In the nineties, designers thought of computers as media, raising a new set of questions. How expressive is the medium? How compelling is the medium? Here we are not thinking so much about a user interacting with or manipulating the computer, but more about them looking at and browsing in the medium.

## Life

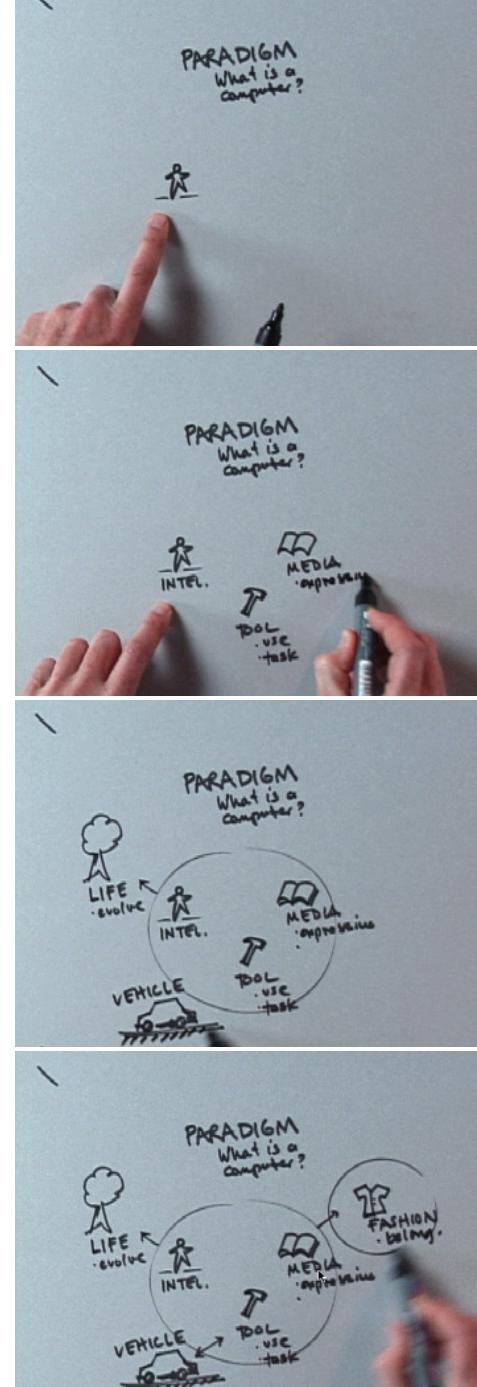
Starting in the mid nineties, people have been talking about computer viruses or computer evolution; they are thinking of artificial life. When the program has been written, it is capable of evolving over time—getting better and adapting. The programmer is in a way giving up responsibility, saying that the program is on its own.

## Vehicle

Another metaphor is the computer as vehicle, and we have to agree on the rules of the road. There has to be some kind of infrastructure that underlies all computer systems. People spend their careers determining the standards that will define the infrastructures, and hence the limitations and opportunities for design.

## Fashion

The media metaphor plays out to computers as fashion. A lot of products are fashion products. People want to be seen with the right computer on. They want to belong to the right in-crowd. Aesthetics can dominate in this world of fashion, as people move from one fashion to another, from one style of interaction to another style.



# Interaction Design Process

BILL VERPLANK SUGGESTS a four-step process. First, the designers are motivated by an error or inspired by an idea and decide what the ideal goal for the design should be. Next they find a metaphor that connects the motivation to the end goal and develop scenarios to help them create meaning. Then they work out step-by-step what the tasks are and find a conceptual model that ties them all together and clarifies the modes. Finally they decide what kind of display is needed, what the control are, and how to arrange them.



## 1. Motivation—errors or ideas

Design ought to start from understanding the problems that people are having, and also from ideals. A lot of people are motivated by problems that they see, breakdowns of one sort or another, errors that they observe. Another place that design starts is with ideas. These are the brilliant concepts, the ideals that we have for making the world wonderful.

## 2. Meaning—metaphors and scenarios

If you can tell a good story about something, or spin a good metaphor, it makes sense to people. This is where the meaning of the design comes from. A clear metaphor is the strange idea that connects two things; for example the cloud and the bolt of lightning, saying—Ah hah! This isn't a computer, it's a desktop! Along with the metaphor, we also need a variety of scenarios, to understand the context of Who is using it, Where are they, and What are they trying to accomplish?

## 3. Modes—models and tasks

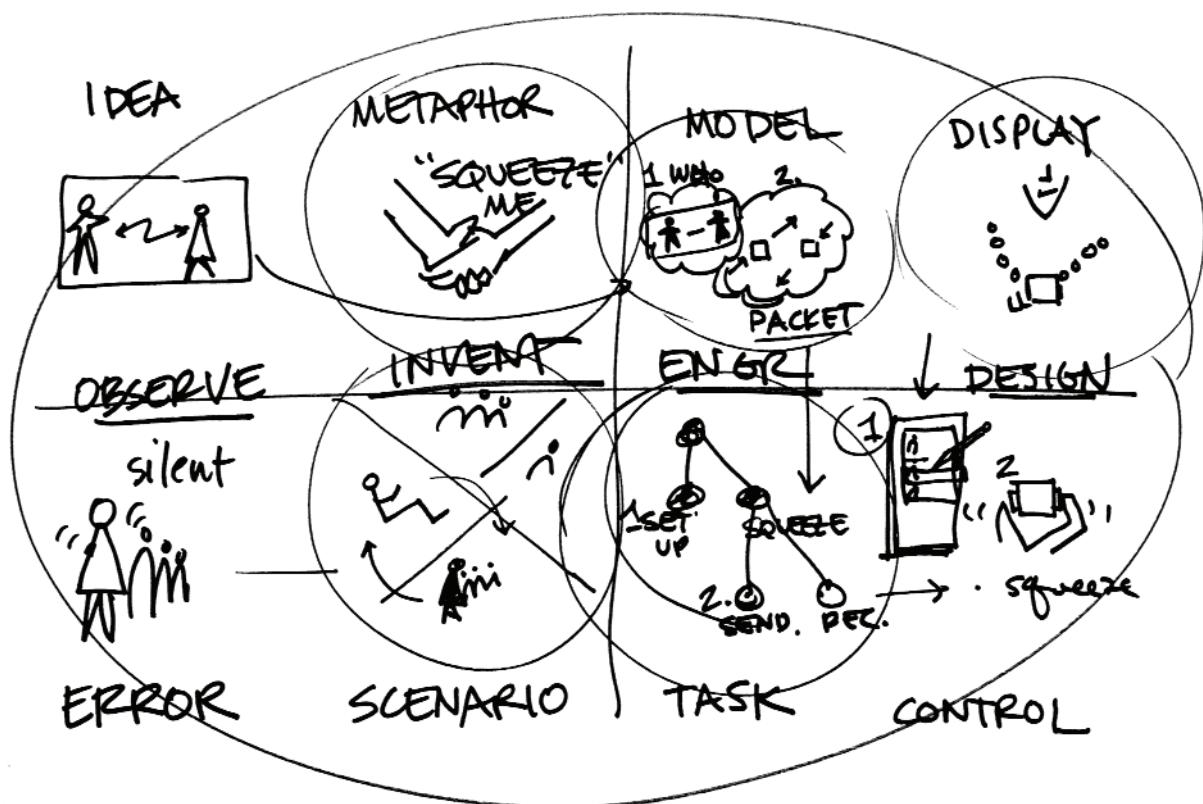
In order to create a conceptual model that users will understand, you have to have a clear picture of what they are thinking about. The mode that they are in depends on what the task is, and what they are trying to accomplish. How they can move from one mode or model to another, or from one environment to another, will then define the tasks. This is the conceptual cognitive science of understanding what the person doing the task needs to know

## 4. Mappings—displays and controls

Often, as an interaction designer, you design some kind of display and some controls. The display is the representation of things that you are manipulating. You need to be able to map the controls to the display. Those mappings can be really complicated with computers, as they can remap things in an instant, giving very strange powerful modes that can select everything or delete everything.

## Process Example, "A Haptic Pager"

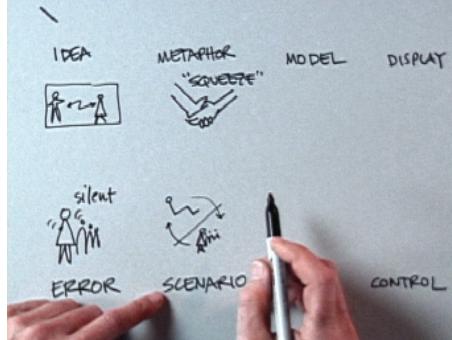
BILL VERPLANK USES a story of the development of a tactile pager to illustrate the interaction design process. The example was created by one of the students at Stanford University, where Bill teaches. It shows the progress of the concept, going from error and idea to display and control, and how metaphor, model, scenario, and task are the core of that process.





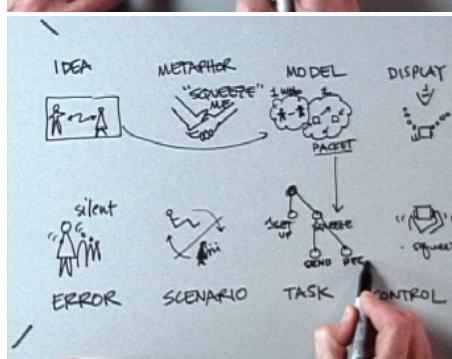
## 1. Error and idea

Celine was annoyed by pagers and cell phones going off when she was waiting for the checkout at a supermarket—and embarrassed when her own went off. **The problem that she wanted to solve was how to make a silent pager that she could feel without having to listen to it.** Her ideal was that she and her friend could be linked, and whenever they wanted they could give each other a **squeeze**.



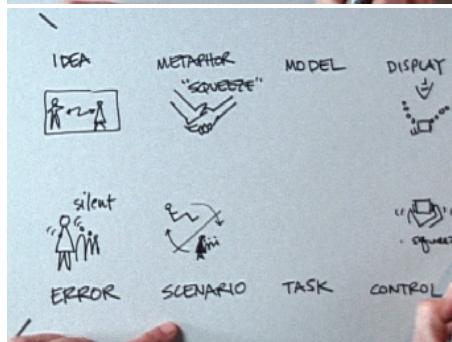
## 2. Metaphor and scenario

**The metaphor is that** she really wants something that is more like holding hands, so that **she can hold hands at a distance and give a squeeze**. She is thinking about a scenario that Fred is at home, and she is taking longer shopping than she had expected. She wants to be able to communicate without interrupting whatever he is doing.



## 3. Model and task

One conceptual model is that she is connected to him and no one else, so that when she squeezes, it is him that is going to feel it. Another is that **when she sends off the squeeze, it is a packet, so she is not squeezing him at that moment**. One task is to set up the connections, and another is to trigger the squeeze, both for sending and receiving.



## 4. Display and control

The idea for the display is that she could wear some kind of necklace that could vibrate, like the vibrating mode in pagers and cell phones, to let her know that someone is calling. **The sending could be controlled by a squeeze**. It turns out that Celine works for a company that makes a handheld computer operated by a stylus, so she uses this to set up the connection.

BILL VERPLANK HAS made connections among people of diverse backgrounds who are trying to understand how to design interactions. He shows his expertise in computer science, grasp of human factors issues, and ability to engage an audience—both with words and drawings—as he explains a concept. Bill helped to move the desktop toward the PC when he worked on the design of Xerox Star graphical user interface, and has helped designers in many different situations since then.

Cordell Ratzlaff,<sup>14</sup> whose interview follows, has been responsible for pushing the design of the operating systems for personal computers to the limits of progress to date. He has taken the desktop from close to the place that Bill Atkinson and Larry Tesler left it to Mac OS X, so far the most advanced graphical user interface there is.





Cordell Ratzlaff

"The big difference with Mac OS X was that it was completely design driven, based on what we thought novice users would need in an operating system," says Cordell Ratzlaff, who managed the Human Interface Group for the design of Macintosh System Software at Apple for five years. He led the team that designed the versions of Mac OS from Mac OS 8 all the way through Mac OS X. Cordell graduated with a degree in psychology from the University of Nebraska. Passionate about the study of human behavior, he eventually became fascinated by technology; he moved into design to combine the two interests. After graduate studies at Ohio State University in industrial engineering, his first job was at the NASA Ames Research Center. Next he designed user interfaces for telecommunication systems at an aerospace company. He ended up at Apple in 1990, in the group that was designing networking and communication products for LAN-based networks. In 1994 he began work on system software at Apple and led the design of four major versions of the operating system, culminating in the innovative Mac OS X. When the design of the new interface was completed and just before it was shown publicly, he went to a startup called GetThere.com to head up the design and development of online travel reservation sites, staying with them until they were established as market leaders in corporate online travel, had gone public, and been acquired by a larger company. He is now creative director for the Digital Media Group at Frog Design, in San Francisco.



# Cordell Ratzlaff

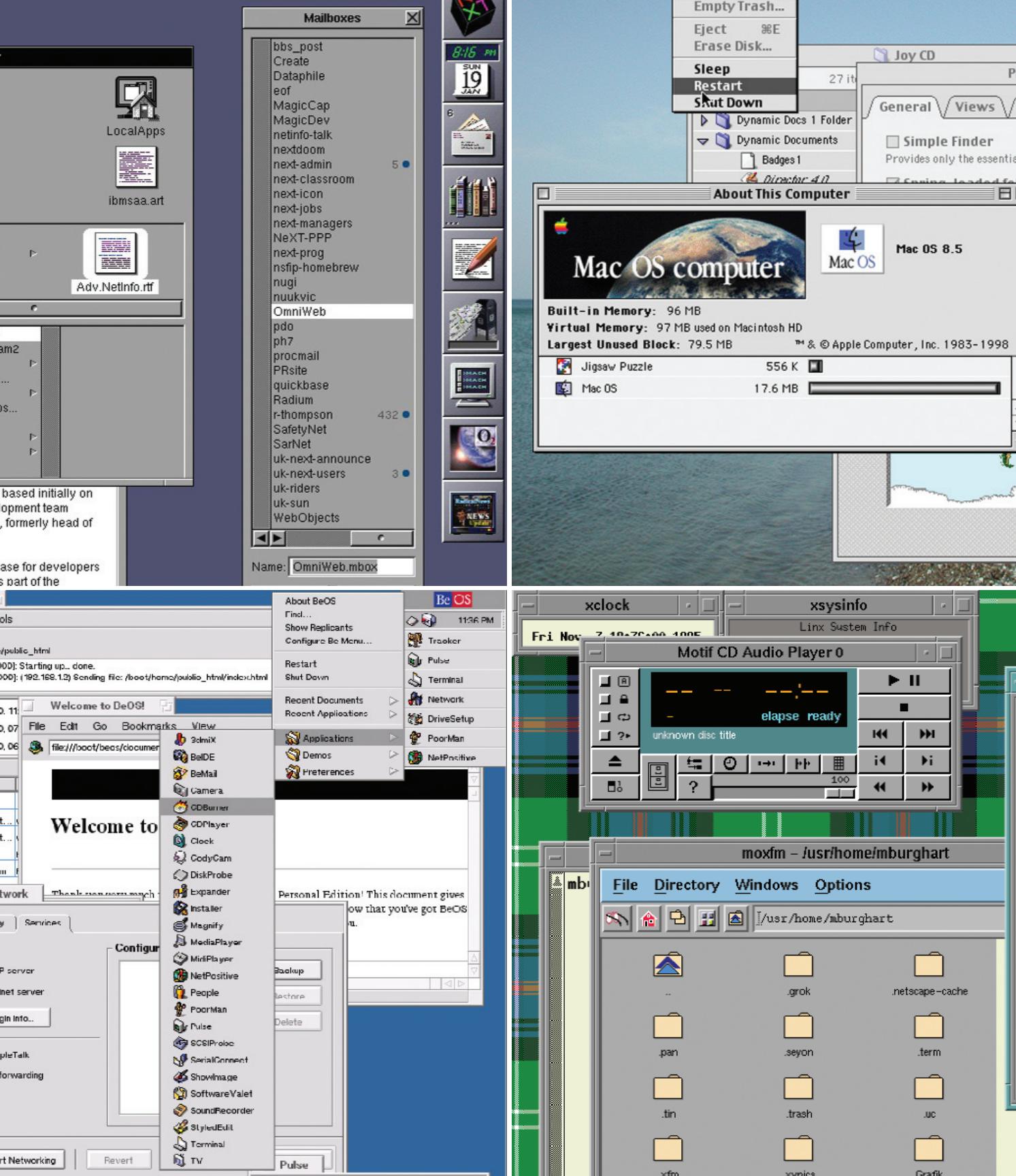
## Mac OS X

- Mac OS X

*Photo*  
Author's screen  
capture

WHEN CORDELL RATZLAFF arrived in 1990, Apple was a great place to work. The good times were rolling. The company had a bigger market share than IBM at 26 percent and large enough profit margins to sponsor generous development budgets—and some good-time extravagance. It was exciting to be a designer at Apple because you had a lot of freedom to explore ideas without much interference. The development organization was fractured into separate teams, with seven or eight different human interface groups.

This situation only lasted a few years before Apple seemed to lose its identity. They started playing catch-up with me-too products, while Windows was becoming more and more popular and other companies were exploiting the Internet. Apple was chasing after too many different things—Newton, digital cameras, printers, projection systems, and other products that never made it to the market. That was when Cordell moved into system software development and took responsibility for the Human Interface Group for Mac OS.



His first project was Mac OS 8, which allowed people to customize the interface more than system 7 and also had richer graphics with more color. For Mac OS 8.5 he took customization further by adding themes. People could choose from a number of prepackaged themes, but they could also change their desktop pattern, color scheme, and the way some of the controls worked.

For Mac OS 8.5 his team designed sounds to enhance the interface. Until that time, the attempts to integrate sound<sup>15</sup> had failed, because people just took sounds and attached them to actions or behaviors, rather than designing them as integral elements of the interactivity. Cordell brought in Earwax Productions,<sup>16</sup> a sound design company, and worked with them to study the events, objects in the interface, gestures, activities—everything that people were trying to do as they interacted with the user interface elements on the screen. They discovered that when people hear the same sound repeatedly, it soon gets annoying. They coordinated the tempo and duration of the sounds with the activity, while varying a number of different parameters, so that there was an element of randomness.

- Operating system user interfaces;  
*clockwise from top left:*  
NeXTSTEP, Mac OS 8.5, Motif, Be OS

In the mid nineties Apple had a project called Copeland, which went on for a number of years. It was meant to be the next-generation operating system for the Macintosh. Eventually everyone realized that Copeland was never going to come to fruition, so Apple started looking for an operating system to buy from outside the company. One candidate was NeXTSTEP, another was Be OS, and a third was Motif on X Windows. They even considered a Java operating system. They eventually decided on NeXTSTEP because it had been out for a number of years and had rock-solid reliability. That decision led to the purchase of NeXT, and Cordell and his team<sup>17</sup> set about integrating NeXTSTEP into Mac OS.

Initially the strategy was to keep the user interface unchanged. They were hoping to take the technology that came with NeXTSTEP and swap it out with Mac OS technology, with the user interface remaining the same so that nobody would ever know that there was different code running underneath. It soon became apparent that it would be impossible to exactly replicate

the Macintosh human interface on top of the NeXTSTEP architecture. For individual applications, like a media player, you can separate the interface from the roots, but for operating systems a lot of the behaviors go right down through the layers to the core of the technology. Cordell remembers his dilemma:

We discovered we could only get 95 percent of the way there, which was the worst possible situation. We were going to end up with something that was almost—but not quite—like the Mac OS, and we knew that was going to be a problem. On the other hand, we saw a lot of interesting capabilities in the NeXTSTEP technology that could be exploited to enhance the Mac OS user experience. I took a couple of designers, and we started exploring some of these ideas—for example, using NeXTSTEP’s advanced graphics system to introduce translucency and real-time shadowing into the user interface.

A few months into this exploration, we had a two-day off-site with the engineering managers from just about every group within Apple. We got together to discuss how we were going to pull off the NeXTSTEP integration—what resources we would need, what the interdependencies were between groups, and to work out how to actually achieve and ship Mac OS X. During the two days, each team was getting up and talking about their particular part of Mac OS X, what their challenges were, and what their needs were from other teams.

I was asked to show some of the new user interface ideas we had, to demonstrate that we were looking beyond just the near-term engineering task that we had in front of us. I was the last person to present at the end of two days, and I got up and showed things like translucent menus, gel-like buttons, and composite shadowing. When I described what we wanted to do and started talking about the technical requirements and what we would need to make that happen, there was literally laughter in the room. We had just gone through two days of realizing how we had this huge mountain ahead of us that we would have to climb, and I was just adding to that. It didn’t go over too well and I left that meeting very depressed.

At that time Apple was a very engineering-driven company. A lot of the products were designed to fit easily into the technical possibilities, so they were easier to develop and to release on time, but they did not necessarily have to be easy or enjoyable to use.



- Mac OS 8.5—themes
- Mac OS 8.5—sounds
- Mac OS 9.2

Then Steve Jobs came back and applied a ruthless rationalization and a much-needed focus for the company. The “Think different” ad campaign communicated the change brilliantly; before it, Apple was scorned for being different from everybody else, but the campaign turned that completely around to say “Being different is good!” When your products are great, they are not for everybody: they are designed for people who are the best in the world. Still, Cordell was understandably nervous:

A couple of weeks later I got a call from Steve Jobs’s admin, saying, “Steve wants to talk to you. He hears you’re heading up the Human Interface Group at Apple, and he has some things he wants to discuss with you.”

I thought, “Oh, great!” This was a time when Steve was reviewing everything that was going on at Apple, projects were getting cut, and people were leaving as a result. I took a couple of our designers, and we went to meet with Steve.

We were sitting in his boardroom, and he comes in, late as usual, and the first words out of his mouth were, “You guys designed Mac OS, huh?”

I said, “Yeah, we did that.”

“I’ve got to tell you, you’re a bunch of amateurs!”

For twenty minutes he just railed on about how bad the Macintosh human interface was and how we were idiots for designing it that way.

Steve hadn’t seen the new work that we had done, so we started talking about some of the ideas that we had come up with, and we were able to turn that conversation around into something more productive. We discussed what was broken in the current human interface and what we could do to fix it. About halfway through that meeting, I thought, “Well, he wouldn’t be wasting his time talking to us about this if he was going to fire us, so I guess we still have our jobs!”

We were able to actually have a pretty good meeting and listed a number of ideas that we wanted to look at some more. At the end Steve said, “Why don’t you go work on these things? Come back in a couple of weeks, and let’s see what you’ve got.”

The next few weeks we worked night and day, building prototypes and fleshing out design ideas. We spent an entire afternoon showing them to Steve, and he was blown away by what he saw. From that



Cordell Ratzlaff at Apple ■  
Steve Jobs ■  
*Photo by Gary Parker, courtesy of Apple*

point on there was never any debate that we were going to put a new human interface on top of Mac OS X.

The designers at Apple, for both physical products and software, find it rewarding to work with Steve Jobs. It may be much more chaotic and challenging, but his leadership encourages them to do their best work. He focuses intensely on design and on building products around his vision of what Apple customers want. He made big changes in the way the development community went about designing, developing, and releasing products:

We did the design first. We focused on what we thought people would need and want, and how they would interact with their computer. We made sure we got that right, and then we went and figured out how to achieve it technically. In a lot of cases when we came up with a design that we knew really worked for people, we didn't know how we were going to build it. We had a design target, and we worked with engineering to reach it. We ended up doing a lot of things that we initially thought were impossible, or would take a long time to do. It was great because we were applying a lot of creativity and ingenuity on the design side and then pushing the engineers to use the same kind of creativity and innovation to make that happen.

Steve Jobs made a choice to focus on the consumer market, and that helped Cordell decide how to simplify the design. Apple has a fanatic fan-base of developers and power users, who hate to lose features that they have grown used to, so Mac OS X initially met with some very strong criticism and resistance. Interaction designers are often not able to stand up to that kind of pressure, and it took firm leadership from the top to allow Cordell and his team to focus on the new customers who did not have much computer experience.

In Mac OS 8.5 there were no less than seven different ways to manage windows. For Mac OS X they created a simplified approach to window management. They eliminated window frames, replacing the heavy structure with a delicate shadow. In 1997 if you looked at operating systems from an aesthetic point of view, whether it was Mac OS 8.5, Windows 95, or



■ Mac OS 8.5 Windows

NeXTSTEP, their appearance was chiseled and beveled. Everything was rectilinear, with quasi-3D shading on the controls and windows. On the interaction side, a lot of the interface elements had crept up and taken over the experience, with thick window frames, big chunky beveled buttons, and big icons. It seemed that the interface elements and controls had become more important than what you really wanted to do with the computer, which was either to look at content, like the Web, or create content, like writing a note or an email.

Cordell saw a great opportunity to change to an appearance that was fresh and fun, in contrast to the existing state of the art. He decided to change from gloomy, square, and beveled, to light, fun, and colorful, with a very fluid expression. He asked, What's the opposite of a computer interface? He came up with things like candy, liquor, and liquids, to inspire a new visual design of the interface. The designers collected magazine ads for liquor, with delicious looking liquids in glasses with ice cubes, sparkling with reflections and highlights. Around this time Apple came out with the original iMac, and translucency was a big feature of the design, so they tried to design an interface that complemented the industrial design:

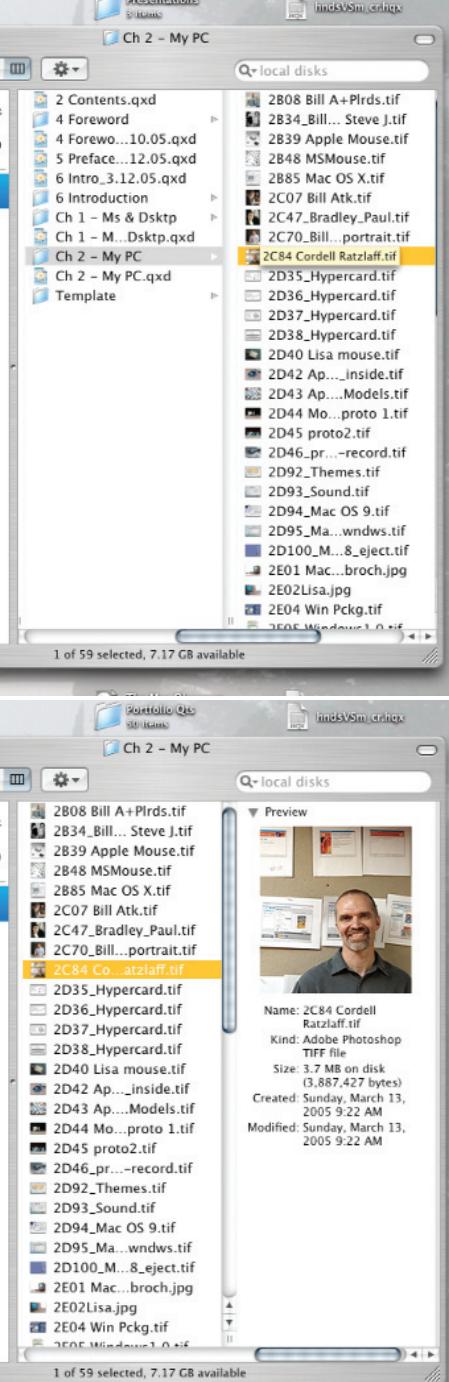
We didn't want to use translucency gratuitously; we wanted to make sure that translucency added value to the experience. A good example of that was window layering. As windows move to the background they become translucent; in fact they become more translucent the farther back they are in the window stack. The translucency not only tied the interface into the industrial design of the product, but also showed off the graphics capabilities of Mac OS X, and provided a useful clue about window ordering.

When Cordell was designing Mac OS X, he was looking for some simple interactive features that were also powerful. He did not want to take away any of the capabilities or functionality of the OS, but to clean out a lot of the detritus that had accumulated over the years. One of these features was the addition of the “column view” in the Finder, which made it easier to manage access to folders and files. The view allows users to navigate deep into the hierarchical file system within a single window. It's much



was a great place to work  
re than IBM at 26%, and 1  
, and some good-time ex  
lot of freedom t System Preferenc

Translucent iMacs ■  
photo courtesy of Apple ■  
Translucency used to show ■  
window ordering ■



- Mac OS X—column view—list
- Mac OS X—column view—thumbnail

quicker, allows for easier backtracking, and doesn't leave the desktop cluttered with extraneous windows that the user needs to clean up later.

Mac OS X is an evolutionary step forward in the series of graphical user interfaces that originated with the desktop and windows metaphors, but it feels like a design for computer interaction, rather than a metaphorical connection to the familiar world of physical objects in offices. We still use words like window, desktop, and folder, but the appearance and behavior of the designs have evolved to a level where they communicate their own attributes rather than the characteristics of a throwback to a physical world.

Cordell believes that design should be driven first by user needs and desires:

There is nothing that I would not consider changing; I think an interface really has to be appropriate for the people who are using it, and the task that they are performing. People don't use a computer to enjoy the operating system; they don't care about setting their system preferences, nor do they care about choosing what kind of scrollbars they want. They use a computer because they want to create something; they want to communicate with somebody; they want to express their own personality, everything from writing a novel to balancing their checkbook—some more fun than others, but it's all about accomplishing something that really doesn't have anything to do with using a computer. The computer is just a tool.

As interaction designers, we need to remember that it is not about the interface, it's about what people want to do! To come up with great designs, you need to know who those people are and what they are really trying to accomplish.

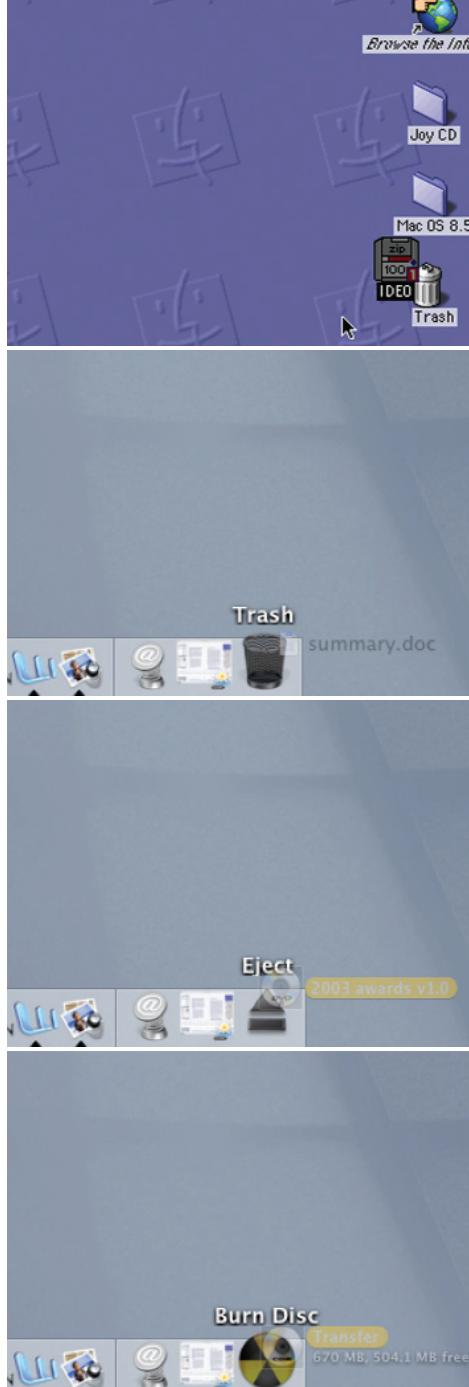
# The Time Dimension

THE DESIGN TEAM developed the concept of a time-based operating system, using not only the two dimensions that you have on the screen, but also the third dimension of time. For example, everybody's favorite complaint about the Macintosh human interface was that dragging a disk icon to the trash would eject it. That confused new users, because normally if you had a file that you wanted to delete, you would drag it to the trash and it would be gone; it seemed so wrong to make the same gesture eject a very valuable disk, full of information. In Mac OS X, depending on what you are doing at the time, different options are available to you. When you select an object that can be ejected, the trashcan changes to an eject icon, making sure that the action and what was showing on the screen are appropriate to the context. Similarly, when you select an empty CD or DVD disk, the trashcan changes to a "burn" icon.

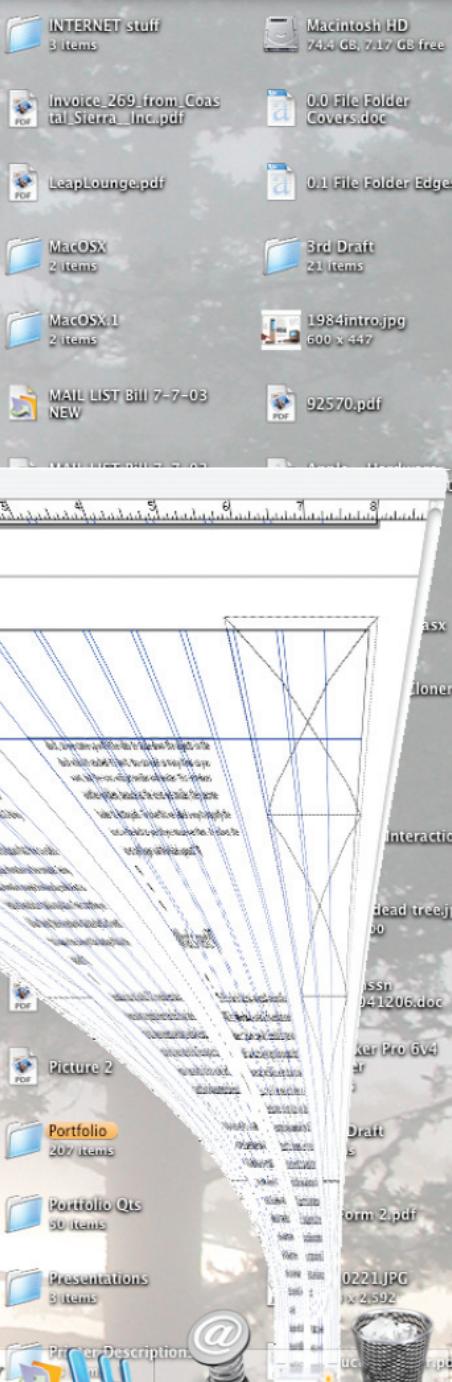
In previous versions of Mac OS, when you selected an application, all of that application's windows came to the forefront. In Mac OS X, those windows appear in the order that you used them. If, for example, you are switching from Photoshop to an HTML editor, the Photoshop window is only one window behind; the system does not bring all of the other HTML editor windows to the front. This keeps the windows ordered in a way that people use them. As long as you can access the close button, even if it is behind several other windows, you can just click on it and it is gone.

The principle of the time-based interface was built on some of the work that had been done in the Apple Advanced Technology Group (ATG) many years before:

There was a project in ATG called Rosebud that explored how the dimension of time could be used to help people find things. This was based on an insight that people sometimes associate information with chronological events or sequences. For example, people may not remember where they saved a particular document on their computer but know that the last time they modified it was two days ago. Or



- Mac OS 8—eject via trash ■
- Mac OS X—trash ■
- Mac OS X—eject ■
- Mac OS X—burn ■



■ Mac OS X—animation to minimize a window

they may remember reading or creating information in relation to other events happening in their lives at that time.

What we found was that a lot of people thought about the content on their computer in terms of time, “When was the last time I read this? When was the last time I changed this? What else was I doing the last time I opened this document?”

Thinking about this, we came to the conclusion that time is a useful way for people to organize information on their computer. Seeing objects and information arranged chronologically could be just as helpful as hierarchical or spatial organization. Time is something we all flow through; there are people who associate objects with events that happen in time, so why not take advantage of that in the interface?

Animation is also used in Mac OS X to provide hints about where an object is going and where it came from. You see that in the dock, when you minimize a window, the window will morph into the place where it rests in the dock. When you restore that window it comes out of the dock in the same way. Cordell explains that the dock was the answer to more than one problem:

We worked on the dock for a long time. It was a way to consolidate a lot of the things that people were using in previous versions of Mac OS. We knew that people wanted an easy way to launch applications and documents. We knew people wanted an easy way to switch between applications. We knew people needed a way to manage their windows and minimize them, so that they could get back to them very quickly. There were several different components in previous versions of Mac OS that allowed people to do that, but they were scattered over various parts of the interface. We were looking for one consolidated, consistent way to combine and simplify them. The dock was a way to achieve that.

One of the biggest challenges we had with the dock was working with a finite screen size. You can keep adding things to the dock, but at some point you run out of space. That’s where animation comes in. As more and more things are added to the dock, it spreads out. What happens when the dock expands to the edge of the screen? You can’t add any more things; you’ve got to take something out!

We didn’t want to limit the number of items people could put in the dock, so we came up with the idea to scale down the objects in

the dock when it reached its limit. You can add as many items as you want, but the icons will get smaller and smaller. This introduces another problem, because as the icons get smaller, they become harder to distinguish. To solve this we added a way to magnify the icons in the dock as you roll your mouse over them. On rollover, the icons get bigger and their label appears.

## What's Next?

THE FINDER FOR the Macintosh was originally developed for a 400K disk with no hard drive, so that all of your applications and documents could live on one disk. A hierarchical file system was added later, when the disk got bigger, to help you organize things in a structured way. Hard drives are huge today, and you have access to your local area network and the Internet, so a hierarchical browser is no longer adequate. In the future, operating systems for personal computers are likely to have “find” functionality that is better integrated. Google<sup>18</sup> uses search techniques that work much better than hierarchical structures would, for the almost infinitely large information source of the Web, but similar methods could also be applied to information on a PC, or local network. Operating systems are likely to be more active in organizing themselves and presenting information to the users when they want it, in a way that suits them. Cordell believes that at some point the whole desktop metaphor is going to go away:

We've got this metaphor that's based on real world objects—like documents, folders, and a trash can—which is good, but it's becoming obsolete. For example, my two daughters, ages eleven and seven, have been using a computer since they were one or two years old. Their first experience with a folder came to them from the computer, not a manila file folder that sits on a real desk. For them, the concept of a folder as a container didn't come from a real-world object; it came from something they saw on a computer. Similarly, the whole application/document model was developed for people who



Mac OS X—dock icons magnifying ■

The screenshot shows a search result for the book "Revolution in The Valley" by Andy Hertzfeld. The page includes the book cover, price (\$16.47), availability, and a "Proceed to Checkout" button. Below the main listing, there's a sidebar with related books and a shopping cart summary.

- Amazon purchase—finding a book
- Amazon purchase—seeing the details
- Amazon purchase—checkout
- Amazon purchase—shipping

were creating content; what most people do today with computers is consume content. If you look at the trend of more people wanting to access information rather than produce information, that could call for a completely different metaphor. I'm not sure what that is yet.

The metaphor of the desktop has already been replaced by the metaphor of the page for browsing the Web. Although you still start from your desktop when you use a PC, as soon as you open a Web browser, you are moving away from file folders toward sites made of pages, with pieces of information linked to each other by cross references; this feels different from documents stacked in folders, and folders collected in volumes. The desk is being invaded by loosely connected pages, which float and flutter in infinite free space, making it very hard to keep tidy!

Transactions are also leaving the desktop. If you go to Amazon.com to buy a book, you step through the interactions page by page. That is different from the desktop and also very different from the experience you have when you go to a physical bookstore. Cordell hopes that future interactive technologies will make the Web a much richer experience:

Maybe it's an agent model, where you just tell your computer, "This is what I want, this is the price I'm willing to pay for it. Go off and do it, and come back to me when you've got it."

You could think of your computer more as a personal assistant that helps you do things. You give it high-level directions, it goes off and does it and comes back to you with the results. Knowledge Navigator<sup>19</sup> was a great attempt to visualize the agent-based interface. I wish someone would build it. So far, all efforts to develop something like that have failed miserably. They come off as either an idiot-savant, who does some things brilliantly but flounders at others; a four-year-old child, who tries to help but only gets in the way; or an amnesiac, who completely forgets what you told it to do the last time.

For agents like the Knowledge Navigator to really work, they've got to be right 99 percent of the time, because you're not going to put up with something that misreads your intentions or interrupts with inane questions. It's a level of trust: "Do I really trust this thing to do the right thing for me?" It doesn't take too many instances of

an agent suggesting something completely out of context, or doing something that you really didn't want it to do, to make you lose confidence in its ability.

Maybe the fallacy of the Knowledge Navigator is having one agent that can do everything for you. Perhaps I need a book-buying agent that's great at going out and finding exactly the book I want, for the lowest price, and it can get it to me the fastest, and it can also recommend other books that might interest me. It just does one thing and it does it very well.

I don't think we are ever going to get computers to replicate the creativity, flexibility, and irrationality of humans. I think that's the great thing about being human!

