1

The Mouse and the Desktop

Interviews with Doug Engelbart, Stu Card, Tim Mott, and Larry Tesler

When you were interacting considerably with the screen, you needed some sort of device to select objects on the screen, to tell the computer that you wanted to do something with them.

Douglas C. Engelbart, 2003, referring to 1964

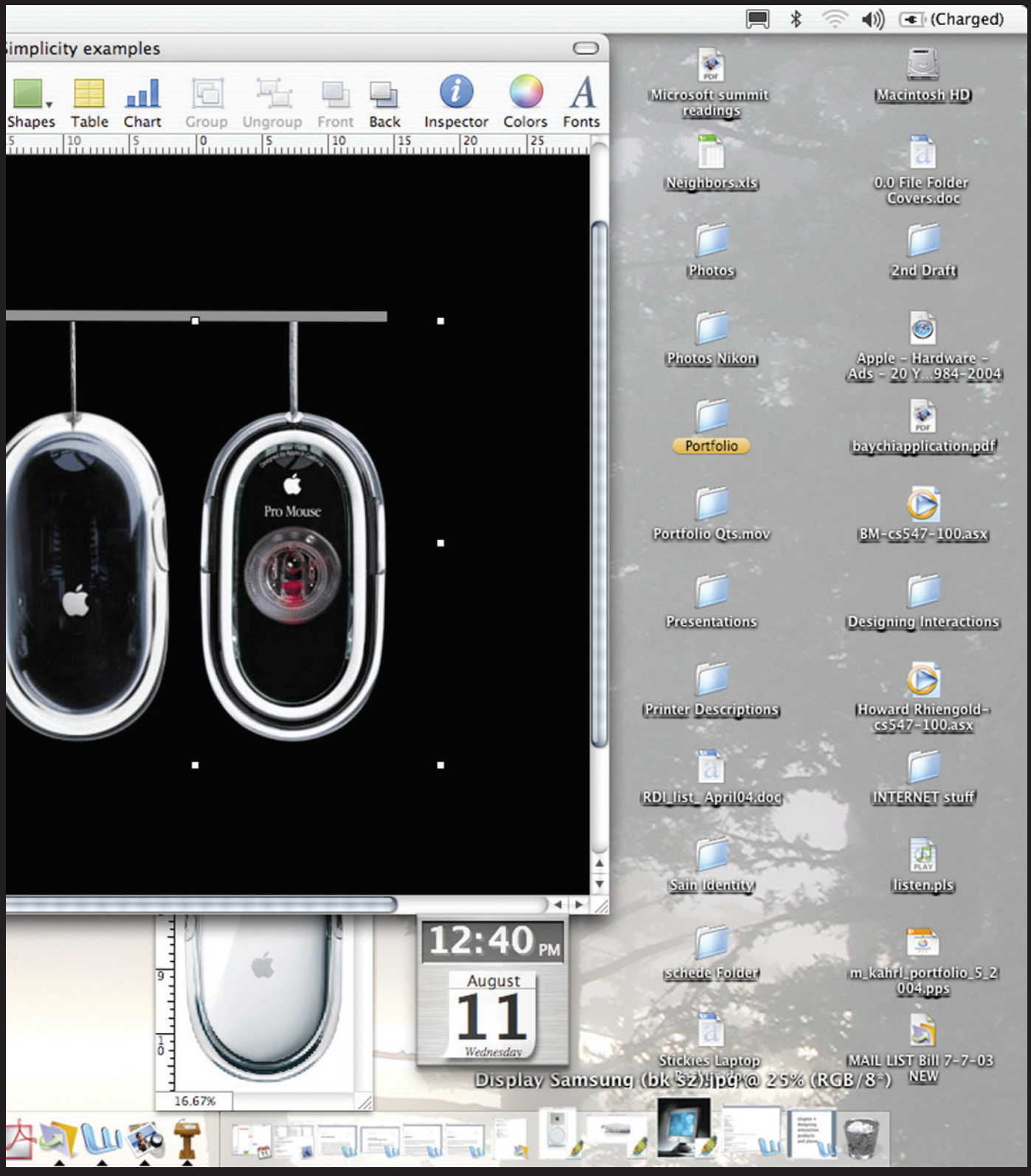
Why a Mouse?

Apple mouse
2002

Photo
Courtesy of Apple

WHO WOULD CHOOSE to point, steer, and draw with a blob of plastic as big and clumsy as a bar of soap? We spent all those years learning to write and draw with pencils, pens, and brushes. Sharpen the pencil to a fine point and you can create an image with the most delicate shapes and write in the tiniest letters; it's not so easy to do that with a mouse.

Doug Engelbart[1] tells the story of how he invented the mouse. When he was a student, he was measuring the area under some complex-shaped curves, using a device with wheels that would roll in one direction and slide sideways in the axis at ninety degrees. He was bored at a conference, and wrote in his notebook about putting two wheels at right angles to track movement on a plane. Years later, when he was searching for a device to select objects on a computer screen, he remembered those notes, and together with Bill English, he built the first mouse. We use the mouse not just because Doug Engelbart invented it, but because it turned out to be the pointing device that performed best for


(Charged)
Simplicity examples
Shapes
Table
Chart
Group
Ungroup
Front
Back
Inspector
Colors
Fonts
Pro Mouse
Microsoft summit readings
Macintosh HD
Neighbors.xls
0.0 File Folder Covers.doc
Photos
2nd Draft
Photos Nikon
Apple - Hardware - Ads - 20 Y...984-2004
Portfolio
baychiapplication.pdf
Portfolio Qts.mov
BM-cs547-100.asx
Presentations
Designing Interactions
Printer Descriptions
Howard Rhiengold-cs547-100.asx
RDI_list_ April04.doc
INTERNET stuff
Sain Identity
listen.pls
schede Folder
m_kahrl_portfolio_5_2004.pps
Stickies Laptop
MAIL LIST Bill 7-7-03 NEW
12:40 PM
August
11
Wednesday
16.67%
Display Samsung (bk sz).jpg @ 25% (RGB/8*)

pointing and clicking on a display, outperforming light pens, cursor keys, joysticks, trackballs, and everything else that was tried in early tests with users. The mouse won because it was the easiest to use.

We understand the reasons for the triumph of the mouse much more clearly from the story of developing the early designs told by Stu Card,[2] who joined Xerox Palo Alto Research Center (PARC) in 1974 and has spent much of his time there perfecting scientific methods to integrate with creative design. He has developed a process to predict the behavior of a proposed design, using task analysis, approximation, and calculation. His idea is to accelerate the movement through the design space by a partnership between designers and scientists, by providing a science that supports design. He tells the story of applying this science to the development of the mouse.

Why a Desktop?

■ Apple Mac OSX desktop with images of Apple mouse

Photo
Author's screen capture

IT SEEMS SURPRISING to find a "desktop" on the spherical surface of a glowing glass display, housed in a bulky plastic box that in itself takes up half your desk. Or is it on the cramped flat screen of your laptop? Who came up with that idea? What were they thinking about, and why did they choose to design a desktop rather than a floor, or a playing field, or a meadow, or a river? Why does this desktop have windows in it? You usually think of windows being on the wall, not all over the surface of your desk. Why does it have a trashcan on it? It would seem more natural to put the trashcan on the floor.

In 1974 Tim Mott[3] was an outsider at Xerox PARC, working for a Xerox subsidiary on the design of a publishing system. He describes how the idea of a desktop came to him as part of an "office schematic" that would allow people to manipulate entire documents, grabbing them with a mouse and moving them around a representation of an office on the screen. They could drop them into a file cabinet or trashcan, or onto a printer. One of the objects in the office was a desktop, with a calendar and clock on it, plus in- and out-baskets for electronic mail.

There were lots of other people at Xerox PARC at that time thinking about desktops and other metaphors for use in the design of graphical user interfaces (GUIs), but Tim was working most closely with Larry Tesler,[4] and the two of them worked out processes for understanding users by talking to them, using guided fantasies, participatory design, and usability testing. Larry describes how he developed these processes and how icons arrived on the desktop. Larry insisted on simplicity and designed interactions that were easy to learn as well as easy to use. He went on to Apple and formed another partnership in the development of the desktop, working with Bill Atkinson[5] to create the designs for Lisa, including the pull-down menus, dialog boxes, and the one-button mouse. These ideas stayed in place as the user's conceptual model for the Macintosh and all of the GUIs that followed, stretching the desktop metaphor almost beyond the breaking point.

NLS, Alto, and Star

When Doug Engelbart invented the mouse, he arrived at the dominant design for input devices in a single leap from the light pen, and the development of the mouse since has been more in the nature of evolution than revolution.[6] Engelbart also invented the point-and-click text editor for the NLS system (oNLine System) that he developed at the Stanford Research Institute (SRI), and that system migrated with members of his design team to the fledgling Xerox PARC and became the foundation of the Alto, the first computer with a GUI. In the versions of NLS that were built at PARC for the Alto, the text-editing demonstrations were impressively fast, with a clattering of keystrokes that sounded businesslike and productive. Direct manipulation made it so easy to pick things up and move them that people would often find an instance of a word they wanted in the text and move it into place, instead of typing it. For programmers, this was a wonderful interaction, but the patience needed to acquire the skills proved a

fatal barrier for novice consumers when computers became accessible to ordinary people. It took the influence of Larry Tesler and Tim Mott to create a text editor and page layout design system that was really easy to use, and this was based on rigorous user testing and rapid iterative prototyping. They came close to the desktop metaphor that survives today.

One of the major innovations of the Alto was the bitmap display, making it easy to show graphics. You could make any dot be black or white, allowing you to put any image, whether a font or a picture, onto the whole screen. The memory to do that for a full-page display was exorbitantly expensive at the time, but it meant that you could translate the tradition of graphic and typographic design to the computer. Earlier computers had primitive and pixelated type fonts, but the bitmap display meant that what was on the display of the computer could look exactly like a book, except that it was only 72 pixels per inch instead of about three times as high a resolution for the printed page. A white background and dark text was used on the screen, so that the displayed image was like the printed result, and the screen size was chosen to be exactly the same as a page of text, enabling the concept of WYSIWYG (What You See Is What You Get).

The idea of the desktop was floating around Xerox PARC as part of the communal consciousness. David Canfield Smith had devised an office metaphor as part of his PhD thesis, "Pygmalion: A Creative Programming Environment," published in 1975. His icons looked like mathematical symbols and boxes that you could type into but defined the characteristics of the icons that have become commonplace. Here is his explanation:

> The entities with which one programs in Pygmalion are what I call "icons." An icon is a graphic entity that has meaning both as a visual image and as a machine object. Icons control the execution of computer programs, because they have code and data associated with them, as well as their images on the screen. This distinguishes icons from, say, lines and rectangles in a drawing program, which have no such semantics. Pygmalion is the origin of the concept of icons as it now appears in graphical user interfaces on personal computers. After completing my thesis, I joined Xerox's "Star" computer project. The

> first thing I did was recast the programmer-oriented icons of Pygmalion into office-oriented ones representing documents, folders, file cabinets, mailboxes, telephones, wastebaskets, etc. These icons have both data (e.g., document icons contain text) and behavior (e.g., when a document icon is dropped on a folder icon, the folder stores the document in the file system). This idea has subsequently been adopted by the entire personal computer and workstation industry.[7]

When Dave Smith had finished his PhD, he was hired to join PARC to work on the Star system. Alan Kay had always talked about the desktop metaphor and he came up with the idea of overlapping windows, from a metaphor of papers on a desktop. "You can have a lot more papers on a desk if they overlap each other, and you can see the corner of one, pull it out and put it on top."

Detailed accounts have been written[8] about how long it took to convince a copier company to commercialize the Alto, but Xerox did make a major effort to develop and market an "Office of the Future" machine. They did a thorough analysis of the business potential to find what the requirements of that machine should be, concluding that people would pay a lot for the Alto technology. Star was conceived in response to this; based on the combination of Smalltalk and Alto, it became a design that fit all of the requirements. Star was a very futuristic machine; when they were asked in the market research surveys, people responded that they would not give up that advanced interactive performance for a much inferior but less expensive machine. After it was launched, the IBM PC came out, and the people who said in the research that they would pay a lot for Star proved to be only willing to pay much less for an inferior interface.

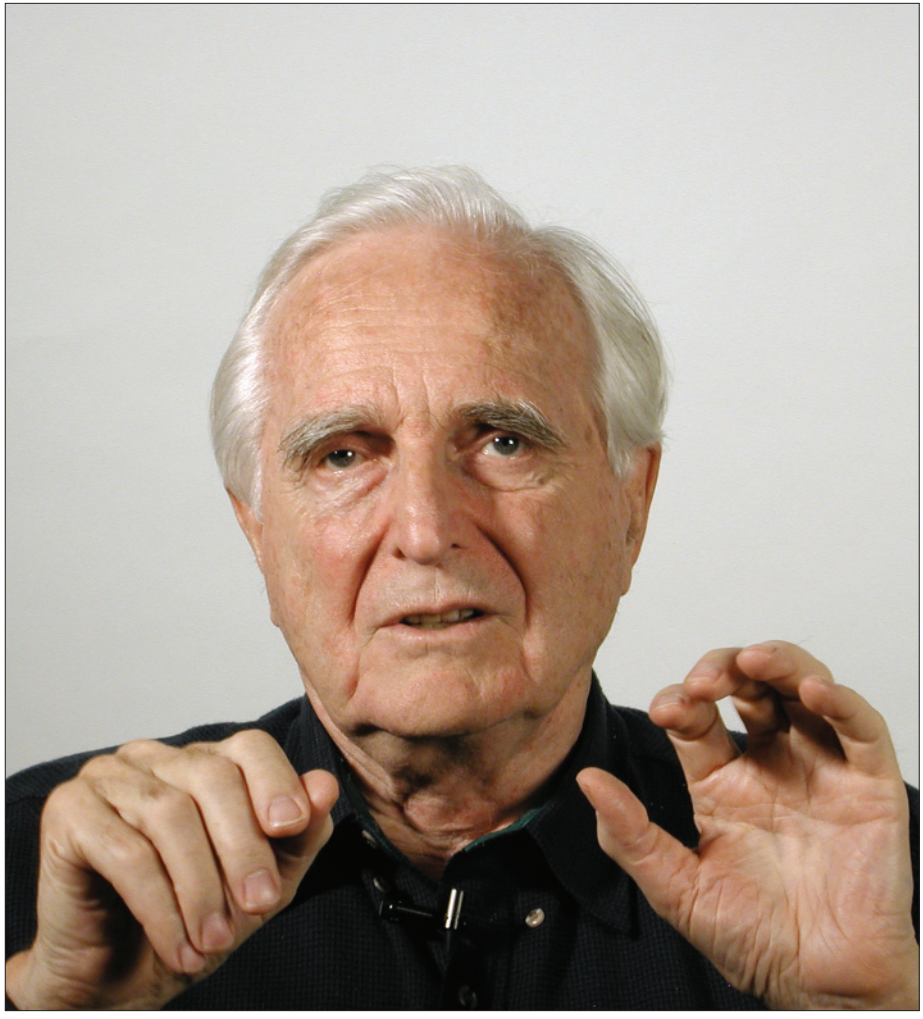
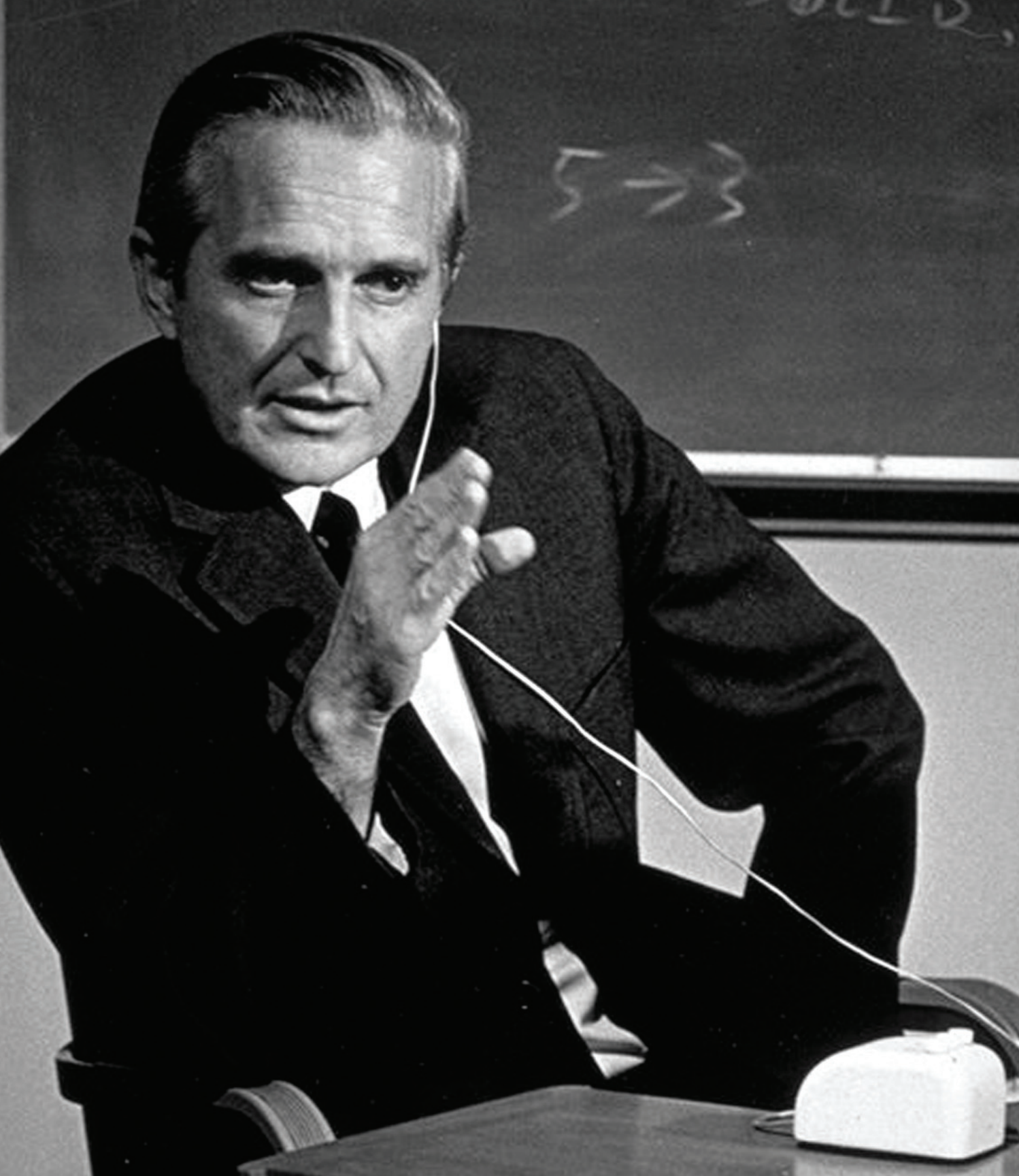


Photo Author

Dr. Douglas C. Engelbart

Doug Engelbart is best known as the inventor of the mouse. At the time of writing, in 2004, at the age of seventy-eight, he is still going to work at his "Bootstrap Alliance," trying to persuade us of the value of his ideas about augmenting the human intellect, and evangelizing the virtues of his "Augment System." His office is located in the headquarters building of Logitech, the world's largest manufacturer of mice. Taking credit as the inventor of such an ubiquitous product is not enough for Doug; in fact, he is charmingly modest about that achievement, preferring to discuss other ideas that have not met with such spectacular success. He has always wanted to create designs that enhance human performance and is not interested in ease of use for the novice. He grew up in Oregon, and his electrical engineering studies were interrupted by service in World War II. During his stint in the Philippines as a naval radar technician, he came across an article by Vannevar Bush in the *Atlantic Monthly*[9] and was inspired to think of a career dedicated to connecting people to knowledge. This idealistic ambition led him to the Ames Laboratory (later NASA Ames Research Center) on the edge of the San Francisco Bay, where he worked on wind tunnel research, and then to the Stanford Research Institute, where he invented the mouse and built up the Augmentation Research Center (ARC) with funding from ARPA. In the early seventies he took several members of his team to Xerox PARC, where he helped put the mouse and the desktop together.



Doug Engelbart

Inventing the Mouse

In 1995 the International World Wide Web Conference committee presented the first SoftQuad Web Award to Dr. Douglas C. Engelbart, to commemorate a lifetime of imagination and achievement and to acknowledge his formative influence on the development of graphical computing, in particular for his invention of the mouse. This is the contribution for which Doug is universally acclaimed. In spite of this, he himself is inclined to give credit to the trackball:

Doug Engelbart conducting a workshop circa 1967–68

Photo Bootstrap Institute

> When I was a senior in electrical engineering, some of the experiments we had to do in the laboratory would end up resulting in funny shaped curves that curled back on themselves, and it was the area under the curve that we were experimenting with. They had an elbow shaped device there, a platform sort of thing that would set down on the table. You would run the pointer around that area, and there was a small wheel next to the pointer, resting on the tabletop, and the other side of the joint there was another one. I couldn't figure out how that would produce the area under the curve.
>
> "I'm not sure myself about the mathematics," said the professor, "but it is a fact that the little wheels will only roll in the axis of the rolling direction, and will slide sideways."

I began to understand that the wheel would roll only as far as you went in the one direction, irrespective of how many sideways movements you made.

I thought of that again one time during a conference on computer graphics, when I was feeling rebellious and bored, so I wrote in my notes about putting two wheels at right angles to each other, so that one would always be measuring how far you went north and south, and the other east and west. It would be easy to convert that into potentiometers and things so that the computer could pick up that signal. It was a very simple idea. At the time I was unaware that that same thing was sitting underneath the tracking ball, and that was how a tracking ball worked. Later on the manufacturers put the wheels against the table, which is exactly like an upside down tracking ball. There should be credit given to the tracking ball, except for my ignorance about it at the time.

It says a great deal about Engelbart's extraordinary modesty that he makes so light of his achievement. It also says a lot about his methodical persistence that he used his moments of boredom at that conference to fill a notebook with ideas and that he remembered what was in that notebook when he was looking for the best input device solutions many years later:

When you were interacting considerably with the screen, you needed some sort of device to select objects on the screen, to tell the computer that you wanted to do something with them. We got some funding in the early sixties, I think it was from NASA, and set up an experimental environment with several different kinds of devices; a tracking ball, a light pen, and things of that sort that were available at the time.

As we were setting up the experiments, I happened to remember some notes that I had made in a pocket notebook some years before, and sketched that out to Bill English, who was the engineer setting up the experiments, and he put one together, with the help of a few draftsmen and machinists. That one was put in the experiments, and happened to be winning all the tests. That became the pointing device for our user interface. Somebody, I can't remember who, attached the name "mouse" to it. You can picture why, because it was an object about this big, and had one button to use for selection, and had a wire running out the back.

■ Graficon experimental pointing device

"It looks like a one-eared mouse!" someone said. Soon all of us just started calling it a mouse.

Thinking of the possible relevance of those orthogonal wheels was the first step; working with Bill English to design an object to contain them was the second. The recognition that this idea might be important for interaction came from the tests that compared the mouse with other possible input devices; it was the people who used it in the tests who proved the point. The designers came up with as many alternatives as they could that seemed plausible, built prototypes and created tasks in the relevant context, and then ran the tests. Here's what Doug says about the testing process:

> We listened to everybody who had strong ideas, and it seem to us worth just testing everything that was available. The light pen had been used by radar operators for years and years, and that seemed to most people would be the most natural way to do it. I couldn't see that, but why argue with them; why not just test and measure? The time it takes to grope for it and lift it up to the screen seemed excessively large, so it didn't do well in the tests.
>
> For the test we had naive users coming in, and we explained everything that would happen so that they weren't surprised. We asked them to put their hands on the keyboard, and all of sudden an array of three-by-three objects would appear at an arbitrary place on the screen, sometimes small objects and sometimes large, and they had to hit a space bar, access the pointing device and go click on it. The computer measured time, overshoot, and any other characteristics we thought were valuable. The assessment just showed the mouse coming out ahead. It was many years later that I heard from Stuart Card, a friend at Xerox PARC, what the human factors explanation was.

There is an objectivity in this process of letting the user decide, the value of which is a recurring theme in this story of designing the desktop and the mouse. Come up with an idea, build a prototype, and try it on the intended users. That has proved, time and time again, to be the best way to create innovative solutions.

First mouse in hand, 1963–64 ■
First mouse ■
First production mouse ■

The Demo that Changed the World

There is a gentle modesty, even diffidence, in the way Doug Engelbart talks, but he holds your attention much more firmly than you would expect from his manner of speech. His passion for philosophy and ideas shines through, with an underlying intensity, almost fanaticism, that is charismatic. He remembers his early motivations:

> My initial framework for thinking about these questions got established in 1951. I had realized that I didn't have any great goals for my career. I was an electrical engineer with an interesting job, recently engaged to be married, but had no picture of the future, and I was embarrassed about this.
>
> "What would be an appropriate career goal for me?" I asked.
>
> "Why don't I design a career that can maximize its benefit to mankind?" I ended up saying.
>
> I was an idealistic country boy. Eventually I realized that the world is getting more complex at an ever more rapid rate, that complex problems have to be dealt with collectively, and that our collective ability for dealing with them is not improving nearly as fast as the complexity is increasing. The best thing I could think of doing was to try and help boost mankind's capability for dealing with complex problems.

By this time he had been working for a couple of years at the Ames Laboratory, in what is now the heart of Silicon Valley but was then still a pleasant agricultural countryside full of orchards. His job researching aerodynamics and wind tunnel testing was interesting and enjoyable, he was engaged to the girl of his dreams, and life might have been good enough; then that idealistic itch to change the world took over, and he started his lifelong search to develop electronic systems that would augment the human intellect. He remembered the "Memex" that Vannevar Bush had described as an "enlarged intimate supplement to a person's memory" that can be consulted with "exceeding speed and flexibility." He felt a kinship for the vision and optimism that Bush communicated and set out to find his own way of realizing an equivalent ambition.

> When I was half way through college, I was drafted for World War II, and had the good fortune to get accepted in a training program that the navy was running for electronic technicians, because the advent of radar and sonar had changed the aspects of navy problems immensely. They had a year-long program which taught me a lot of practical things about electronics and exposed me to the fact that the electronics of radar could put interesting things on the screen, so I just knew that if a computer could punch cards or send information to a printer, then electronics could put anything you want on the screen. If a radar set could respond to operators pushing buttons or cranking cranks, certainly the computer could! There was no question in my mind that the engineering for that would be feasible, so you could interact with a computer and see things on a screen. That intuitive certainty made me change my career path totally to go after it, but I had an extremely difficult time conveying that conviction to anybody else successfully for sixteen years or more.

His first step in that sixteen-year path of dogged determination was to leave his job and go to the graduate school at the University of California at Berkeley, where one of the earliest computers was being constructed. His fixed idea that people should be able to interact with computers directly did not fit with the prevailing view, so he started to get a reputation as an eccentric. Once he had his PhD, he started to look around for a place that would be more accepting of his vision than the UC Berkeley community. He talked to Bill Hewlett and David Packard, but although they were enthusiastic about his ideas, they were determined to focus on laboratory instruments rather than computers.

He finally landed a job at SRI, whose leaders were interested in researching possible uses for computers in both military and civilian applications. He started there at the end of 1957, soon after Sputnik had been launched, and the space race was getting under way. After learning the ropes at SRI for a year and a half, he started to lobby for the opportunity to start his own lab to experiment with new ways of creating and sharing knowledge by combining man and machine. His wish was granted when the U.S. Air Force Office of Scientific Research provided a small grant, and he settled down to the task of articulating his views.

"I wrote a paper that was published in 1962 called 'Augmenting the Human Intellect: A Conceptual Framework'[10] that steered my life from that point forward." In his paper he defined four areas in which human capabilities could be augmented:

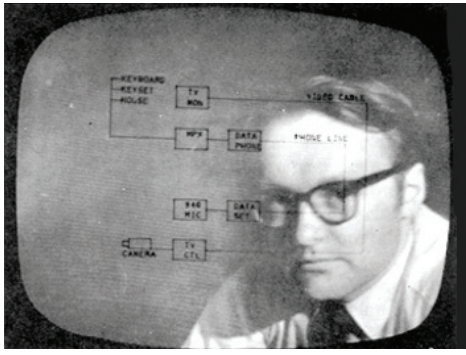
> 1. **Artifacts**—physical objects designed to provide for human comfort, the manipulation of things or materials, and the manipulation of symbols.
>
> 2. **Language**—the way in which the individual classifies the picture of his world into the concepts that his mind uses to model that world, and the symbols that he attaches to those concepts and uses in consciously manipulating the concepts ("thinking").
>
> 3. **Methodology**—the methods, procedures, and strategies with which an individual organizes his goal-centered (problem-solving) activity.
>
> 4. **Training**—the conditioning needed by the individual to bring his skills in using augmentation means 1, 2, and 3 to the point where they are operationally effective.
>
> The system we wish to improve can thus be visualized as comprising a trained human being, together with his artifacts, language, and methodology. The explicit new system we contemplate will involve as artifacts computers and computer-controlled information storage, information handling, and information display devices. The aspects of the conceptual framework that are discussed here are primarily those relating to the individual's ability to make significant use of such equipment in an integrated system.

In this short quote one can see the seeds of triumph and tragedy. The triumph is Doug's powerful vision of a complete system, where people and computers are engaged in a symbiotic relationship for human benefit, working cohesively as an integrated system. From this came the mouse and the other elements of interactive computing that he pioneered. The tragedy is that training is a necessary component of the system. He developed concepts for experts, and the pursuit of the highest capability drove the design criteria; it was therefore inevitable that training would be needed to reach the level of proficiency that

would let people benefit from this capability. That proved a barrier to acceptance by ordinary people, and as computers became less expensive and more accessible, the barrier got in the way more and more.

But we are getting ahead of ourselves in the story. Let's go back to 1964 when, to the surprise of the SRI management, the Defense Advanced Research Projects Agency (DARPA) offered to fund the Augmentation Research Center (ARC) to the tune of half a million dollars a year, as well as providing a new time-sharing computer system, worth another million. Engelbart's energetic lobbying for funding and his flow of papers describing the high-level potential of automation had not been ignored by everyone, so now he had the resources he needed to move from theory to practice. He put together a stellar team of engineers for both hardware and software and set about developing NLS. Bill English was a partner for Doug in much of the work they did, leading the hardware development as the team grew to seventeen people. He joined ARC in 1964 and was the perfect complementary talent, having the technical ability to implement many of the ideas that were expressed by his boss as high-level abstractions. After four years of development, Doug took a chance to show the computer science community what he had been doing:

> For the Fall Joint Computer Conference in 1968, I stuck my neck out and proposed giving a real-time demonstration, if they would give me a whole hour-and-a-half conference session. We had a timesharing computer supporting our laboratory, and small five-inch diagonal screens that were high resolution, but worked by moving the beam around (vector graphics). We put a TV camera in front of the screen and used a TV display for a larger size image. We rented microwave links (from the Menlo Park civic auditorium) up to San Francisco, and borrowed an enormous video projector, three feet by two feet by six feet, to project onto a twenty-foot screen for the audience to see, using a very novel way of converting the video sweep into modulated light.
>
> Bill English, the genius engineer that I worked with (on the mouse also) managed to make all this work. He built a backstage mixing booth, where he could select from four video feeds, one from



Bill English ■

each of the linked displays, one looking at me, and one overhead showing my hands working. He could select from the feeds so you could see a composite image in real time. He had experience doing the stage work for amateur plays, so he was the director. I had written a script for different people to come onto the stage, so he put a speaker in my ear to let me hear his cues: sometimes it was so distracting that I would fumble words.

We were able to show high-resolution links, graphics for schematic diagrams of what was going on, the faces of members of our team in the Menlo Park Laboratory, as well as the screens that they were looking at. We had cursors controlled by two people simultaneously interacting on the screen; one guy started buzzing at my cursor as if in a fight. The audience all stood up and applauded at the end of the demo.

This was the demo that changed the world. The computer science community moved from skepticism to standing ovation in an hour and a half, and the ideas of direct manipulation of a graphical user interface became lodged in the communal consciousness. This was not punched cards and Teletypes. It was something entirely different. Doug sat alone at a console in the middle of the stage, with the twenty-foot screen behind him showing the view from the video feeds. He was wearing a short-sleeved white shirt and a thin tie, with a microphone dangling on his chest, and a headset like an aircraft controller's. The overhead camera showed his right hand using a mouse, seen for the first time by most of the audience, to point and select with; a standard typewriter keyboard in the center and a five-key command pad under his left hand. In his calm but mesmerizing voice, he described and demonstrated an amazing array of functions on the NLS system. Words were manipulated with full-screen text editing, including automatic word wrap, corrections and insertions, formatting, and printing. Documents were planned and formatted using headings and subheadings. Links were demonstrated between one document and another, and collaboration between remote participants was demonstrated in real time:

One of the basic design principles that we started with was that you want to be able to talk about any other knowledge object out there.

You want your links to point in high resolution, for example to a given word in another document, so you want a link addressing string that will let you get there. You also should be able to have optional alternative views.

"I just want to see that one paragraph," I might say.

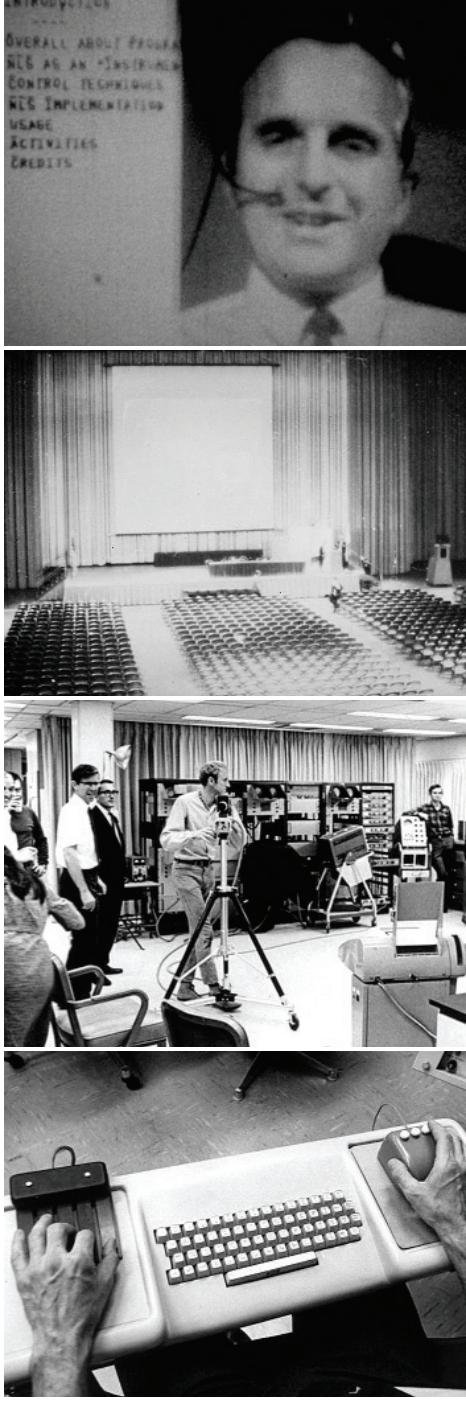
"Okay! When I get there, I'd like to have certain terms highlighted."

"Okay! I'd also like to know who else in my group has links to it, and what they say about it."

I wrote in 1962 that we are all used to the idea that we can diagram sentences, based on the syntactical rules for a properly structured sentence. Now we might want to see a similar set of rules for the structure of an argument, so the computer has a graphic diagram of the argument with nodes connected to other arguments, expressions and statements, and meaningful links connecting them.

The demo was truly amazing, proving that interactive computing could be used for real-time manipulations of information in ways that very few people had imagined before. The assumption that high levels of training would always be acceptable did not get in the way until ordinary people tried to become users of NLS.

The demo also positioned Doug and his band at ARC to receive continuing funding for their research until 1975. His team grew to thirty-five people at one point. In 1969 they were connected to ARPAnet as one of the original nodes of the military research connected network, which eventually developed into the Internet. NLS grew in sophistication and content as time went on but remained essentially the same in concept. In 1971 a group of the best people at ARC, including Bill English, were tempted away from SRI by the opportunities at the new Xerox PARC, where so many exciting things seemed to be about to happen. This was the start of a slide for Doug Engelbart, during which his long-held dreams seemed to have less and less influence. You can feel the frustration behind his words as he describes the determined pursuit of his ideals and bemoans the success of the desktop:



Demo at Joint Computer Conference, 1968 ■

I've been pursuing for fifty years something that required higher and higher levels of capability. "How do you achieve capability," I was

asking, "and when you achieve capability levels as high as you can get, then how do you reduce the learning costs in a reasonable way, but not try to set what I think of as an artificial level of learnability ease, and have to keep your capability enhancements within that level?"

In the business world, I understand, that is awkward to try to do, because you are competing with other people for sales, and people will try computer interfaces that will strike customers as easy to use early on when they purchase them. I don't object to having a difference, but I feel that the world should recognize that there are really high levels of capability there to pursue that will be very important for society to have reached. That's been my pursuit.

Many years ago it became clear to me that what you need to do is develop a basic software structure that will have file designs, capabilities, and properties that the very expert person could use. Then it is easy enough to support the beginner, or pedestrian user, by plugging a very simple user interface with simple operations on the front, but they can both work on the same materials.

Yes, you can point with a GUI, I admit, but our system had an indefinite number of verbs and nouns that you could employ.

There's no way that pointing and clicking at menus can compete with that. You wouldn't want to give someone directions by that limited means.

It is easy to understand the idea of going for the best, of catering to the expert user, and then providing a path to get there from a simple user interface designed for the beginner. In practice, however, this has proved to be the wrong way round, as it's not easy to get something right for the beginner when your design is already controlled by something that is difficult to learn. Look, for example, at the use of the five-key keypad for typing text. Like the stenographer's keyboard used for recording court proceedings, it enables impressive typing speeds when you have been trained long enough to become expert.

Quite a few users adopted the chording key set that I built for myself. You could type any of the characters in the alphabet with one hand, and give commands with the three-button mouse in the other hand at the same time as pointing. This gave a much richer vocabulary and a much more compact way to evoke it than the GUI.

This is how the interactions were designed. On the mouse, one button was to click, another was called *command accept*, and the third was called *command delete*. If you wanted to delete a word, you would hit the middle button on the keypad, which was the letter *d*. It was *d* because it is the fourth letter in the alphabet, and this was a binary coding, 1, 2, 4, 8, 16. If it was the letter *f*, it was the sixth letter, so you'd hit the 2 and the 4 keys at the same time. Then you pointed at and clicked the beginning of the thing you wanted deleted, then you pointed at and clicked the end of the thing you wanted deleted, and if you hadn't made any mistakes, you would hit the *command accept* key on the mouse. It was *d*, point/click, point/click, *command accept*. If you made a mistake at any point, you would hit the *command delete* key to back you up one step.

That process was complicated to learn, and it took a long time for most people to memorize the binary-based, five-finger alphabet. Alternatively, they could invoke commands with keypad, or even the keyboard, the mouse for pointing, and a conventional keyboard in between for typing text. This would have been a very good solution for people with four hands, but is not as fast as the chorded keyboard and three-button mouse, as it takes longer to move your hands to and from the keyboard.

Doug Engelbart strives consistently toward a goal of the best possible performance, and his intuitions and insights have set the scene for the dominance of the mouse and the desktop. His influence has been limited by his decision to design for people as determined and proficient as he is himself, rather than for those who require an easy-to-use system.

Photo Author

Stu Card

At first meeting, Stu Card seems to be a serious person. He looks at you intensely from behind his glasses and speaks in bursts, often veering off on a new tangent of connected thought. You have to concentrate hard to keep pace with him, but when you do, the reward is immediate, as he has thought everything through and arrived at a beautifully balanced view of the whole picture. Occasionally his face breaks into an impish grin, and you see that there is a rich sense of humor under the seriousness. He joined Xerox PARC in 1974, with probably the first-ever degree in human-computer interaction. As a freshman in high school, Stu built his own telescope, grinding the mirror himself. His ambition to be an astronomer led him to study physics, but he was always very interested in computers. In the eighth grade he read navy circuit manuals about how to build flip-flops out of vacuum tubes. His first evening course in computing was to aim the college telescope in the direction defined by a computer program that he wrote. At graduate school at Carnegie Mellon University he had designed his own program, studying computer science, artificial intelligence, and cognitive psychology. After graduation he was offered three jobs; PARC was the most interesting, had the highest salary, and was in California, so it was an easy decision. Doug Engelbart and Bill English had brought the mouse to PARC from SRI, and Stu was assigned to help with the experiments that allowed them to understand the underlying science of the performance of input devices.




XEROX

XEROX

Stu Card

A Supporting Science

Xerox mice; *clockwise from top right:* existing concept 1—flat concept 2—puck concept 3—pen

When Stu joined PARC in 1974, he set out to invent a supporting science for the design of human-computer interactions. This made his position somewhat different from the other researchers. There had been several attempts to develop similar sciences, for example human factors, but that focused too much on the evaluative side, waiting until the structure of the design was already complete and then measuring the result. Stu was more interested in contributing to the design process at the beginning, when the significant choices were still in flux and the science would be able to influence the outcome before much work was complete:

> Newell, who was a consultant at PARC at the time, wanted to try to do a kind of applied psychology. The idea was that computer science is a very asymmetrical discipline. If you take things like time-shared operating systems or programming languages, there is obviously a computer part to them, like how you parse them and do the compiling. There is also obviously a human part; what kind of languages can people program most easily, and what kind of errors do

they make, and how can you make them more efficient? All of the work that had been done in computer science was on the computer side and none on the applied psychology side. Information processing psychology showed real promise as a theory, but there were problems with psychology; it tends to be faddish and impractical and hard to build on.

The idea was that any science worth its salt should have practical applications, allowing you to test the theory in a more pragmatic context than a journal article. It would really have to work, as you were not immune to the hostile reactions of the people that you were trying to do this for. It would be difficult to do this in a university, because universities, especially psychology departments, would not tolerate applied work. In order to do basic research, you had to do it in a place like PARC.

"Design is where all of the action is!" was one of our slogans.

If this was going to work at all, you had to have something that could be used as you were designing. This did not mean that you could do all of design from science. You could have the equivalent role to that of structural engineering in relation to architecture. You could have a technical discipline that would support the design activity. In fact, we had a particularly concrete notion of what kind of supporting science this would be; task analysis, approximation, and calculation. The idea was that you would be able to look at a situation, make zero parameter computations about it, and then say things about what would happen in that situation without running full experiments to see, rather with just occasional experiments to spot check. The idea was that this would give you lots of insight that it would otherwise be hard to have gotten.

"Don't bug us for ten years," we said, "and at the end we'll deliver this to you."

The wonderful thing about PARC was that they said, "Sure." We had to make arguments up front, but once they were made, we got ten years to do this. I've always appreciated the freedom that I got to do this at PARC.

To think that design is where all the action is was very forward-looking at that time, when innovation was usually thought to come from genius or pure scientific research. The core skills of design are synthesis, understanding people, and iterative prototyping. Even now, the idea that these skills are central to innovation is not very widely accepted, but you will see evidence

of their significance again and again throughout this book. The unique contribution that Stu Card made was to create a supporting science, connecting the theoretical underpinnings of research to the pragmatic synthesis of design.

Bill English was Stu's first boss at PARC. Bill and Doug Englebart had coinvented the mouse when they were at SRI, with Doug providing the idea and Bill the engineering development and prototyping. A large part of English's group had come over to PARC from SRI. They wanted to do more experiments on the mouse to determine whether it or some other device was really the better one. There were devices like rate-controlled isometric joysticks, almost identical to the one in the IBM ThinkPad keyboard today; there were various kinds of trackballs; and there were many versions of buttons and keys. Stu remembers the experiments:

> English was pretty busy, so I agreed to help him do the experiments. He set up the usual kind of A-versus-B experiments between devices, but the problem with them was that he did not really know why one was better than the other, as the results varied when you changed the context. Since we were trying to do the science of this stuff, I modeled each one of the devices so that I had the empirical differences between them, and I was trying to figure out how numerically to account for why those differences occurred. The most interesting model was the model of the mouse.
>
> Fitts's[11] law says that the time to point goes up as the log of the ratio between the distance and the size of the target. What's interesting about this law is that the slope of that curve is about ten bits per second, which is about what had been measured for just moving the hand alone without a device. What's interesting about that is that the limitation of the device is not then in the device itself, but in the eye-hand coordination system of the human. What that meant was that the device was nearly optimal. The hand was showing through the machine instead of operating the machine at a loss, and so if you were to introduce this onto the market, nobody would be likely to come up with another device to beat you. We could know this theoretically once we had this one empirical fact.

Stu went down to El Segundo to meet the Xerox engineers who were trying to invent a pointing device for the office system

- Xerox Alto
- Xerox Star
- Xerox Star mouse

that was planned. They resisted the idea of something outside the normal cabinet full of standard electronic racks that needed a work surface and a connection cable and would have to be packaged separately. Stu presented the results of his tests, with a lot of interruptions.

"Why didn't you run it this way?"

"Who were your subjects?"

"Why didn't you have eighty-year-old grandmothers try it?"

There was some flack and hostility, but when Stu gave his theoretical explanation of his supporting science, the room fell silent and he won the day. Xerox put a mouse on the market, and what Stu had predicted was true. It is still the most successful pointing device to this day.

In introducing the Star system, which was the next step after the Alto toward commercialization, there was a certain circuit in the machine, and Stu was asked whether the circuit would be fast enough to track the mouse. If they had to make it faster, they would have to rip out the existing circuit and put in a more expensive one. Stu discussed this over lunch one day with a colleague, and because he had a theory of the mouse he was able to whip out a napkin and make a calculation, which indicated that the circuit was going to be too slow. He went into the laboratory and took a little video of a mocked up task to check the theoretical curve, and that confirmed the problem, so they had to change to the more expensive circuit. It was possible to arrive at this definitive result four hours from the initial formulation of the problem because there was a theory. There was no need to run lots of user experiments and counterbalance them and do the analysis. In a design context you want to very rapidly say, "I believe it should be thus because of such a thing," and then do a little checking to be sure that you didn't leave something out.

Stu demonstrated that you could use the same theory to understand how to develop a better pointing device:

> Just as we showed the advantages of the mouse, you could use the same theory to show that you could beat the mouse. The mouse was optimum, but it was based on movements of the wrist. If you looked at Langoff's thesis, he had measured Fitts's law slopes for different

sets of muscles. In particular, when you put the fingers together you can maybe double the bandwidth.

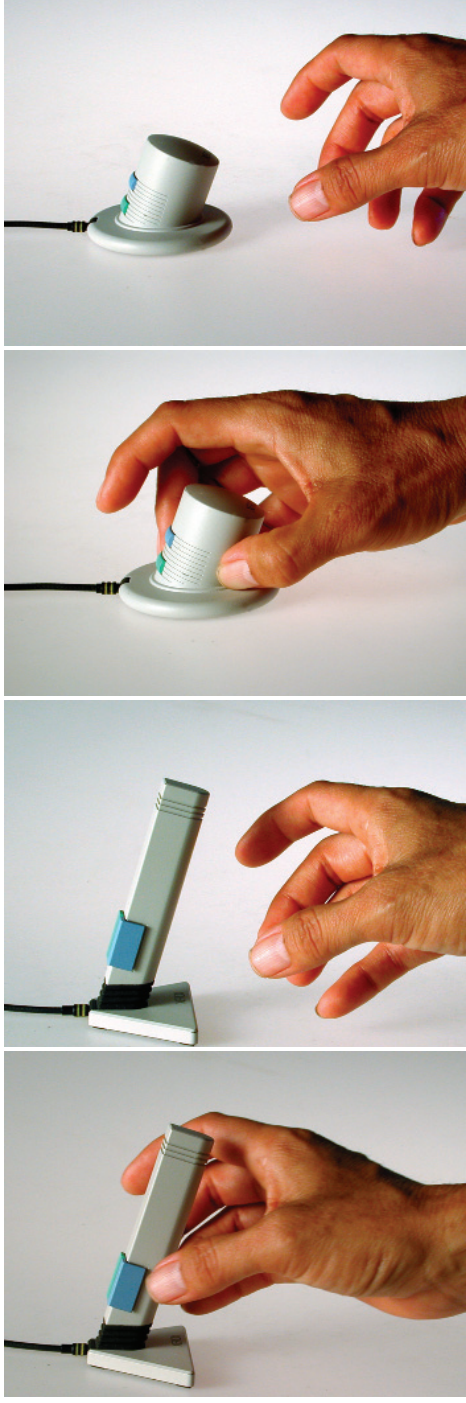
"That's where you want to put your transducers!" I said.

If you see these maps of the motor cortex and how much of it is devoted to different sets of muscles, you want to put you transducer in an area that is covered by a large volume of motor cortex. You could probably do one with the tongue too, because there's a lot of space devoted to that. That's another example of how having some notion of the theory of this thing gives you some structuring of the design space.

In the mid eighties ID TWO[12] was engaged by Xerox to design the enclosures for a new workstation and wanted to push the design of the mouse farther. Stu Card was able to help the designers think about new concepts for pointing devices. He demonstrated one idea with a pencil stuck onto an eraser, to make it easier to grasp when you are only seeing it out of the corner of you eye. He believed that it had become feasible to build a mouse like this because of the purely optical mouse that had been developed at PARC by this time. The whole mouse was on a chip, so most of the case was empty, and that meant that you could shape the casework to be optimal to the shape of the hand, instead of needing to make it big enough to cover the mechanism.

As soon as Stu explained how to structure the design space from his theory, the designers were able to quickly create alternative designs for pointing devices. They produced three models, based on progressively more radical assumptions. One was a conventional mouse that was flatter and more elegant, the next was a puck shape, like a top hat of about an inch diameter, with a skirt at the base for the fingers to rest on. The most radical one was the penlike device with a weighted base, causing it to stand upright on the desk, giving form to Stu's idea of the pencil stuck into the eraser. At the end of the project Stu concluded:

> This was my ideal model of how the supporting science could work. It required good designers to actually do the design, but what we could do was help structure the design space so that the movement through that design space was much more rapid. The science didn't design the mouse, but it provided the constraints to do it.



Xerox puck and pen mouse concept designs ■

Tim Mott

Tim Mott created the concept of "guided fantasies" to learn about user needs, and was one of the very first people to apply rigorous user testing to the design of user interfaces. He studied computer science at Manchester University in England in the sixties and found a job with a publishing company called Ginn, near Boston, that was owned by Xerox. This led him to work with the researchers at Xerox PARC, and he collaborated with Larry Tesler to design a publishing system that included a new desktop metaphor; together they invented a user-centered design process. In the late seventies he became more interested in designing processes for management and business and honed his management skills working for Versatec, another Xerox subsidiary that manufactured electrostatic printers and plotters. In 1982 he cofounded Electronic Arts (EA) and set about building a set of processes to enable the creation and production of really rich interactive entertainment experiences—as soon as the supporting hardware was available. From the very beginning, they built the company with people who were just crazy about games. Once EA was successful, Tim went on to run a small company called Macromind, whose founders had invented a user interface design tool called Director, leading the company to expand into multimedia and become Macromedia. He was a founding investor in Audible, setting the precedent for the MP3 players that came later, and moving from "Books on tape" to the spoken word Web site that supports public radio. He is a pilot, flying jets as well as the single-engine back-country plane that he loves to fly over wild mountain scenery.

Tim Mott

Tim put the editors in front of a display with a keyboard and a mouse. Nothing was on the display and no programs were running, but he asked them if they could walk him through the process, imagining what it would be like to use that hardware to edit. This was 1974, before word processors, so they were using typewriters, pencils, and erasers.

■ Tim Mott in 1974

Guided Fantasy

Ginn was one of several companies in the publishing industry that Xerox acquired around 1970. They were based in Lexington, Massachusetts, in one of the first buildings equipped with office systems furniture from Herman Miller. When the Ginn management team found out how their contribution to Xerox corporate finance was being spent, they put in a request:

> We're taxed for your research center; every division of Xerox has to pay money to support Xerox PARC research, and we want something back! What are you going to do for us?

This challenge eventually reached Bill English, who assigned the task of developing a publishing system for them to Larry Tesler. Larry started working with Ginn and writing specifications for their system and suggested that they hire somebody; they could send the new person to PARC for a year or so to work on the design of the new product. As it moved from design to implementation, he could write some of the code and then return to Ginn to provide support. They found Tim Mott at Oberlin College, where he was working on the help desk for a mainframe. Larry remembers interviewing Tim:

> When we talked to him, we realized that this was the perfect guy; he writes good code, he's really fast, he understands about usability because he helps customers all the time, he loves doing support and he loves programming, and he thinks it's an interesting problem. We hired Tim Mott, and we got ten times more than we bargained for.

Tim spent a little time at Ginn, observed how they worked, started thinking about how to make it easier for the publishing team to get their job done, and then flew out to PARC. He remembers his first encounter with POLOS (PARC On Line Office System):

> When I got out there, what I found was that, in my judgment at any rate, the system was completely unusable by anyone other than the people who built it. Their background was in building editing systems and design systems for themselves and for other computer professionals, and at least in my judgment it just wasn't going to be usable by editors and graphic designers who worked in a publishing company, so a month after getting to Palo Alto, I wrote a letter of resignation to the executive editor back in Boston, saying, "This is not a project that you should be doing." That letter found its way to Bob Taylor, who was running the computer science lab at the time. I spent some time with Bob, and he said, "Why don't you figure out what it is that we should be doing then?"

That was a challenge that Tim couldn't resist, so he teamed up with Larry Tesler, and together they set about discovering what the people who worked at Ginn really needed. Tim went back to Lexington and put the editors in front of a display with a keyboard and a mouse. Nothing was on the display and no programs were running, but he asked them if they could walk him through the process, imagining what it would be like to use that hardware to edit. This was 1974, before word processors, so they were using typewriters, pencils, and erasers. He used the "guided fantasy" technique that Larry had told him about, explaining that the mouse could be used to position a pointer on the screen and that the text would be on the screen. The editors described the process that they used at that time with paper and pencil. Together they imagined typing in the text and creating a manuscript, and

then editing that manuscript using the mouse and keyboard in the same way that they would use a pencil.

The Alto and the POLOS in 1974 both had mice on them based on the Engelbart mouse developed at SRI, and the Alto display was already a bitmap display, but the design of editing programs was still based on the prior generation of character displays; no one had really thought about how to use the characteristics of the bitmap display to create a more flexible editing environment.

None of the text editors that had been designed for computers up to that point had a space between characters. They were based on character matrix displays that didn't allow for an editing mark to be placed between the character cells. If you wanted to mark an insertion point in the text, you either selected the character before where you wanted to add the new text, and said *append*, or you selected the character afterwards and said *insert*. Tim explains the difficulty that this caused:

> That was one of the first things that I got from working with these editors. One of the concepts that they worked with was the space between characters, or the space between words.
>
> "I want to use the mouse and put this insertion point or this caret between these two characters, and then I just want to type in the new text," they said.
>
> When it came to deleting text, they talked about wanting to strike through it, just like they would with a pencil. They wanted to use the mouse to draw through the text. Up until that point, the way that a span of text had been selected was to mark the beginning point, and mark the end point and say "select." No one had actually used a mouse to draw through text.
>
> These techniques that we see in all the word processing programs today came directly from working with people who spent their entire lives editing text and asking them, "How do you want to do that?" Once Larry and I hit on this idea of having people talk about how they would want to do the work, the design itself became pretty simple. There was a new design methodology that came out of it. We talked about the design process as one that began with building a conceptual model that the user had today.

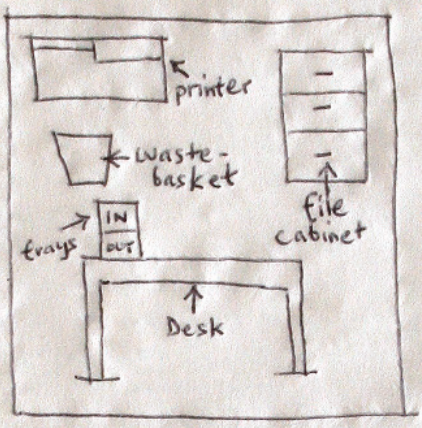
The Alto didn't have a user interface; rather, it came with a toolbox. You could build any kind of program you wanted, based on this very flexible architecture, and people started building applications on it. The Smalltalk system incorporated menus and editing techniques. There was a graphics program that William Newman built called Markup and a paint program that Bob Flegal created—applications started springing up.

Tim and Larry took the code base from Bravo,[13] a text editor that already existed at PARC, and put a completely different user interface on top of it. They called their design "Gypsy"; the implementation program was straightforward, but they worked hard. They shared one Alto between them, working fourteen-hour shifts so that they would overlap by an hour at both ends and could tell each other what they had done. In this way they could work on the same code around the clock and could protect their access to the computer, as there were only four Altos at PARC when they started.

Using "drag-through" selection for text was not the only innovation that came out of the work on Gypsy. The origin of the cut-and-paste metaphor is described by Larry in his interview, as is Tim's idea for double-clicking the mouse. There was also the first dialog box; this was a little bar with a place for typing commands such as *Find*. It was more like a noniconic toolbar of today, as in Apple's Safari browser.

Gypsy had a file directory system with versions and drafts, not just a list of files. You could have versions of the document and drafts of the version; it remembered them all and organized them all for you. It had bold, italic, and underline—used very much like they are today. You could select something by dragging through it or clicking the two ends, and do the equivalent of *Control b* for bold. They changed the name of the control key to *look* with a paper label, so you said *look b* for bold.

Larry had come up with the "guided fantasy" technique in 1973, and he and Tim developed other user-centered methods with frequent usability testing in the fall of 1974 and the spring of 1975; their work really caught the imagination of the people in the PARC community. In the spring of 1975 Tim took the text-



Office Schematic

PRINT, FILE, DELETE, MAIL

↓

all are inter-doc actions

—11—

INTRA-DOC use Cut & Paste physical metaphor

What's analog ↓ for INTER-DOC ??

Grab & Move !!!

editing program back to Lexington to try it out with the people at Ginn. For the first test he picked the most senior editor, figuring that, since she was the most entrenched both in her work habits and the company, if he could win her over, it would be smooth sailing from then on. After the first day she said, "You know, I think the quality of my work will be better going forward, because it's just so much easier to edit with this than it is with a typewriter and a pencil!" The approach had proved itself!

The Desktop (Office) Metaphor

In parallel with completing the text editor, Tim and Larry were pretty far along in designing a page layout system for graphic designers. They were still struggling with the issue of how to think about the user interface for documents and files, and the actions that take place on an entire document, rather than on the pages or text within a document. Tim describes his moment of inspiration:

Tim Mott's reconstruction of his sketch on the bar napkin

Photo
Author

> I was in a bar late one afternoon waiting for a friend, doodling on a bar napkin and thinking about this problem. I was just obsessed with this design at the time; I was just consumed by it. I was thinking about what happens in an office. Someone's got a document and they want to file it, so they walk over to the file cabinet and put it in the file cabinet; or if they want to make a copy of it, they walk over to the copier and they make a copy of it; or they want to throw it away, so they reach under their desk and throw it in the trash can.
>
> I'm sitting there thinking about this and I'm doodling. What ended up on the bar napkin was what Larry and I called the "Office Schematic." It was a set of icons for a file cabinet, and a copier, or a printer in this case, and a trash can. The metaphor was that entire documents could be grabbed by the mouse and moved around on the screen. We didn't think about it as a desktop, we thought about it as moving these documents around an office. They could be dropped into a file cabinet, or they could be dropped onto a printer, or they could be dropped into a trashcan.

> The desktop was part of the design, and on it there were those things that you might normally find on a desktop like a calendar, a clock, and baskets for incoming and outgoing mail; the notion was that we would be able to use that as the controlling mechanism for electronic mail.

When Larry heard about the idea and saw the bar napkin, he showed Tim the illustrations from "Pygmalion,"[14] Dave Canfield Smith's thesis. He said that people had tried to implement similar designs before, but they had always attempted to do very complex, three dimensional, true-to-life simulations, as opposed to just a simple two-dimensional iconic representation. The simplicity of the representation was the breakthrough! Somewhere in one of Tim's notebooks in Xerox there is a bar napkin covered in the doodle of his office metaphor, complete with a desktop and trash can.

Photo Author

Larry Tesler

When he was at Apple, Larry Tesler had a license plate saying "NO MODES," emphasizing his passion for designing software that would be simple and easy to use. He had been writing code since he was in high school and worked at the computer center at Stanford while studying there in the early '60s. He founded his own company to offer programming services while he was in his junior year and soon discovered that his customers had a different way of thinking from the software engineers in the Computer Science Department. He realized that the best way to design the software was with participation from the customer, and he developed techniques for watching how people did things and designing software that allowed people to use new technology in familiar ways. He learned to create prototypes rapidly and to test them with the intended users early and often. After working at the Stanford Artificial Intelligence Laboratory, he went to Xerox PARC in 1973 in time to be a key player in the development of the desktop and desktop publishing. In 1980 he moved to Apple, where he was core to the design of Lisa. He invented cut-and-paste and editable dialog boxes, and he designed the Smalltalk browser. He simplified the use of the mouse by reducing the controls to a single button. He insists on truth and accuracy and is willing to challenge anybody's assumptions about the best way to do things, always thinking from basic principles. After a three-year stint at Amazon, running the usability and market research groups, he is now helping Yahoo improve the design of their online interactions, as VP of user experience and design.


HOW DO I GET
MONTH
CALIFORNIA
YEAR
NOMODES
OUT OF THIS MODE ?

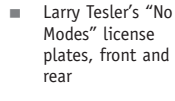

INSERT
FIND
FEB
CALIFORNIA
CA 96
H 1920079
NOMODES
REPLACE
FILE NAME

Larry Tesler

"Iconic naming systems will be explored. A picture of a room full of cabinets with drawers and file folders is one approach to a spatial filing system."

From a white paper called OGDEN (Overly General Display Environment for Non-programmers), by Larry Tesler and Jeff Rulifson

Participatory Design



Larry Tesler's "No Modes" license plates, front and rear

Photo
Author

LARRY WAS QUICK to realize the value of participatory design and usability testing. His first lesson occurred while he was still studying at Stanford, when he was asked to get involved with a project to organize the "card stunts" at football games and to automate the production of instruction cards. At halftime during the football game, when the students flipped up the cards, instead of hand-written instructions they would be computer-generated. Students from the Computer Science Department had already written a programming language on punch cards, with all numeric commands. An art student would draw polygons to represent the shapes and colors and describe an animation, and then a computer science student would code all the instructions to generate the instructions and print out the cards. Finally the cards would be torn along perforations and set out on the seats. Larry was persuaded to take over the coding role.

"The trouble is that the art students have a great deal of difficulty with this language, so you end up coding all the stunts," he was warned; "if you don't want to code all the stunts, you better find a way of making it easier to use!"

He started talking with the art students and trying to figure out what they wanted. It was a different world! Over the next three years, he reworked this language several times, until even the art students could use it.

In 1963, when he started a software consulting business in parallel with his studies, he developed a rapport with users and worked out how to ask them the right questions, so that he could make programs they could use. He did not have to run the software for them; they could do it themselves. One of the programs he wrote was a room-scheduling program for the Stanford Psychiatry Department. They were trying to arrange for patients who came in to get assigned to rooms with doctors. They had scheduling problems because appointments overlapped, and rooms had to be used at certain times during the week for other things.

> One of the psychiatrists and I designed it together. That worked out really well, as they needed no help from me at all to use this. That's when I realized that the best way to design the software is with the customer, which is now called participatory design. With the art students and the card stunts, I would watch how they used it and see when they got confused. That was my first experience with what we call usability tests today, observing people and seeing what the problem was and then going and fixing it, and realizing that I could make something simpler. That was really the beginning of my interest in usability.

■ Larry Tesler at Stanford, 1961

The Future System Will Use Icons

Experience at the Stanford Artificial Intelligence (AI) Lab gave Larry lots of exposure to the computer science community in the San Francisco Bay area. He got interested in typography and developed a publication language called Pub. It was what today would be called a markup language with embedded tags and scripting, but this was in 1971. It only ran on the PDP 6, which limited it to use in universities. A lot of graduate students used it for their theses in the major universities that were on the ARPAnet. On request, he distributed it in source-code, so it was actually one of the first open-source markup languages.

He decided this was really the wrong way to do it, and that the right way to do it was to have an interactive layout system, not a language, so that anybody could use it. He was putting together a catalog for a volunteer group one day in 1970, using X-ACTO knives and glue, pasting up the pages with type-scripts and remembers saying to a friend:

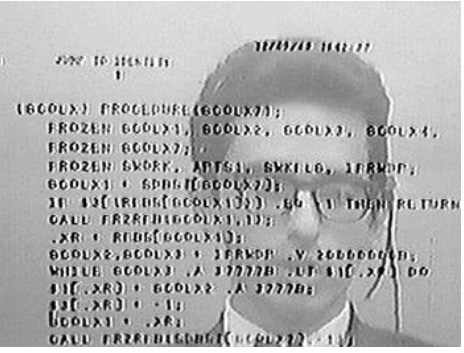
> You know, this is not the way this is going to be done in the future. In the future you're going to have a big screen in front of you, and you're going to be taking these excerpts from documents, and you're going to be pasting them into pages, without glue, and it'll be perfect. We won't have all these alignment problems! And then when you're done you're going to print on a phototypesetter. It'll all be done interactively.

He was looking for the right place and opportunity to turn this vision into reality, and in 1973 he joined Xerox PARC with that expectation in mind. When he got there, he said that he wanted to work on interactive document formatting. "We'll get to that, but first you need to work on this distributed office system we're building," Bill English replied. This was disappointing, but he was allowed to spend about half his time working with Alan Kay's team, which was much more exciting as they were starting to develop Smalltalk. He got involved there developing the first modeless editor.

Larry Tesler at Stanford AI Lab, 1971 ■
Larry Tesler at Xerox PARC, 1974 ■

Initially I started working with a guy named Jeff Rulifson, who had worked with Bill English and Doug Engelbart at SRI. "You two go off and do some visioning about the office system of the future," Bill said. That was pretty exciting, so we went off and wrote a little white paper on it called OGDEN (Overly General Display Environment for Non-programmers). A lot of it was kind of nuts and all wrong, but there were some really good things in there. Jeff had just read a book about semiotics, and he said that it had talked about signs and symbols and icons and so on. The author of the book had said that icons would be a really appropriate thing to use in an interface for a computer system, because they would remind people of a concept, and they would grasp the idea quickly by looking at the icon without having to read a word. Jeff thought that we should incorporate that, so we put it in our white paper, and said, "Iconic naming systems will be explored. A picture of a room full of cabinets with drawers and file folders is one approach to a spatial filing system."

We had a diagram with an idea of what it might look like. It didn't look at all like the Mac or Windows; it looked more like General Magic, with a 3D, or perspective, desk, a file cabinet in the corner, and things on the desk, and a trash can next to the desk. The idea was that you would point at stuff that looked semirealistic, representational, but not equal-sized icons like we have today. Alan Kay loved that idea and had been thinking of similar things, and he passed it on to Dave Canfield Smith when Dave came looking for a thesis to do, and Dave actually implemented the first thing that's more like what we think of as icons today.



■ Jeff Rulifson at Doug Engelbart's 1968 demo

The Five-Minute Learning Curve

With the exception of Jeff Rulifson and Larry, everybody at PARC was extremely wedded to the NLS interface that was developed in Engelbart's team. Jeff was the least wedded to it, perhaps because he had come up with it.

"Why is everybody so excited about this interface?" Larry said to Jeff, who laughed.

"I don't know, because it was never designed!" Jeff replied, "When they implemented the system, they didn't have a user interface; they just knew what they wanted it do."

Jeff kept asking what the user interface was going to be.

"Well, I haven't designed that yet," was the reply, "I'll do that later."

Jeff was assigned the task of testing the system, so he took the procedures in the language. There were commands like *delete*, *left parenthesis*, *character number*, *right parenthesis*, so he made something that was absolutely literally the same thing, only intended for the programmers testing the software. They all started using it and learned it, and then they started getting excited about how you could type the letters on the keypad by using binary code. What started as a testing language acquired a mystique; although it was never designed for end users and never analyzed scientifically, its aficionados grew to believe that it was an ideal interface for a system to augment the intellect. Perhaps it was. The trouble was that it was really hard to learn, and they admitted that.

Larry set out to prove that you could learn something in a day. He developed a little text-editing program that he called "Mini Mouse." It didn't use the NLS keypad at all and only used one button on the mouse. Before doing it he decided that he wanted to observe a user, and used a technique similar to his "guided fantasy." He describes working with a secretary who had just started at PARC and was not yet influenced by the programs that were being used:

> I sat her down in front of a screen, and did what's now called a "blank screen experiment."

"Imagine that there is a page on the screen, and all you've got is this device that you can use to move a cursor around, and you can type," I said. "You've got to make some changes to this document. How would you do it?"

I gave her a paper document with lots of markups on it for reference, and asked her to imagine that is was on the screen. She just designed it right there!

"I would point there, and then I would hit a delete key," she said.

To insert, she would point first and then start typing. She'd never been contaminated by any computer programs before, so I wrote all this down, and I thought, "That sounds like a pretty good way to do it!" Later I also tested other new people like that and got similar kinds of results. I wrote a report about this. Up to that time, Bill English had been extremely skeptical about what I was doing. He seemed offended that I was challenging the "perfect" NLS editing language, but when he read my document a light went off, and he came to me and said, "This is really good!"

In the past, the programmers at SRI and PARC had thought that usability tests would be extremely expensive and time consuming and had done very few of them. Larry felt that he had to demonstrate that testing did not have to be difficult or expensive, so he built Mini Mouse in Smalltalk, like a typewriter. It turned out to be so easy to use that they could take people who were walking by in the building, and ask them to try it, and they were able to learn to operate it in five minutes—compared with a week to get comfortable doing comparable tasks using NLS.

Double-click, Cut, Paste, and Cursors

The Mini Mouse triumph earned Larry the freedom to work more independently and to focus on text-editing software. When the gauntlet was thrown down by Ginn to help them with a publishing system, he was only too pleased to take on the task. He found Tim Mott, and they became partners in creating a new state

of the art for text editing and graphic layout, as described in Tim Mott's interview. Larry tells the story of inventing the double-click:

> When I had written up Miki Mouse, which was going to be the next thing after Mini Mouse, I was trying to decide what to do with all this hardware. We had three buttons on the mouse. I really wanted it to be a single-button mouse, because I wanted to be able to use other things like tablets, touch screens and light pens, and you couldn't do that with a multibutton design. Another reason was that when people were using the software, they thought of the mouse as being held by the pointing hand, with the other hand being for commands, either on the keyboard or on the five-key keypad. I thought if we could separate that out really cleanly we would reduce the common mistake of hitting the wrong button on the mouse. One in five times it was, "Oops, hit the wrong button." This was why you had to practice a lot to master NLS, as so often you hit the wrong button.
>
> "Maybe we could use the first button to select the cursor location between characters," I thought, "and the second button to select a word, and the third to select a sentence or something." I was with Tim and said, "I really don't like this, because I'd really rather just have one button."
>
> One morning he came in and said, "I've got it; double-click! You click twice in rapid succession in the same place to get a word, and three times to get a sentence."
>
> "Twice maybe, but not three times, surely!" I said. I gave him all the reasons that it wouldn't work, but when I closed my eyes and tried to envision it: "Yes it just feels right," I had to admit, "double-click to select a word; it just feels right."
>
> We had already been implementing Gypsy at that point, and some of it was working enough so that we could try it. We brought in the secretaries and had them try it, and everybody thought it was good. There were lots of detailed technical arguments from the programmers, but basically it became a good thing right away.

One day Larry saw a novel way of handling "delete" and "insert" that had been developed by Pentti Kanerva, who was a software expert at Stanford University. It was done by first selecting what you wanted to delete and then saying *escape-d-*


PASTE
FAST
CUT

return for delete, and then by moving the cursor where you wanted it and then saying *escape-o* for "oops," which meant "undo last delete." If you made a mistake, you just hit L, which meant insert, and it inserted whatever was in the buffer wherever the cursor was. The cursor was likely to be where it was before, so it came back.

The magic of this idea was that if you moved the cursor somewhere else after you said delete, it would move it there, so you didn't need a *move* command. When Larry was working on Gypsy, he remembered that and thought that he could use it to avoid a special mode for moving. The trouble was that it was not good for copying, because you would need to copy it to the buffer.

> We were doing this for a publisher, Ginn and Company, and we were planning to implement "cut" and "paste" to move things around in the document.
>
> "Why don't we use the same words, *cut* and *paste*, for this process of delete and insert," I said to Tim, "and that way if you want to copy, you can say *copy* and *paste*."
>
> Tim agreed, so we named our commands, *cut/paste* and *copy/paste*. Because of that, I seem to have the reputation for originating cut-and-paste, but it was really Pentti Kanerva's idea that I renamed and reduced the number of keystrokes from six to two.

■ The original "cut" and "paste" labels written by Larry, stuck onto the NLS keyset to first test the idea

Photo
Nicolas Zurcher

There was no cursor in NLS: you just pointed at things and they became the first item (character, word, and so forth) or the last item of the selection. Every system that had a cursor took a character and either underlined it or showed it in reverse video; normally if you were doing white on green, you'd show green on white. The problem with that was that if you moved the cursor somewhere and started typing, it wasn't clear where the character would go. The way it worked in most systems was that the character that you had inverted would change as you typed, so you would overtype things. If you wanted to insert, you had to go into a mode, like control-i or an insert key, and then everything you typed would get inserted before the character. Larry found in experiments that a lot of people expected it to go after the

character, and some people were just confused. He didn't want something you had to learn; he wanted it to be obvious. He was puzzling about this, and brought it up it during a meeting at PARC.

A few days later he bumped into Peter Deutsch in the hallway, who was not a user interface guy in any way; he was more of a systems guy.

"I think I have a solution to your problem of insertion for the cursor," he said. "Put the cursor between characters, not on a character."

"That's a nice idea, but how do you know which character you mean?" Larry said.

"Well, if you click on the right half of the character, the cursor goes after it, and if you're on the left half of the character, it goes before it."

"Oh, of course, that's so simple. It's the answer." Larry thought, "I just need a way to show that there's this place between characters. In this publishing system they're used to using this caret mark to show that something needs to be inserted, so I can use that."

He started playing around with the caret mark, and finally came up with a caret that would work, overcoming the space management problem, and in the early systems they used a little caret mark as the symbol. Later, Dan Ingalls came up with the idea of just doing a vertical line between characters and implemented that for Smalltalk. They made it easier to see by making it blink on and off, to avoid confusion with a vertical bar. Larry tried other alternatives to make it more visible, for example curling the top and bottom like a sideways H, so that it didn't obscure the character you were looking at, was easy to find on the screen, and had a clear meaning. The vertical line was the design that survived.

Smalltalk Browser

Larry had been trying to get into Alan Kay's group from the moment he arrived at PARC. As a result of the success of the Gypsy project, he was given the chance, and he started working on Smalltalk. He describes his delight in working for Alan and how he was challenged to design a browser:

> Alan was always a great research manager; he would always challenge us, and would never be satisfied with anything. He was very good at rewarding people and acknowledging their work.
>
> He would say things like, "We still don't have a good way to query databases," or "We still don't have a good way to deal with animation."
>
> One of his favorites was, "You know, you can go to a library and you can browse the shelves, but we don't have any good way to browse."
>
> Diana Merry, who had converted from being a secretary and become a Smalltalk programmer, said to me one day, "You know, we still don't have a good way to browse."
>
> "I'm so tired of hearing that, Diana," I said, "Alan says that every month, and I'm so tired of it! It isn't such a hard problem."
>
> "Well then, go do it!" she said.
>
> "If I do it, will you promise never to say this again?"
>
> She assured me that she wouldn't, so I had to figure out how to browse. I was thinking of graphical metaphors with books on bookshelves, and then thought that maybe I should start with something closer to home. We programmers are always trying to understand other people's code, scrolling through big files of source code all the time trying to find stuff, and we don't know what it's called, so we can't use the search command. I thought that if I could figure out a way to browse through code, we could maybe generalize it to work for other things too, plus it would be a cool thing to have for myself, so I decided to build a code browser.
>
> I think this is one of those things that a lot of designers do. You just kind of close your eyes and vision, what would it look like? Instead of putting the secretaries in front of a blank screen, for a programmer's tool, I'm designing for myself, so I can be the subject.

Close your eyes. Blank screen: what does it look like? Smalltalk had classes of objects, and methods, which were routines or procedures.

"Okay, I'll have a text list of classes, and then when I click one of them, somewhere it'll show all the methods," I thought. "Then when I click the method, it'll show me the code for that method. Then I can scroll through the lists, and browse around. That's pretty simple!"

I got a piece of paper and drew it out and had the usual width problems with lack of room for three columns, so I put the two lists of classes and methods across the top, to allow a full width window for the code. It was a window broken up into three pieces, so I'll call them "panes," so we can have windows and panes. Each pane will have its own scroll bar so you can browse before you pick one to see the code. While we're at it, once you're looking at the code, I might as well let you edit it, using the standard Smalltalk editing facilities of drag through, cut-and-paste and all. I built it in somewhere between one and two weeks; it was one of those "Aha" kind of things, and it immediately started getting used by everyone. There were over a hundred classes and over a hundred methods; Dan Ingalls made a big improvement to it by dividing them into categories, so we ended up with four panes across the top. That became the classic Smalltalk browser.

This was the first browser design. It was a welcome change for Larry to design something for himself as a software engineer. He had become so wedded to the ideas of participatory design and user testing, that it seemed surprisingly straightforward to be designing for the user in the mirror. He went on to use the Smalltalk browser to build a point-and-click debugger and inspector. Other people have used the same metaphor to browse around databases. Although Web browsers are fundamentally more like Engelbart's NLS, which used links to jump between documents, frames in Web browsers resemble Larry Tesler's browser panes from 1976.

The Brain Drain from PARC

Around 1980, lots of people started leaving Xerox PARC.[15] At the beginning, when PARC was founded, the most inventive minds in technology had been attracted by the promise of creating something that would change the world and by the chance to work with and learn from each other. They had been together long enough to be strengthened by their collective experience, and many of them were itching for the opportunity to see their ideas move from research to reality. Xerox was struggling to maintain dominance in the copier market, as Japanese competitors were overtaking them at the lower end of the market and gradually moving up. The only part of the research at PARC that Xerox really took advantage of was the laser printing technology, and although that produced enough revenue to justify the investment in the research center many times over, it was increasingly clear to the people there that concepts for the "office of the future" were on hold. There was also a sense of opportunity in Silicon Valley, as the venture capital firms were starting to fund new-start companies for the development of products rather than just for chips. It was natural for all those brilliant researchers to want to take the next step in their careers, so they were looking around.

As it happened, Larry Tesler, Alan Kay, and Doug Fairbairn all resigned from Xerox on the same day, for three different reasons, to go three different places, without knowing one another's plans. Larry was headed for Apple, where he started in July 1980. He had participated in the famous demo of PARC technology to Steve Jobs and the team from Apple and was getting to know more about them:

> I was blown away by the Apple people. I had been trying to convince people at Xerox about my belief in the future of personal computers, but the Apple people appreciated everything in the demo that I thought was important. I agreed with all the things that they said should be done next.
>
> "They think more like me than the PARC people do." I felt, "I'm in the wrong place! I should join a company like Apple."

Two years before I joined Apple, when the company had only thirty employees, a friend had dragged me along to a company picnic, but they still seemed to me like a bunch of hobbyists. Between the picnic and the demo, they had hired all these computer scientists and software engineers who were very sophisticated.

"These guys really know what they're doing," I thought, "and they want to do the right thing!"

When I got to Apple, I said once again, "What I want to work on is publishing systems," but they asked me first to work on Lisa. They put me in the research group consisting of three people: Bruce Daniels, who was a systems guy; David Hough, who was a numerical analyst; and me.

"I don't really understand what kind of research we're going to do," I said after I had been there a week. "We don't have time to do research. Apple is not ready to do research. It's too early in the history of the company."

They agreed, so we went to the management.

"We've just dissolved our research group," we said, "so you should just put us in the Lisa group."

They did, so I started talking to Bill Atkinson about what I thought should be different. Bill was a neuro-psychologist by training, with a great combination of psychology, design (he's now a photographer), and programming. He was a great integrator; he was a sponge. At first I didn't think he was hearing people's ideas, but he was processing them and synthesizing them in a wonderful way. It would be great to be inside Bill Atkinson's brain!

This was the start of a great collaboration between Larry Tesler and Bill Atkinson. They formed a tight bond and creative partnership and went on to design some of the most significant and long-lasting interactions in the world of desktops and windows. An interview with Bill Atkinson follows in chapter 2, and he and Larry tell some of these stories together.

2

My PC

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


hello.

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

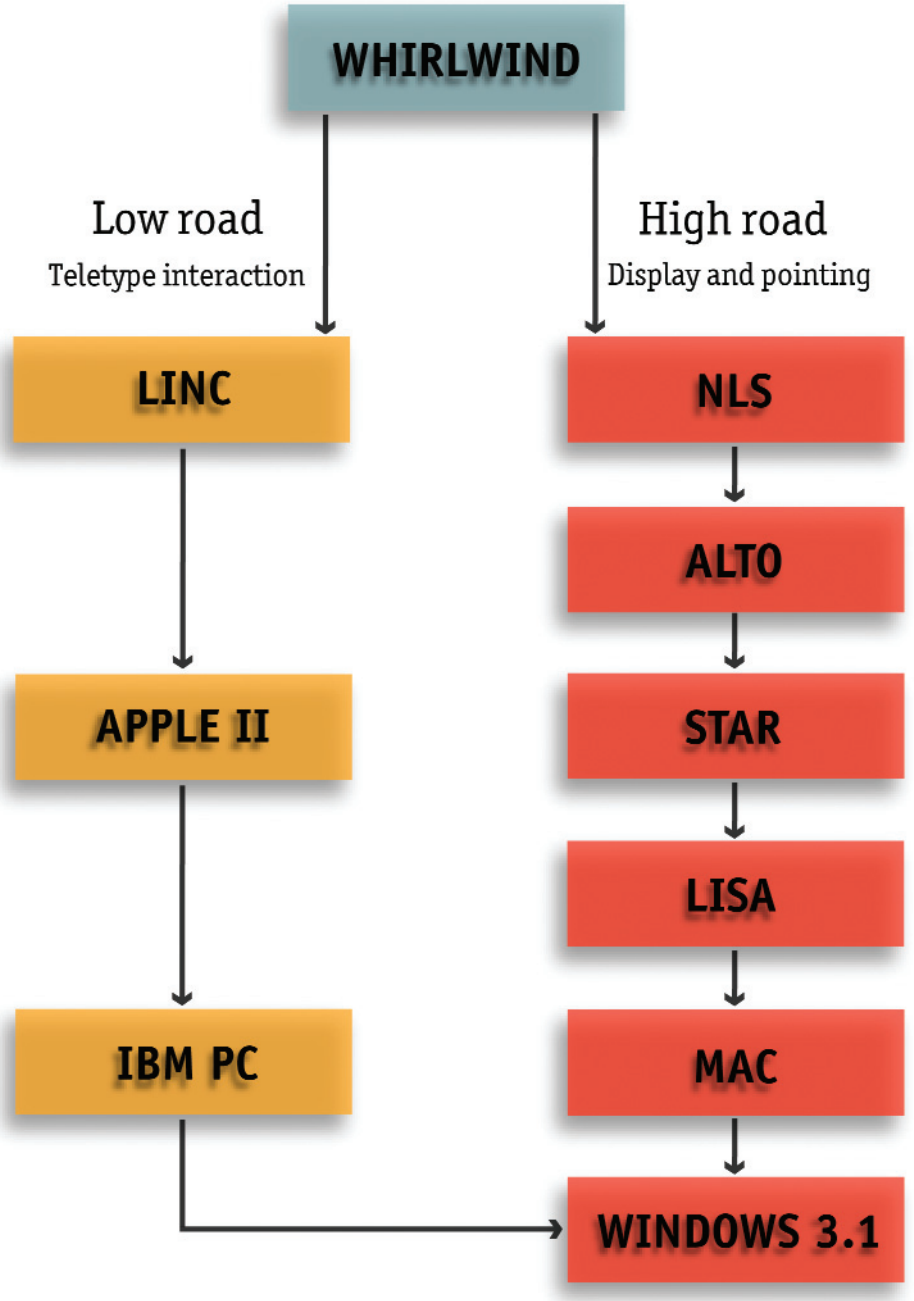
THE DESKTOP AND mouse are the dominant designs for the personal computer. In chapter 1, we looked at the invention of the original concepts at Xerox PARC and the roots of the design thinking that defined the interactions. This chapter follows the development of the designs for the first versions that 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.

■ Apple Mac and Lisa, from 1984 brochures

Photos
Courtesy of Apple

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


WHIRLWIND
Low road
Teletype interaction
High road
Display and pointing
LINC
APPLE II
IBM PC
NLS
ALTO
STAR
LISA
MAC
WINDOWS 3.1

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[1] 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[2] 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.

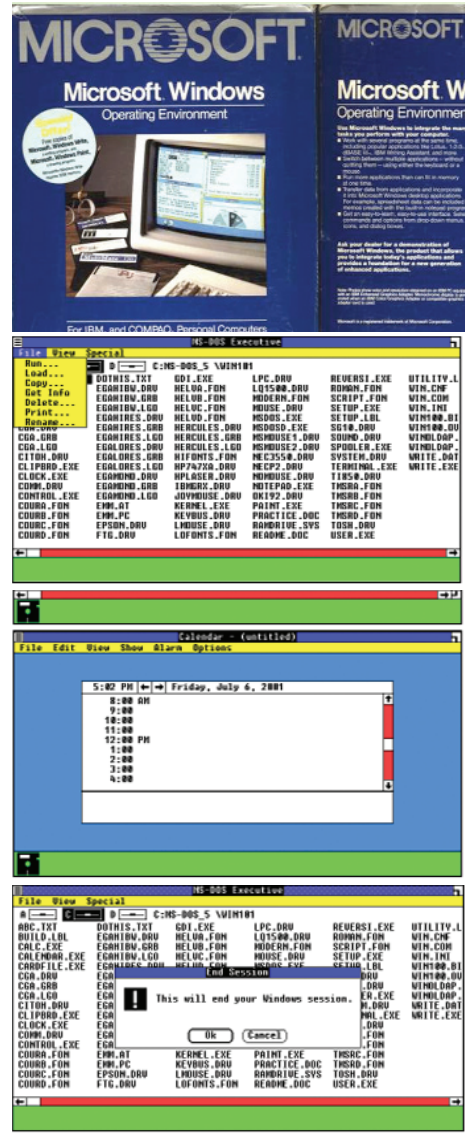
Both Lisa and Mac were very different from Star—full of new interactive features and dramatically less expensive to manufacture. Lisa was a wonderful accomplishment of interaction

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[3] 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.

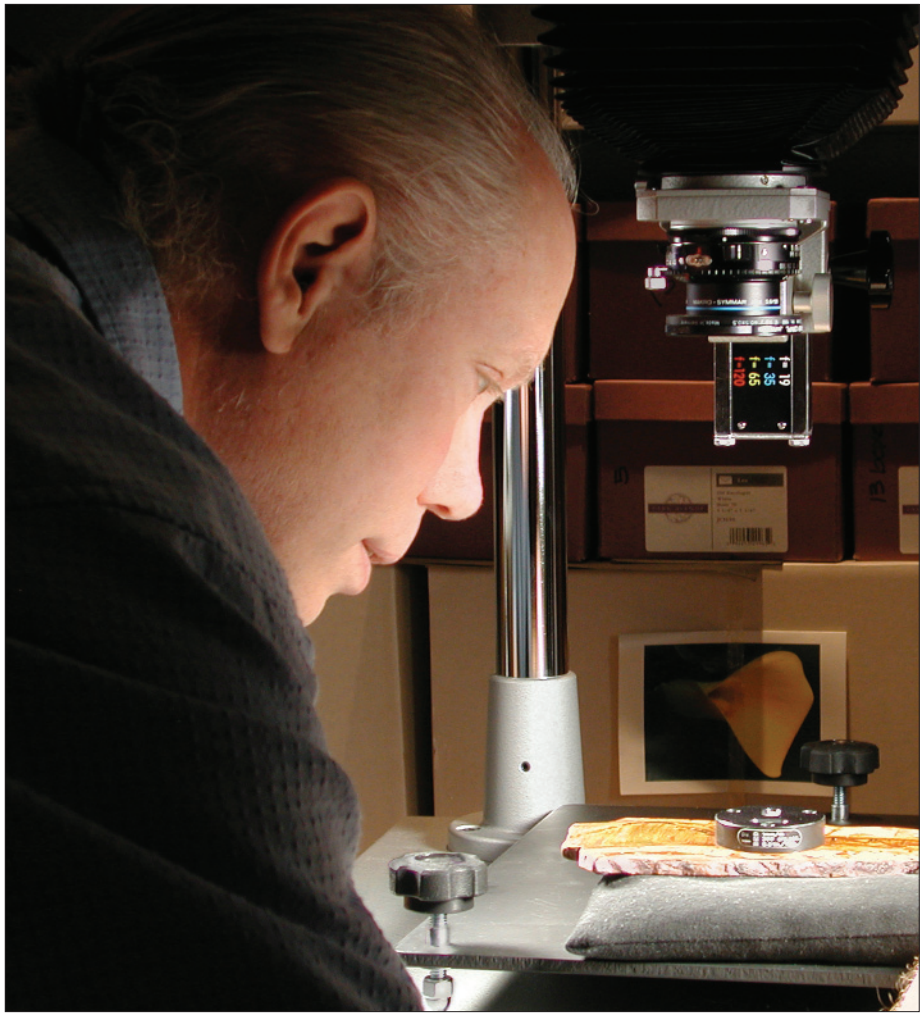
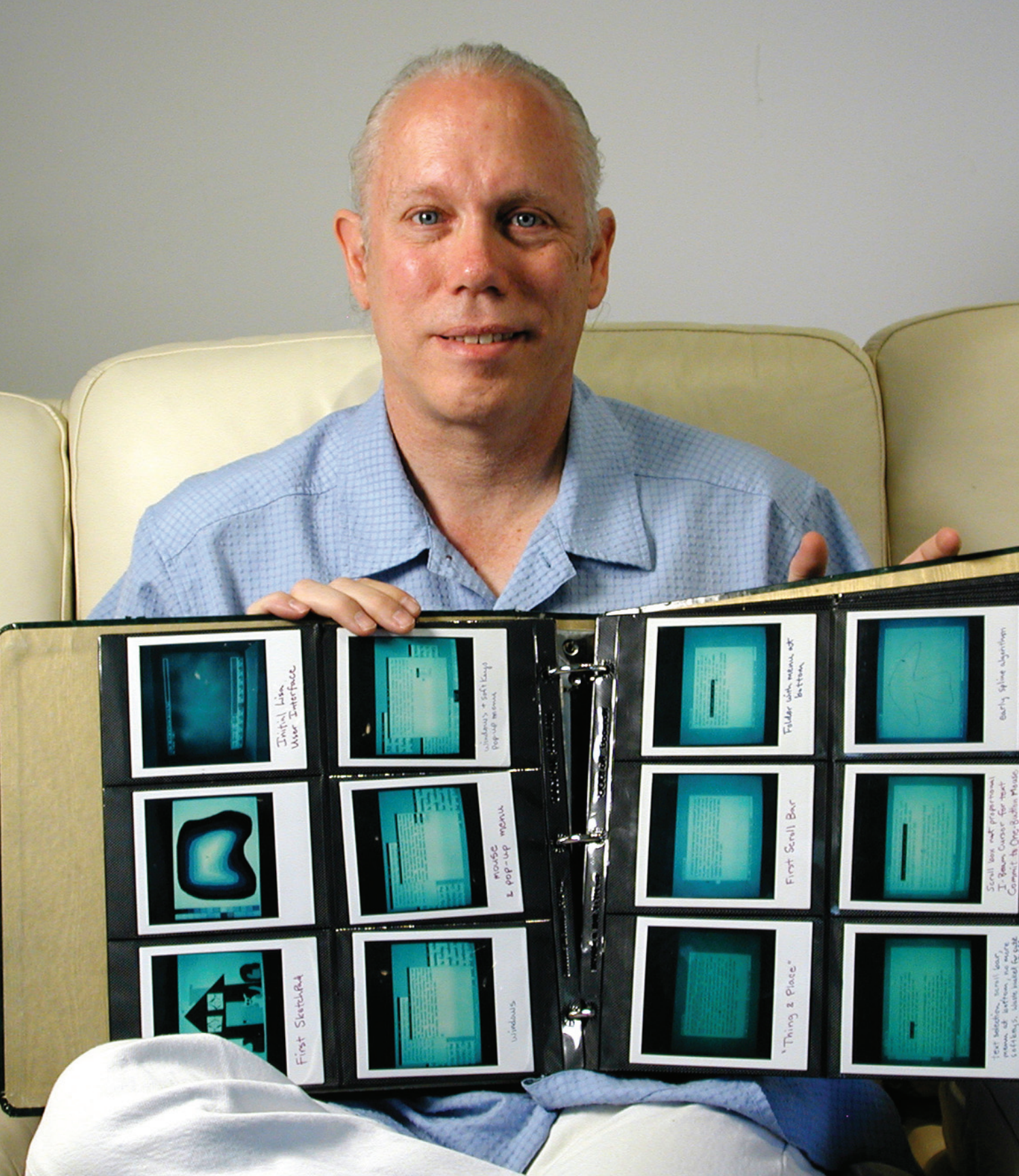


Photo Author

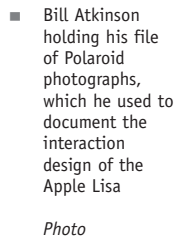
Bill Atkinson

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[4] 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 Lisa User Interface
Windows + Soft Keys
Folder with menu at bottom
Early Spline algorithm
mouse & pop-up menus
First Scroll Bar
First Sketchpad
Windows
"Thing & Place"

Bill Atkinson

Apple Lisa



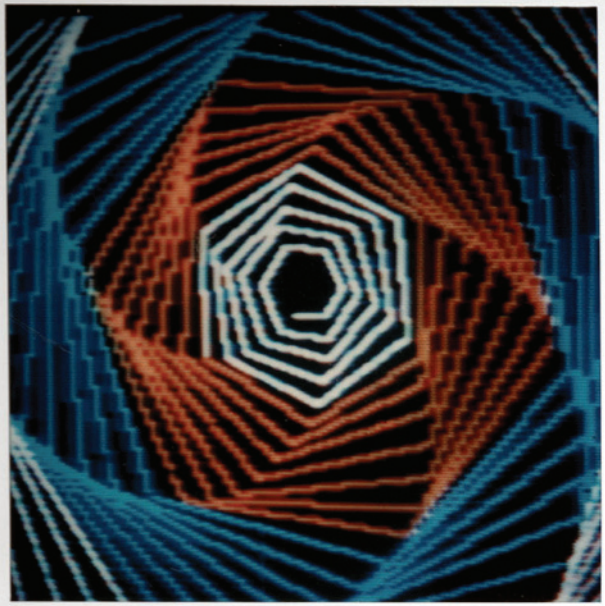
Bill Atkinson holding his file of Polaroid photographs, which he used to document the interaction design of the Apple Lisa

Photo
Author

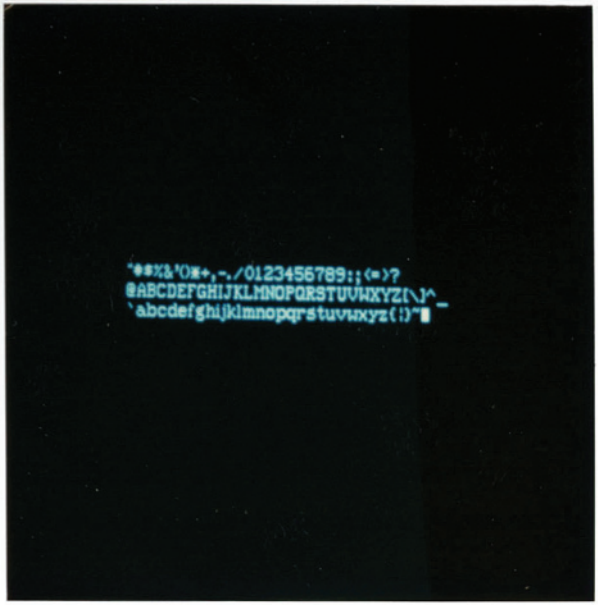
As a boy, Bill was fascinated by chemistry. He combined different ingredients to make homemade gunpowder and rigorously tested for the optimum mix to power his skyrockets. This passion for research and experiment pushed him toward a career in neurochemistry. As he studied the human brain, the other side 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*[5] makes you think hard about the underlying concepts behind today's user interfaces. Bill describes Jef:

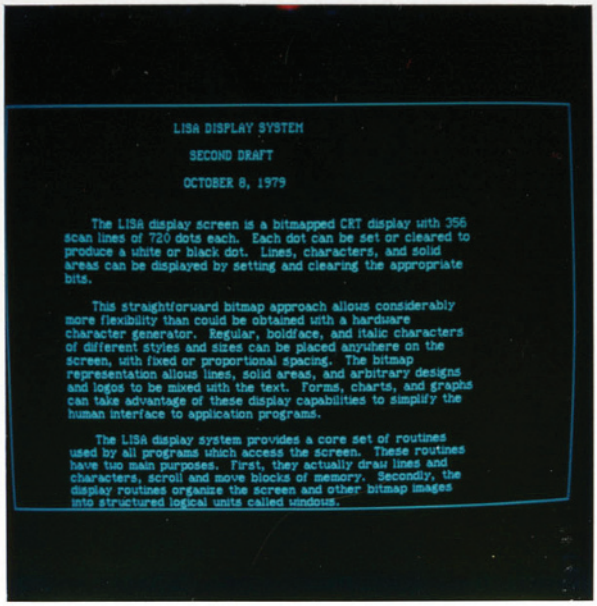
> Jef Raskin was a very brilliant and very independent-thinking professor. I remember one time he got in trouble with the campus computing center because he used his computing budget to buy a minicomputer, terminals, and bean bags for his students. This was at



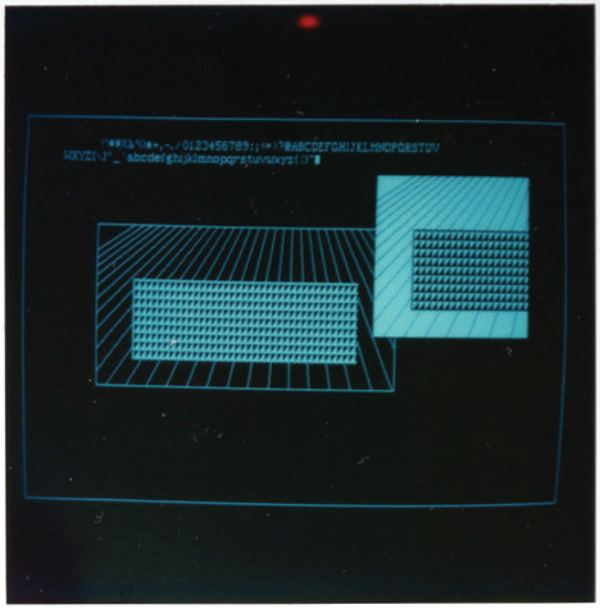
Apple II graphics for Pascal



First Characters on Lisa



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.

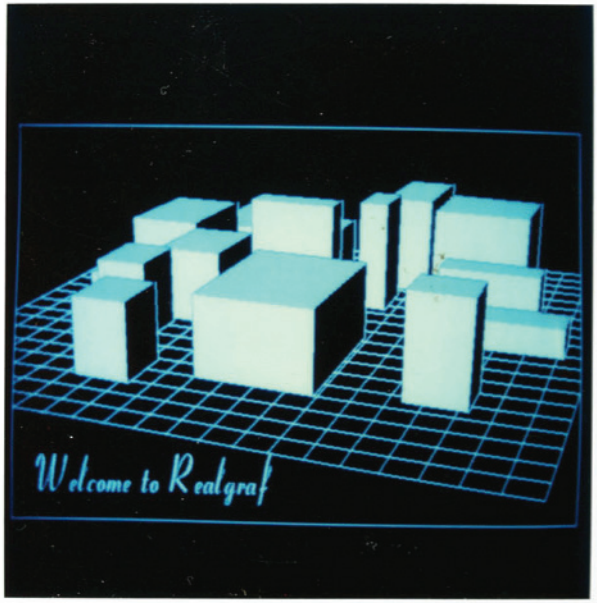
Bill picked up programming and computer science as a peripheral aspect of his studies, but his habitual thoroughness pushed him to master both the science and the black arts of writing code for software. Jef Raskin saw the latent design talent in his young student and thought of him when Apple needed 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:

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

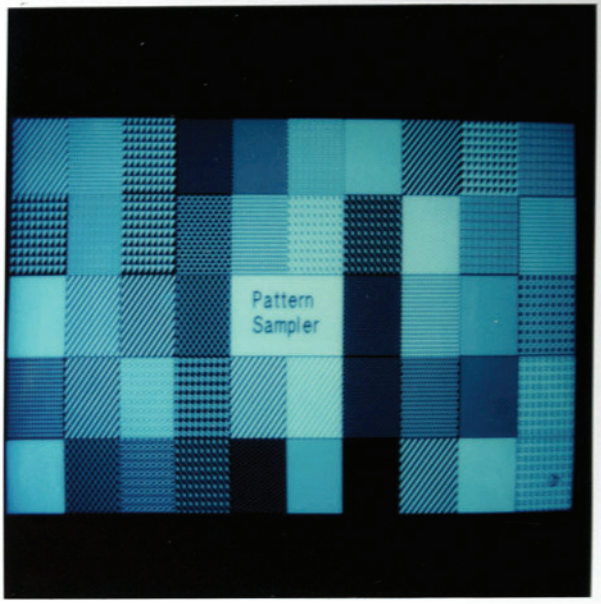
Photos
Bill Atkinson

> 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.



Three-D Graphics

Halftone Images



Patterns

25×80 + Softkeys
80×9 - 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:

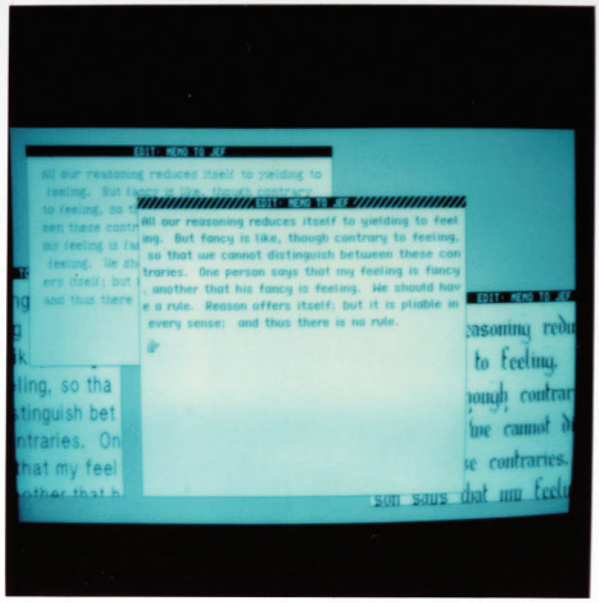
> The people at Xerox had done some wonderful work on using a mouse and using windows, and that was work that presaged what we did at Apple. What we did was not just to replicate that, but to start anew with a fresh research project, asking the question, "What is the best user interface for this whole 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.

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

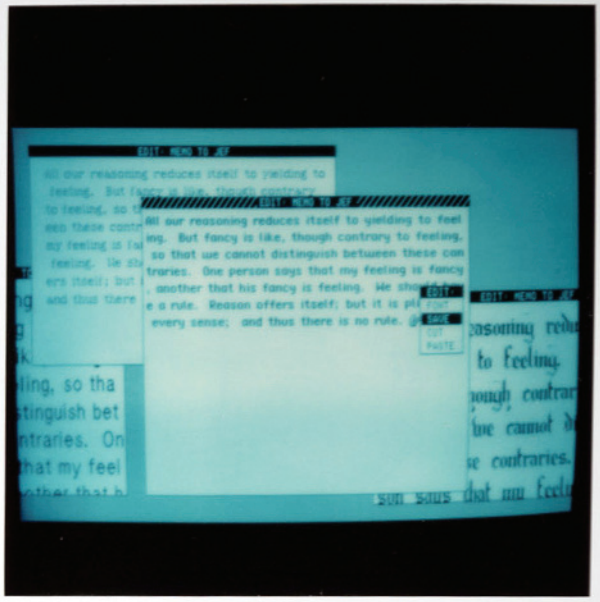
Photos
Bill Atkinson

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

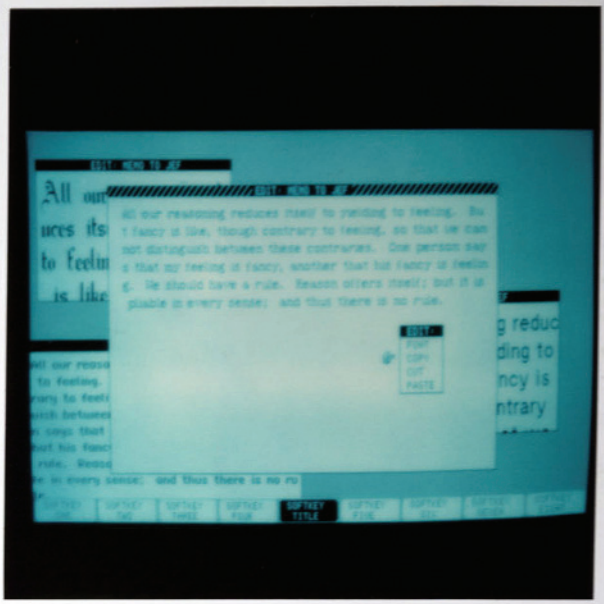
> The people who had come from Xerox had a rule that we could not ever disclose what we had done at Xerox. When I got there, I asked them how this worked.



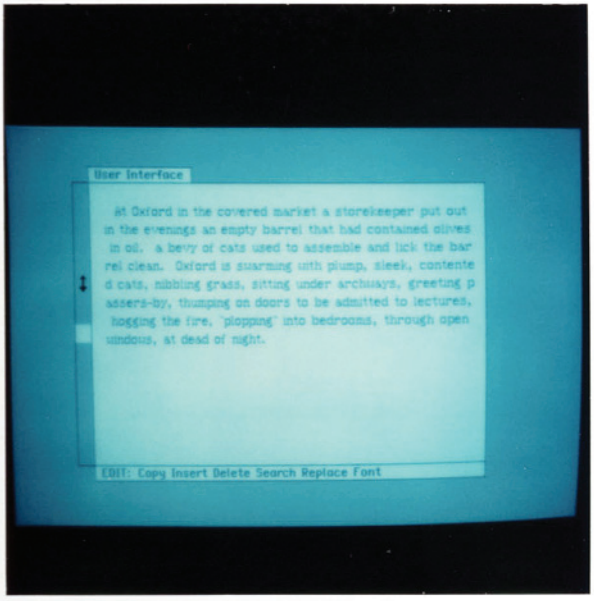
windows



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.

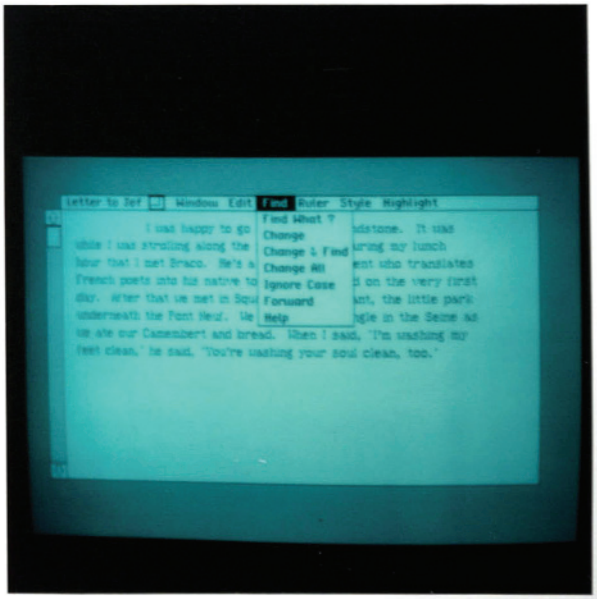
In some cases the PARC precedent spurred the Apple team to invent new designs that were not part of the Alto, as for example with the partly covered window. Bill Atkinson saw overlapping windows during his hour and a half at PARC, and he thought he saw text and graphics being refreshed in one of the 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:

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

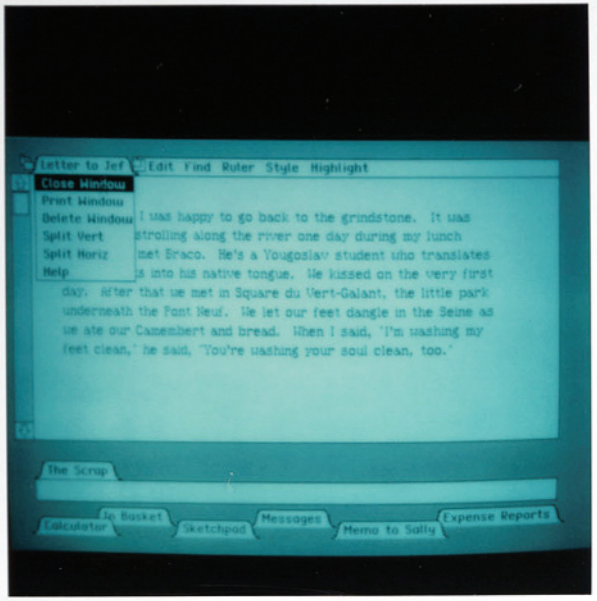
Photos
Bill Atkinson

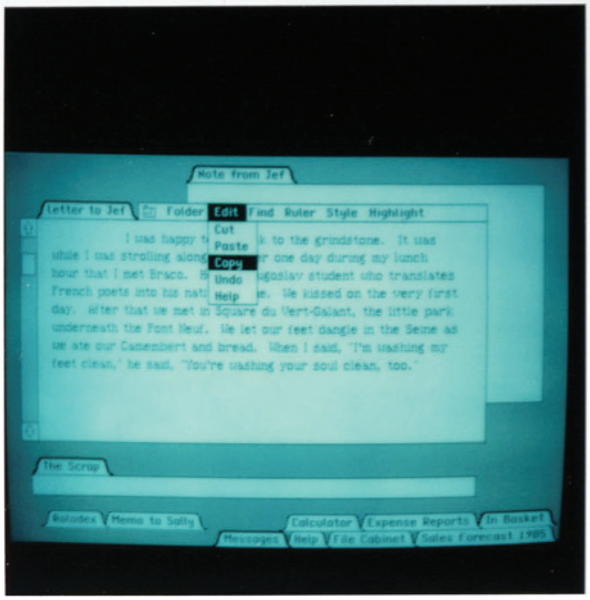
No, no! We didn't have a clever way of doing that; when you wanted to draw into a window with another one on top of it, you had to completely redraw the one that was behind, and then the other one on top.

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



Pull down menus, grow icon next to title, seperate 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

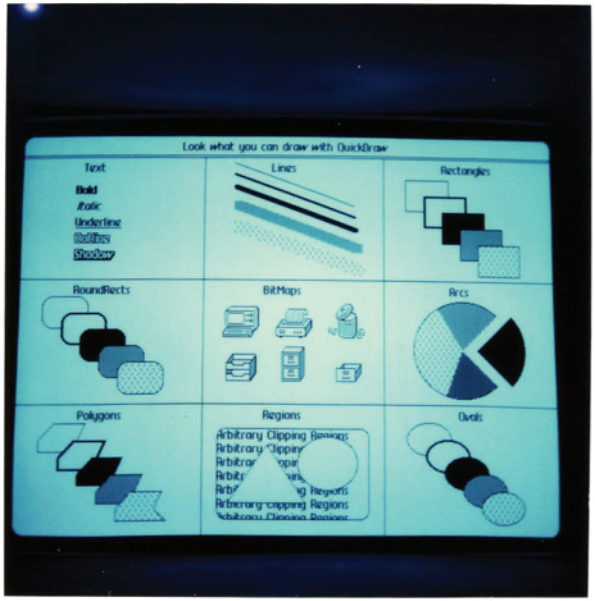
As the first user interface specification for Lisa was being formulated, Bill and Larry formed a close partnership and developed a round-the-clock working relationship, similar to the way that Larry had worked with Tim Mott on Gypsy. Bill would work nights and Larry would work days. During the night 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.

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.

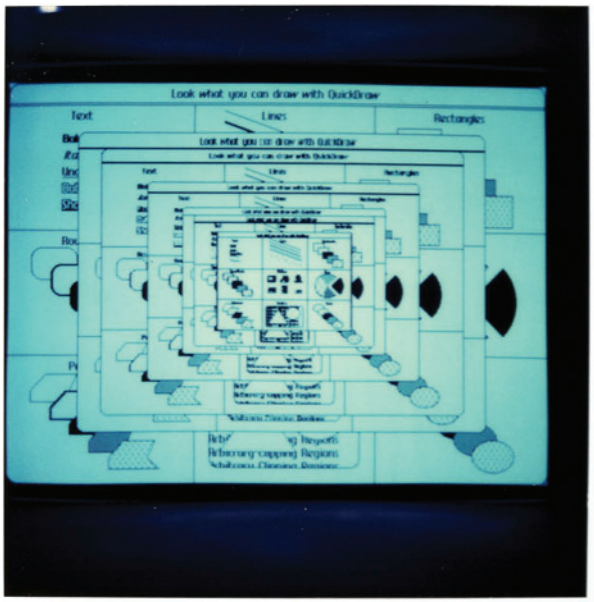
■ Polaroids of development of designs for pull-down menus

Photos
Bill Atkinson

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

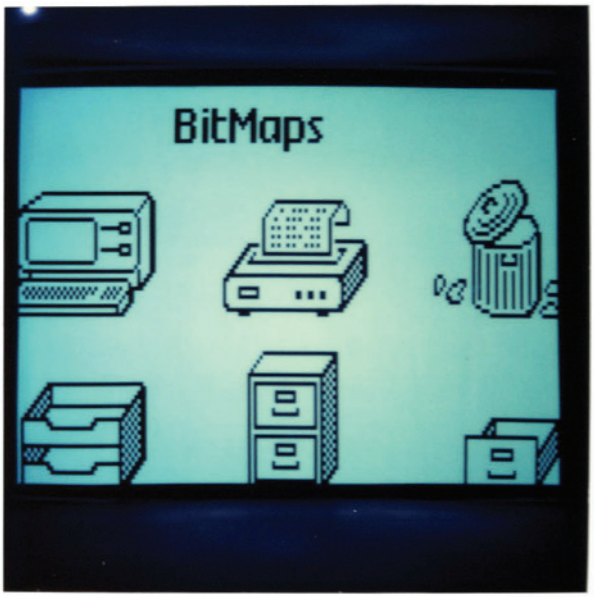


QuickDraw Sampler



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.

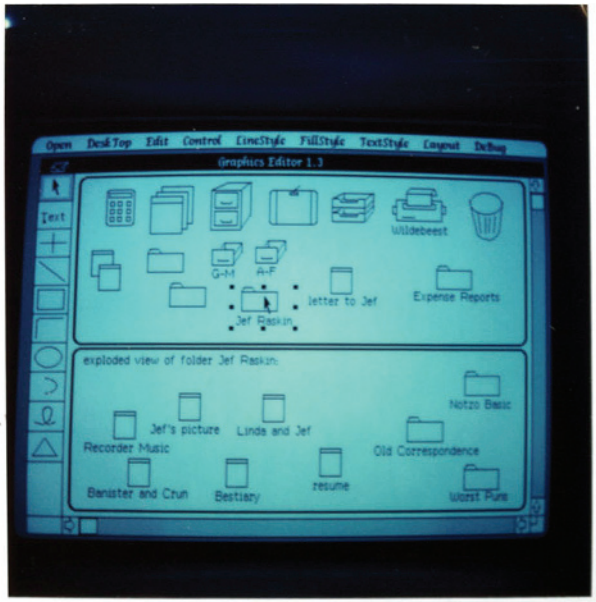
■ Polaroids of QuickDraw manipulations, bitmaps of icons

Photos
Bill Atkinson

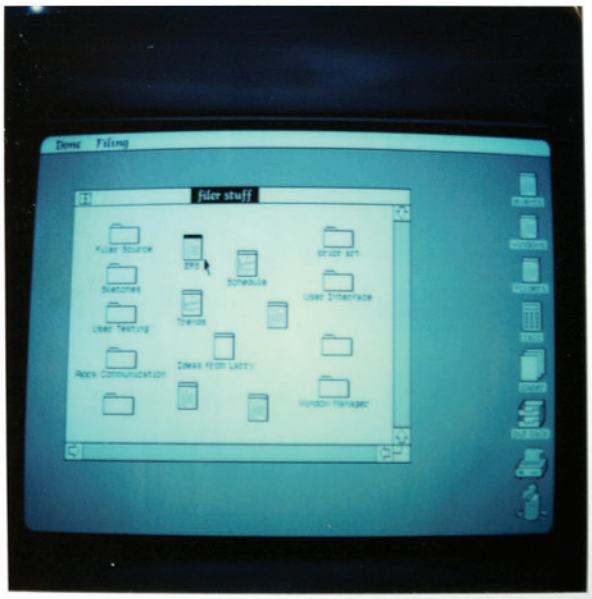
"Why don't we move it to the top of the window instead of the bottom?" Larry suggested.

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:

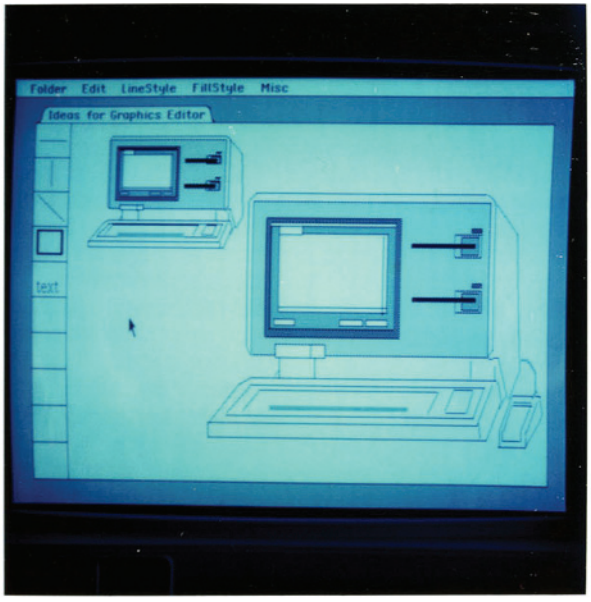
> In one night he developed the entire pull-down menu system! Everything! He hadn't just moved it to the top of the screen; he had the idea that as you scanned your mouse across the top, each menu would pop down, and they would ruffle as you went back and forth,



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 you 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.[6] 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 and Bill Atkinson, holding a Mac, in 1983

Photos
Bill Atkinson

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

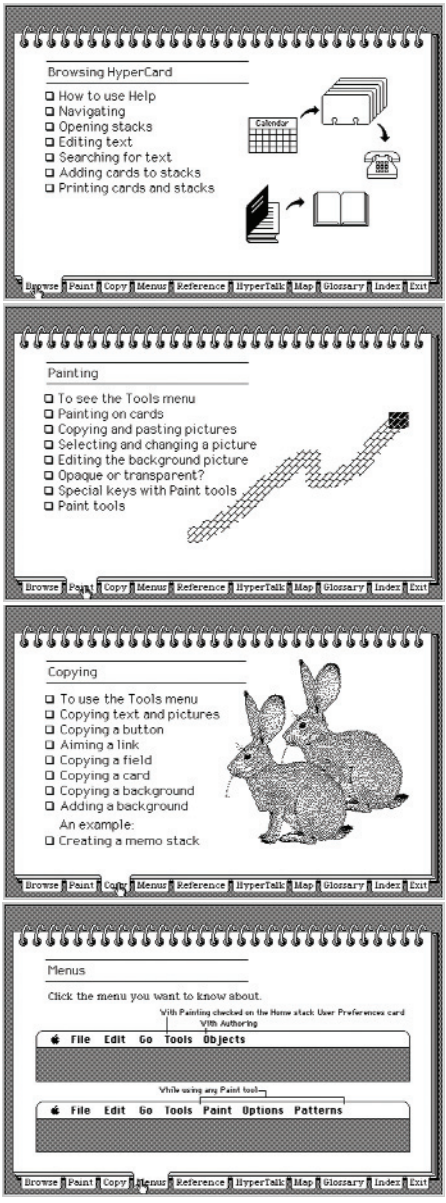
friendly, and the physical design had a perky cuteness that was very appealing. The famous 1984 Super Bowl ad by Ridley Scott had an enormous effect, and all aspects of the supporting materials, such as packaging and print matter, had panache. Steve also managed to develop a culture for the company that 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[7] 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.



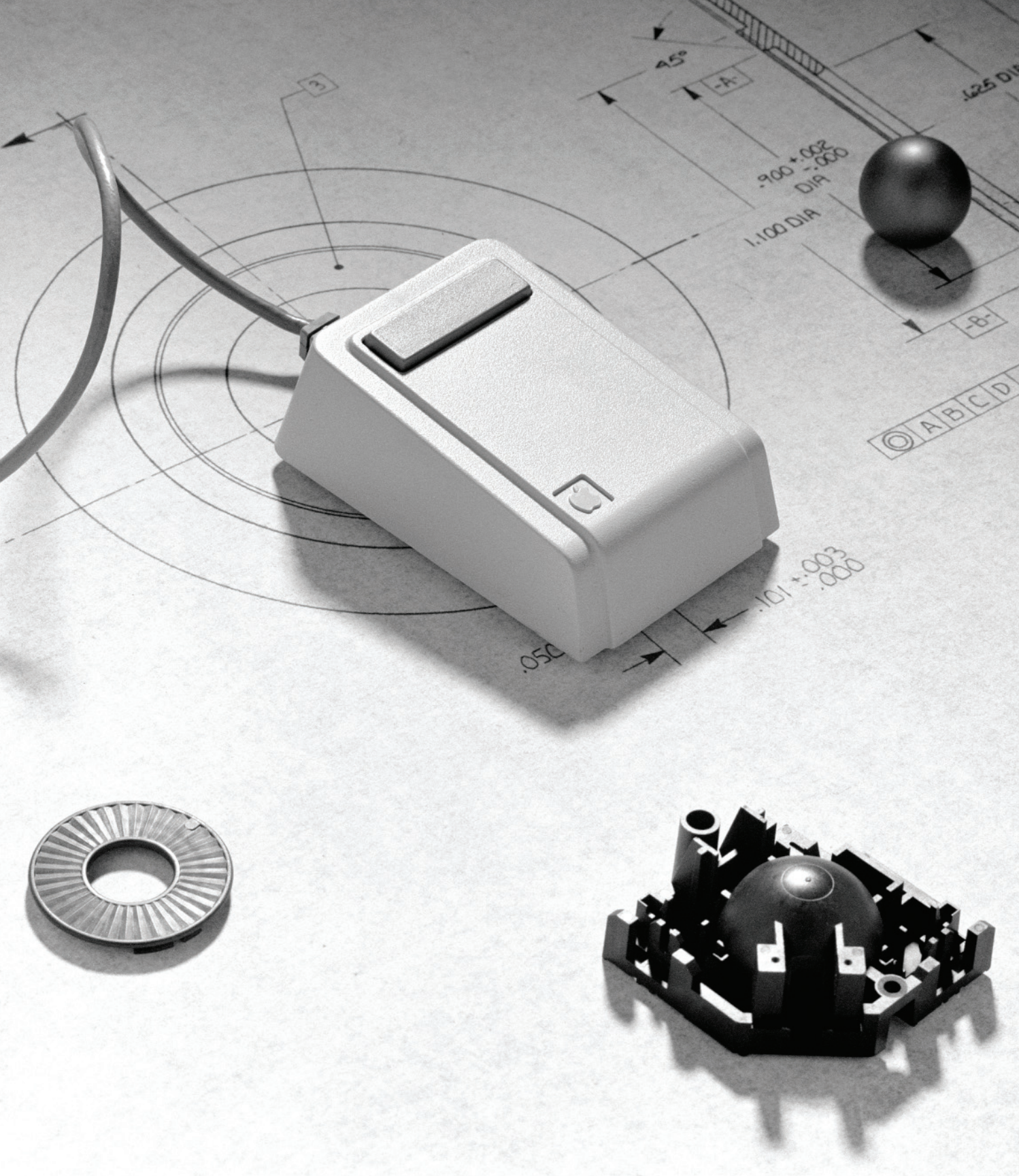
- HyperCard help stack—browse
- HyperCard help stack—paint
- HyperCard help stack—copy
- HyperCard help stack—menus

HyperCard

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.


45°
-A-
.900 +.002 -.000
DIA
1.100 DIA
-B-
A B C D
.050
.101 +.003 .000

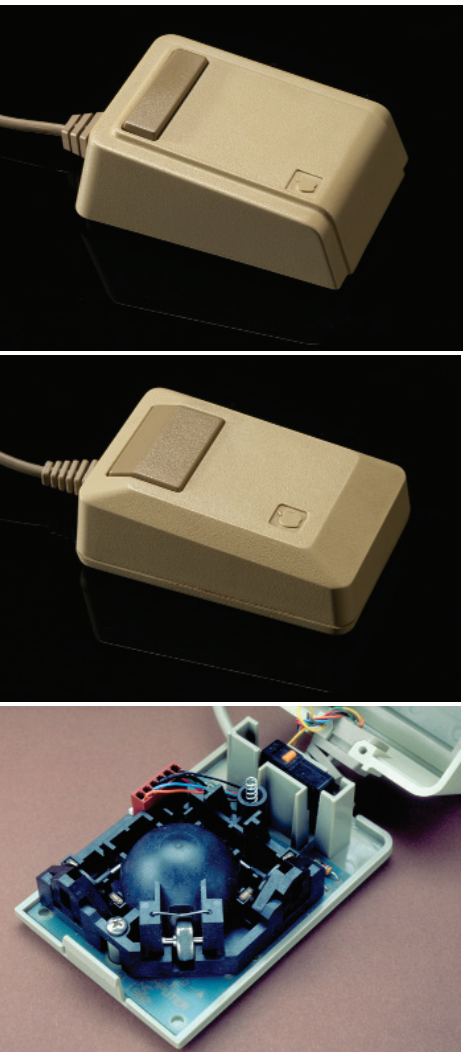
Apple Mice

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!)."[8] 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



- Original Lisa mouse
- Original Mac mouse
- Common mechanism

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[9] 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[10] 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[11] 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

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.

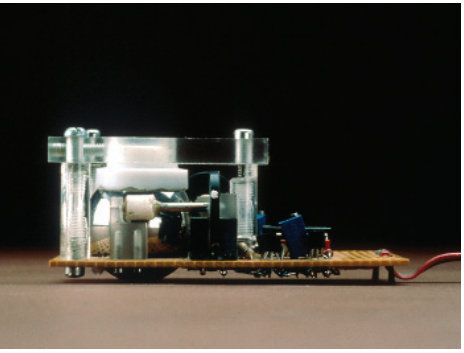
Larry and Bill ran some experiments and proved that the single button was acceptable. Then they went through the whole interface and found alternatives to using the second button, the most controversial of which was using the shift key to find the other end of a large selection. Larry wrote a single-page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

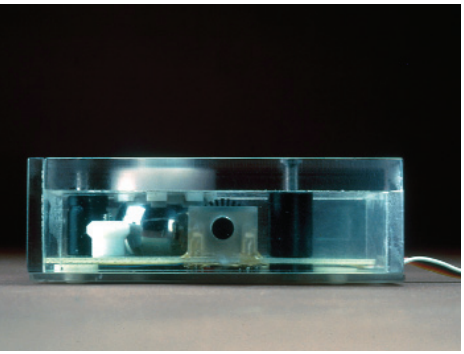
> This is what we need to do! We need to make this for the average person and this is the kind of simplification we need to have. We're going to go with the one-button mouse!

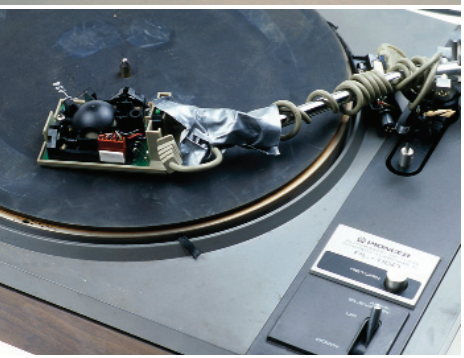
Larry remembers:

> 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."[12]

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.

Photo Author

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.


Microsoft

Paul Bradley

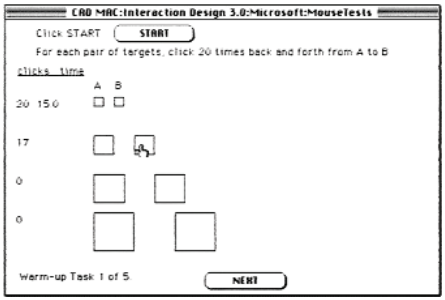
Microsoft Mouse

Microsoft mouse with development models

Photo
Rick English

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,[13] 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.

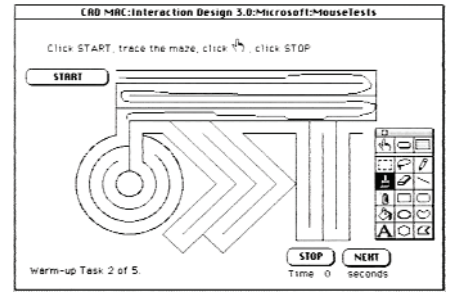
Next they were asked to trace their way through a maze to reveal steering ability; this showed a dramatic advantage in moving the ball to the front of the mouse, so that it lay between thumb and finger, rather than in the conventional location further back.

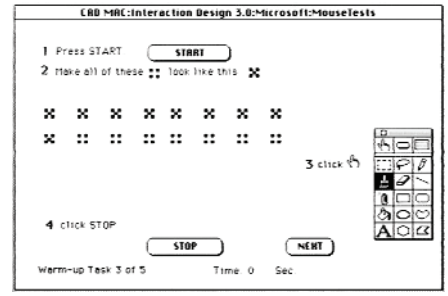
A task to test precision asked the user to position the cursor exactly in the center of arrays of dots, and then to click; this revealed the importance of having the buttons on the top rather than on the side.

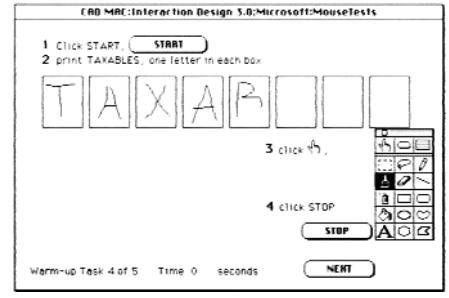
Writing the word "TAXABLES," with one character in each of eight boxes, required repeated short strokes with the button down and then up; for this task a mouse is much better than a trackball, where the buttons are hard to hold while rolling the ball.

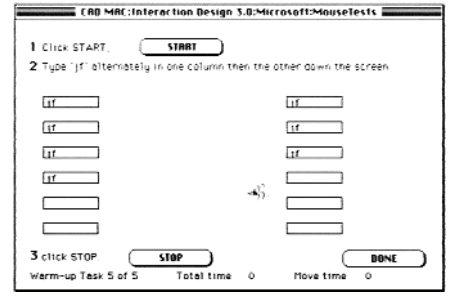
The final task required typing, then pointing, then typing, then pointing, again and again; it measured homing time as you move from keyboard to pointing device, showing that the mouse did just as well as devices with fixed positions.

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.







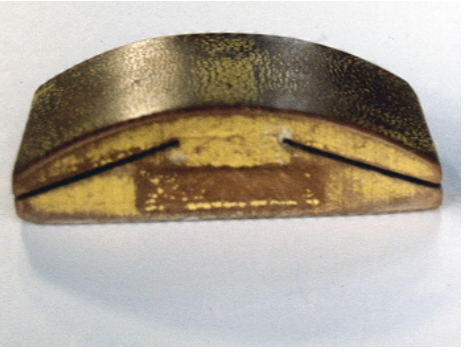


Test for mouse designs—maze task ■
Test for mouse designs—precision task ■
Test for mouse designs—writing task ■
Test for mouse designs—homing task ■

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.



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

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

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.

■ Underside of mouse

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

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.

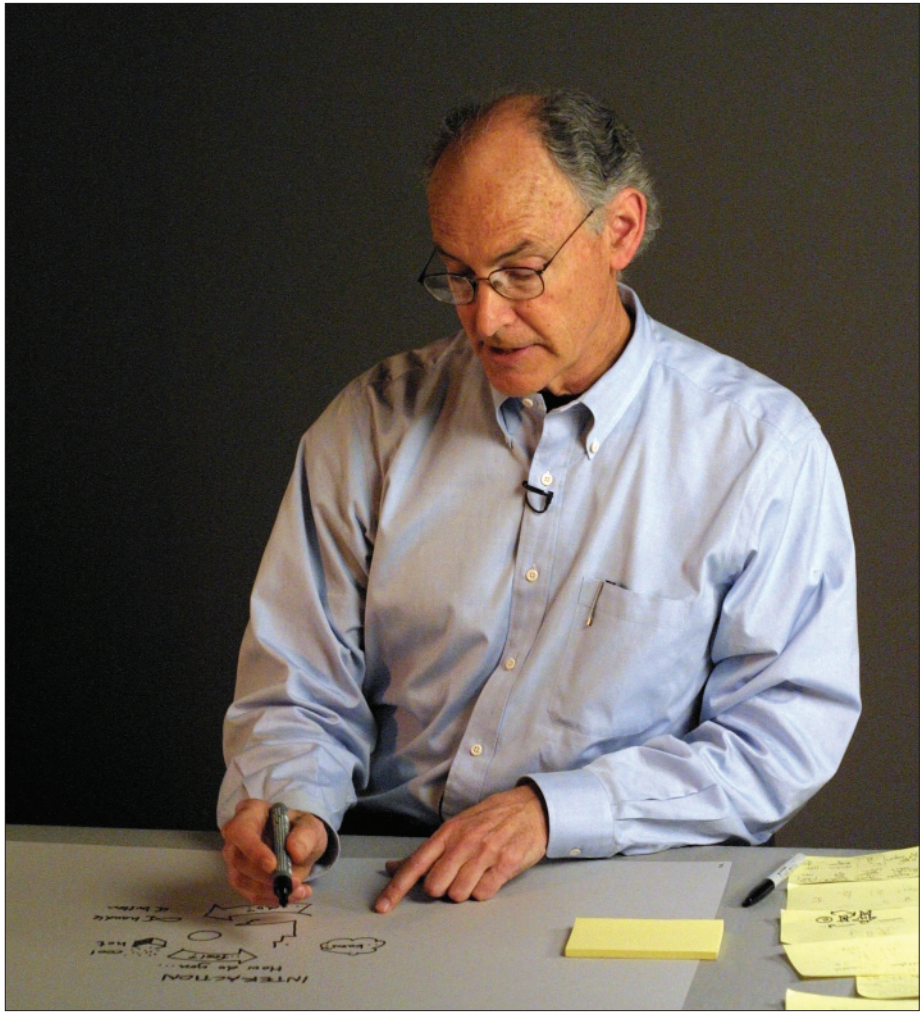


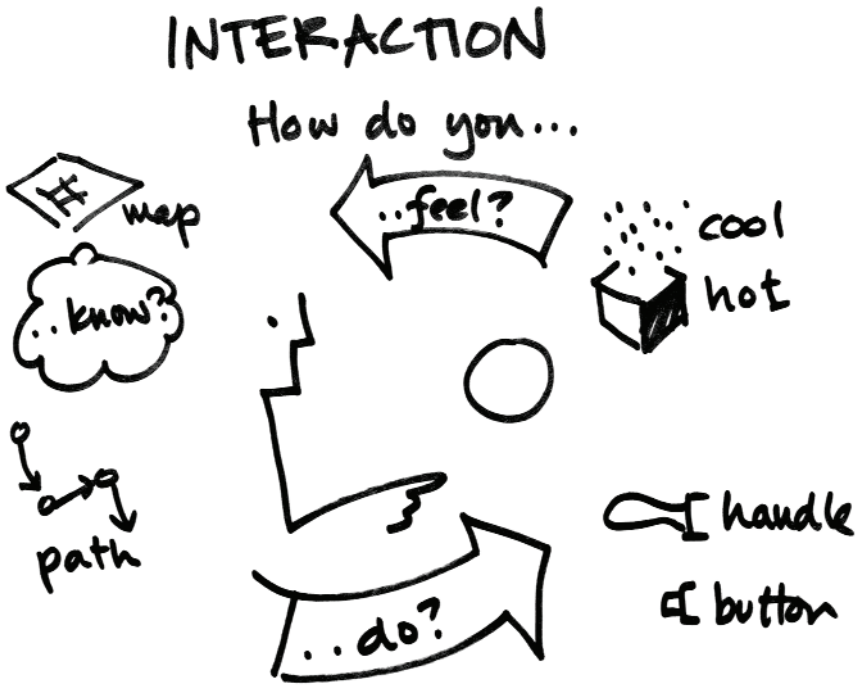
Photo Craig Syverson

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?"

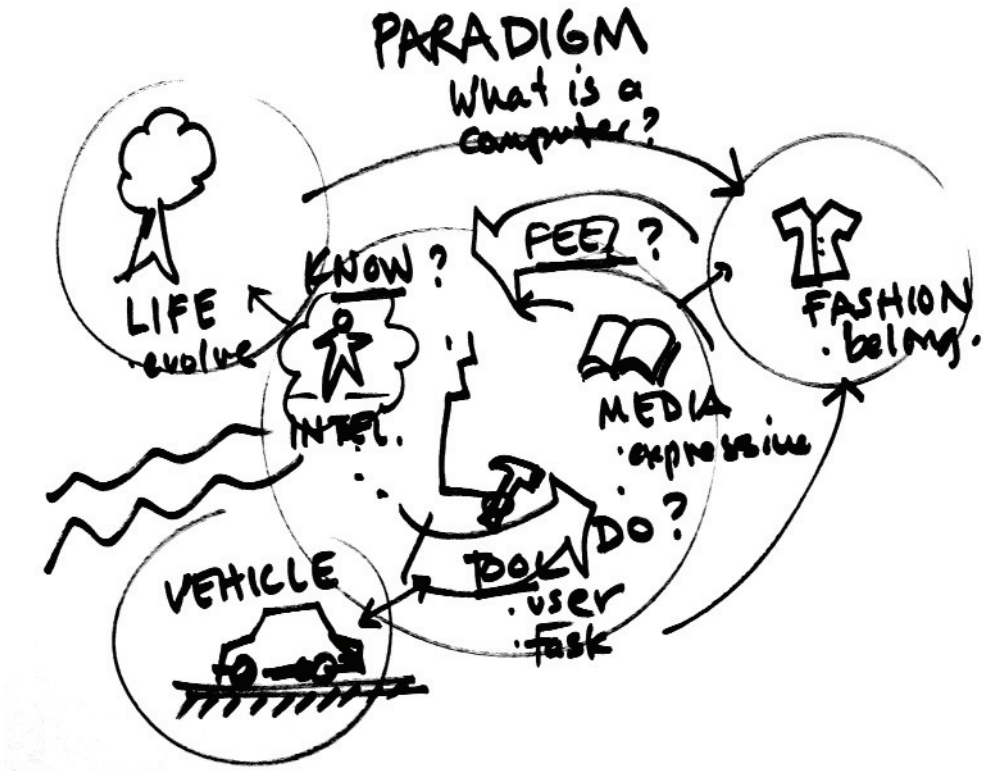
How do you get feedback? McLuhan made the distinction between what he called "fuzzy," or "cool," media, and "distinct," or "hot," media. Early TV was a cool medium, with its fuzzy images. Cool media draw you in. A book with careful printing or a gravestone with carved lettering is hot, or immutable—you 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

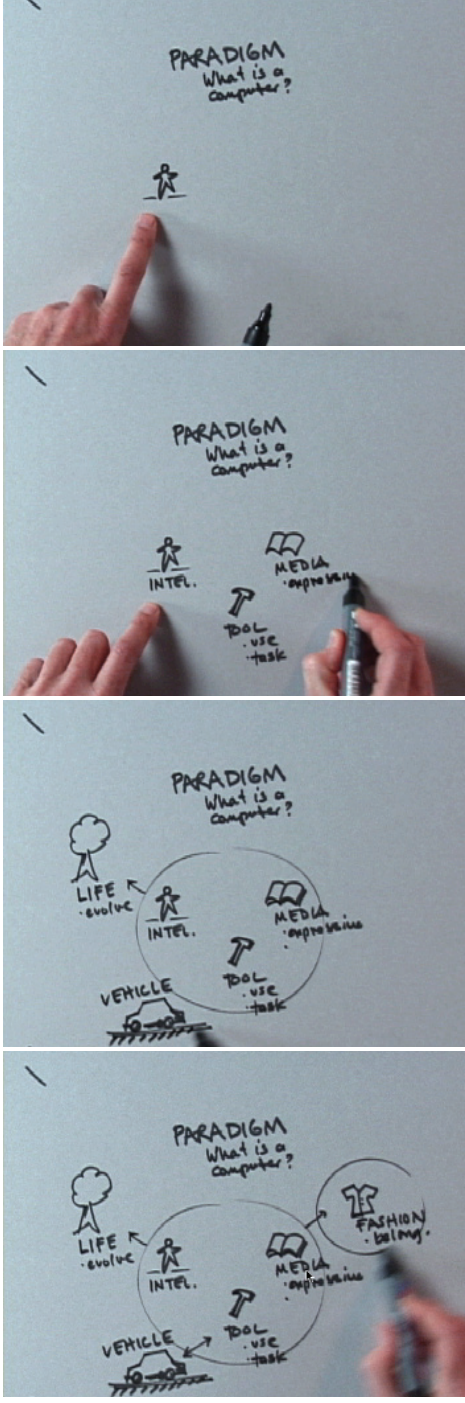
Starting in the mid nineties, people have been talking about computer viruses or computer evolution; they are thinking of artificial life. When the program has been written, it is capable of evolving over time—getting better and adapting. The programmer is in a way giving up responsibility, saying that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

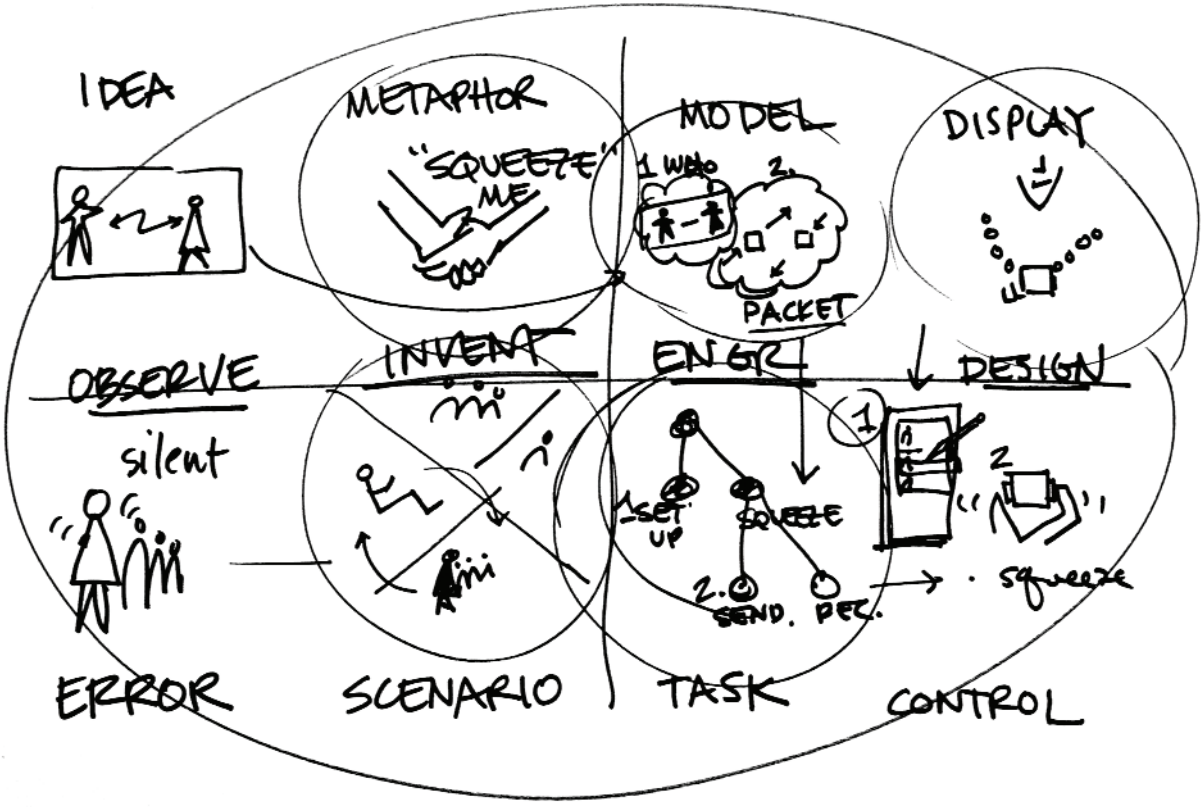
In order to create a conceptual model that users will understand, you have to have a clear picture of what they are thinking about. The mode that they are in depends on what the task is, and what they are trying to accomplish. How they can move from one mode or model to another, or from one environment 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

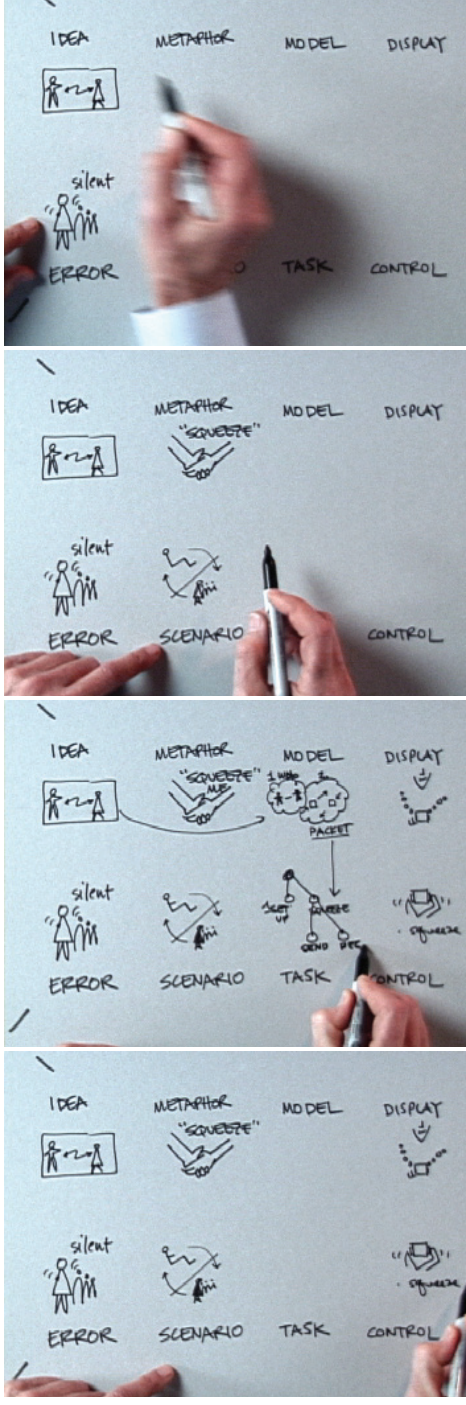
The metaphor is that she really wants something that is more like holding hands, so that she can hold hands at a distance and give a squeeze. She is thinking about a scenario that Fred is at home, and she is taking longer shopping than she had expected. She wants to be able to communicate without interrupting 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,[14] 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.

Photo Author

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.


Stickies File Edit Note Color Window
About This Mac
Mac OS X
Version 10.2.8
Memory: 512 MB
Processors: Dual 1 GHz PowerPC G4
More Info...
™ & © Apple Computer, Inc. 1
Mac OS X
When Cordell
were rolling. The c
margins to sponsor
exciting to be a desi
interference from ar
was a great place t
re than IBM at 26%
, and some good-ti
lot of freedom te
Syste
iPhoto
04 Paradigms
00:04:39
Show All
Displays
Sound
Network
Startu
Personal
Desktop
Dock
General
Hardware
CDs & DVDs
ColorSync
Displays
Internet & Network
Internet
Network
QuickTime
System
Accounts
Classic
Date & Time
Back
video
Brenda Laurel
Compilations
Desktop Folder
Durrell Bishop
Excerpts Radio Watch
Excerpts reel intros
Connect to laptop: computer
name "Bill Moggridge",
Password "mogric"
or Karinmogg
1111aa
or alexmoggridge

Cordell Ratzlaff

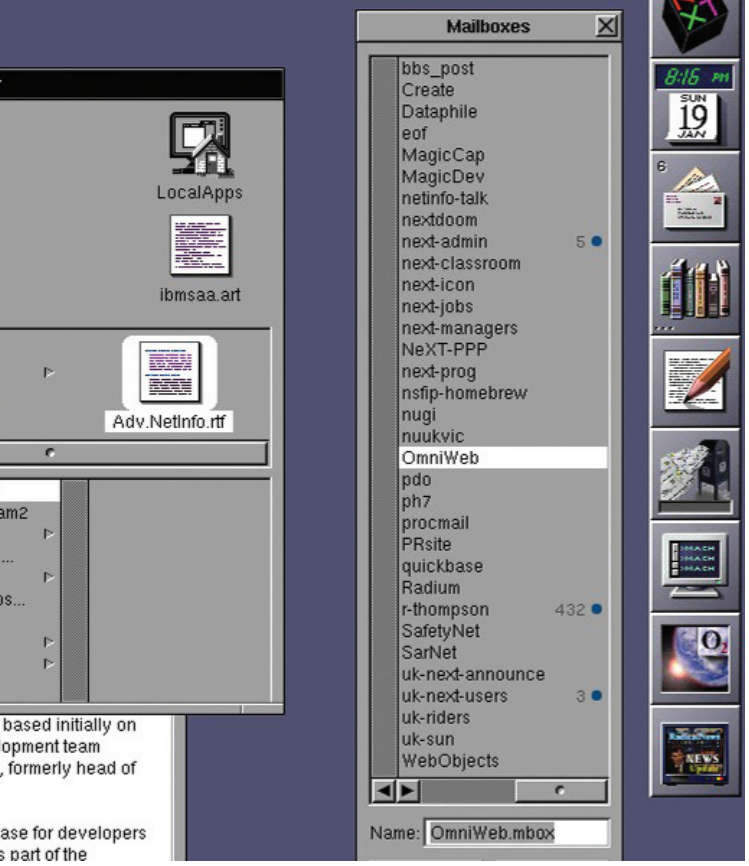
Mac OS X

■ Mac OS X

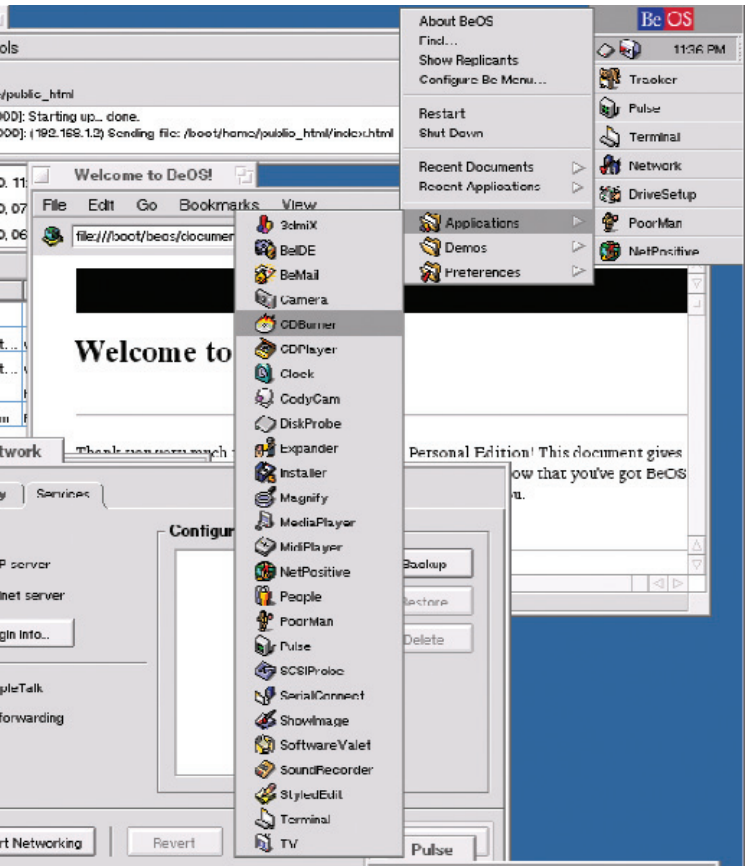
Photo
Author's screen capture

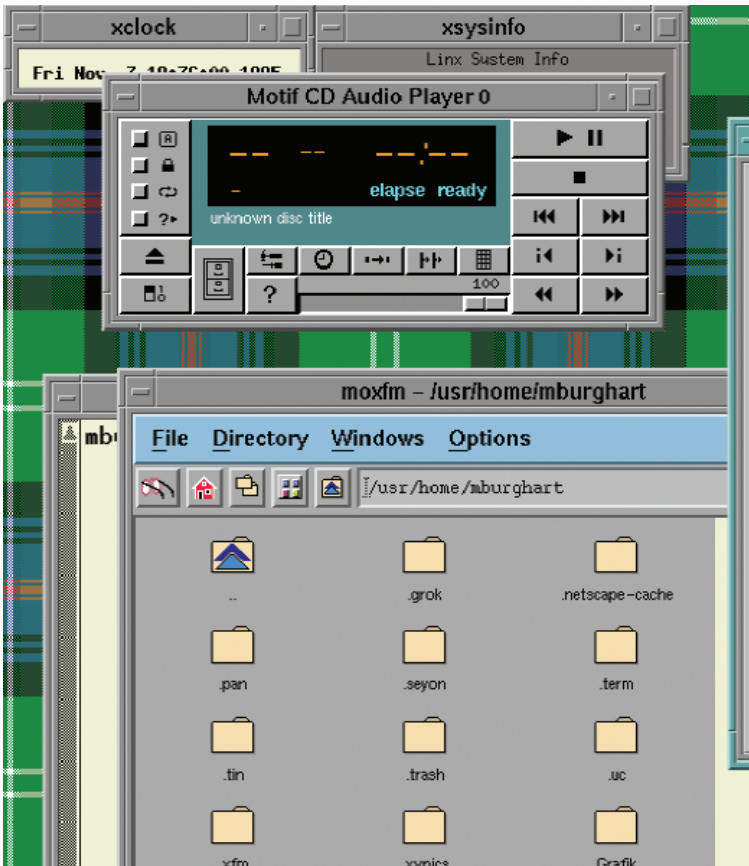
When Cordell Ratzlaff arrived in 1990, Apple was a great place to work. The good times were rolling. The company had a bigger market share than IBM at 26 percent and large enough profit margins to sponsor generous development budgets—and some good-time extravagance. It was exciting to be a designer at 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.


Mailboxes
bbs_post
Create
Dataphile
eof
MagicCap
MagicDev
netinfo-talk
nextdoom
next-admin
next-classroom
next-icon
next-jobs
next-managers
NeXT-PPP
next-prog
nsfip-homebrew
nugi
nuukvic
OmniWeb
pdo
ph7
procmail
PRsite
quickbase
Radium
r-thompson
SafetyNet
SarNet
uk-next-announce
uk-next-users
uk-riders
uk-sun
WebObjects
Name: OmniWeb.mbox
LocalApps
ibmsaa.art
Adv.NetInfo.rtf


Empty Trash...
Eject ⌘E
Erase Disk...
Sleep
Restart
Shut Down
Joy CD
General
Views
Simple Finder
Dynamic Docs 1 Folder
Dynamic Documents
Badges1
About This Computer
Mac OS computer
Mac OS 8.5
Built-in Memory: 96 MB
Virtual Memory: 97 MB used on Macintosh HD
Largest Unused Block: 79.5 MB
™ & © Apple Computer, Inc. 1983-1998
Jigsaw Puzzle 556 K
Mac OS 17.6 MB


About BeOS
Find...
Show Replicants
Configure Be Menu...
Restart
Shut Down
Recent Documents
Recent Applications
Applications
Demos
Preferences
Tracker
Pulse
Terminal
Network
DriveSetup
PoorMan
NetPositive
Welcome to BeOS!
File Edit Go Bookmarks View
Welcome to
3dmiX
BeIDE
BeMail
Camera
CDBurner
CDPlayer
Clock
CodyCam
DiskProbe
Expander
Installer
Magnify
MediaPlayer
MidiPlayer
NetPositive
People
PoorMan
Pulse
SCSIProbe
SerialConnect
ShowImage
SoftwareValet
SoundRecorder
StyledEdit
Terminal
TV


xclock
xsysinfo
Linux System Info
Motif CD Audio Player 0
elapse ready
unknown disc title
moxfm – /usr/home/mburghart
File Directory Windows Options
/usr/home/mburghart
.grok
.netscape-cache
.pan
.seyon
.term
.tin
.trash
.uc

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[15] 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,[16] 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

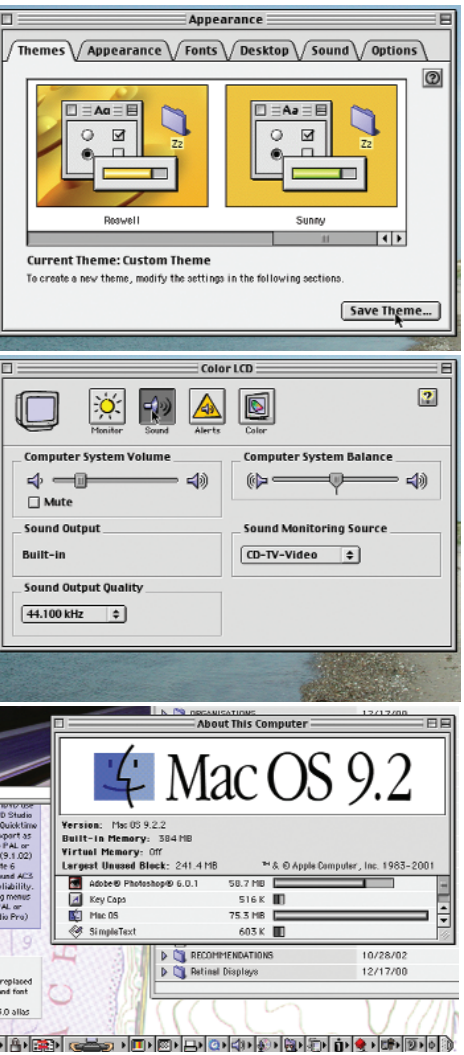
In the mid nineties Apple had a project called Copeland, which went on for a number of years. It was meant to be the next-generation operating system for the Macintosh. Eventually everyone realized that Copeland was never going to come to fruition, so Apple started looking for an operating system to 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[17] 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

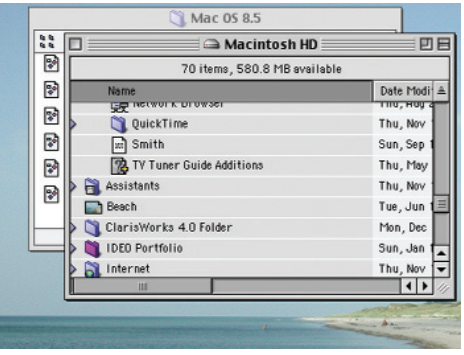
> point on there was never any debate that we were going to put a new human interface on top of Mac OS X.

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



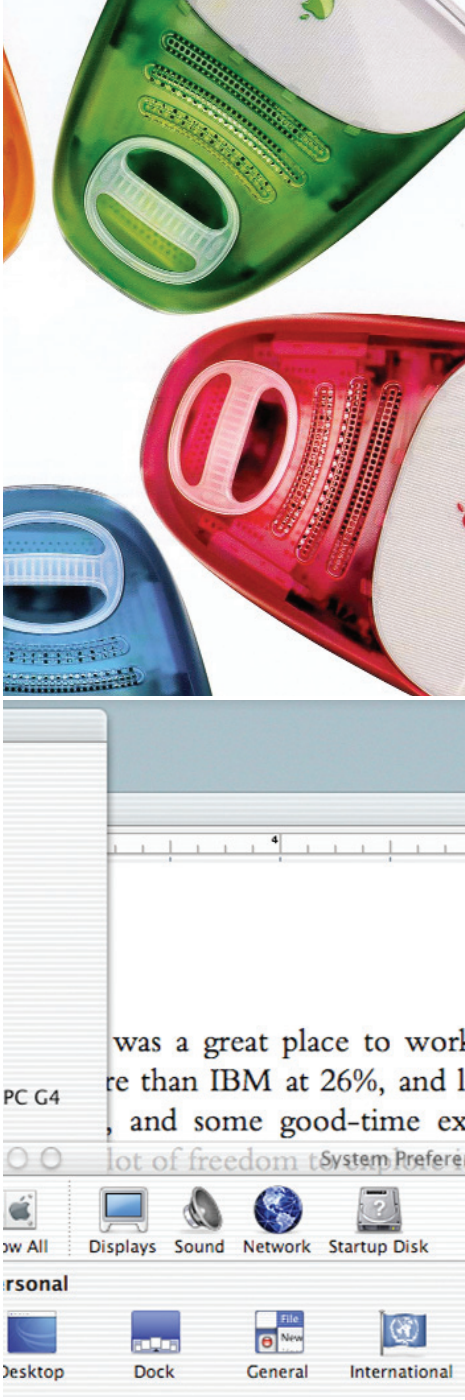
■ Mac OS 8.5 Windows

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.

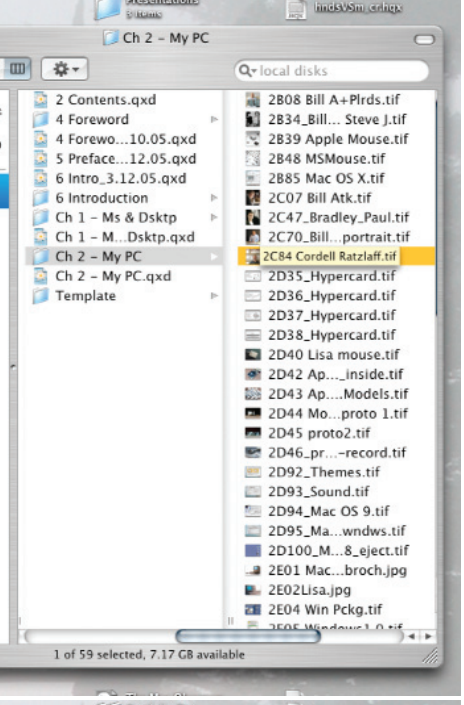
Cordell saw a great opportunity to change to an appearance that was fresh and fun, in contrast to the existing state of the art. He decided to change from gloomy, square, and beveled, to light, fun, and colorful, with a very fluid expression. He asked, What's the opposite of a computer interface? He 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:

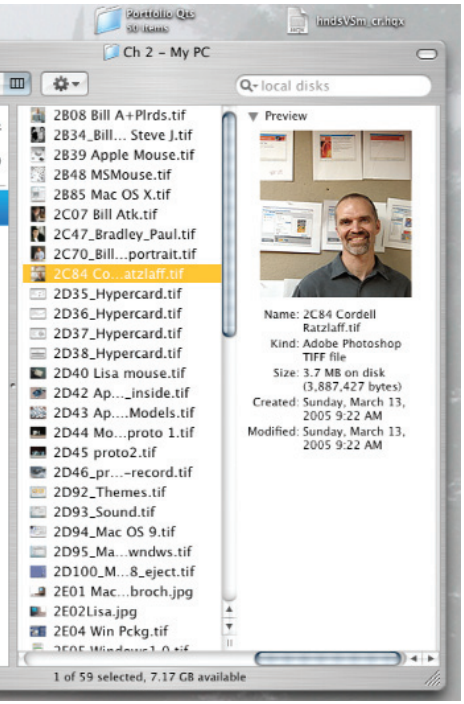
> We didn't want to use translucency gratuitously; we wanted to make sure that translucency added value to the experience. A good example of that was window layering. As windows move to the background they become translucent; in fact they become more translucent the farther back they are in the window 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



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.

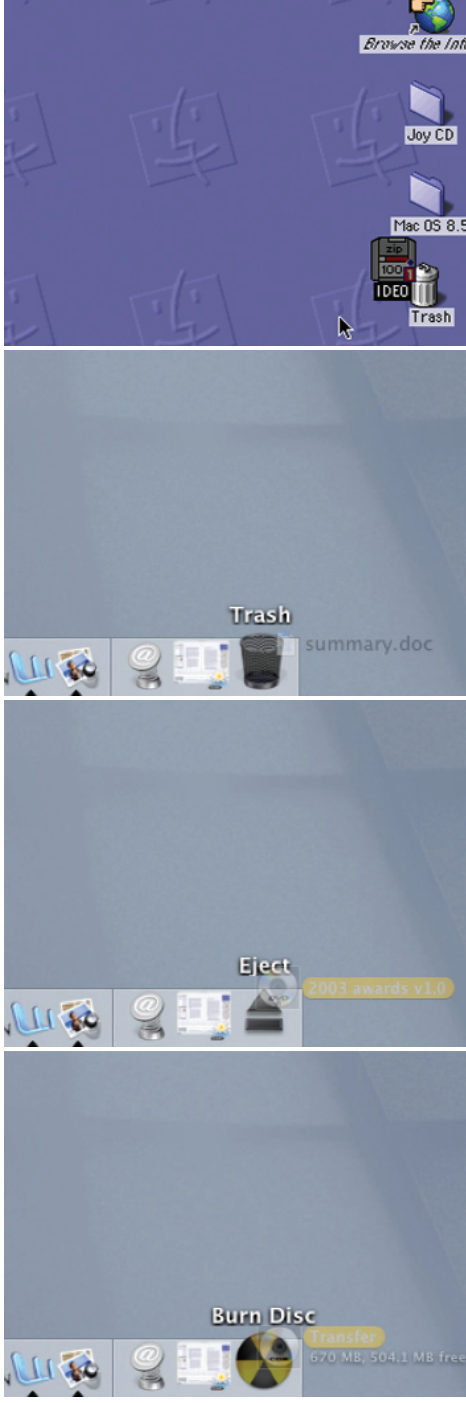
The Time Dimension

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.

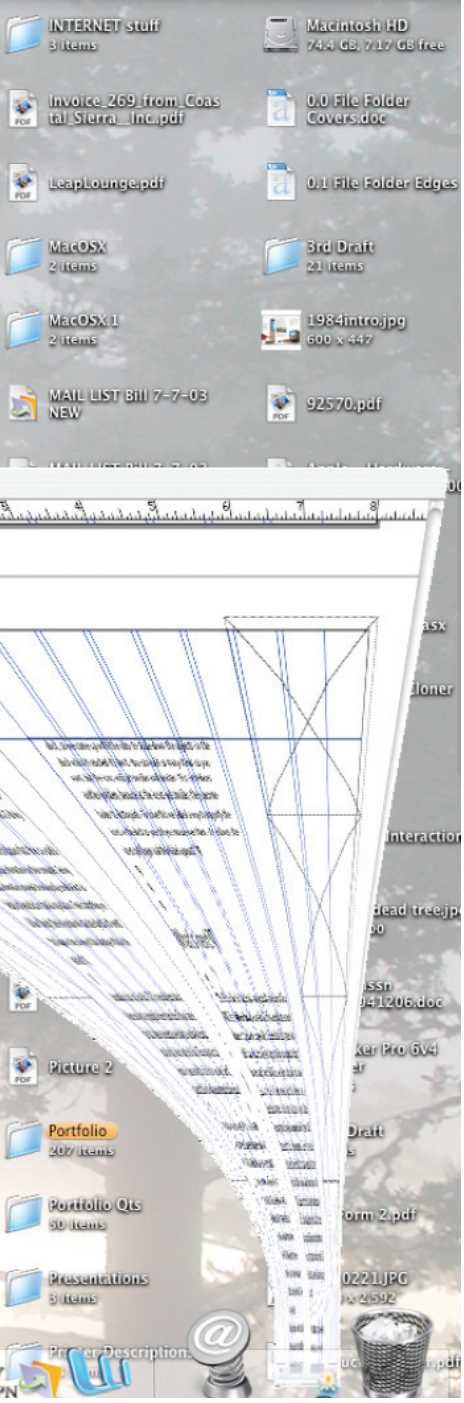
In previous versions of Mac OS, when you selected an application, all of that application's windows came to the forefront. In Mac OS X, those windows appear in the order that you used them. If, for example, you are switching from Photoshop to an HTML editor, the Photoshop window is only one window behind; 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

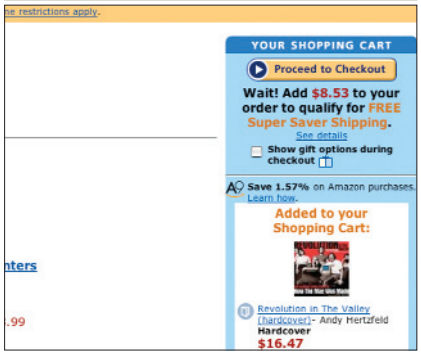
the dock when it reached its limit. You can add as many items as you want, but the icons will get smaller and smaller. This introduces another problem, because as the icons get smaller, they become harder to distinguish. To solve this we added a way to magnify the icons in the dock as you roll your mouse 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[18] 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 ■



- 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[19] 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!