Animation at the Interface

Ronald Baecker
Ian Small

Dynamic Graphics Project
Computer Systems Research Institute
and Department of Computer Science
University of Toronto

DESPITE MANY YEARS OF ADVANCES in computer graphics hardware and software, and in human interface technologies, designs, and styles, our user interfaces are still primarily static. The purpose of this chapter is to review ways in which dynamic imagery and animation have been used in interfaces to date, and to sketch some ways in which they could be used to enrich the interfaces of the future. Our thesis is that current uses of animation at the interface have barely scratched the surface of what is possible and interesting.

What Is Animation?

> Animation is not the art of *DRAWINGS-that-move* but the art of *MOVEMENTS-that-are-drawn*. What happens between each frame is more important than what exists on each frame. Animation is therefore the art of manipulating the invisible interstices that lie between frames. The interstices are the bones, flesh, and blood of the movie, what is on each frame, merely the clothing. [Norman McLaren, 1968, as quoted in Baecker (1969)]

Animation is "the graphic art which occurs in time" [Martin, 1969]. It is a dynamic visual statement, form and structure evolving through movement over time.

The Saturday morning cartoons with which we are all familiar barely scratch the surface of the possibilities of animation. Yes, animation can be used for entertainment and for storytelling, but it can also be used to

Published in The Art of Human-Computer Interface Design, Brenda Laurel (ed) Addison Wesley 1990

establish a feeling or a mood, as a diversion, for drama, as identification, for selling or persuasion, and for explanation and teaching.

Inanimate objects can be energized with feeling and emotion [*Luxo Jr.*, Lasseter, Reeves, et al., 1986; *Red's Dream*, Lasseter, Reeves, et al., 1987]. Orders of magnitude of time and space can be compressed and made accessible [*Cosmic Zoom*, Verrall, 1968; *Powers of Ten*, Eames, 1971]. Time and space can be combined in innovative ways [*Pas de Deux*, McLaren, 1967].

It is also important to realize that there are numerous animation techniques [Laybourne, 1979] with which one can produce effective dynamic imagery that are far simpler than the full motion of two-dimensional or three-dimensional character animation. Thus, we should look to the language of cinema for models of how our interfaces could behave. This suggests that we include, as global changes to the entire screen, the *cut*, the *fade in (fade out)*, the *dissolve*, the *wipe*, the *overlay*, and the *multiple exposure*. Locally, within a region of the screen, interfaces should allow the *pop on (pop off)*, the *pull down (pull up)*, the *flip*, and the *spin*. Finally, they should allow, as either global or local phenomena, *reverse video*, *color changes*, *scrolling*, *panning*, *zooming in (zooming out)*, and *close-ups*. We shall see below how such effects can make an interface more memorable and vivid, more captivating and enjoyable to use.

How Is Computer Animation Produced?

Animation in all forms and media, whether based on traditional painted cels, clay models, or computer-generated imagery, whether rendered in flipbooks, recorded on videotape, or displayed on a computer screen, is composed of sequences of static images changing rapidly enough to create the illusion of a continuously changing picture.

Animation depends on the fact that images formed on the human retina persist for some time after the source of the image has disappeared. Because of this, a set of images rapidly displayed in succession appear to blend together into a continuum. The speed at which images must be displayed in order to achieve apparently continuous imagery depends on the persistence time. For example, standard film speed is 24 frames (distinct static images) per second, and the NTSC video standard specifies 30 frames per second (which is actually produced as 60 fields per second, a field being either all the even or all the odd scan lines of the image). Animated sequences presented at 15 frames per second will be perceived by the average viewer as being jerky.

Animation produced on modern computers is fettered by the discrete nature of the computer's display. A modern display is composed of a set of colored or monochrome pixels arranged in a rectangular array called a raster. The contents of the raster are stored in a hardware device called a frame buffer. Images are created by setting each individual pixel in the raster to an appropriate color; when viewed as a whole, the collection of densely packed pixels can appear to represent a continuous image. Although this concept sounds crude in comparison to high-grade film stock, surprisingly good results can be obtained if care is taken in the way images are created, especially when a series of images is used to create an animated sequence.

While the pixel and the raster are the basic tools for displaying computer imagery, the step from a set of static images to an animated display is not a simple one. Effective animation on a computer display generally depends on some degree of hardware support, a number of software tricks, or a combination of both. Although hardware support can speed up the rate at which animation can be produced, the clever use of software techniques can also accelerate and improve a given platform's animation potential. These techniques include the use of *double buffering* [Baecker, 1979], *color map animation* [Shoup, 1979], and the use of *incremental updates* to redraw only those portions of a frame which have changed, a technique pioneered by the programmers of early videogames.

Animations can be specified in any number of ways (they can even be described in words, although they lose most of their magic in the telling), and computers can generate animation using a variety of techniques. *Picture-driven animation* [Baecker, 1969] creates dynamic sequences through the appropriate selection and positioning in each frame of one animation cel chosen from a family of similar cels. *Keyframing* [Burtnyk and Wein, 1971] relies on the specification of static images at a number of particular frames (keyframes). Based on the surrounding keyframes, the computer can then interpolate all the frames lying in between, a process called in-betweening. Both of these processes actually have their roots in traditional cel animation production. *Procedural animation* [Reynolds, 1982] is generated automatically from a procedural description of the animation; often the procedure will take a set of parameters, allowing it to produce an entire family of similar animation sequences, the particulars of which depend on the precise values of the parameters.

Each of these techniques has a particular application to and impact on the production of computer animation. The exact mix of hardware support and software availability determines to a significant degree the type, quality, and volume of computer animation that a particular environment can support. But in deciding what kinds of images we actually want to produce, hardware and software considerations are secondary to the *purpose* of the image and the *user's tasks and goals.* How can we employ animation to create more engaging, useful, and usable interfaces? We will examine three roles for animation: to reveal structure, process, and function. For each role, several subroles will be illustrated by scenarios.

Animation of Structure: Exploring Complex Environments

The field of real-time three-dimensional computer graphics has revolutionized design, simulation, and testing. The automotive design team of today can create a complete model of a hypothetical car and animate it in real time on their workstations, placing it in different environments and viewing it from different angles. Only ten years ago, a similar process would have required the construction of a scale or life-size mock-up, a much more expensive and time-consuming process. Our new capabilities accelerate and cheapen the design process, and encourage more alternatives to be explored in greater detail than was previously feasible.

Scenario, Animation of Viewpoint: Explore viewpoints to determine the relationship between an entity and its environment.

Consider the creation of new entities for which spatial considerations are important. The appearance of a hypothetical object from multiple viewpoints, whether the object is considered independently or in relationship to a surrounding environment, is of prime importance to its design. Significant questions include:

- What can be seen and how does it look from a particular viewpoint?
- Are there viewpoints from which the object is particularly unappealing or from which the object clashes with its environment?
- What is the overall impact that the new object has on its environment as one moves through that environment?

These questions can best be answered using animated walkthroughs or flybys. Numerous applications for the animation of viewpoint may be found in CAD systems and CAD applications.

A simple example of the use of animation viewpoints is provided by John Danahy's study of the impact of a proposed building project in the Ottawa Parliamentary Precinct [Danahy, 1988]. Figure A in the color insert illustrates Ottawa's Parliament Hill shown with and without a proposed new building, and also from two different perspectives. The street-level view makes it clear that the proposed new building will block pedestrian sightlines to existing key buildings. The ability to animate the viewpoint is crucial to achieving such insights.

Scenario, Animation of Appearance: Visualize how an object, structure, or system would look under various conditions of light and shade.

Consider the computer simulation of any complex object or structure. Depending on the nature of the object, the user may need to know the following:

- What does the object look like under different lighting conditions?
- How does the object interact with its surroundings under different lighting conditions?

Animating the appearance of the object will help to answer both of these questions. By animating the addition, movement, change of spectral qualities, and removal of different lights, we can simulate the object under different lighting conditions and in transition between those conditions.

Figure B in the color insert shows a computer model of the downtown core of Toronto at two different times of the day, producing radically different configurations of light and shadow.

Scenario, Animation of Alternative Futures: EXAMINE ALTERNATIVE FUTURES TO EXPERIENCE HYPOTHETICAL ENVIRONMENTS VISUALLY OR TO SIMULATE ACTIVITIES TOO DANGEROUS OR DIFFICULT TO VIEW DIRECTLY.

Consider a situation that can evolve in a number of different ways. It is valuable to be able to simulate and visualize these possible futures in order to determine which is most desirable. Questions prompting the simulation and visualization include:

- What are the potential futures and their associated costs and benefits?
- What is the visual impact of each?
- What elements do each of the potential futures have in common?

Questions such as these can be addressed by animating each of the alternative futures. In essence, such animation is the visual equivalent to asking "What if?" questions of a spreadsheet.

John Danahy has made extensive use of the animation of alternative futures in considering various planting schemes for softening the visual impact of power lines on the surrounding environment [*Danahy, 1988*]. *Figure C in the color insert presents four images from a study prepared for the Ontario Hydroelectric Commission in which the effects of different kinds of vegetation were simulated over the entire growth cycle.*

Of the three areas we discuss, animation of structure and its use in applications represents by far the most mature and widespread use of computer animation. It has been spurred for the past decade by the introduction and development of sophisticated hardware accelerators for assisting in the production of real-time animation. This highly specialized field has been embraced by the scientific and engineering communities, who view the use of this technology as a natural and logical extension of their work.

Animation of Process: Visualizing Algorithms and Programs

Another role for animation is in revealing or explaining complex processes such as computer programs. This is part of an activity that we call *program visualization* [Baecker, 1986].

A program is a computation that executes and unfolds over time, so it is natural to use animation as a method of portrayal. Our goal could be to explain the process, to simplify it to make it accessible, or perhaps to reveal its full complexity gradually and in a layered manner.

SCENARIO, ANIMATION OF ALGORITHMS: SIMULATE AND VISUALIZE A CLASS OF ALGORITHMS.

To understand how a particular algorithm or class of algorithms works, it often helps to construct a visual model of what happens when the program executes. Often, these models can be useful in comparing several algorithms that are similar but slightly different. These visual models are designed to address some of the questions frequently asked about algorithms:

- What are the basic characteristics of the algorithm?
- How exactly does the algorithm achieve its goal?
- Is the algorithm implemented recursively, as a loop, or in some other manner?
- What type of data structures does the algorithm use?
- How fast does the algorithm run with respect to other algorithms in the same class?

Each of these questions will be of particular interest depending on the programmer's requirements. Answers will be provided compellingly and convincingly by a successful animation of the algorithm, which can reveal an algorithm's traits, basic strategy, and overall performance characteristics.

In the film Sorting Out Sorting *[Baecker, 1981], a number of different sorting algorithms are animated and compared. The dynamic visual representations of the executing algorithms convey their basic structure in an extremely effective manner, as shown in figure D in the color insert. This figure presents six still images from the algorithm "race" held at the end of the movie, in which the nine algorithms race against each other on identical data, compellingly illustrating the performance differences between the algorithms.*

Another visual experiment included in Sorting Out Sorting *proved instructive. At the end of the film, we included as a recapitulation a 12-times speeded-up version of the entire movie. Animation sequences literally whiz by, but, because we've seen them before in slow motion, they are meaningful even at high speed. They become, in a sense, dynamic pictograms, animated icons representing the algorithms.*

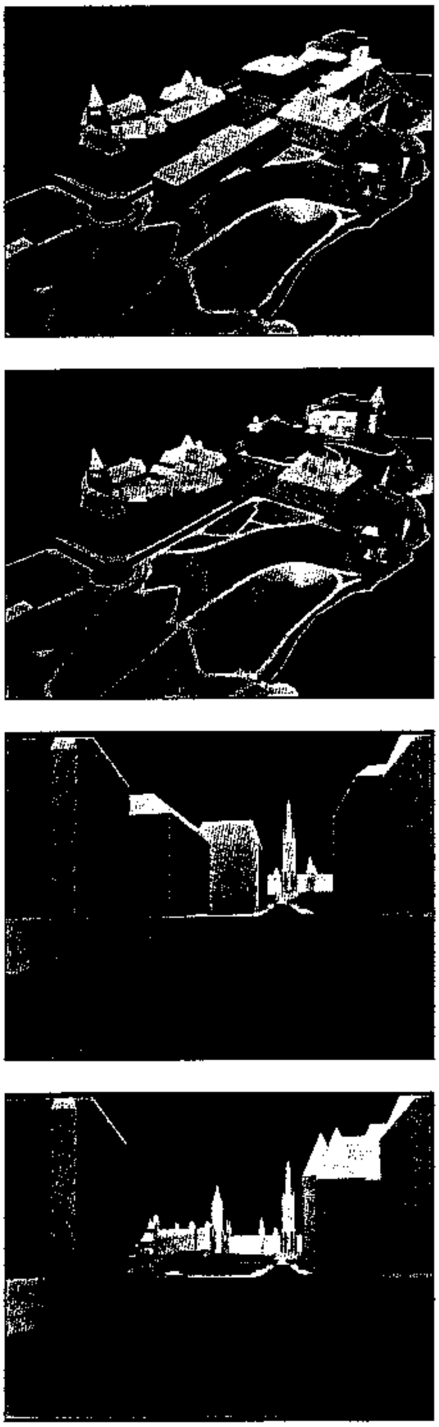


Figure A: Ottawa's Parliament Hill shown from two different perspectives. Images courtesy of the Centre for Landscape Research, University of Toronto.

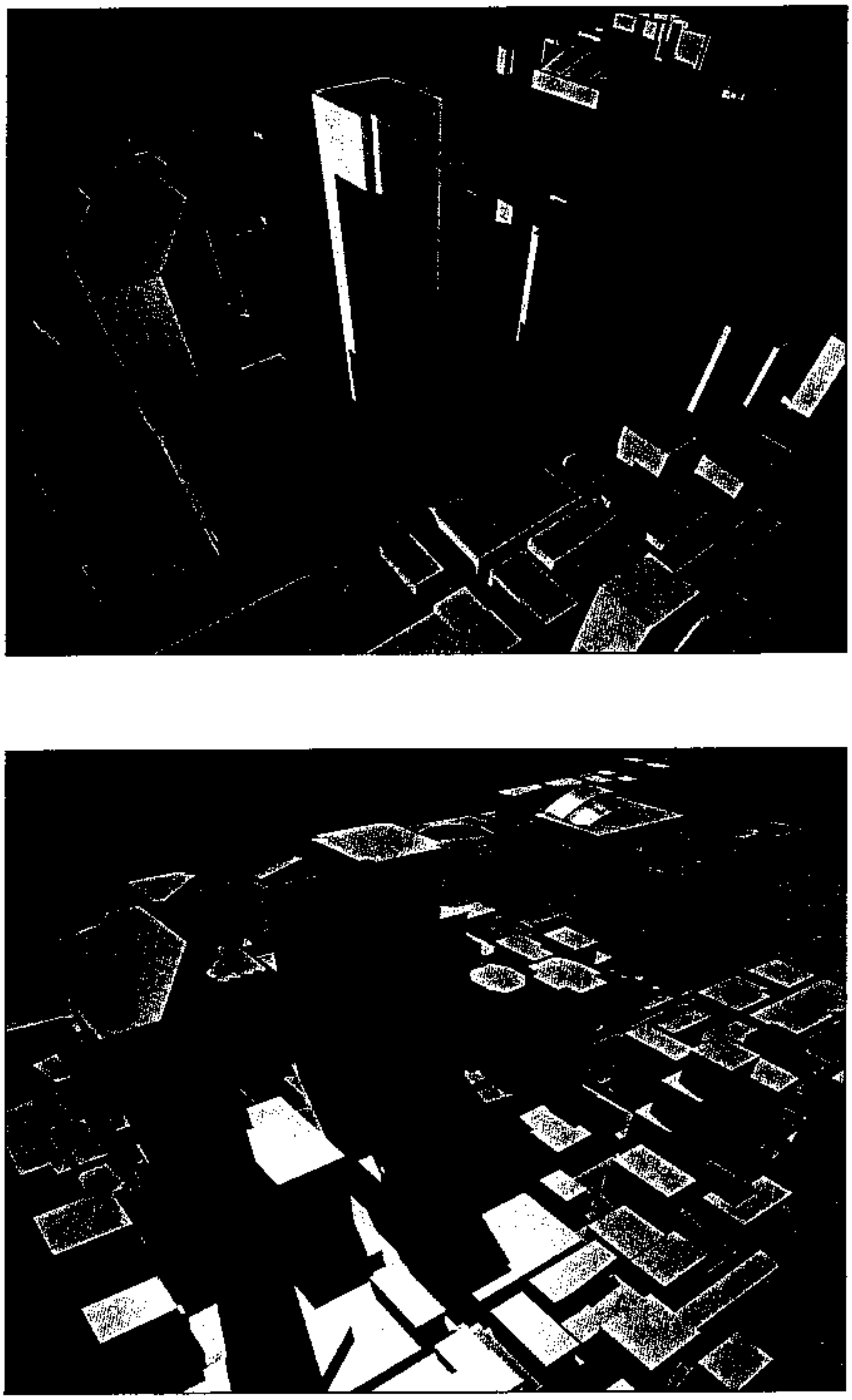
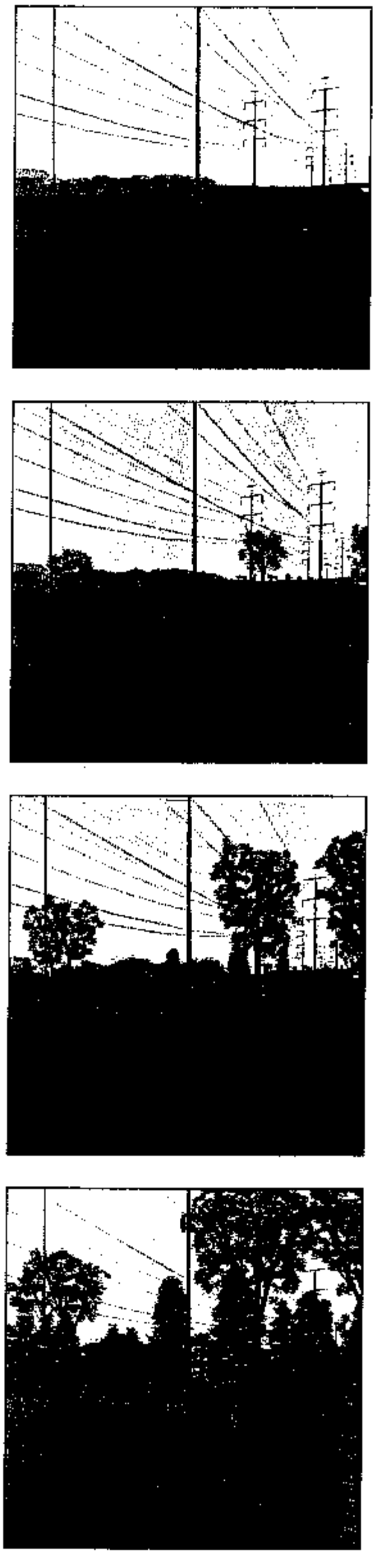


Figure B: A model of the downtown core of Toronto. B1 (top) shows the long shadows characteristic of the early morning, and B2 (bottom) represents the city at noon. Images courtesy of the Centre for Landscape Research, University of Toronto.

Figure C: An illustration of how a proposed planting scheme will help screen a person's view of an electric transmission line over a 15-year period as the vegetation matures. The images (read top to bottom) show a view before planting, after planting, after five years, and after fifteen years. Images courtesy of the Centre for Landscape Research, University of Toronto.



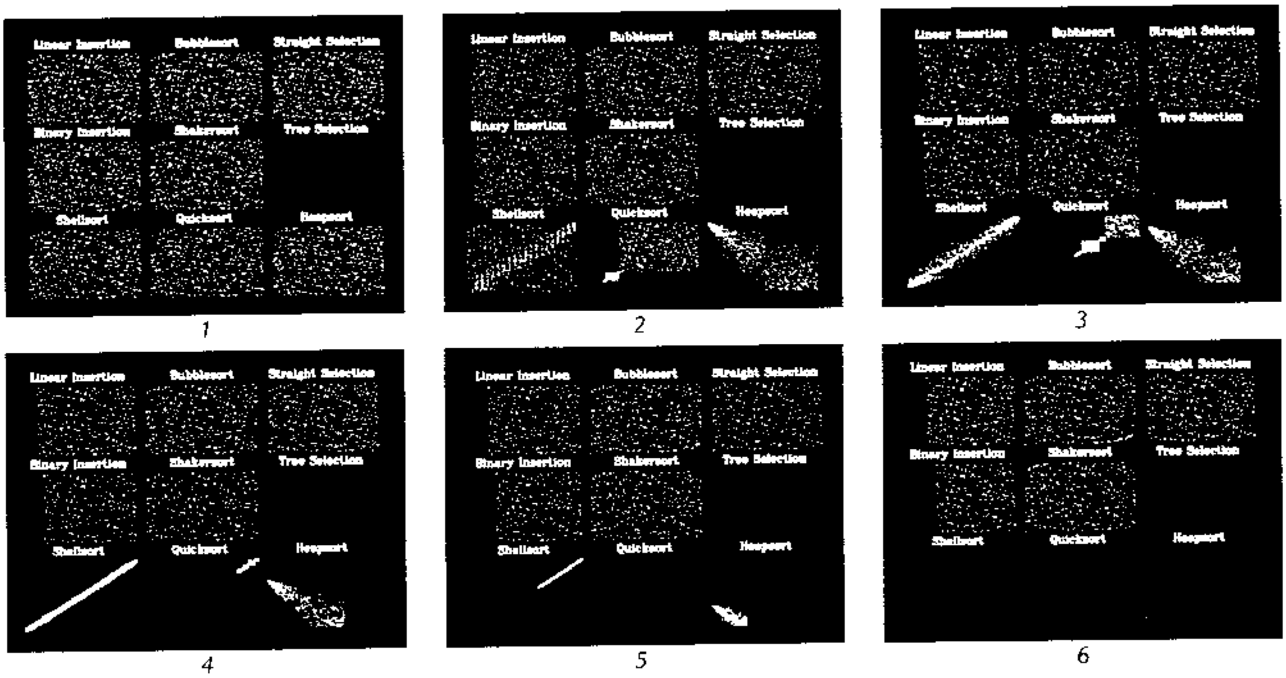


Figure D: An illustration of how interesting features of nine different sorting algorithms are vividly portrayed by their animations. For example:

- *The movement of data is depicted by the movement of dots. A color change (from yellow to red) denotes when an item reaches its ultimate "sorted" position.*
- *The recursive behavior of the* quicksort *and its use of partitioning are clearly visible (D2–D4).*
- *The property of* shellsort *in which, through multiple passes, it pushes all of the data closer and closer to the "sorted" state is apparent (D2–D5).*
- *The paradoxical tendency of* heapsort *to move large items closer and closer to the front before switching them to their correct position toward the back of the data is vividly shown (D2–D5).*
- *The five "n squared" algorithms are immediately distinguishable from the four that are "n log n" (D3–D6).*

Frames are excerpted from Sorting Out Sorting, *produced and directed by Ronald Baecker, Bynamic Graphics Project, Computer Systems Research Institute, University of Toronto, 1981. Distributed by Morgan Kaufmann, Publishers.*

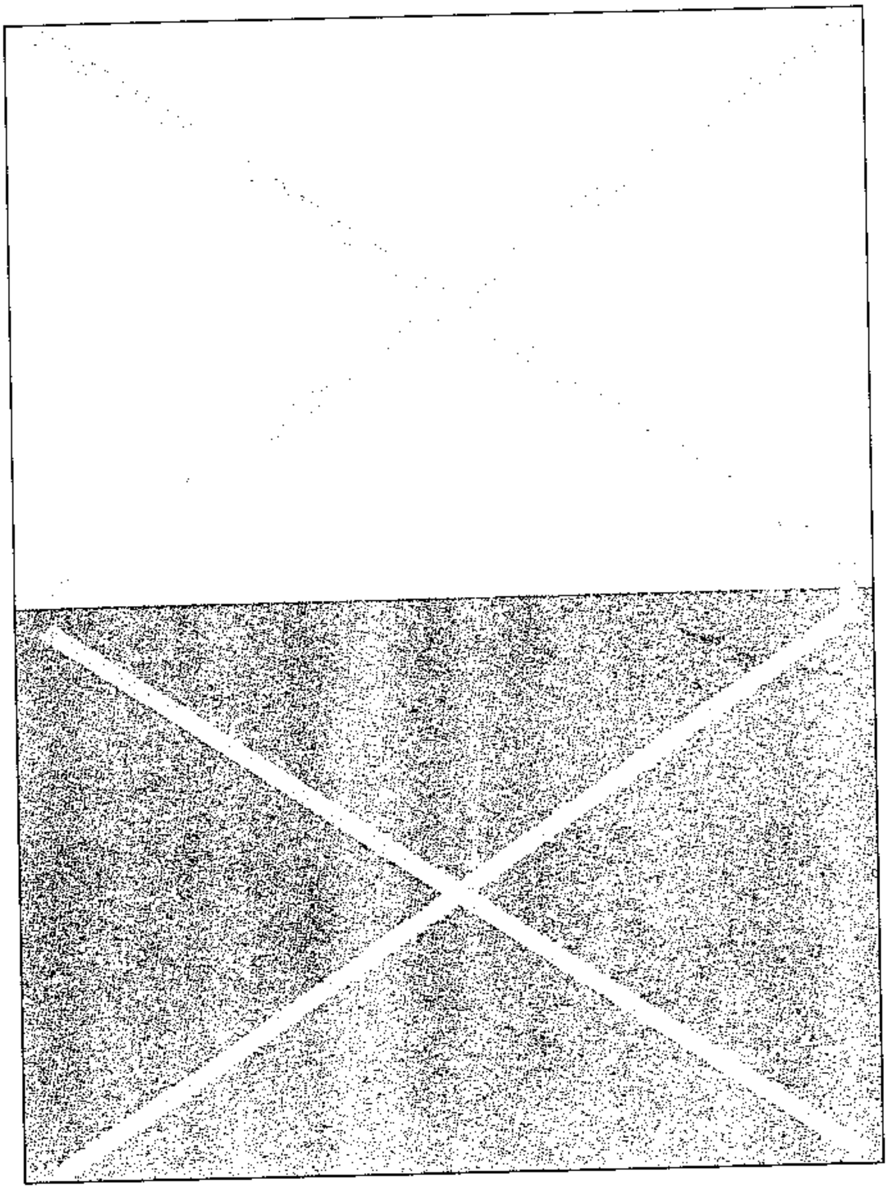


Figure E: Color appearances change based on their relationship to other colors. Each "X" appears to be the color of the opposite background, however, both "X's" are the same color. Reprinted courtesy of Yale University Press.

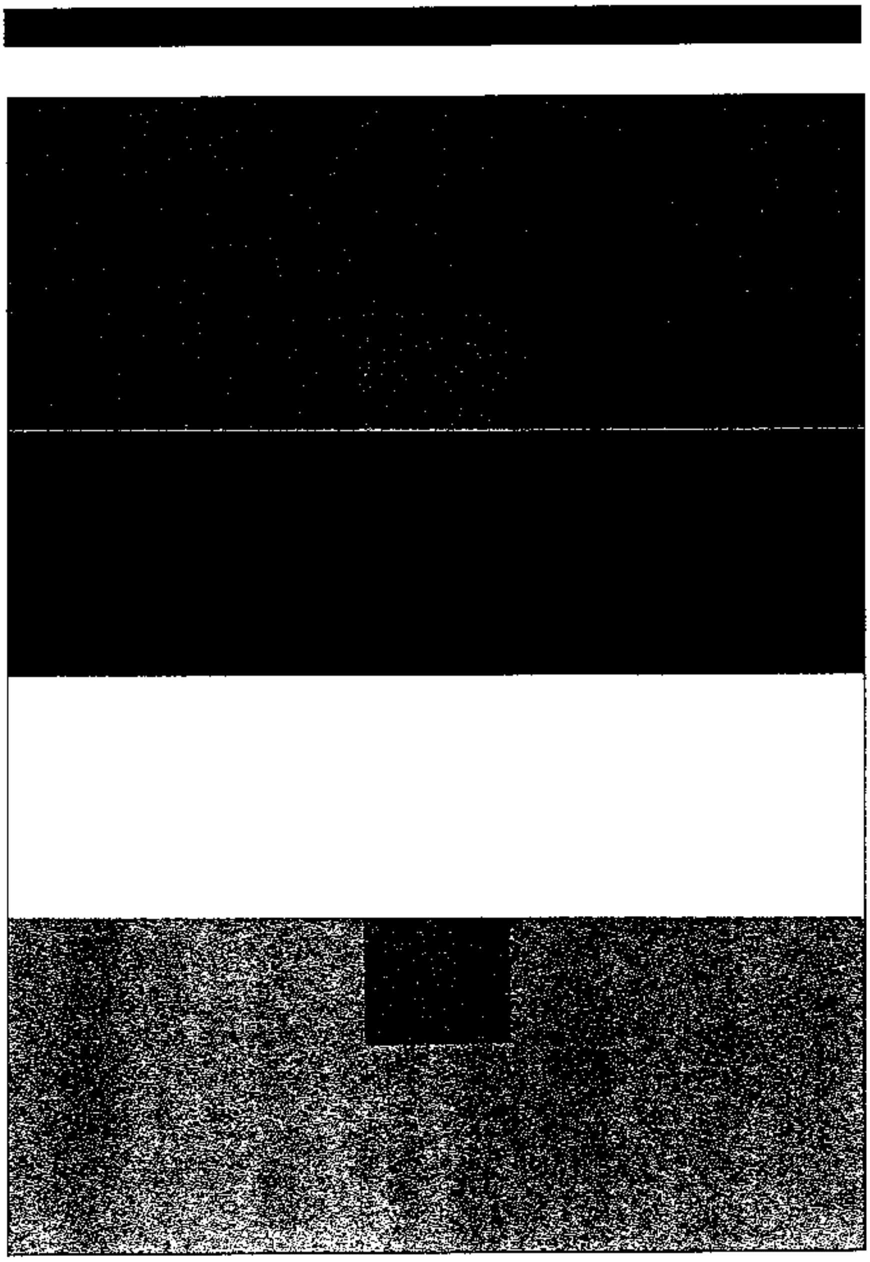


Figure F: The small brown squares appear to be two distinct colors, but they are identical. Once again, placement and relationships with other colors affect interpretation. Reprinted courtesy of Yale University Press.

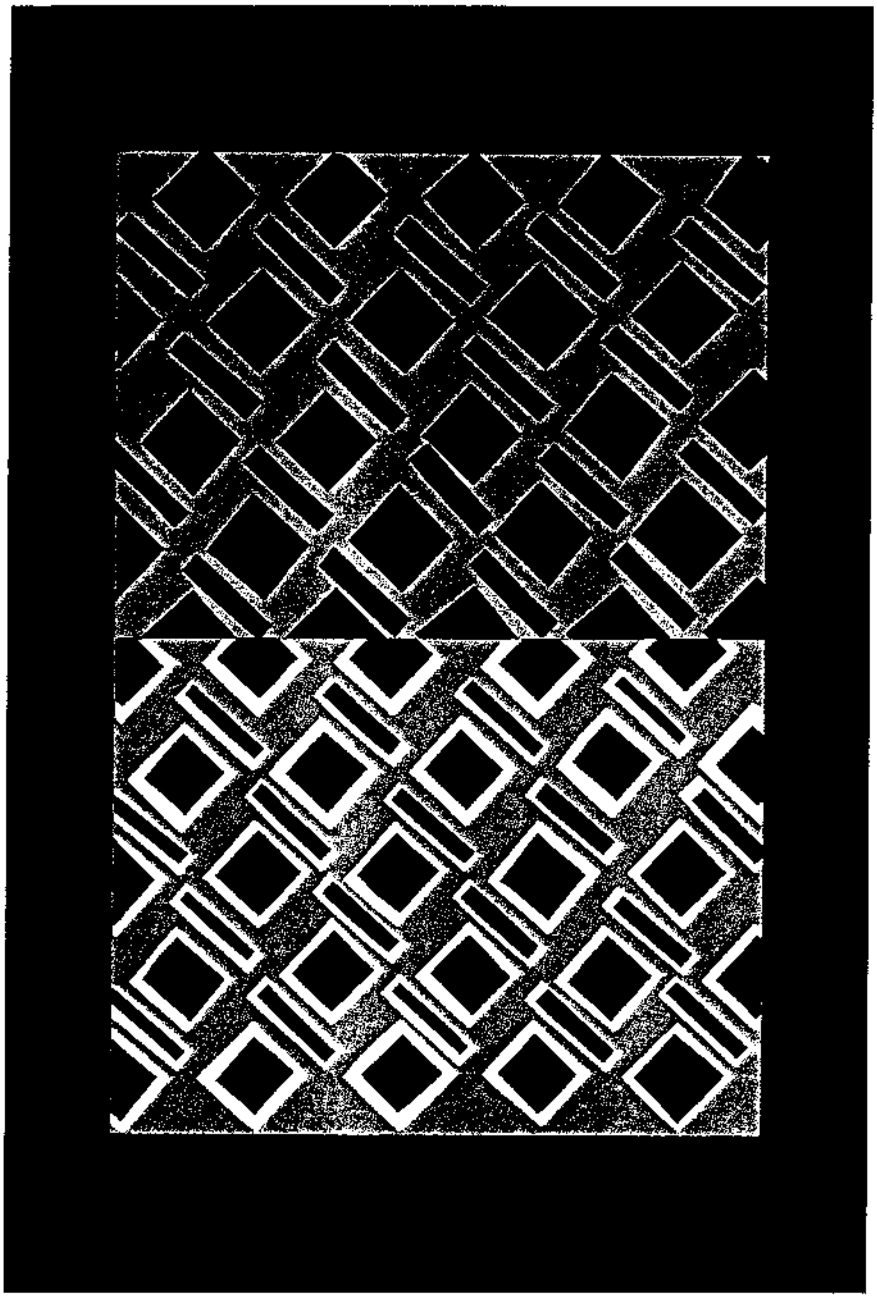


Figure G: When any color is added or changed in a design, the overall effect must be reconsidered. By altering only one color in this pattern (black changes to white), the entire character of the pattern is transformed. Reprinted courtesy of Yale University Press.

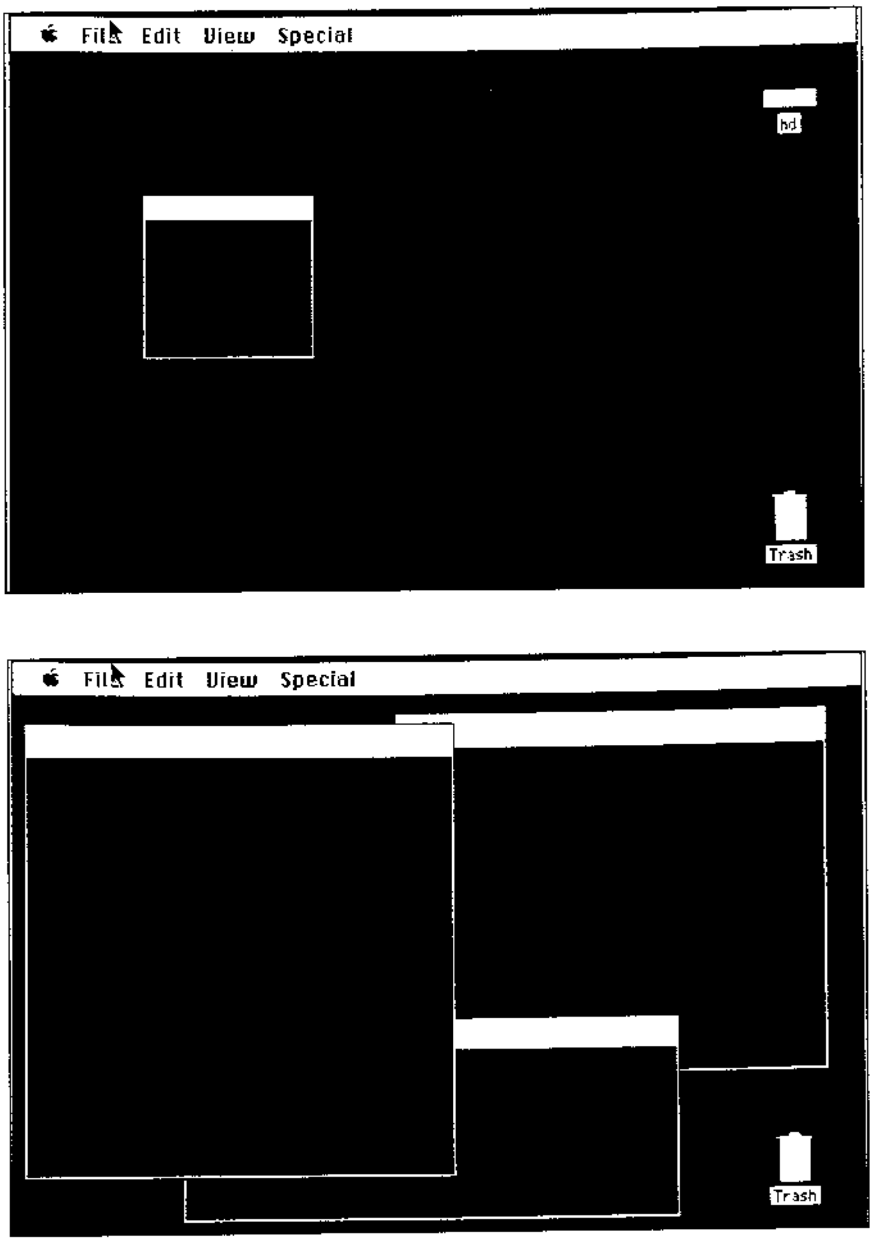


Figure H: How should colors be chosen for dynamically changing displays? When the size and placement of elements are altered, the overall color effect can change. This figure shows two abstracted Macintosh desktops. A user might select colors while looking at the window configuration of H1 (top), only to find these colors have a different effect in the H2 layout (bottom). The opening of several windows changes the apparent saturation of both the red in the background and the blue in the window interiors. Also, note that the user's work environment changes from primarily red to blue.

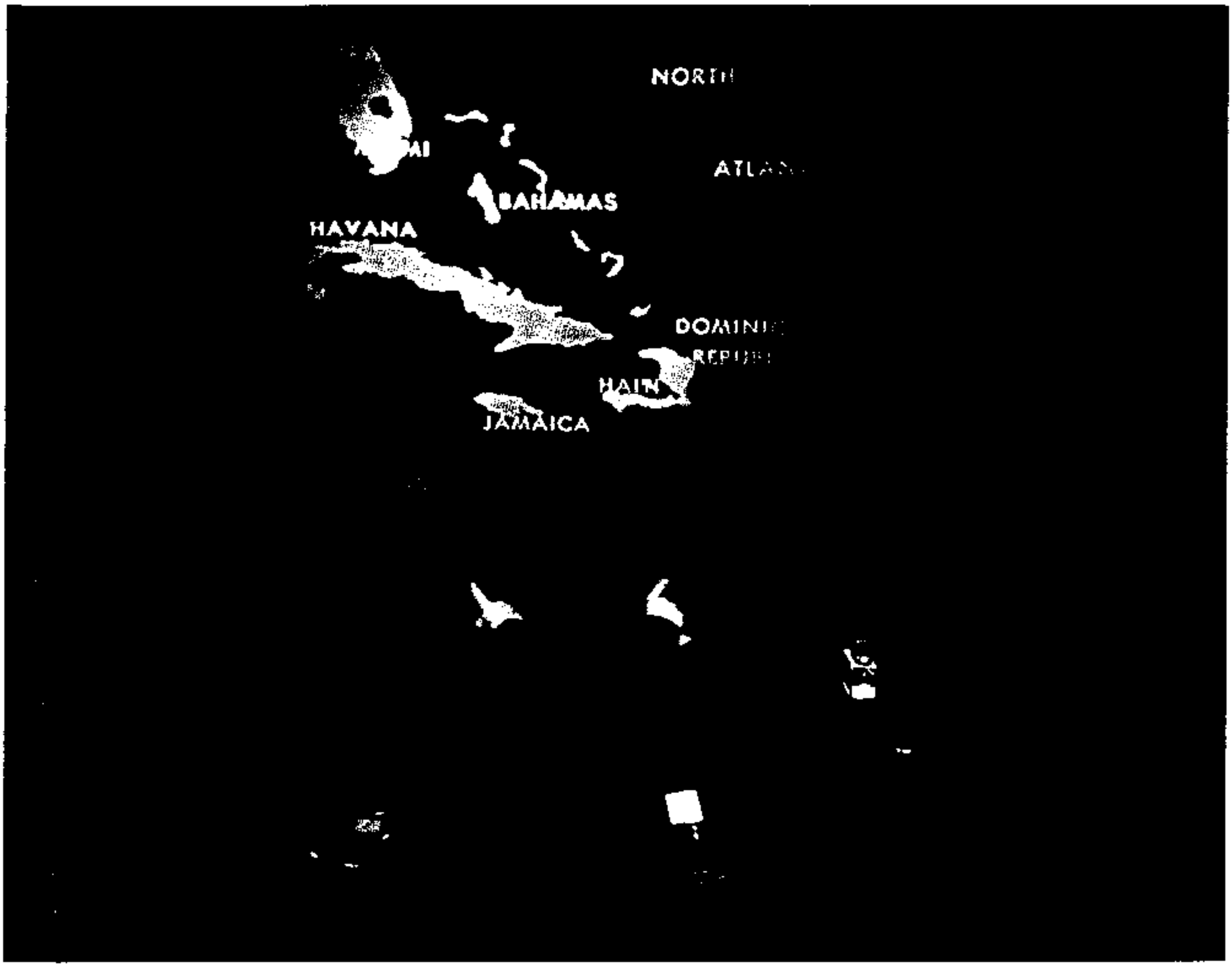


Figure K: Put That There. A time-lapse photograph shows a user "dragging" a triangle across a map of the Caribbean.

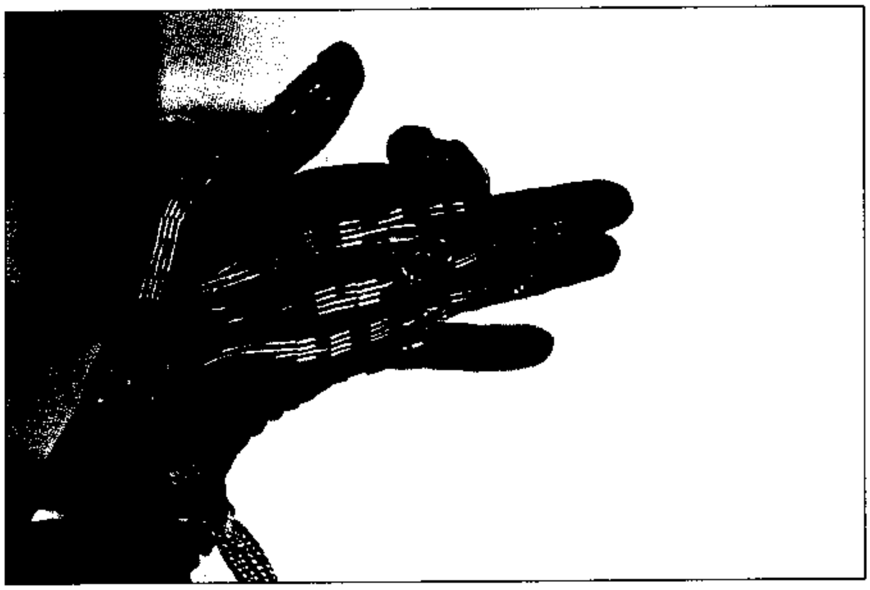
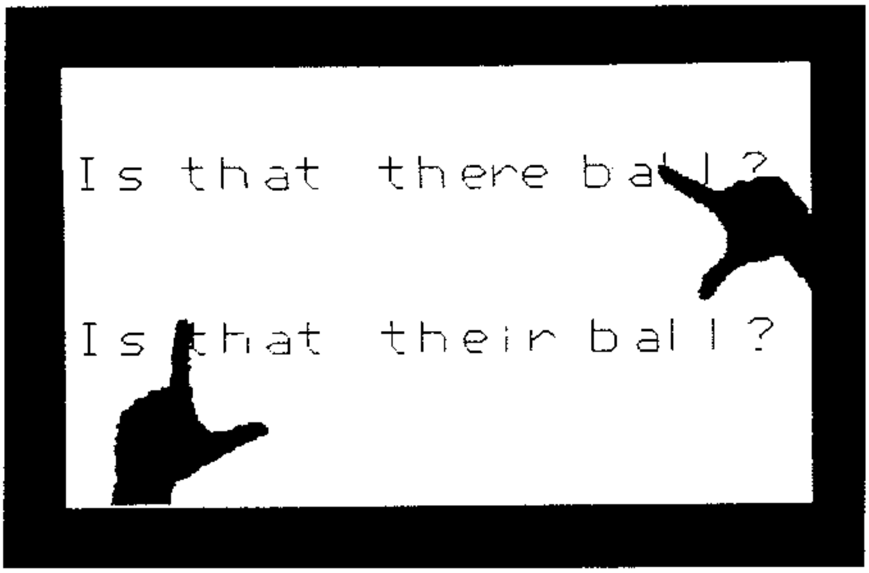


Figure L: The DataGlove™ (© VPL Research, Inc., 1988). Image courtesy of VPL Research.



Is that there ball?

Is that their ball?

Figure M: VIDEOPLACE Telecommunications. A teacher and student in different locations discuss a homework assignment. Each sees a composite image of the teacher's and student's hands pointing to relevant information (© Myron W. Krueger).

Figure N: A one-finger drawing produced by a user standing in the middle of the room.

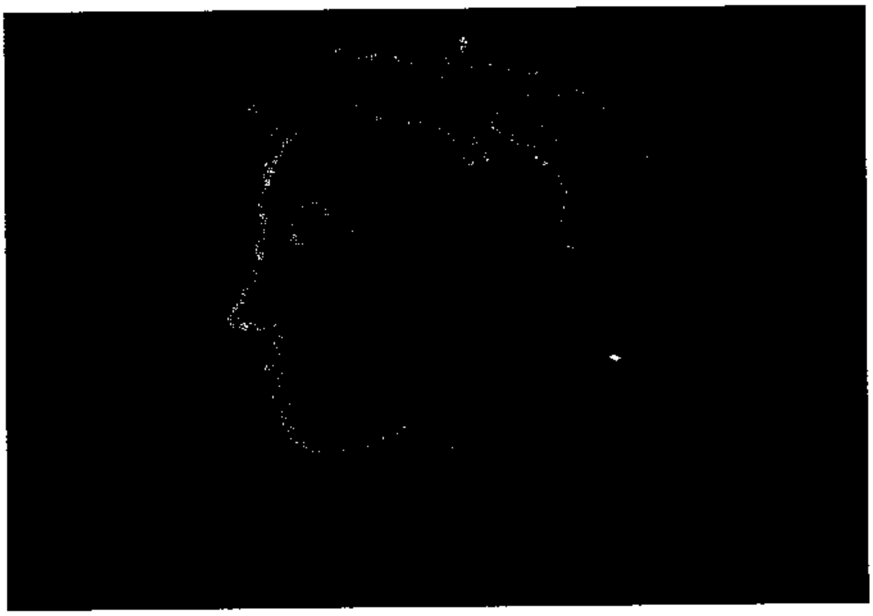
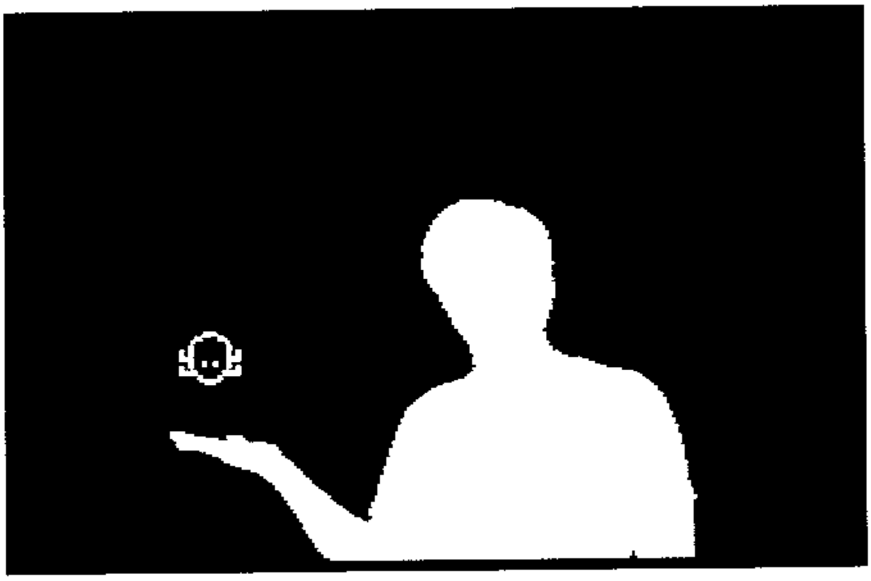


Figure O: CRITTER, a computer-generated playmate.



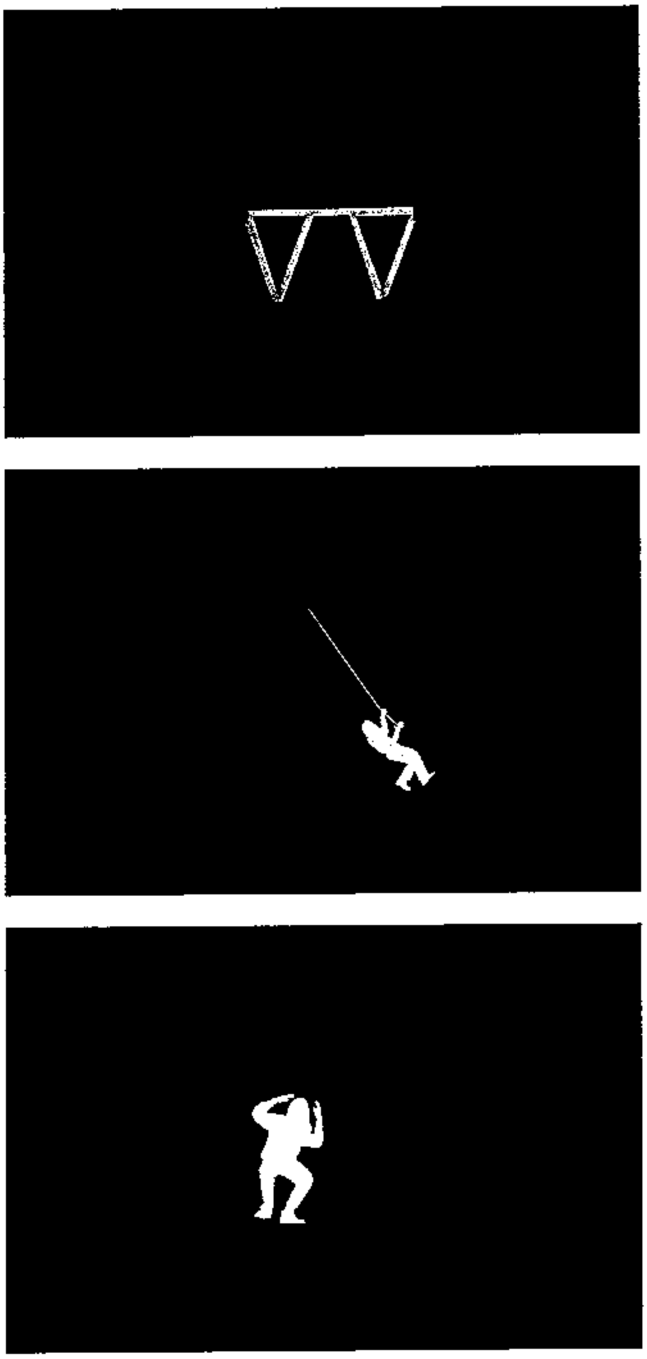


Figure P: Users interact via telecommunications at VIDEOPLACE.

Figure Q: A user can create a spline curve using thumbs and forefingers with VIDEODESK.

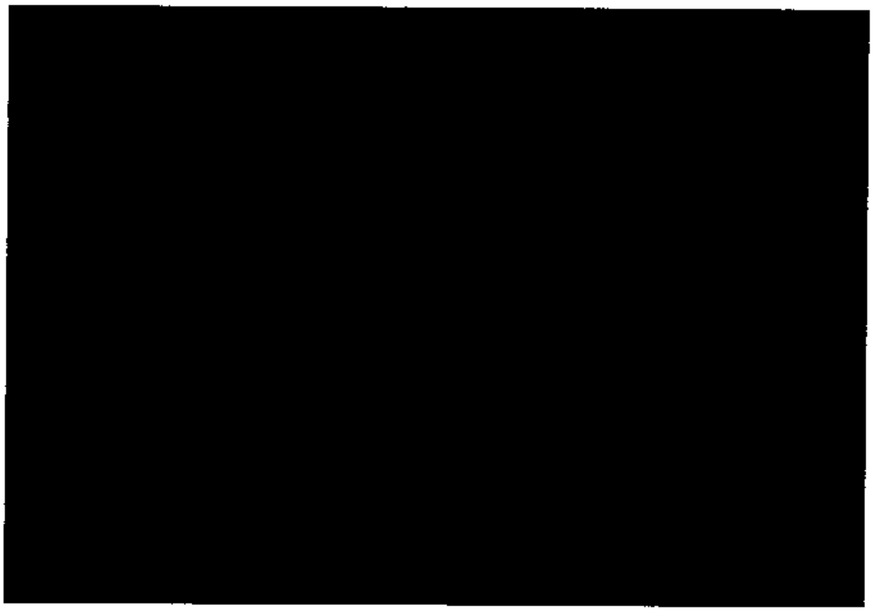
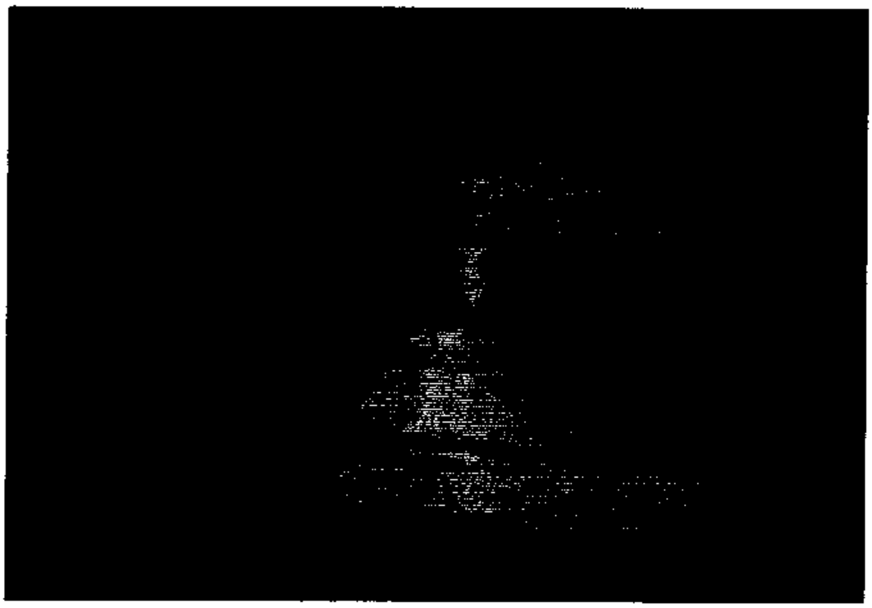


Figure R: Three-dimensional solids can be easily created "by hand" with VIDEODESK.



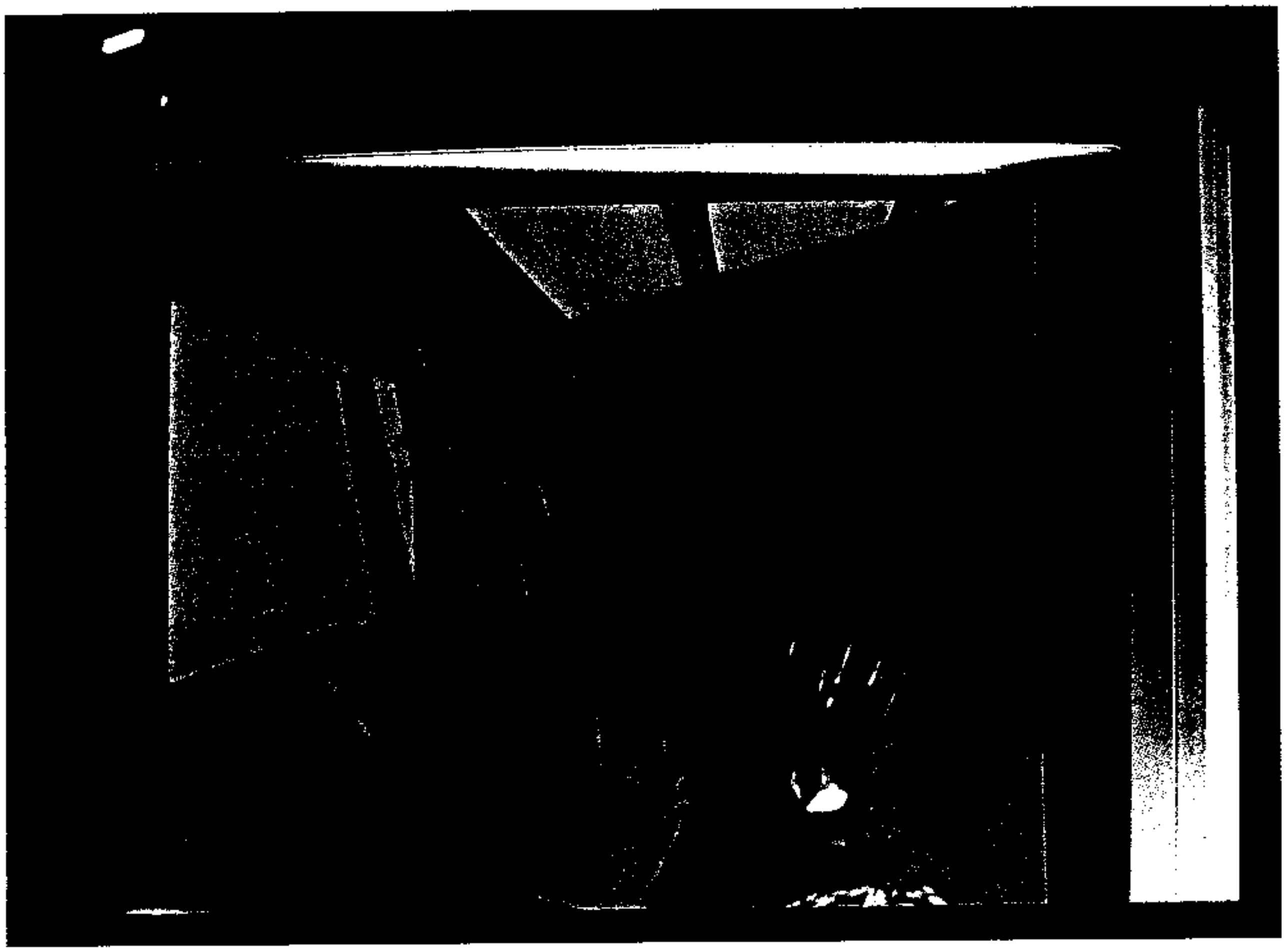
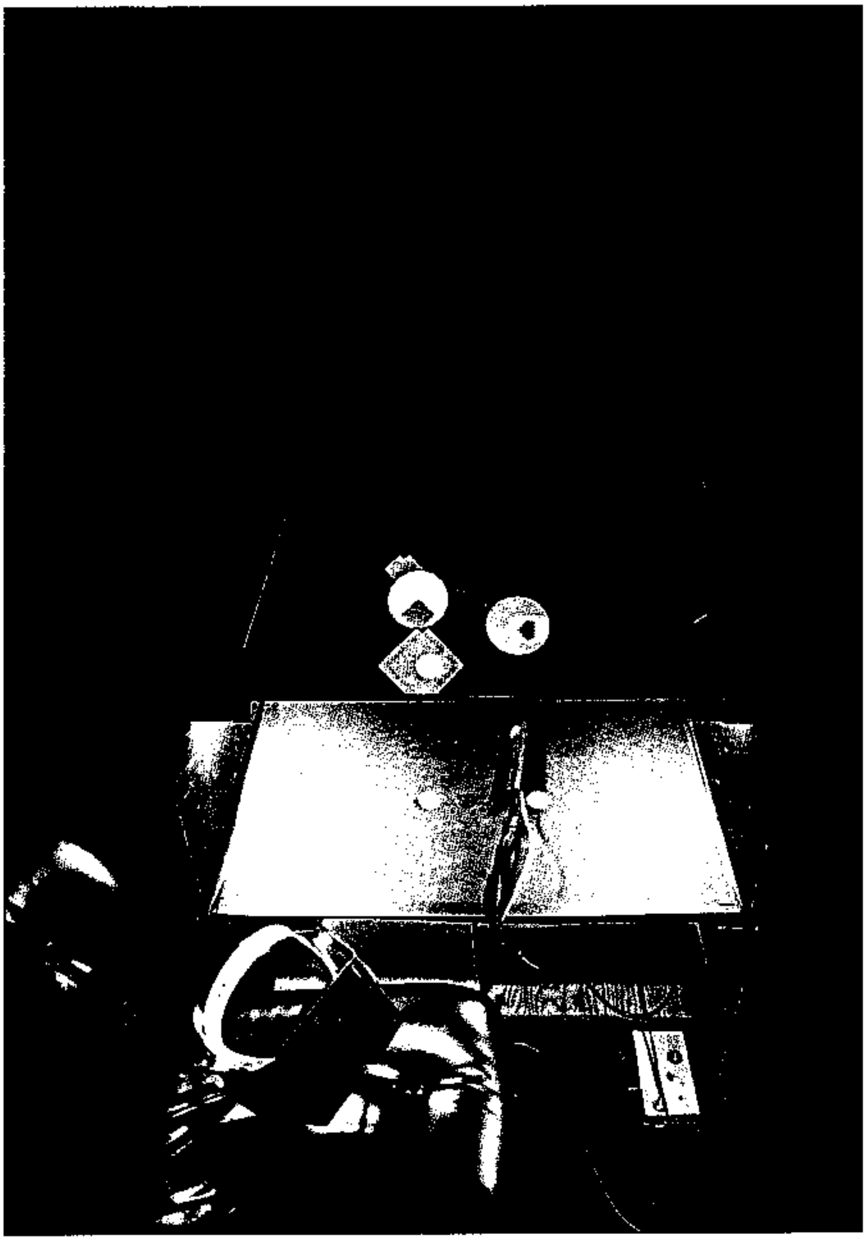


Figure 5: The VPL DataGlove™ Model 2 (© VPL Research 1989).

Figure T: Workstation prototype, developed at the MIT Architecture Machine Group, for three-dimensional drawing and manipulating virtual objects in a limited virtual environment (© Scott Fisher 1982).



Sorting Out Sorting was produced laboriously, by constructing and tailoring programs to visualize each specific algorithm to be animated. Other investigators have attempted to build tools to aid the process of program animation—for example, Balsa-II [Brown, 1988] and Movie and Stills [Bentley and Kernighan, 1987]. We also have looked at this problem. The result was LOGOmotion [Baecker and Buchanan, submitted for publication], a prototype system for the graceful and unobtrusive animation of algorithms. To use LOGOmotion, we merely indicate which procedures and data structures we want to observe, and LOGOmotion constructs a default animation of the execution of a LOGO program. If the default animation is not satisfactory, we describe how to improve it by writing additional code in LOGO.

Movies such as *Sorting Out Sorting* and systems such as Balsa-II and LOGOmotion are important because they encourage the use of animation as a vehicle for visualizing and aiding the comprehension of complex programs. How can we understand the behavior of distributed computing systems operating in dozens of sites, multiprocessor systems consisting of hundreds of integrated processors and memories, or expert systems operating with thousands of rules? Perhaps, once we learn how to define and construct animations of such systems, we will be better able to master their complexity.

Animation of Function: Making Interfaces More Comprehensible

Even if the computations underlying a program are simple, the interface to it may appear to be complex. Animation can help cut through the complexity of an interface. Animation can:

- Review what has been done
- Show what can be done
- Show what cannot be done
- Guide a user as to what to do
- Guide a user as to what not to do

In other words, animation can help us review the past, understand the present, and describe the future. It can help us answer the questions: "How did I get here?", "Where am I?", and "Where am I going?". We shall describe eight uses of animation—animation as:

Identification:	What is this?
Transition:	From where have I come, to where have I gone?
Choice:	What can I do now?
Demonstration:	What can I do with this?
Explanation:	How do I do this?

Feedback:	What is happening?
History:	What have I done?
Guidance:	What should I do now?

SCENARIO, ANIMATION AS IDENTIFICATION: IDENTIFY AN APPLICATION QUICKLY AND VIVIDLY WHEN IT IS INVOKED.

Consider a user starting an application with which he is unfamiliar: the user is unsure of whether he has selected the correct application, and of whether the application does what he wants. Furthermore, he may have to wait while the application is loaded and initialized before finding out. Many home computer games have made use of this waiting period, which is an excellent time for:

- Identifying which application has been started
- Marketing and calling attention to the manufacturer's name
- Conveying to the user a sense of what the application does

These tasks can be accomplished effectively with animation. Although the application name and manufacturer can be identified by static type, an attractive animated presentation is more likely to be remembered. Although text and static pictures can inform the user of the application's function, an animation can convey information more vividly and more rapidly.

A typical example of this use of animation occurs in the Robotropolis Preview in the computer game Robot Odyssey 1 [The Learning Company, 1984]. The user is treated to a one-minute animated preview of highlights from the game, is introduced to the game's portrayal of him and his three robot companions, and is introduced to the task of ascending upwards through the five levels of Robotropolis with the help of the robots.

This type of animation is already in occasional use, particularly in video games, but it is not yet pervasive in any particular environment.

SCENARIO, ANIMATION AS TRANSITION: ORIENT THE USER DURING TRANSITIONS FROM ONE PROCESS TO ANOTHER.

Animation can be used successfully to portray and clarify transitions in the state of both the working environment and individual processes. In such cases, animation attempts to:

- Keep the user aware of changes to the working environment
- Locate and identify newly created entities within the environment
- Cue the user to new or old areas of interest.

These goals can all be achieved by simple animation effects.

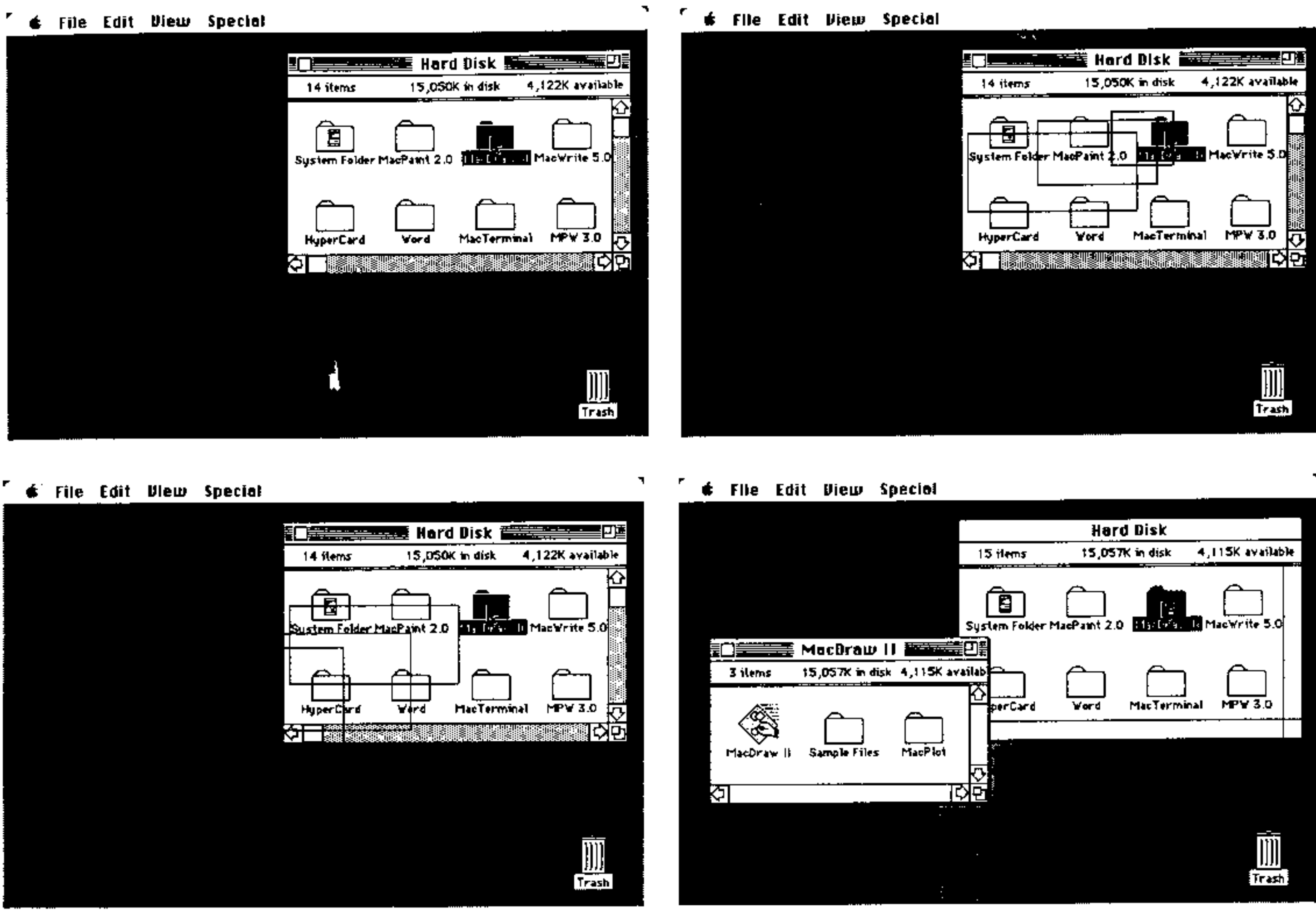


Figure 1: Transition to a new folder on the Macintosh Desktop (read right to left).

A good illustration is the zoom used in the Macintosh desktop environment (figure 1). The outline zoom that accompanies the opening (and closing) of an icon orients the user to the location and origin of the new window that appears on the desktop. This is particularly helpful in a crowded environment. If the new window were to appear without the opening zoom, it would be more difficult for the user to determine that he had indeed opened the correct icon. The closing zoom assists in informing the user where he was working before he started the process that has just been completed.

This type of animation is already in widespread use on the Macintosh, perhaps because of its relative ease of implementation and uncontroversial nature. In fact, HyperCard provides users with a variety of special visual effects to highlight transitions, including zooms, wipes, and dissolves.

Scenario, Animation as Choice: Provide an overview of complex menus.

Typical pull-down and pop-up menus occupy a fair amount of screen space, especially on personal computers, with their relatively small displays. This can result in a loss of context, as the information that relates to the user's menu choice ends up being hidden behind the menu. Furthermore, complex hierarchies of menus require large numbers of menu selections in order to arrive at the desired menu choice. Animated menus represent one potential solution to these problems.

The basic idea is simple. Instead of the menu and its choices being displayed statically, different sets of choices, submenus, and other items can be animated by displaying them in succession using techniques such as paging, flipping, cycling, scrolling, or three-dimensional rotation. The benefits that can result from such techniques include:

- Reducing the screen space required by the menu and the degree to which the menu obscures other material on the screen
- Allowing rapid display of the entire menu set and of the relationships between menu items

Consider, for example, a desktop publishing system in which there are large numbers of combinations of typeface, weight, slant, and point size. Conventionally, we could make a set of independent selections, not knowing if the combination is valid, or use a hierarchic set of menus designed to allow only valid choices. An alternative is to animate the set of all valid combinations (figure 2).

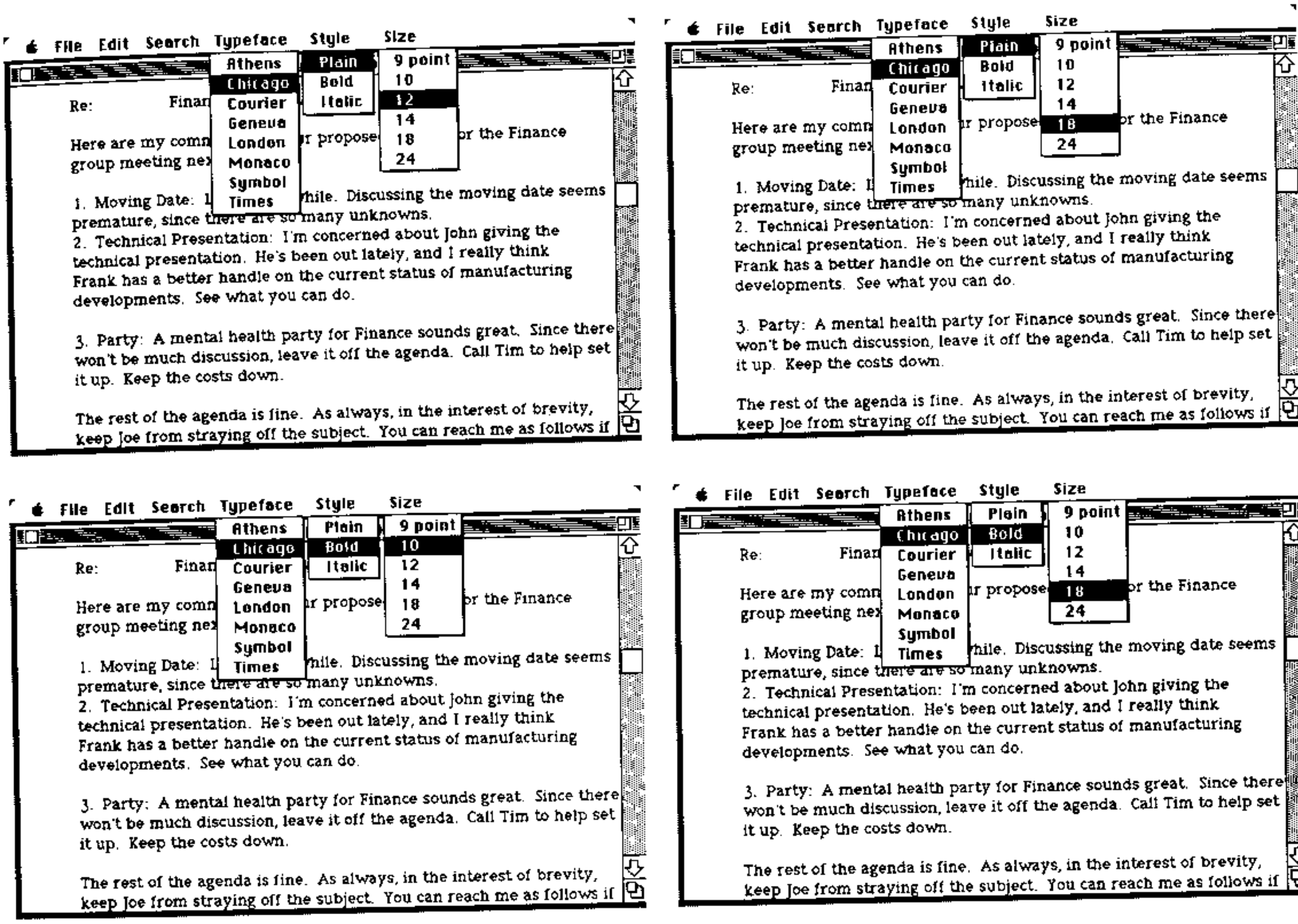
Animations of this kind are possible with current technology; research is required to determine if they are effective.

Scenario, Animation as Demonstration: Improve the information content of icons or symbols used to perform actions.

Although desktop environments often make use of icons to represent actions that can be performed, the static nature of these icons limits their information content. Instead, imagine the possibilities of using dynamically changing pictograms—animated icons. Such use of animation could:

- Increase the amount of information that is contained in the iconic representation
- Clarify the function of the icon through the use of animated rather than static symbols

Both of these items are best achieved through the use of carefully scripted animation that is tailored to the particular icon. The animation of an application icon allows more information about the application to be con-



Re: Finan

Here are my comn r propose r the Finance group meeting ne

1. Moving Date: L hile. Discussing the moving date seems premature, since there are so many unknowns.
2. Technical Presentation: I'm concerned about John giving the technical presentation. He's been out lately, and I really think Frank has a better handle on the current status of manufacturing developments. See what you can do.

3. Party: A mental health party for Finance sounds great. Since there won't be much discussion, leave it off the agenda. Call Tim to help set it up. Keep the costs down.

The rest of the agenda is fine. As always, in the interest of brevity, keep Joe from straying off the subject. You can reach me as follows if

Figure 2: An animated menu set describing valid typographic choice (read right to left).

veyed to the user at an earlier stage; the concepts are similar to those described in the section entitled "Animation as Identification." The animation of action icons is more radical, however.

In many paint programs, most of the icons are self-evident, but some are not. Obscure icons include the Paint Can, the Spray Can, and the Eraser. Even more problematic is the trio of the Paint Brush, the Pencil, and the Closed Curve. It is not at all obvious from their icons how the attributes of brush shape, line thickness, and fill texture affect these three drawing operations. A great deal of trial-and-error may be required to understand what is going on. Animation can clarify underlying relationships and introduce functionality (see figure 3).

This type of complex animation requires significant computing power and some degree of operating system support, particularly if animation of a number of different icons is to exist concurrently. Again, research is required to determine the perceptual and cognitive implications of such animation.

Scenario, Animation as Explanation: Provide more interesting, understandable, and communicative tutorials and explanations.

Animation is a compelling communications medium, comparing favorably with the more commonly used static media involving text and pictures. As a result, a natural application of animation is for tutorials and explanations, such as the "Guided Tours" that are distributed with some Apple products. Whereas animation as demonstration provides only a superficial introduction to function, animation as explanation presents a detailed tutorial illustrating technique. These tutorials differ from error and help messages (see below) in that the situations they portray are purely hypothetical and are used only for purposes of illustration. Animated tutorials can:

- Depict complex sequences of steps and evolving situations more realistically and convincingly than textual explanations
- Maintain the user's interest by being lively and enjoyable

Consider the processes of Copy and Paste in the Macintosh. An animated tutorial such as that shown in figure 4 could communicate very effectively the concept that the Clipboard holds only the result of the most recent Cut or Copy operation, highlighted in figures 4a and 4c by the dashed line, and not also the result of a preceding action.

Such animations must be designed well in order to be effective, the large investment of time required usually precluding their use. Tools for accelerating the authoring of these animations must therefore be developed.

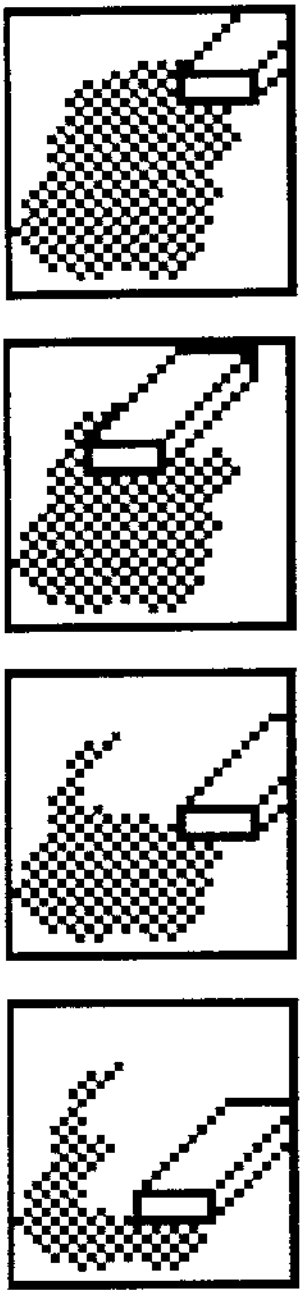


Figure 3: An animated icon describing the Eraser in a paint program.

Scenario, Animation as Feedback: Provide current information concerning the status of a system or its background processes.

A considerable amount of information about processes running in the background can be conveyed to the user through an iconic representation of each process. This approach was advocated by Myers [1984] in his design of the Sapphire windowing system for the Perq workstation operating environment. Figure 5 shows a Sapphire icon at four stages in the execution of a C compiler processing the file foo. The upper horizontal bar is a percent-done process indicator [Myers, 1985] showing the progress of the compiler on a set of files. The three stars indicate that the window associated with the icon is not displayed; as can be seen from figure 5d, the window was brought on to the screen after the keyboard input request signaled in figure 5c. In addition, the exclamation mark in figure 5d indicates that the process needs attention.

We can extend these ideas and adapt them to create a more animated interface. Animated icons associated with processes could convey such information as:

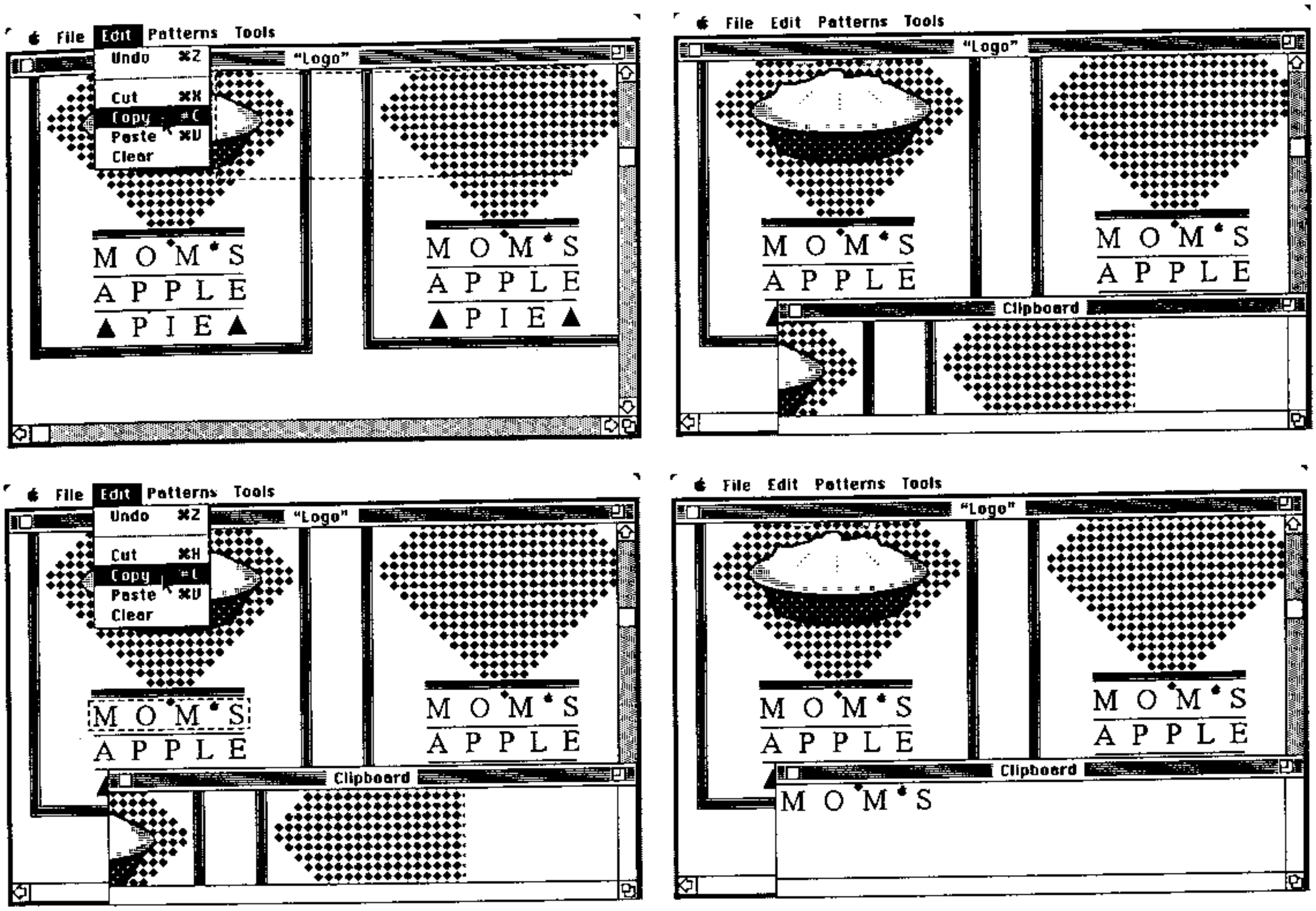


Figure 4: An animated tutorial: only the last Copy appears in the Clipboard (read right to left). This figure is taken from Brad Myers' article, "The User Interface of Sapphire," IEEE Computer Graphics and Applications, Volume 4, Number 12 (December 1984), © 1984 IEEE.

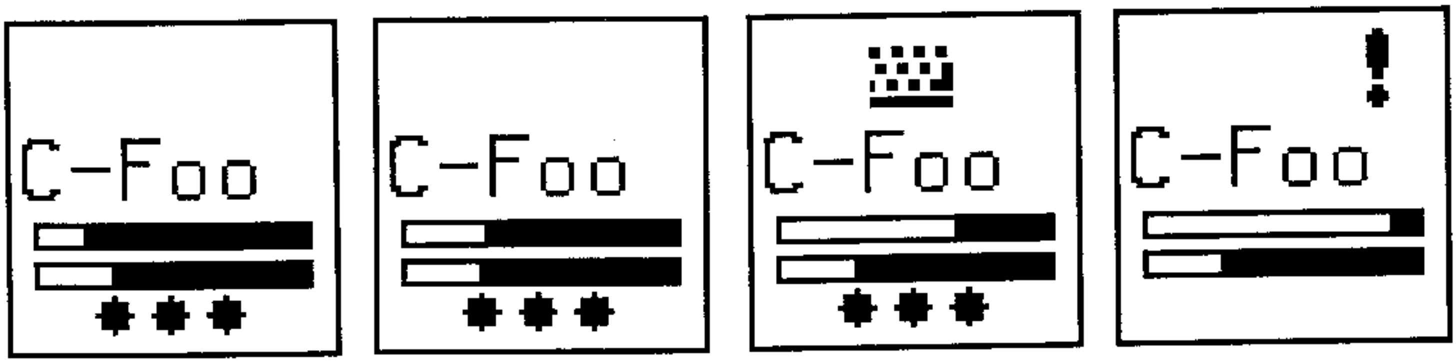


Figure 5: An animated icon giving feedback to Sapphire users.

- Whether the process is active at any given instant in time
- What portion of the computer's time is being devoted to the process
- What balance of other computer resources is required by the process
- How close the process is to completion, as is done, for example, with file copying in the Macintosh Finder and with pagination and file saving in Microsoft Word

Simple animation can communicate such information to the user, using scaling, transparency, color animation, and bar graphs similar to Myers's. Imagine an icon that needs attention changing color to become progressively more red as it is ignored for an increasingly long period of time. Indeed, color could also be used to convey the balance of resources required by the process, with a process that is totally compute-bound being blue, a process totally I/O-bound being yellow, and more balanced processes having a color in between. Similarly, an icon's size could be used to convey an idea of how much compute power the process is consuming.

Obviously, attempting to communicate all this information for each process would lead to both a pulsating display and a riot of information that the user would not be able to assimilate easily. Thus, good design is essential. Given sufficient compute power and operating system support, such animation could be provided easily.

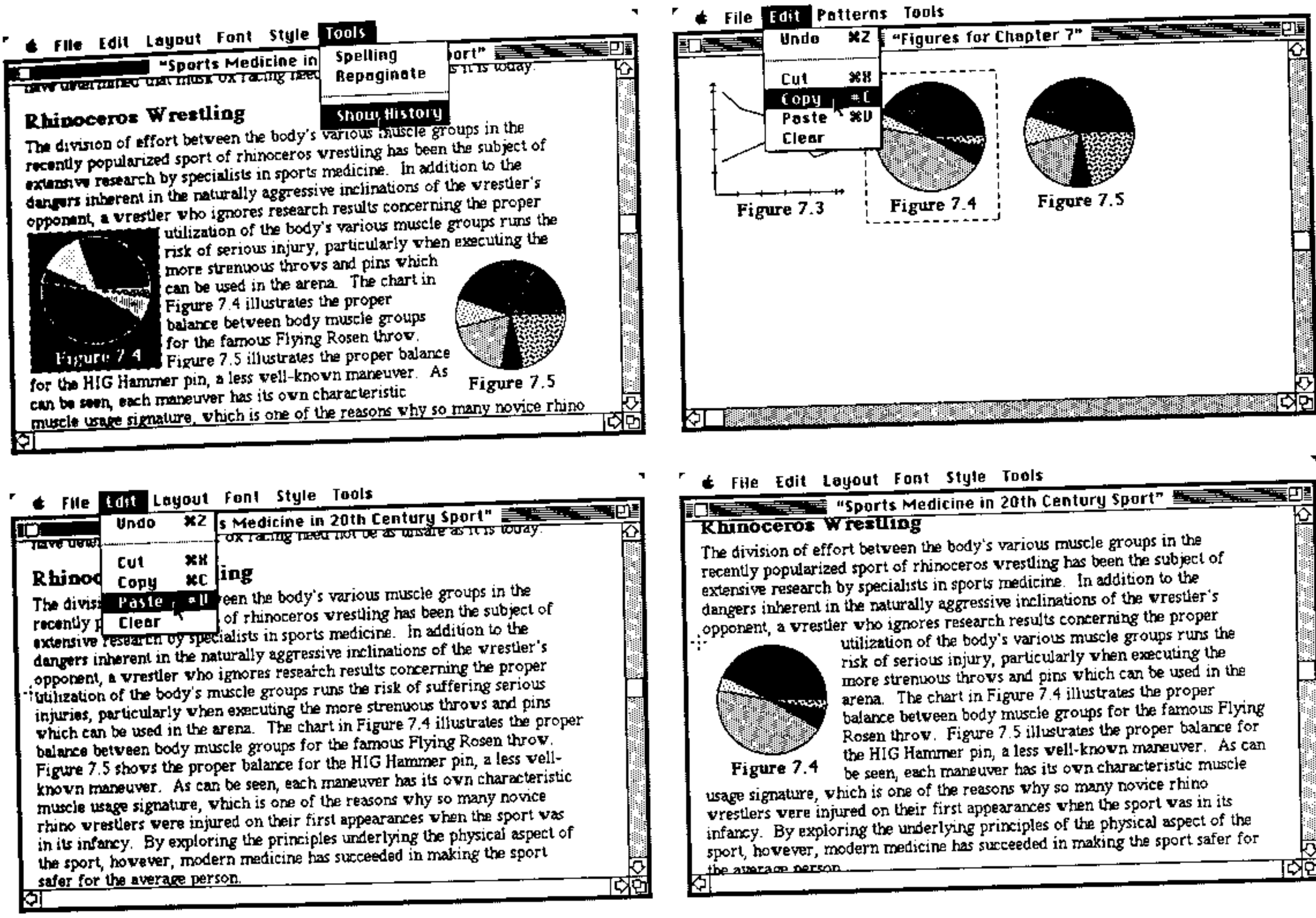
Scenario, Animation as History: PROVIDE ACCESS TO INTERACTION HISTORY AND TO HISTORICAL CONTEXT.

Users frequently get lost in applications. All of a sudden, they realize that they have no idea where they are or how they got there. The availability of history is important to the user in these situations, for it can:

- Show exactly what steps were taken to get to the current state
- Demonstrate the precise results of specific actions recently undertaken
- Provide rapid overviews of general trends in the user's recent activities

Consider the process of copying a portion of a graphic document into a word processing document using Cut and Paste. An animated history such as that shown in figure 6 could allow the user to review how a picture was included in the document and from where it came.

Animated historical overviews can be produced easily by supplying a history of the user's actions to a procedural animation package that interprets the transactions in the historical log to produce descriptive animation specific to the actual application or environment being used. These *animated histories* are almost exact playbacks of the user's recent (or not so recent) activities. Of interest is the ability to play these histories backwards or forwards, to concentrate on specific segments, and to play them at


File Edit Layout Font Style Tools
Spelling
Repaginate
Show History
"Sports Medicine in 20th Century Sport"
Figure 7.4
Figure 7.5
File Edit Patterns Tools
Undo ⌘Z
Cut ⌘X
Copy ⌘C
Paste ⌘V
Clear
"Figures for Chapter 7"
Figure 7.3
Figure 7.4
Figure 7.5
File Edit Layout Font Style Tools
Undo ⌘Z
Cut ⌘X
Copy ⌘C
Paste ⌘V
Clear
"Sports Medicine in 20th Century Sport"
File Edit Layout Font Style Tools
"Sports Medicine in 20th Century Sport"
Figure 7.4

Rhinoceros Wrestling

The division of effort between the body's various muscle groups in the
recently popularized sport of rhinoceros wrestling has been the subject of
extensive research by specialists in sports medicine. In addition to the
dangers inherent in the naturally aggressive inclinations of the wrestler's
opponent, a wrestler who ignores research results concerning the proper
utilization of the body's various muscle groups runs the
risk of serious injury, particularly when executing the
more strenuous throws and pins which
can be used in the arena. The chart in
Figure 7.4 illustrates the proper
balance between body muscle groups
for the famous Flying Rosen throw.
Figure 7.5 illustrates the proper balance
for the HIG Hammer pin, a less well-known maneuver. As
can be seen, each maneuver has its own characteristic
muscle usage signature, which is one of the reasons why so many novice rhino

Rhinoc ing

The divis een the body's various muscle groups in the recently p of rhinoceros wrestling has been the subject of extensive research by specialists in sports medicine. In addition to the dangers inherent in the naturally aggressive inclinations of the wrestler's opponent, a wrestler who ignores research results concerning the proper utilization of the body's muscle groups runs the risk of suffering serious injuries, particularly when executing the more strenuous throws and pins which can be used in the arena. The chart in Figure 7.4 illustrates the proper balance between body muscle groups for the famous Flying Rosen throw. Figure 7.5 shows the proper balance for the HIG Hammer pin, a less well-known maneuver. As can be seen, each maneuver has its own characteristic muscle usage signature, which is one of the reasons why so many novice rhino wrestlers were injured on their first appearances when the sport was in its infancy. By exploring the principles underlying the physical aspect of the sport, however, modern medicine has succeeded in making the sport safer for the average person.

Rhinoceros Wrestling

The division of effort between the body's various muscle groups in the recently popularized sport of rhinoceros wrestling has been the subject of extensive research by specialists in sports medicine. In addition to the dangers inherent in the naturally aggressive inclinations of the wrestler's opponent, a wrestler who ignores research results concerning the proper utilization of the body's various muscle groups runs the risk of serious injury, particularly when executing the more strenuous throws and pins which can be used in the arena. The chart in Figure 7.4 illustrates the proper balance between body muscle groups for the famous Flying Rosen throw. Figure 7.5 illustrates the proper balance for the HIG Hammer pin, a less well-known maneuver. As can be seen, each maneuver has its own characteristic muscle usage signature, which is one of the reasons why so many novice rhino wrestlers were injured on their first appearances when the sport was in its infancy. By exploring the underlying principles of the physical aspect of the sport, however, modern medicine has succeeded in making the sport safer for the average person.

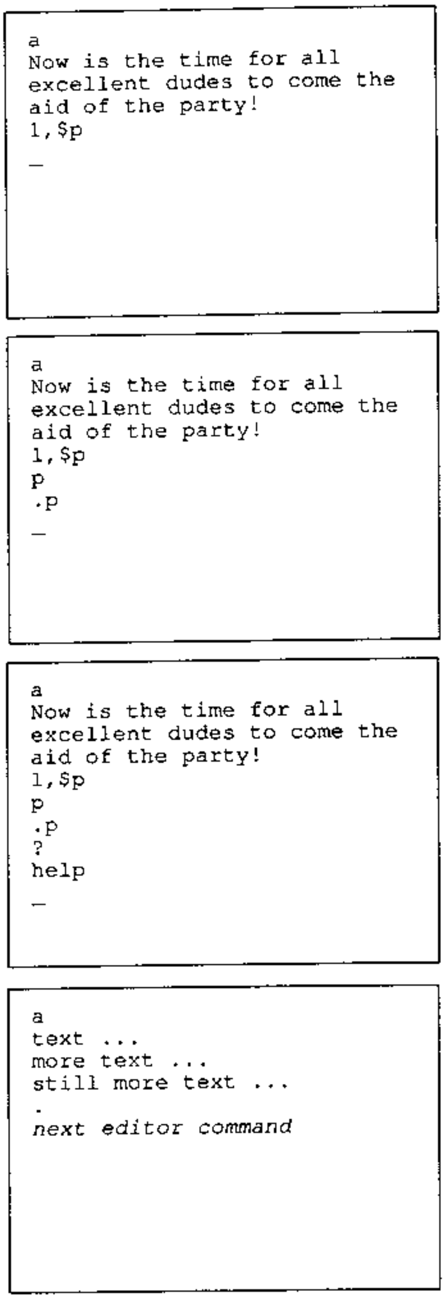
Figure 6: An animated history showing a Copy and Paste sequence (read right to left).

arbitrary speeds. This latter ability allows the user to compress a great deal of history into a short period of time, which is useful for refreshing familiar material or scanning for general trends. More abstract historical animation is more complex to produce but can rely on similar strategies [Kurlander and Feiner, 1988; Sukaviriya, 1988].

Although simple, this approach requires extensive disk space for storing the transaction logs, which makes it infeasible for current floppy-disk-based home computer applications. For scientific workstations, however, animated histories are closer to becoming a reality.

Scenario, Animation as Guidance: Improve the delivery of error messages and help and thereby accelerate the correction of mistakes.

Computers, even those with what are considered user-friendly environments, tend to produce error messages that are abrupt, unhelpful, or cryptic. The careful use of animation in these messages can aid in making them appear less intimidating, can convey more information, and can help users determine how to correct mistakes more quickly. Such animation can:



```
a
Now is the time for all
excellent dudes to come the
aid of the party!
1,$p
_
```

```
a
Now is the time for all
excellent dudes to come the
aid of the party!
1,$p
p
.p
_
```

```
a
Now is the time for all
excellent dudes to come the
aid of the party!
1,$p
p
.p
?
help
_
```

```
a
text ...
more text ...
still more text ...
.
next editor command
```

Figure 7: Animated guidance in response to a UNIX mode error.

- Indicate the set of steps that produced the error
- Show what happened as a result of these incorrect actions
- Show what would have happened had the user followed various similar courses of action
- Provide exact instruction demonstrating the steps required to achieve a desired result

The first two stages can be achieved easily using the techniques described earlier for producing animated histories. The third and fourth stages require the application to be able to determine possible courses of action, but these steps can use the same animation capability. In this case, the application produces alternative future histories that can then be displayed using the animated history mechanism, thus producing animated futures.

Even providing only the first two stages would be valuable. A common experience when one is confused and stuck is to describe the problem to someone else. Often, the other person doesn't even get a chance to respond, because in the very act of articulating what has been done, one often realizes the mistake that has been made.

Consider, as an example, the common append mode problem of the UNIX editor ed (figure 7). The user is typing text, but fails to realize that he has neglected to exit from append mode (7a). He becomes confused because the system appears mute (7b), and asks for help. The system then reviews his recent actions (7c), notes that he is in append mode (that is, executing an append command), and plays an animation showing typical uses of append (7d). "Aha!" says the user, immediately seeing the mistake.

The automatic generation of such guidance requires a system to "understand" what a user is trying to do, what has gone wrong, and what the user should do to fix it. This will likely require the application of artificial intelligence techniques.

The examples of *animation of function* appear simple. Yet appearances can be deceptive. There are difficult problems in the design of appropriate, effective animation. There are also difficult problems in implementation. Unlike *animation of structure* and *animation of process*, where we expect to require significant resources and where we are willing to wait for suitable effects, these new examples must execute instantaneously. If response is sluggish, users will retreat to the fast, safe world of static text and static pictures.

Conclusions

We have described a variety of roles for animation at the interface. *Animation of structure* and *animation of process* are relatively conventional roles that emerge out of several decades of work in interactive three-dimensional computer graphics and in program visualization. Far newer and therefore in a sense much more exciting, however, is *animation of function*. Here we are not only using animation at the interface but also animation in and of the interface.

But many questions remain. How do we design such animations so that they are clear and comprehensible, attractive and appealing? How do we prevent animation from becoming too complex to be effective? It must not become too busy and too distracting, either spatially—in that too much is going on in parallel—or temporally—in that too much is changing too quickly. Such questions can be answered only through the extensive development of prototypes and through user testing.

Can we develop tools to enable the automatic or semi-automatic construction of interface animations? Or is this premature—need we focus all our energies for the moment on developing the art of describing algorithms and interfaces visually, on advancing the art of *program illustration*?

If we can answer some of these questions and further explore the ideas developed in this paper, *program illustrators* of the future, both human and automated, will be able to employ animation, along with other media such as audio and video, to make interfaces to computer systems more enjoyable and more comprehensible.

Acknowledgments

Support from the Natural Sciences and Engineering Research Council of Canada, from the Information Technology Research Centre of the Province of Ontario, and from Apple Computer is gratefully acknowledged. Work on this paper was begun at the University of Toronto and was completed while the authors were visiting the Human Interface Group of Apple Computer.