Exhibit 1014 – Part 8

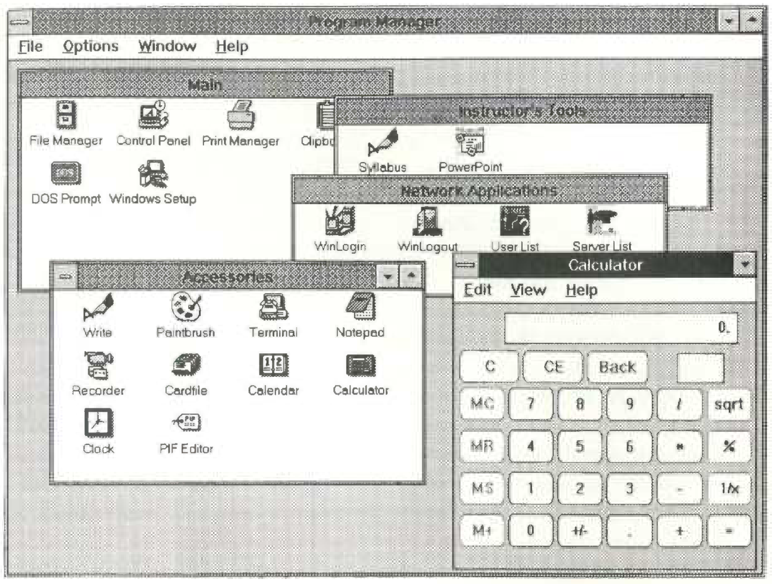


Figure 5.11

Microsoft Windows 3.0 and other systems on IBM's PS/2 offered variations on the direct-manipulation style popularized by the Macintosh. (Screen shot ©1985–1991 Microsoft Corporation. Reprinted with permission from Microsoft Corporation, Redmond, WA.)

ing, and customizing information," HyperCard quickly spawned variants such as SuperCard and ToolBook. Each has a scripting language to enable users to create appealing graphics applications.

Checkbook maintenance and searching can be done in a direct manipulation manner by displaying a checkbook register with labeled columns for check number, date, payee, and amount, as is done in the Quicken product from Intuit, Inc. Changes can be made in place, new entries can be made at the first blank line, and a check mark can be made to indicate verification against a monthly report or bank statement. Users can search for a particular payee by filling in a blank `payee` field and then typing a `?`.

Bibliographic searching has more elaborate requirements, but a basic system could be built by first showing the user a wall of labeled catalog-index drawers. A cursor in the shape of a human hand might be moved over to the section labeled `Author Index` and to the drawer labeled `F-L`. Depressing the button on the joystick or mouse would cause the drawer to open, revealing an array of index cards with tabs offering a finer index. By moving the cursor-hand and depressing the selection button, the user would make the indiviaual index cards appear. Depressing the button while holding a card would cause a copy of the card to be made in the user's notebook, also represented on the screen. Entries in the notebook might be edited to create a printed bibliography, or combined with other entries to perform set intersections or unions. Copies of entries could be stored on user files or transmitted to colleagues by electronic mail. It is easy to visualize many alternate approaches, so careful design and experimental testing would be necessary to sort out the successful comprehensible approaches from the idiosyncratic ones.

Why not do airline reservations by showing the user a map and prompting for cursor motion to the departing and arriving cities? Then, use a calendar to select the date, and a clock to indicate the time. Seat selection is done by showing the seating plan of the plane on the screen, with a diagonal line to indicate an already-reserved seat.

The term *direct manipulation* is accurately applied to describe the programming of some industrial robot tools. The operator holds the robot "hand" and guides it through a spray painting or welding task while the controlling computer records every action. The control computer can then operate the robot automatically and repeat the precise action as many times as necessary.

Why not teach students about polynomial equations by letting them move sliders to set values for the coefficients and watch how the graph changes, where the y-axis intercept occurs, or how the derivative equation reacts (Shneiderman, 1974). Similarly, direct manipulation of sliders for red, green, and blue is a satisfying way to explore color space. Slider-based dynamic queries are a powerful tool for information exploration (Section 11.8).

These ideas are sketches for real systems. Competent designers and implementers must complete the sketches and fill in the details. Direct manipulation has the power to attract users because it is comprehensible, natural, rapid, and even enjoyable. If actions are simple, reversibility ensured, and retention easy, then anxiety recedes and satisfaction flows in.

5.3 Explanations of Direct Manipulation

Several authors have attempted to describe the component principles of direct manipulation. An imaginative observer of interactive system designs, Ted Nelson (1980), perceives user excitement when the interface is constructed by what he calls the *principle of virtuality*—a representation of reality that can be manipulated. Rutkowski (1982) conveys a similar concept in his *principle of transparency*: "The user is able to apply intellect directly to the task; the tool itself seems to disappear."

Heckel (1991) laments that "Our instincts and training as engineers encourage us to think logically instead of visually, and this is counterproductive to friendly design." He suggests that thinking like a filmmaker can be helpful for interactive systems designers: "When I design a product, I think of my program as giving a performance for its user."

Hutchins et al. (1986) review the concepts of direct manipulation and offer a thoughtful decomposition of concerns. They describe the "feeling of involvement directly with a world of objects rather than of communicating with an intermediary," and clarify how direct manipulation breaches the *gulf of execution* and the *gulf of explanation*.

These writers and others (Ziegler and Fahnrich, 1988; Thimbleby, 1990; Phillips and Apperley, 1991) support the growing recognition that a new form of interactive system is emerging. Much credit also goes to the individual designers who have created systems that exemplify aspects of direct manipulation.

Another perspective on direct manipulation comes from the psychology literature on *problem-solving* and *learning research*. Suitable representations of problems have been clearly shown to be critical to solution finding and to learning. Polya (1957) suggests drawing a picture to represent mathematical problems. This approach is in harmony with Maria Montessori's teaching methods for children (Montessori, 1964). She proposed use of physical objects, such as beads or wooden sticks, to convey such mathematical principles as addition, multiplication, or size comparison. Bruner (1966) extended the physical-representation idea to cover polynomial factoring and other mathematical principles. Carroll, Thomas, and Malhotra (1980) found that subjects given spatial representation were faster and more successful in problem solving than were subjects given an isomorphic problem with a temporal representation. Similarly, Te'eni (1990) found that the feedback in direct-manipulation designs was effective in reducing users' logical errors in a task requiring statistical analysis of student grades. The advantage appears to stem from having the data entry and display combined in a single location on the display. Deeper understanding of the relationship between problem solving and visual perception can be obtained from Arnheim (1972) and McKim (1972).

Physical, spatial, or visual representations also appear to be easier to retain and manipulate than do textual or numeric representations. Wertheimer (1959) found that subjects who memorized the formula for the area of a parallelogram, $A = h \times b$, rapidly succeeded in doing such calculations. On the other hand, subjects who were given the structural understanding of cutting off a triangle from one end and placing it on the other end could more effectively retain the knowledge and generalize it to solve related problems. In plane-geometry theorem proving, spatial representation facilitates discovery of proof procedures over a strictly axiomatic representation of Euclidean geometry. The diagram provides heuristics that are difficult to extract from the axioms. Similarly, students are often encouraged to solve algebraic word problems by drawing a picture to represent that problem.

Papert's (1980) LOGO language creates a mathematical microworld in which the principles of geometry are visible. Based on the Swiss psychologist Jean Piaget's theory of child development, LOGO offers students the opportunity to create line drawings easily with an electronic turtle displayed on a screen. In this environment, users derive rapid feedback about their programs, can determine what has happened easily, can spot and repair errors quickly, and can gain satisfaction from creative production of drawings. These features are all characteristic of a direct-manipulation environment.

5.3.1 Problems with direct manipulation

Spatial or visual representations are not necessarily an improvement over text, because they may be too spread out, causing off-page connectors on paper or tedious scrolling on displays. In professional programming, use of high-level flowcharts and database-schema diagrams can be helpful for some tasks, but there is a danger that they will be confusing. Similarly, direct-manipulation designs may consume valuable screen space and thus force valuable information offscreen, requiring scrolling or multiple actions. Studies of flowchart usage in programming (Shneiderman, 1982) and in business graphics (Tullis, 1981) show that these visual representations can produce poorer performance than the equivalent program text or tabular presentation, probably because of the low density of information in the visual displays. For experienced users, a tabular textual display of 50 document names may be more appealing than only 10 document graphic icons with the names abbreviated to fit the icon size.

A second problem is that users must learn the meaning of components of the visual representation. A graphic icon may be meaningful to the designer, but may require as much or more learning time than a word. Some airports that serve multilingual communities use graphic icons extensively, but the meanings of these icons may not be obvious. Similarly, some computer terminals designed for international use have icons in place of names, but the meaning is not always clear.

A third problem is that the visual representation may be misleading. Users may grasp the analogical representation rapidly, but then draw incorrect conclusions about permissible actions. Ample testing must be carried out to refine the displayed objects and actions and to minimize negative side effects.

A fourth problem is that, for experienced typists, moving a mouse or raising a finger to point may sometimes be slower than typing. This problem is especially likely to occur if the user is familiar with a compact notation, such as arithmetic expressions, that is easy to enter from a keyboard, but may be more difficult with mouse-based selection. The keyboard remains the most effective direct-manipulation device for some tasks.

Choosing the right objects and actions is not an easy task. Simple metaphors, analogies, or models with a minimal set of concepts seem most appropriate to start. Mixing metaphors from two sources may add complexity that contributes to confusion. The emotional tone of the metaphor should be inviting rather than distasteful or inappropriate (Carroll and Thomas, 1982)—sewage-disposal systems are an inappropriate metaphor for electronic message systems. Since the users may not share the metaphor, analogy, or conceptual model with the designer, ample testing is required. For help in training, an explicit statement of the model, the assumptions, and the limitations is necessary.

5.3.2 The SSOA model explanation of direct manipulation

The attraction of direct manipulation is apparent in the enthusiasm of the users. The designers of the examples in Section 5.2 had an innovative inspiration and an intuitive grasp of what users would want. Each example has features that could be criticized, but it seems more productive to construct an integrated portrait of direct manipulation:

1. Continuous representation of the objects and actions of interest
2. Physical actions or presses of labeled buttons instead of complex syntax
3. Rapid incremental reversible operations whose effect on the object of interest is immediately visible

Using these three principles, it is possible to design systems that have these beneficial attributes:

- Novices can learn basic functionality quickly, usually through a demonstration by a more experienced user.
- Experts can work rapidly to carry out a wide range of tasks, even defining new functions and features.
- Knowledgeable intermittent users can retain operational concepts.
- Error messages are rarely needed.
- Users can immediately see if their actions are furthering their goals, and, if the actions are counterproductive, they can simply change the direction of their activity.
- Users experience less anxiety because the system is comprehensible and because actions can be reversed so easily.
- Users gain confidence and mastery because they are the initiators of action, they feel in control, and the system responses are predictable.

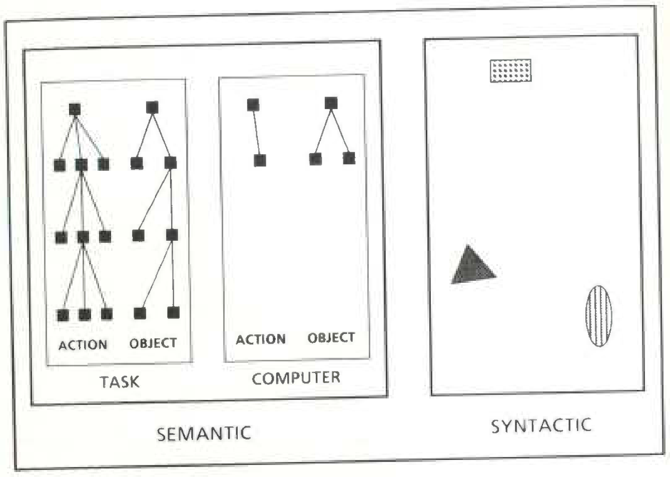


Figure 5.12

Users of direct-manipulation systems, may need to have substantial task-domain semantic knowledge. However, users must acquire only a modest amount of computer-related semantic knowledge and syntactic knowledge.

The success of direct manipulation is understandable in the context of the SSOA model. The object of interest is displayed so that actions are directly in the high-level task domain. There is little need for the mental decomposition of tasks into multiple commands with a complex syntactic form. On the contrary, each action produces a comprehensible result in the task domain that is visible immediately. The closeness of the task to the action syntax reduces operator problem-solving load and stress. This principle is related to the principle of stimulus–response compatibility in the human-factors literature. The task semantics dominate the users' concerns, and the distraction of dealing with the computer semantics and the syntax is reduced (Figure 5.12).

Dealing with representations of objects may be more "natural" and closer to innate human capabilities: Action and visual skills emerged well before language in human evolution. Psychologists have long known that spatial relationships and actions are grasped more quickly with visual rather than linguistic representations. Furthermore, intuition and discovery are often promoted by suitable visual representations of formal mathematical systems.

The Swiss psychologist Jean Piaget described *four stages of development*: *sensorimotor* (from birth to approximately 2 years), *preoperational* (2 to 7 years), *concrete operational* (7 to 11 years), and *formal operations* (begins at approximately 11 years) (Copeland, 1979). According to this theory, physical actions on an object are comprehensible during the concrete operational stage, and children acquire the concept of conservation or invariance. At about age 11, children enter the formal-operations stage, in which they use symbol manipulation to represent actions on objects. Since mathematics and programming require abstract thinking, it is more difficult for children, and a greater effort must be made to link the symbolic representation to the actual object. Direct manipulation brings activity to the concrete-operational stage, thus making certain tasks easier for children and adults.

It is easy to envision use of direct manipulation in cases where the task is confined to a small number of objects and simple actions. In complex applications, it may be more difficult to design a direct-manipulation interface. On the other hand, display editors provide impressive functionality in a natural way. The limits of direct manipulation will be determined by the imagination and skill of the designer. With more examples and experience, researchers should be able to test competing theories about the most effective metaphor or analogy. Familiar visual analogies may be more appealing to users in the early stages of learning about the system; more specific, abstract models may be more useful during regular use.

5.4 Visual Thinking and Icons

The concepts of a *visual language* and of *visual thinking* were promoted by Arnheim (1972), and were embraced by commercial graphic designers (Verplank, 1988), semiotically oriented academics (*semiotics* is the study of signs and symbols), and data-visualization gurus. The computer provides a remarkable visual environment for revealing structure, showing relationship, and enabling interactivity that attracts users with artistic, right-brained, holistic, intuitive personalities. The increasingly visual nature of computer interfaces can sometimes challenge or even threaten the logical, linear, text-oriented, left-brained, compulsive, rational programmers who were the heart of the first generation of hackers. Although these stereotypes—or caricatures—will not stand up to scientific analysis, they do convey the dual paths that computing is following. The new visual directions are sometimes scorned by the traditionalists as *WIMP* (windows, icons, mouse, and pull-down menus) interfaces, whereas the command-line devotees are seen as inflexible, or even stubborn.

There is evidence that different people have different cognitive styles, and it is quite understandable that individual preferences may vary. Just as there are multiple ice-cream flavors or car models, so too there will be multiple interface styles. It may be that preferences will vary by user and by tasks. So

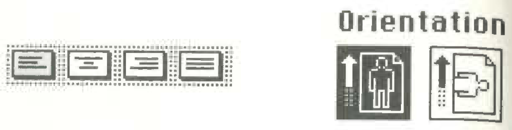


Figure 5.13

A small set of small, but effective, icons with no labels to convey the formatting actions of left, center, right, and dual justification, and a pair of action icons plus a label to convey printing orientation. (©Apple Computer, Inc., Cupertino, CA. Used with permission.)

respect is due to each community, and the designer's goal is to provide the best of each style and the means to cross over when desired.

The conflict between text and graphics becomes most heated when the issue of *icons* is raised. Maybe it is not surprising the dictionary definitions of *icon* usually refer to religious images, but the central notion is that an icon is an image, picture, or symbol representing a concept (Gittins, 1986; Rogers, 1989). In the computer world, icons are usually small (approximately 1 inch square, or 64 by 64 pixels) representations of a file or program (an object or action; see Figures 5.13, 5.14, and 5.15). Smaller icons are often used to save space or to be integrated in other objects, such as a window border (see Chapter 9). It is not surprising that icons are often used in painting programs to represent the tools or actions (lasso or scissors to cut out an image, brush for painting, pencil for drawing, eraser to wipe clean), whereas word

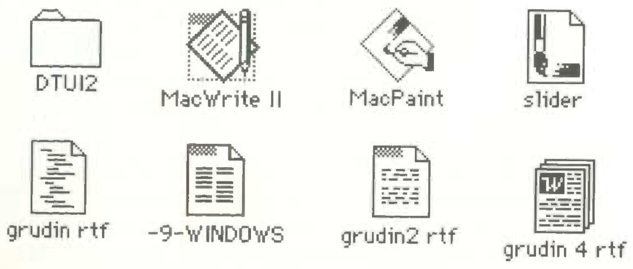


Figure 5.14

Macintosh icon collection: folder with the manuscript for this book, applications MacWrite II and MacPaint, a MacPaint image. On the second line are four documents in different formats, as indicated by their varied styles. (Courtesy of Claris Corp. Microsoft Word: Screen shot ©1984–1990 Microsoft Corporation. Reprinted with permission from Microsoft Corporation, Redmond, WA.)

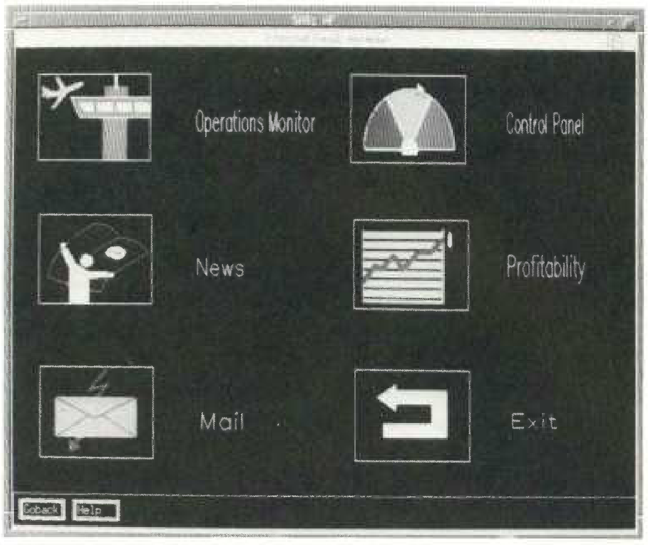


Figure 5.15

SAS uses icons with labels to the right to identify different applications. (Screens reprinted by permission. Copyright 1991 by SAS Institute Inc.)

processors usually have textual menus for their actions. This difference appears to reflect the differing cognitive styles of visually and textually oriented users, or at least differences in the tasks. Maybe, while you are working on visually oriented tasks, it is helpful to "stay visual" by using icons, whereas, while you are working on a text document, it is helpful to "stay textual" by using textual menus.

For situations where both a visual icon or a textual item are possible—for example, in a directory listing—designers have two interwoven issues: how to decide between icons and text and how to design icons. The well-established highway signs are a useful source of experience. Icons are unbeatable for showing things such as a road curve, but sometimes a phrase such as `ONE WAY—DO NOT ENTER` is more comprehensible than an icon. Of course, the smorgasbord approach is to have a little of each (as with, for example, the octagonal STOP sign) and there is evidence that icons plus words are effective in computing situations (Norman, 1991). So the answer to the first question (deciding between icons and text) depends not only on the users and the tasks, but also on the quality of the icons or the words that are proposed. Textual menu choices are covered in Chapter 3; many of the principles carry over. In addition, these icon-specific guidelines should be considered:

- Represent the object or action in a familiar and recognizable manner.
- Limit the number of different icons.
- Make the icon stand out from its background.
- Consider three-dimensional icons; they are eye-catching, but also can be distracting.
- Ensure that a single selected icon is clearly visible when surrounded by unselected icons.
- Make each icon distinctive from every other icon.
- Ensure the harmoniousness of each icon as a member of a family of icons.
- Design the movement animation: when dragging an icon, the user might move the whole icon, just a frame, possibly a grayed-out version, or a black box.
- Add detailed information, such as shading to show size of a file (larger shadow indicates larger file), thickness to show breadth of a directory folder (thicker means more files inside), color to show the age of a document (older might be yellower or grayer), or animation to show how much of a document has been printed (a document folder is absorbed progressively into the printer icon).
- Explore the use of combinations of icons to create new objects or actions—for example, dragging a document icon to a folder, trashcan, outbox, or printer icon has great utility. Can a document be appended or prepended to another document by pasting of adjacent icons? Can security levels be set by dragging a document or folder to a guard dog, police car, or vault icon? Can two database icons be intersected by overlapping of the icons?

Marcus (1992) applies semiotics as a guide to four levels of icon design:

1. *Lexical qualities*: Machine-generated marks—pixel shape, color, brightness, blinking
2. *Syntactics*: Appearance and movement—lines, patterns, modular parts, size, shape
3. *Semantics*: Objects represented—concrete, abstract, part–whole
4. *Pragmatics*: Overall legible, utility, identifiable, memorable, pleasing

He recommends starting by creating quick sketches, pushing for consistent style, designing a layout grid, simplifying appearance, and evaluating the designs by testing with users. We might consider a fifth level of icon design:

5. *Dynamics*: Receptivity to clicks—highlighting, dragging, combining

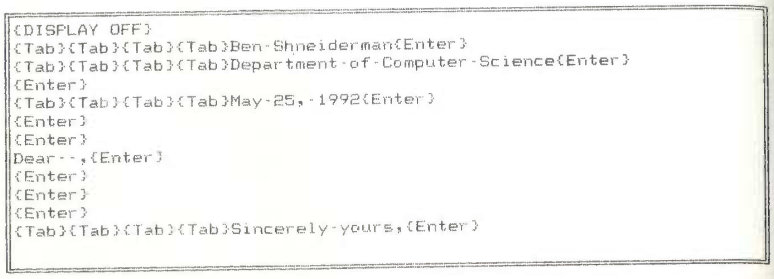
The dynamics of icons might also include a rich set of gestures with a mouse, touchscreen, or pen. The gestures might indicate copy (up and down), delete (a cross), edit (circle), etc. Icons might also have associated sounds. For example, if each document icon had a tone associated with it (the lower the tone, the bigger the document), then, when a directory was opened, each tone might be played simultaneously or sequentially. Users might get used to the symphony played by each directory and could detect certain features or anomalies.

Icon design becomes more interesting as computer hardware improves and as designers become more creative. Animated icons that demonstrate their function improve online help capabilities (see Chapter 12). Beyond simple icons, there have been increasing numbers of visual programming languages (Chang, 1990; Shu, 1988; Glinert et al., 1990) and specialized languages for mechanical engineering, circuit design, and database query (see Chapter 11).

5.5 Direct-Manipulation Programming

Performing tasks by direct manipulation is not the only goal. It should be possible to do programming by direct manipulation as well, for at least some problems. Robot programming is sometimes done by moving the robot arm through a sequence of steps that are later replayed, possibly at higher speed. This example seems to be a good candidate for generalization. How about moving a drill press or a surgical tool through a complex series of motions that are then repeated exactly? How about programming a car by driving it once through a maze and then having the car repeat the path? In fact, these direct-manipulation programming ideas are implemented in modest ways with automobile radios that the user presets by turning the frequency control knob and then pulling out a button. When the button is depressed, the radio tunes to the frequency. Some professional television-camera supports allow the operator to program a sequence of pans or zooms and then to replay it smoothly when required.

Programming of physical devices by direct manipulation seems quite natural, but an adequate visual representation of information may make direct-manipulation programming possible in other domains. Several word processors allow users to create macros by simply performing a sequence of commands that is stored for later reuse. WordPerfect enables the creation of macros that are sequences of text, special function keys such as TAB, and other WordPerfect commands (Figure 5.16). EMACS allows its rich set of functions, including regular expression searching to be recorded into macros. Macros can invoke each other, leading to complex programming possibilities. These and other systems allow users to create programs with



```
{DISPLAY OFF}
{Tab}{Tab}{Tab}{Tab}Ben·Shneiderman{Enter}
{Tab}{Tab}{Tab}{Tab}Department·of·Computer·Science{Enter}
{Enter}
{Tab}{Tab}{Tab}{Tab}May·25,·1992{Enter}
{Enter}
{Enter}
Dear··,{Enter}
{Enter}
{Enter}
{Enter}
{Tab}{Tab}{Tab}{Tab}Sincerely·yours,{Enter}
```

Figure 5.16

This WordPerfect macro produces a template of a letter.

nonvarying action sequences using direct manipulation, but strategies for including loops and conditionals vary. EMACS allows macros to be encased in a loop with simple repeat factors. By resorting to textual programming languages, EMACS and WordPerfect allow users to attach more general control structures.

Spreadsheet packages, such as LOTUS 1-2-3, Excel, and Quattro, have rich programming languages and allow users to create portions of programs by carrying out standard spreadsheet operations. The result of the operations is stored in another part of the spreadsheet and can be edited, printed, and stored in a textual form. Macro facilities in graphic user interfaces are more challenging to design than are macro facilities in traditional command interfaces, but MacroMaker and Tempo2 on the Macintosh, and the Hewlett-Packard NewWave Agent facility on the IBM PCs, are current standouts. The `MACRO` command of Direct Manipulation Disk Operating System (DMDOS) (Iseki and Shneiderman, 1986) was an early attempt to support a limited form of programming for file movement, copying, and directory commands.

A delightful children's program, Delta Drawing from Spinnaker, enables children to move a cursor and to draw on the screen by typing `D` to draw one unit, `R` to rotate right 30 degrees, and so on. The 40 commands provide rich possibilities for drawing various kinds of screen images. In addition, Delta Drawing allows users to save, edit, and then invoke programs. For example, a user can draw a circle by saving the program consisting of a `D` and a `R`. Invoking the program with the argument `12` then produces a rough 12-sided circle.

Smith (1977) inspired work in this area with his Pygmalion system that allowed arithmetic programs to be specified visually with icons. A number of research projects have attempted to create direct manipulation programming systems (Rubin et al., 1985). Halbert's Smallstar (1984) was a programming-by-example system to enable programming of Xerox Star actions. Maulsby and Witten (1989) developed a system that could induce or infer a program from examples, questioning the users to resolve ambiguities. Myers (1991) coined the phrase *demonstrational programming* to characterize these efforts as programming-by-example or programming-with-examples in which users can create macros by simply doing their tasks and letting the system construct the proper generalization automatically to form a macro. Cypher (1991) built and ran a usability test with seven subjects for his EAGER system that monitored user actions within HyperCard. When EAGER recognized two similar sequences, a small smiling cat appeared on the screen to offer the users help in carrying out further iterations. Cypher's success with two specific tasks is encouraging, but more work is needed to generalize this approach.

If designers are to create a general tool that works reliably in many situations, it they must meet the *five challenges of programming in the user interface* (PITUI) (Potter, 1992):

1. Sufficient computational generality (conditionals, iteration)
2. Access to the appropriate data structures (file structures for directories, structural representations of graphical objects) and operators (selectors, Booleans, specialized operators of applications)
3. Ease in programming (by example, by demonstration, modularity, argument passing) and editing programs
4. Simplicity in invocation and assignment of arguments (direct manipulation, simple library strategies with meaningful names or icons, in-context execution, and availability of result)
5. Low risk (high probability of bug-free programs, halt and resume facilities to permit partial executions, undo operations to enable repair of unanticipated damage)

The goal of PITUI is to allow users easily and reliably to repeat automatically the actions that they can perform manually in the user interface. Rather than depending on unpredictable inferencing, users will be able to indicate their intentions explicitly by manipulating objects and actions. The design of direct-manipulation systems will undoubtedly be influenced by the need to support PITUI. This influence will be a positive step that will also facilitate history keeping, undo, and online help.

5.6 Home Automation

Internationally, many companies have logically concluded that the next big market will be the inclusion of richer controls in homes. Simple ideas such as turning off all the lights with a single button or remote control of devices (either from one part of the home to another, from outside, or by programmed delays) are being extended in elaborate systems that channel audio and video throughout the house, schedule lawn watering as a function of ground moisture, offer video surveillance and burglar alarms, and provide multiple-zone environmental controls plus detailed maintenance records. Demonstrations such as the Smart House project in Upper Marlboro, Maryland, and installations such as those by Custom Command Systems, are a testing ground for the next generation.

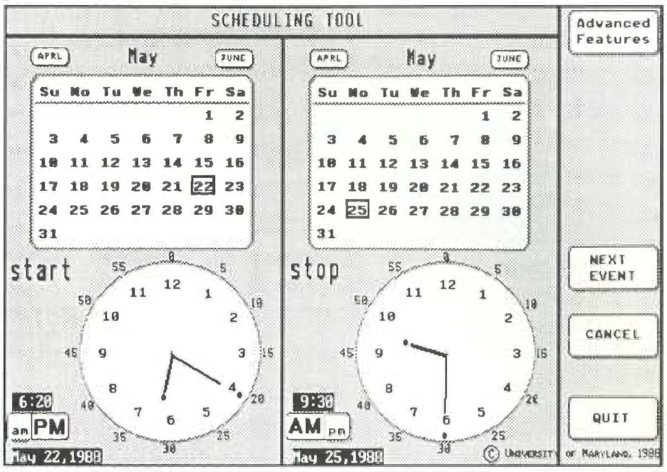
Some futurists and marketing specialists promote voice controls and home robots, but the practical reality is more tied to traditional pushbuttons, remote controllers, telephone keypads, and especially touchscreens, with the latter proving to be the most popular. Installations with two to 10 touchscreens spread around the house should satisfy most homeowners. Providing direct-manipulation controls with rich feedback is vital in these applications. Users are willing to take training, but operation must be rapid and easy to remember even if used only once or twice a year (such as spring and fall adjustments for daylight-savings time).

Our studies (Plaisant et al., 1990; Plaisant and Shneiderman, 1991) explored four touchscreen designs, all based on direct-manipulation principles, for scheduling operations such as VCR recording or light switching:

1. A digital clock that the user sets by pressing step keys (similar to onscreen programming in current video-cassette players)
2. A 24-hour circular clock whose hands can be dragged with the fingers
3. A 12-hour circular clock (plus A.M.–P.M. toggle) whose hands can be dragged with the fingers (Figure 5.17)
4. A 24-hour time line in which ON–OFF flags can be placed to indicate start–stop times (Figure 5.18)

Our results indicated that the 24-hour time line was easiest to understand and use. Direct-manipulation principles were central to this design; users selected dates by touching a monthly calendar, and times by moving the ON or OFF flags on to the 24-hour time line. The flags were an effective way of representing the ON or OFF actions and of specifying times without use of a keyboard. The capacity to adjust the flag locations incrementally, and the ease of removing them, were additional benefits. We are extending the design to accommodate more complex tasks, such as scheduling and synchronization of multiple devices, searching through schedules to find dates with specific events, scheduling repeated events (close curtains every night at dusk, turn lights on every Friday night at 7 P.M., record status monthly), and long-duration events. A generalization of the flags-on-a-line idea was applied to heating control, where users specified upper and lower bounds by dragging flags on a thermometer.

Many interesting product suggestions have emerged during our trials—for example, an alarm clock that would ring only on weekdays, thus

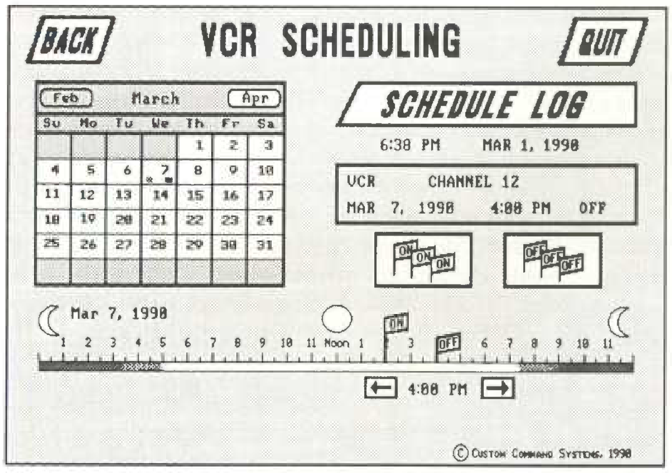


start: 6:20 PM, May 22,1988

stop: 9:30 AM, May 25,1988

Figure 5.17

This scheduler shows two calendars for start and stop dates plus, two 12-hour circular clocks with hands that can be dragged to set start and stop times. (© University of Maryland, 1988.)



6:30 PM MAR 1, 1990

VCR CHANNEL 12
MAR 7, 1990 4:00 PM OFF

Mar 7, 1990

4:00 PM

Figure 5.18

This 24-hour time line scheduler was most successful in our usability studies. The users select a date by pointing to the calendar, and then drag ON and OFF flags to the 24-hour time lines. The feedback is a red line on the calendar and the time lines. (© 1990 Custom Command Systems, College Park, MD.)

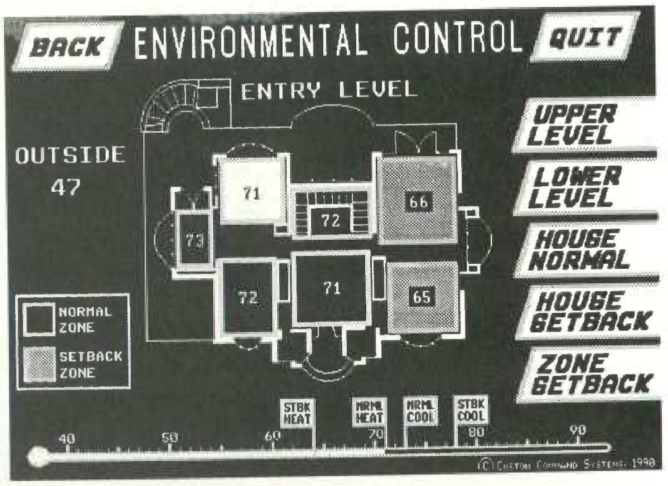


Figure 5.19

Direct-manipulation designs emphasize task-domain graphics, such as this floorplan of a private home used to set temperatures. (© 1990 Custom Command Systems, College Park, MD.)

avoiding the oversight that leads to a ruined Saturday morning when the alarm rings at 6 A.M. Other innovations include programming appliances such as washers, dryers, and ovens to run at night, when electricity costs might be lower. Utility companies are eager to see such devices installed to reduce peak loads.

Since so much of home control involves the room layouts and floorplans, many direct-manipulation actions take place on a display of the floorplan (Figure 5.19), with selectable icons for each status indicator (such as burglar alarm, heat sensor, or smoke detector), and for each activator (such as curtain or shade closing and opening motors, airconditioning or heating vent controllers, or audio and video speaker or screen). People could route sound from a compact-disc player located in the living room to the bedroom and kitchen by merely dragging the CD icon into those rooms. Sound volume control can be accomplished by having the user move a marker on a linear scale.

The simple act of turning a device ON or OFF proved to be an interesting problem. Wall-mounted light switches typically show their status by up for ON and down for OFF. Most people have learned this standard and can get what they want on the first try, if they know which switch to throw to turn on a specific light. Laying out the switches to reflect the floorplan does solve the problem quite nicely (Norman, 1988). Visitors may have problems

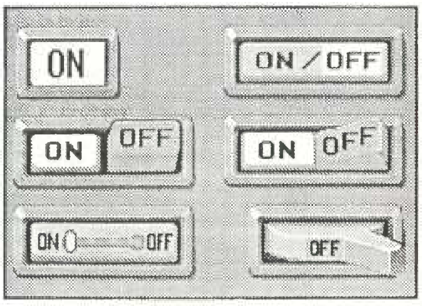


Figure 5.20

Varying designs for toggle buttons using three-dimensional graphic characteristics, designed by Catherine Plaisant.

because, in some countries, ON and OFF are reversed or the up–down switches have been replaced by push buttons. To explore possibilities, we constructed six kinds of touchscreen ON–OFF buttons with three-dimensional animation and sound (Figure 5.20). There were significant differences in user preferences, with high marks going to the simple button, rocker, and multiple push buttons. The multiple push buttons have a readily comprehensible visual presentation, and they generalize nicely to multiple state devices (`OFF`, `LOW`, `MEDIUM`, `HIGH`).

Controlling complex home equipment from a touchscreen by direct manipulation reshapes how we think of homes and their residents. New questions arise, such as whether residents will feel safer, be happier, save more money, or experience more relaxation. Are there new notations such as petri net variants or role–task diagrams for describing home automation and the social relations among residents? The benefits to users with disabilities and elderly users were often on our minds as we designed these systems, since these people may be substantial beneficiaries of this technology, even though initial implementations are for the healthy and wealthy.

5.7 Remote Direct Manipulation

There are great opportunities for the remote control of devices if acceptable user interfaces can be constructed. If designers can provide adequate feedback in sufficient time to permit effective decision making, then attractive applications in office automation, computer-supported collaborative work, education, and information services may become viable. Remote-controlled environments in medicine could enable specialists to provide consultations more rapidly, or allow surgeons to conduct more complex procedures during operations. Home-automation applications could extend remote operation of telephone-answering machines to security and access systems, energy control, and operation of appliances. Scientific applications in space, underwater, or in hostile environments can enable new research projects to be conducted economically and safely (Sheridan, 1987; Uttal, 1989).

In traditional direct-manipulation systems, the objects and actions of interest are shown continuously; users generally point, click, or drag, rather than type; and feedback, indicating change, is immediate. However, when the devices being operated are remote, these goals may not be realizable, and designers must expend additional effort to cope with slower response, incomplete feedback, increased likelihood of breakdowns, and more complex error recovery. The problems are strongly connected to the hardware, physical environment, network design, and the task domain.

5.7.1 The Corabi telepathology workstation

Telemedicine is the practice of medicine over communication links. The physician specialist being consulted and the patient's primary physician are in different locations. Corabi International Telemetrics developed the first telepathology system (Weinstein et al., 1989) that allows a pathologist to examine tissue samples or body fluids under a remotely located microscope. The transmitting workstation has a high-resolution camera mounted on a motorized light microscope. The image is transmitted via broadband satellite, microwave, or cable. The consulting pathologist at the receiving workstation can manipulate the microscope using a keypad, and can see a high-resolution image of the magnified sample. Both physicians talk by telephone to coordinate control and to request slides that are placed manually under the microscope (see Figure 5.21).

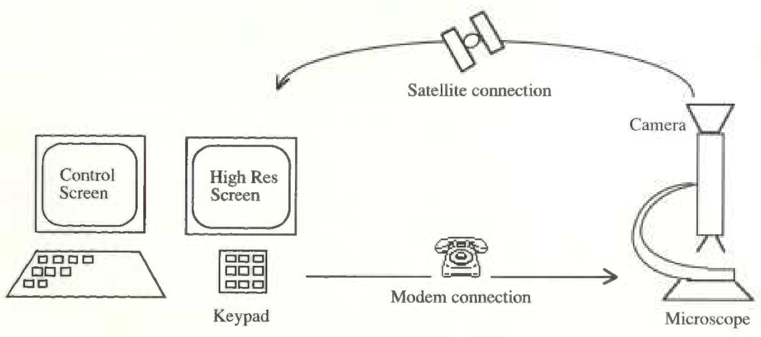


Figure 5.21

A simplified diagram of a telepathology system showing control actions sent by telephone, and images sent by satellite.

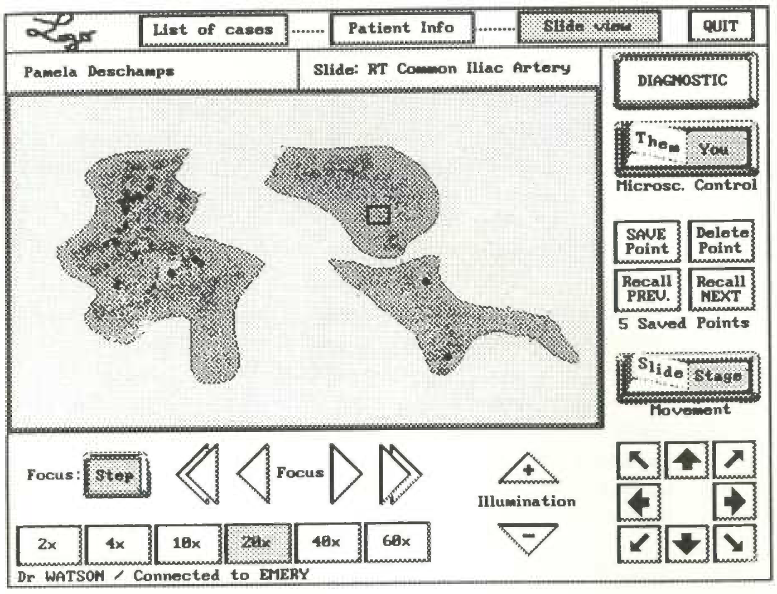


Figure 5.22

The control screen of a telepathology system showing the global view of the slide and the microscope stage position marked by a rectangle. Controls are for illumination, focus, magnification, and movement of the stage.

The pathologist uses a keypad (with arrows keys and function keys) and a large number of buttons and toggles to operate the microscope controls for

- Magnification (three or six objectives)
- Focus (coarse and fine bidirectional control)
- Illumination (bidirectional adjustment continuous or by step)
- Position (two-dimensional placement of the slide under the microscope objective)

A proposed improvement for the control screen consolidates the multiple controls (Figure 5.22).

5.7.2 Typical problems of remote direct manipulation

The architecture of remote environments introduces several complicating factors:

- *Time delays:* The network hardware and software cause delays in sending user actions and receiving feedback: *transmission delays*, the time it takes for the command to reach the microscope (in our case, transmitting the command through the modem), and *operation delays*, the time until the microscope responds (Van de Vegte et al., 1990). These delays in the system prevent the operator from knowing the current status of the system. For example, if a positioning command has been issued, it may take several seconds for the slide to start moving. As the feedback appears showing the motion, the users may recognize that they are going to overshoot their destination, but a few seconds will pass before the stopping command takes effect.

- *Incomplete feedback:* Devices originally designed for direct control may not have adequate sensors or status indicators. For instance, our microscope can report its current position, but it does so so slowly that it cannot be used continuously. Thus, it is not possible to indicate on the control screen the exact current position relative to the start and desired positions.
- *Feedback from multiple sources:* Incomplete feedback does not imply no feedback. The image received on the high-resolution screen is the main feedback to evaluate the result of an action. In addition, the microscope can occasionally report its exact position, allowing recalibration of the status display. It is also possible to indicate the estimated stage position during the execution of a movement. This estimated feedback can be used as a progress indicator whose accuracy depends on the variability of the time delays. To comply with the physical incompatibility between the high-resolution feedback (analog image) and the rest of the system (digital), the multiple feedbacks are spread over several screens. Thus, the pathologists are forced to switch back and forth between multiple sources of feedback, increasing their cognitive load.
- *Unanticipated interferences:* Since the devices operated are remote, and may be also operated by other persons in this or another remote location, unanticipated interferences are more likely to occur than in traditional direct-manipulation environments. For instance, if the slide under the microscope were moved (accidentally) by a local operator, the positions indicated might not be correct. A breakdown might also occur during the execution of a remote operation, without indicating this event properly to the remote site. Such breakdowns require increased status information for remote users and additional actions that allow for correction.

One solution to these problems is to make explicit the network delays and breakdowns as part of the system. The user sees a model of the starting state of the system, the action that has been issued, and the current state of the system as it carries out the action. It may be preferable to provide spatially parametrized positioning actions (for example, move of a distance $+x,+y$, or move to a fixed point (x,y) in a two-dimensional space), rather than providing temporal commands (for example, start moving right at a 36° angle from the horizontal). In other words, the users specify a destination (rather than a motion), and wait until the action is completed before readjusting the destination if necessary. In general, we try to turn the remote environment as much as possible into a direct-manipulation environment by applying the same basic principles.

5.7.3 The roots of a theory of remote direct manipulation

The theory of remote direct manipulation is rooted in two domains that, so far, have been independent. The first root of direct manipulation is in the context of personal computers and is often identified with the desktop metaphor and office automation. The second root is in process control where human operators control physical processes in complex environments. Typical tasks are operating power or chemical plants, controlling manufacturing, flying airplanes, or steering vehicles. If the physical processes take place in a remote location, we talk about *teleoperation* or *remote control*. To perform the control task, the human operator may interact with a computer, which may carry out some of the control tasks without any interference by the human operator. This idea is captured by the notion of *supervisory control* (Sheridan, 1988). Although supervisory control and direct manipulation stem from different problem domains and are usually applied to different system architectures, there is a strong resemblance.

Traditional direct manipulation can also be interpreted as a teleoperation, especially with high-speed networking that provides client–server environments. Files that appear on a screen may come from a remote computer, and the software may be distributed throughout the network. Messages and documents can be sent to or retrieved from remote machines, printers, or file servers. Even the letters on a display may be composed of font descriptions stored in a remote location from where the keystrokes are issued. Thus, the essential components of a teleoperation environment—such as sensors, displays, controls, remote effectors or tools, and communication links—are involved.

Remote direct manipulation (as well as supervisory control) cannot be taken as a design criterion that either is or is not fulfilled. One interface can be slightly more direct than another. Similarly, the control can be felt to be more or less remote. Thus, remote direct manipulation denotes a range of possible solutions rather than a binary variable. Direct manipulation is still an imprecise and subjective concept, although it has proved eminently useful in stimulating designers, revising existing systems, training designers, and comparing systems. The connection between direct manipulation and supervisory control seems promising.

5.8 Virtual Reality

Flight-simulator designers use many tricks to create the most realistic experience for fighter or airline pilots. The cockpit displays and controls are taken from the same production line as the real ones. Then, the windows are replaced by high-resolution computer displays, and sounds are choreographed to give the impression of engine start or reverse thrust. Finally, the vibration and tilting during climbing or turning are created by hydraulic jacks and intricate suspension systems. This elaborate virtual or artificial reality may cost almost $100 million, but it is a lot cheaper, safer, and more useful for training than the $400-million jet that it simulates. Of course, home videogame players have purchased millions of $30 flight simulators that run on their personal computers. Flying a plane is a complicated and specialized skill, but simulators are beginning to appear for more common—and for some surprising—tasks under the alluring name of *virtual reality* (VR).

Far above the office desktop, much beyond multimedia, and high above the hype of hypertext, the gurus and purveyors of virtuality are donning their EyePhones and DataGloves to explore the total-immersion experience. Whether soaring over Seattle, bending around bronchial tubes to find lung cancers, or grasping complex molecules, the cyberspace explorers (cybernauts, VR voyagers, or Chipster Columbuses) are playing Lewis and Clark to a grand Montana of the mind. The imagery and personalities involved in virtual reality are often colorful, and journalists have enjoyed the fun as much as the developers and promoters (Stewart, 1991; Rheingold, 1991). More sober researchers have tried to present a balanced view by conveying enthusiasm while reporting on problems (Brooks, 1988; Mercurio and Erickson, 1990).

The technology can be exotic and expensive. EyePhones comprise a pair of small color LCD computer displays (360 x 240 pixels) mounted in a soft swimmask (it weighs about 4 pounds). DataGloves are made of fiber optic sensors sewn to spandex that measure the degree of bending of each finger joint (Figure 5.23). Three-dimensional position and orientation information about the head and hand are captured by Polhemus trackers. However, clever researchers are quickly finding ways to bring down the cost (Pausch, 1991) and to apply VR ideas to existing applications.

Architects have been using computers to draw three-dimensional representations of building for more than a decade. Most of their design systems show the building on a standard or slightly larger display, but adding a large screen projector to create a wall-sized image gives prospective clients a more realistic impression. Now add animation that allows clients to see what

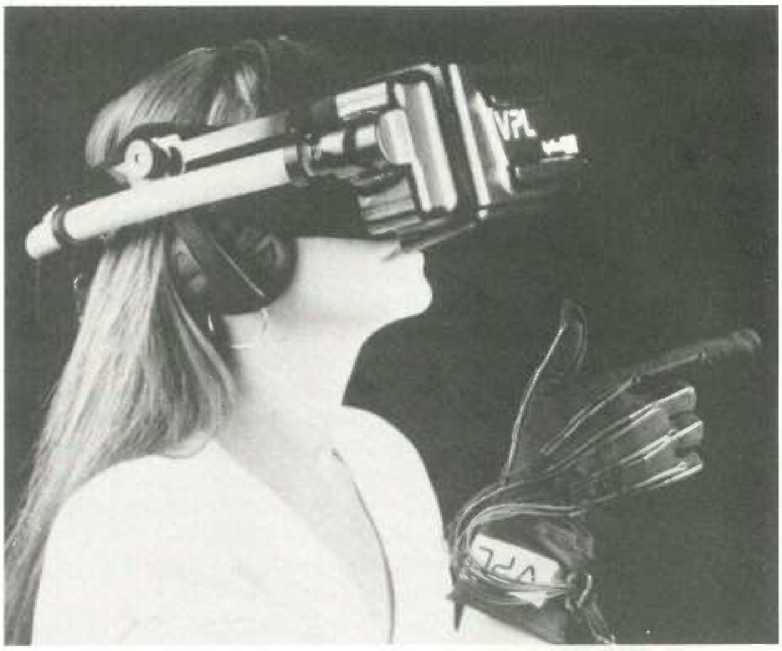


Figure 5.23

EyePhones and DataGlove from VPL Research are components of many virtual-reality (VR) systems. (EyePhones and DataGlove, VPL Research, Inc., Redwood City, CA.)

happens if they move left or right, or approach the image. Then enable clients to control the animation by walking on a treadmill (faster walking brings the building closer more quickly), and allow them to walk through the doors or up the stairs. Finally, replace the large-screen projector with EyePhones, and monitor head movement with Polhemus trackers. Each change takes users a bit farther along the range from "looking at" to "being in." Bumping into walls, falling down stairs, meeting other people, or having to wait for an elevator could be the next variations.

The architectural application is a persuasive argument for "being in" because we are used to "being in" buildings and moving around them. On the other hand, for many applications "looking at" is often more effective, which is why air-traffic–control workstations place the viewer above the situation display. Similarly, the large wraparound movie screens that put viewers "in" race cars or airplanes are special events compared to the more common "looking at" television experience. The Living Theater of the 1960s created an involving theatrical experience and "be-ins" were popular, but most theater goers prefer to take their "suspension of disbelief" experiences from the "looking at" perspective (Laurel, 1991).

It remains to be seen whether doctors, accustomed to "looking at" a patient, really want to crawl through the patient's lungs or "be in" the patient's brains. Modern surgical procedures and technology can benefit by "looking at" video images from inside a patient's heart taken through fiber-optic cameras and by remote direct-manipulation devices that minimize the invasive surgery. There are more mundane applications to such video and fiber-optic magic; imagine the benefits to household plumbers of being able to see lost wedding rings around the bends of a sink drain or to see and grasp the child's toy down the pipes of a clogged toilet.

Other concepts that were sources for the current VR excitement include artificial reality, pioneered by Myron Krueger (1980; 1991). His VideoPlace and VideoDesk installations with large-screen projectors and video recognition sensors combined full-body movement with projected images of light creatures that walked along a performer's arm or multicolored patterns and sounds generated by the performer's movement. Similarly, Vincent Vincent's demonstrations of the Mandala system carried performance art to a new level of sophistication and fantasy (Color Plate 4).

The telepresence perspective is, as Brenda Laurel of Telepresence Research says, "Be here there"—or is it "Be there here"? Either way, the message is to break the physical limitations of space and to act as though the user is somewhere else. Practical thinkers immediately grasp the connection to remote direct manipulation, remote control, and remote vision (Sheridan, 1987), but the fantasists quickly recognize the potential to escape current reality and to visit science-fiction worlds, cartoonlands, previous times in history, galaxies with different laws of physics, or unexplored emotional territories. And while telepresence trippers are changing the world around them, they can also change their own body features or personality.

Scientific analyses of VR user-interface issues may be too sober a process for those who are enjoying their silicon trips, but it may aid in choosing the appropriate applications and refining the technologies. The direct-manipulation principles and the SSOA model are helpful in designing VRs. Users should be able to select actions rapidly by pointing or gesturing, with incremental and reversible control, and display feedback should occur immediately to convey the sense of causality. Complex syntax should never be used, computer concepts should be minimized, and the users should be in the task domain as much as possible. The world of actions should contain visual representations of the objects *and* actions of interest; the surgeon's instruments should be readily available or easily called up by spoken command or gesture. Similarly, an architect walking through a house with a client should be able to pick up a window-stretching tool to try out a larger window, or a room-painting tool to change the wall colors while leaving the windows and furniture untouched. Since applications may require numerous tools, a menu-selection method or tool box is needed.

VR scenarios are such fun to get into, but the scientific analysis of components is a useful complement:

- *Visual display:* The normal computer display at a normal viewing distance (70 centimeters) subtends an angle of about 5 degrees, large screen displays can cover a 20- to 30-degree field, and the EyePhones cover 100 degrees horizontally and 60 degrees vertically. The EyePhones are mounted so as to block other images, so the effect is more dramatic, and head motion produces new images, so the users can get the impression of 360 degree coverage. Flight simulators also block extraneous images, but they do so without forcing the users to wear the sometimes-cumbersome EyePhones. Could a room with four walls of rear-projected displays and a black ceiling offer satisfying experiences? It seems interesting to consider the direct opposite of "being in": Palmtop computers or watch-sized computers exemplify the extreme of "looking at" because of their small displays. Are there personality correlates for users who want to "be in" versus those who prefer to "look at"?

  As hardware technology improves, it will be possible to provide more rapid and higher-resolution images. Most researchers agree that the displays must approach real time (probably under 100 millisecond delay) in presenting the images to the users. Low-resolution displays are acceptable while users or the objects are moving, but when users stop to stare, higher resolution is necessary to preserve the sense of "being in." Improved hardware and algorithms are needed to display rough shapes rapidly and then to fill in the details when the motion stops. A further requirement is that motion be smooth; both incremental changes and continuous display of the objects of interest are required.

  Graphics researchers have been perfecting image display to account for various lighting effects, textured surfaces, reflections, and shadows. All these phenomena need to be included in VR applications. Data structures and algorithms for zooming in or panning across an object rapidly and smoothly are still needed. Dealing with three-dimensional models to permit or prevent bumping into or going through are a good challenge for computer scientists interested in computational geometry. And why limit thinking to three dimensions? Four-dimensional realities may become more understandable after a

VR visit (Banchoff, 1990). And why stop at four dimensions? Multidimensional statistics problems are common in meteorology, sociology, and economics (Lewis et al., 1991). Feiner and Beshers (1990) built a six-dimensional financial data model that the user manipulated with a DataGlove and viewed with a StereoGraphics CrystalEyes LCD stereo system.

- *Head position sensing:* The EyePhones and other head-mounted displays (HMDs) can provide differing displays depending on head position. Look to the right, and you see a forest; look to the left, and the forest gives way to a city. The Polhemus tracker requires mounting on the user's head, but other devices embedded in a hat or eyeglasses are possible. Video recognition of head position is possible. Sensor precision should be high (within 1 degree) and rapid (within 100 milliseconds). Eye tracking to recognize the focus of attention might be useful, but it is difficult to accomplish while the user is moving and wearing EyePhones.
- *Hand-position sensing:* The DataGlove is a highly innovative invention; it surely will be refined and improved beyond its current low resolution. It may turn out that accurate measurement of finger position is required only for one or two fingers or for only one or two joints. Hand orientation is provided by a Polhemus tracker mounted on the glove or wrist. Sensors for other body parts such as knees, arms, or legs may yet find uses. The potential for sensors and tactile feedback on more erotic body parts has been referred to by more than one journalist.
- *Force feedback:* Hand-operated remote-control devices for performing experiments in chemistry laboratories or for handling nuclear materials provide force feedback that gives users a good sense of when they grasp an object or bump into one. Force feedback to car drivers and pilots is carefully configured to provide realistic and useful tactile information. Recreating such experiences from purely software sources is a novelty, but the University of North Carolina's success with force feedback to indicate docking of two complex molecules has stimulated several researchers (Brooks, 1988). It might be helpful for surgeons to receive force feedback as they perform novel procedures or practice difficult operations. Remote handshaking as part of a video conference has been suggested, but it is not clear that the experience could be as satisfying as the real thing.
- *Sound input and output:* Sound output adds realism to bouncing balls, beating hearts, or dropping vases, as videogame designers have found out long ago. Making convincing sounds at the correct moment with

full stereo effect is possible, but it too is hard work. The digital sound hardware is adequate, but the software tools are still inadequate. Music output from virtual instruments is promising; early work simulates existing instruments such as a violin, but novel instruments seem likely to follow quickly. Speech recognition may complement hand gestures in some applications. Discrete spoken commands are useful to solve the "two-cursor problem" (Brooks, 1988): one cursor points at an object or a direction, while the other cursor selects an action from a menu. Two DataGloves could be used, but speech recognition plus one DataGlove seems a workable alternative.

- *Other sensations:* The tilting and vibration of flight simulators might provide other clues and experiences for VR designers. Could a tilting and vibrating virtual roller coaster become popular if users could travel at 60, 600, or 6000 miles per hour and crash through mountains or go into orbit? Other effects such as a throbbing disco sound and strobe lights could also amplify some VR experiences. Why should we not include real gusts of air, made hot or cold to convey the virtual weather? Finally, the power of smells to evoke strong reactions has been understood by writers from Proust to Gibson. Olfactory computing has been discussed, but appropriate and practical applications have yet to be found.
- *Cooperative and competitive virtual reality:* Computer-supported cooperative work (Chapter 10) is a lively research area, so why not cooperative VRs, or as one developer called it, "virtuality built for two." Two people at remote sites could work together in the same VR, seeing each other's actions and sharing the experience. Competitive games such as virtual racquetball have been built for two players. Software for training Army tank crews took on a much more compelling atmosphere when the designs shifted from playing against the computer to shooting at other tank crews and worrying about their attacks. The realistic sounds created such a sense of engagement that crews experienced elevated heart rates, more rapid breathing, and increased perspiration. Presumably, VRs could also bring relaxation and pleasant encounters with other people. VR is an inadequate substitute for being there, but children who appreciate telephone calls from their parents, might like a VR call from their parents while the parents are away on business.

VR's compelling attraction should be acknowledged, and its positive aspects applied. Cyberphilia is spreading rapidly on an international basis and will inevitably find its niches—whether only as an amusement, or instead as a widely applied technology, remains to be seen.

Table 5.1

Direct manipulation definition with benefits and concerns.

Direct Manipulation

Definition

- Visual representation (metaphor) of the world of action
  Objects and actions are shown
  Analogical reasoning is tapped
- Rapid, incremental, and reversible actions
- Typing replaced by pointing and selecting
- Results of actions visible immediately

Benefits

- Control and display compatibility
- Less syntax, resulting in reduced error rates
- More preventable errors
- Faster learning and higher retention
- Exploration encouraged

Concerns

- Increased system resources
- Cumbersome actions
- Weak macro techniques
- Difficult history and tracing
- Difficult for visually impaired users

5.9 Practitioner's Summary

Among interactive systems that provide equivalent functionality and reliability, some systems emerge to dominate the competition. Often, the most appealing systems have an enjoyable user interface that offers a natural representation of the task objects and actions—hence the term *direct manipulation* (Table 5.1). These systems are easy to learn, use, and retain over time. Novices can acquire a simple subset of the commands and then progress to more elaborate operations. Actions are rapid, incremental, and reversible, and can be performed with physical actions instead of complex syntactic forms. The results of operations are visible immediately, and error messages are needed less often.

Just because direct-manipulation principles have been used in a system does not ensure that system's success. A poor design, slow implementation, or inadequate functionality can undermine acceptance. For some applications, menu selection, form fillin, or command languages may be more appropriate. However, the potential for direct-manipulation programming, remote direct manipulation, and virtual reality is great. Many new products will certainly emerge. Iterative design (Section 13.2) is especially important in testing direct-manipulation systems, because the novelty of this approach may lead to unexpected problems for designers and users.

5.10 Researcher's Agenda

Research needs to be done to refine our understanding of the contribution of each feature of direct manipulation: analogical representation, incremental operation, reversibility, physical action instead of syntax, immediate visibility of results, graceful evolution, and graphic form. The relative merits of competing analogical or metaphorical representations could be better understood through experimental comparisons (Bewley et al., 1983). Reversibility is easily accomplished by a generic UNDO command, but designing natural inverses for each operation may be more attractive. Can complex actions always be represented with direct manipulation, or is there a point at which command syntax becomes appealing?

As the hardware improves, we can begin to realize the fantasies of user-interface researchers. Beyond the office desktops, the laptops, and the kitchen countertops, there is the allure of telepresence and virtual realities. The playful aspect should be pursued, but the challenge is to find the practical situations in which people need and want to be in and part of the application, not just to be observers.

If researchers and designers can free themselves to think visually, then the future of direct manipulation is promising. Tasks that could have been performed only with tedious command or programming languages may soon be accessible through lively, enjoyable interactive systems that reduce learning time, speed performance, and increase satisfaction.

References

Arnheim, Rudolf, *Visual Thinking*, University of California Press, Berkeley, CA (1972).

Banchoff, Thomas F., *Beyond the Third Dimension: Geometry, Computer Graphics, and Higher Dimensions*, Scientific American Library, New York (1990).

Bewley, William L., Roberts, Teresa L., Schroit, David, and Verplank, William L., Human factors testing in the design of Xerox's 8010 "Star" Office Workstation, *Proc. CHI '83 Conference—Human Factors in Computing Systems*, ACM, New York (1983), 72–77.

Brooks, Frederick, Grasping reality through illusion: Interactive graphics serving science, *Proc. CHI '88 Conference—Human Factors in Computing Systems*, ACM, New York (1988), 1–11.

Brooks, Kenneth P., Lilac: A two-view document editor, *IEEE Computer 24*, 6 (June 1991), 7–19.

Bruner, James, *Toward a Theory of Instruction*, Harvard University Press, Cambridge, MA (1966).

Card, Stuart K., Robertson, George G., and Mackinlay, Jock D., The Information Visualizer, an information workspace, *Proc. CHI '91 Conference—Human Factors in Computing Systems*, ACM, New York (1991), 181–188.

Carroll, J. M., The adventure of getting to know a computer, *IEEE Computer 15* (1982), 49–58.

Carroll, John M. and Thomas, John C., Metaphor and the cognitive representation of computing systems, *IEEE Transactions on Systems, Man, and Cybernetics, SMC-12*, 2 (March/April 1982), 107–116.

Carroll, J. M., Thomas, J. C., and Malhotra, A., Presentation and representation in design problem-solving, *British Journal of Psychology 71* (1980), 143–153.

Chang, Shi-Kuo (Editor), *Principles of Visual Programming*, Prentice-Hall, Englewood Cliffs, NJ (1990).

Copeland, Richard W., *How Children Learn Mathematics* (Third Edition), MacMillan, New York (1979).

Crawford, Chris, *The Art of Computer Game Design: Reflections of a Master Game Designer*, Osborne/McGraw-Hill, Berkeley, CA (1984).

Cypher, Allen, EAGER: Programming repetitive tasks by example, *Proc. CHI '91 Conference—Human Factors in Computing Systems*, ACM, New York (1991), 33–39.

Feiner, Steven and Beshers, Clifford, Worlds within worlds: Metaphors for exploring *n*-dimensional virtual worlds, *Proc. User Interface Software and Technology '90*, ACM, New York (1990), 76–83.

Gittins, David, Icon-based human–computer interaction, *International Journal for Man–Machine Studies 24* (1986), 519–543.

Glinert, Ephraim P., Kopache, Mark E., and McIntyre, David W., Exploring the general-purpose visual alternative, *Journal of Visual Languages and Computing 1* (1990), 3–39.

Halbert, Daniel, *Programming by Example*, Ph.D. dissertation, Department of Electrical Engineering and Computer Systems, University of California, Berkeley, CA; available as Xerox Report OSD-T8402, Xerox PARC, Palo Alto, CA (1984).

Heckel, Paul, *The Elements of Friendly Software Design: The New Edition*, SYBEX, San Francisco (1991).

Herot, Christopher F., Spatial management of data, *ACM Transactions on Database Systems 5*, 4, (December 1980), 493–513.

Herot, Christopher, Graphical user interfaces. In Vassiliou, Yannis (Editor), *Human Factors and Interactive Computer Systems*, Ablex, Norwood, NJ (1984), 83–104.

Hollan, J. D., Hutchins, E. L., and Weitzman, L., STEAMER: An interactive inspectable simulation-based training system, *AI Magazine* (Summer 1984), 15–27.

Hutchins, Edwin L., Hollan, James D., and Norman, Don A., Direct manipulation interfaces. In Norman, Don A. and Draper, Stephen W. (Editors), *User Centered*

System Design: New Perspectives on Human–Computer Interaction, Lawrence Erlbaum Associates, Hillsdale, NJ (1986), 87–124.

Iseki, Osamu and Shneiderman, Ben, Applying direct manipulation concepts: Direct Manipulation Disk Operating System (DMDOS), *Software Engineering Notes 11*, 2, (March 1986), 22–26.

Krueger, Myron, *Artificial Reality*, Addison-Wesley, Reading, MA (1980).

Krueger, Myron, *Artificial Reality II*, Addison-Wesley, Reading, MA (1991).

Laurel, Brenda, *Computers as Theater*, Addison-Wesley, Reading, MA (1991).

Lewis, J. Bryan, Koved, Lawrence, and Ling, Daniel T., Dialogue structures for virtual worlds, *Proc. CHI '91 Conference—Human Factors in Computing Systems*, ACM, New York (1991), 131–136.

McKim, Robert H., *Experiences in Visual Thinking*, Brooks/Cole, Monterey, CA (1972).

Malone, Thomas W., What makes computer games fun? *BYTE 6*, 12 (December 1981), 258–277.

Marcus, Aaron, *Graphic Design for Electronic Documents and User Interfaces*, ACM Press, New York (1992).

Margono, Sepeedeh and Shneiderman, Ben, A study of file manipulation by novices using commands versus direct manipulation, *Twenty-sixth Annual Technical Symposium*, ACM, Washington, DC (June 1987), 154–159.

Maulsby, David L. and Witten, Ian H., Inducing programs in a direct-manipulation environment, *Proc. CHI '89 Conference—Human Factors in Computing Systems*, ACM, New York (1989), 57–62.

Mercurio, Philip J. and Erickson, Thomas D., Interactive scientific visualization: An assessment of a virtual reality. In Diaper, D., Gilmore, D., Cockton, G., and Shackel, B. (Editors), *Human–Computer Interaction: Interact '90*, North-Holland, Amsterdam, The Netherlands (1990), 741–745.

Montessori, Maria, *The Montessori Method*, Schocken, New York (1964).

Morgan, K., Morris, R. L., and Gibbs, S., When does a mouse become a rat? or... Comparing performance and preferences in direct manipulation and command line environment, *The Computer Journal 34*, 3 (1991), 265–271.

Myers, Brad A., Demonstrational programming: A step beyond direct manipulation. In Diaper, Dan and Hammond, Nick (Editors), *People and Computers VI*, Cambridge University Press, Cambridge, England (1991), 11–30.

Nelson, Ted, Interactive systems and the design of virtuality, *Creative Computing 6*, 11, (November 1980), 56 ff., and 12 (December 1980), 94 ff.

Norman, Donald A., *The Psychology of Everyday Things*, Basic Books, New York (1988).

Norman, Kent, *The Psychology of Menu Selection: Designing Cognitive Control at the Human/Computer Interface*, Ablex, Norwood, NJ (1991).

Papert, Seymour, *Mindstorms: Children, Computers, and Powerful Ideas*, Basic Books, New York (1980).

Pausch, Randy, Virtual reality on five dollars a day, *Proc. CHI '91 Conference—Human Factors in Computing Systems*, ACM, New York (1991), 265–270.

Plaisant, Catherine and Shneiderman, Ben, Scheduling ON–OFF home control devices: Design issues and usability evaluation of four touchscreen interfaces, *International Journal for Man–Machine Studies* (in press).

Plaisant, C., Shneiderman, B., and Battaglia, J., Scheduling home-control devices: A case study of the transition from the research project to a product, *Human-Factors in Practice*, Computer Systems Technical Group, Human-Factors Society, Santa Monica, CA (December 1990), 7–12.

Polya, G., *How to Solve It*, Doubleday, New York, (1957).

Potter, Richard, Just in Time Programming: Long Overdue, Ph.D. Dissertation, Department of Computer Science, University of Maryland, College Park, MD (1992).

Rheingold, Howard, *Virtual Reality*, Simon and Schuster, New York (1991).

Roberts, Teresa L., Evaluation of Computer Text Editors, Ph.D. dissertation, Department of Computer Science, Stanford University. Available from University Microfilms, Ann Arbor, MI, Order Number AAD 80-11699. Stanford, CA (1980).

Rogers, Yvonne, Icons at the interface: Their usefulness, *Interacting with Computers 1*, 1 (1989), 105–117.

Rubin, Robert V., Golin, Eric J., and Reiss, Steven P., Thinkpad: A graphics system for programming by demonstrations, *IEEE Software 2*, 2 (March 1985), 73–79.

Rutkowski, Chris, An introduction to the Human Applications Standard Computer Interface, Part 1: Theory and principles, *BYTE 7*, 11 (October 1982), 291–310.

Sheridan, T. B., Teleoperation, telepresence, and telerobotics: Research needs for space. In Sheridan, T. B., Kruser, D. S., and Deutsch, S. (Editors), *Human Factors in Automated and Robotic Space Systems*, Proceedings, National Research Council, Washington, DC (1987), 279–291.

Sheridan, T. B., Task allocation and supervisory control. In Helander, M. (Editor), *Handbook of Human–Computer Interaction*, Elsevier Science Publishers, Amsterdam, The Netherlands (1988), 159–173.

Shneiderman, Ben, A computer graphics system for polynomials, *The Mathematics Teacher 67*, 2 (1974), 111–113.

Shneiderman, Ben, Control flow and data structure documentation: Two experiments, *Communications of the ACM. 25*, 1 (January 1982), 55–63.

Shu, Nan C., *Visual Programming*, Van Nostrand Reinhold, New York (1989).

Smith, David Canfield, *Pygmalion: A Computer Program to Model and Stimulate Creative Thought*, Birkhauser Verlag, Basel, Switzerland (1977).

Smith, D. Canfield, Irby, Charles, Kimball, Ralph, Verplank, Bill, and Harslem, Eric, Designing the Star user interface, *BYTE 7*, 4 (April 1982), 242–282.

Stewart, Doug, Through the looking glass into an artificial world—via computer, *Smithsonian*, (January 1991), 36–45.

Temple, Barker and Sloane, Inc., The benefits of the graphical user interface, *Multimedia Review* (Winter 1990), 10–17.

Thimbleby, Harold, *User Interface Design*, ACM Press, New York (1990).

Tullis, T. S., An evaluation of alphanumeric, graphic and color information displays, *Human Factors* 23 (1981), 541–550.

Uttal, W. R., Teleoperators, *Scientific American 261*, 6 (December 1989), 124–129.

Van de Vegte, J. M. E., Milgram, P., Kwong, R. H., Teleoperator control models: Effects of time delay and imperfect system knowledge, *IEEE Transactions on Systems, Man, and Cybernetics 20*, 6 (November/December 1990), 1258–1272.

Verplank, William L., Graphic challenges in designing object-oriented user interfaces. In Helander, M. (Editor), *Handbook of Human–Computer Interaction*, Elsevier Science Publishers, Amsterdam, The Netherlands (1988), 365–376.

Weinstein, R., Bloom, K., Rozek, S., Telepathology: Long distance diagnosis, *American Journal of Clinical Pathology*, 91 (Suppl 1) (1989), S39–S42 .

Wertheimer, M., *Productive Thinking*, Harper and Row, New York (1959).

Ziegler, J. E. and Fähnrich, K.-P., Direct manipulation. In Helander, M. (Editor), *Handbook of Human–Computer Interaction*, Elsevier Science Publishers, Amsterdam, The Netherlands (1988), 123–133.

Zloof, M. M., Query-by-Example, *Proceedings of the National Computer Conference*, AFIPS Press, Montvale, NJ (1975), 431–438.

Zloof, M. M., Office-by-Example: A business language that unifies data and word processing and electronic mail, *IBM Systems Journal 21*, 3 (1982), 272–304.