Exhibit 1014 – Part 7

Grudin, Jonathan and Barnard, Phil, When does an abbreviation become a word and related questions, *Proc. CHI '85 Conference on Human Factors in Computer Systems*, ACM, New York (1985), 121–126.

Hanson, Stephen J., Kraut, Robert E., and Farber, James M., Interface design and multivariate analysis of UNIX command use, *ACM Transactions on Office Information Systems* 2, 1 (1984), 42–57.

Hauptmann, Alexander G. and Green, Bert F., A comparison of command, menu-selection and natural language computer programs, *Behaviour and Information Technology* 2, 2 (1983), 163–178.

Hayes-Roth, Frederick, The knowledge-based expert system: A tutorial, *IEEE Computer 17, 9* (September 1984), 11–28.

Jarke, Matthias, Turner, Jon A., Stohr, Edward A., Vassiliou, Yannis, White, Norman H., and Michielsen, Ken, A field evaluation of natural language for data retrieval, *IEEE Transactions on Software Engineering SE-11*, 1 (January 1985), 97–113.

Kraut, Robert E., Hanson, Stephen J., and Farber, James, M., Command use and interface design, *Proc. CHI '83 Conference on Human Factors in Computing Systems*, ACM, New York (1983), 120–123.

Landauer, T. K., Calotti, K. M., and Hartwell, S., Natural command names and initial learning, *Communications of the ACM 26*, 7 (July 1983), 495–503.

Ledgard, H., Whiteside, J. A., Singer, A., and Seymour, W., The natural language of interactive systems, *Communications of the ACM 23* (1980), 556–563.

Napier, H. Albert, Lane, David, Batsell, Richard R., and Guadango, Norman S., Impact of a restricted natural language interface on ease of learning and productivity, *Communications of the ACM 32*, 10 (October 1989), 1190–1198.

Norman, Donald, The trouble with UNIX, *Datamation 27* (November 1981), 139–150.

Pausch, Randy and Leatherby, James H., An empirical study: Adding voice input to a graphical editor, *Journal of the American Voice Input/Output Society 9*, 2 (July 1991), 55–66.

Roberts, Terry, *Evaluation of Computer Text Editors*, Ph. D. dissertation, Department of Computer Science, Stanford University, Stanford, CA (1980).

Rosenberg, Jarrett, Evaluating the suggestiveness of command names, *Behaviour and Information Technology 1* (1982), 371–400.

Rosson, Mary Beth, Patterns of experience in text editing, *Proc. CHI '83 Conference on Human Factors in Computing Systems*, ACM, New York (1983), 171–175.

Scapin, Dominique L., Computer commands labelled by users versus imposed commands and the effect of structuring rules on recall, *Proc. Conference on Human Factors in Computer Systems*, available from ACM, Washington, DC (1982), 17–19.

Schneider, M. L., Ergonomic considerations in the design of text editors, In Vassiliou, Y. (Editor), *Human Factors and Interactive Computer Systems*, Ablex, Norwood, NJ (1984), 141–161.

Schneider, M. L., Hirsh-Pasek, K., and Nudelman, S., An experimental evaluation of delimiters in a command language syntax, *International Journal of Man–Machine Studies* 20, 6 (June 1984), 521–536.

Shneiderman, Ben, *Software Psychology: Human Factors in Computer and Information Systems*, Little, Brown, Boston (1980).

Small, Duane and Weldon, Linda, An experimental comparison of natural and structured query languages, *Human Factors* 25 (1983), 253–263.

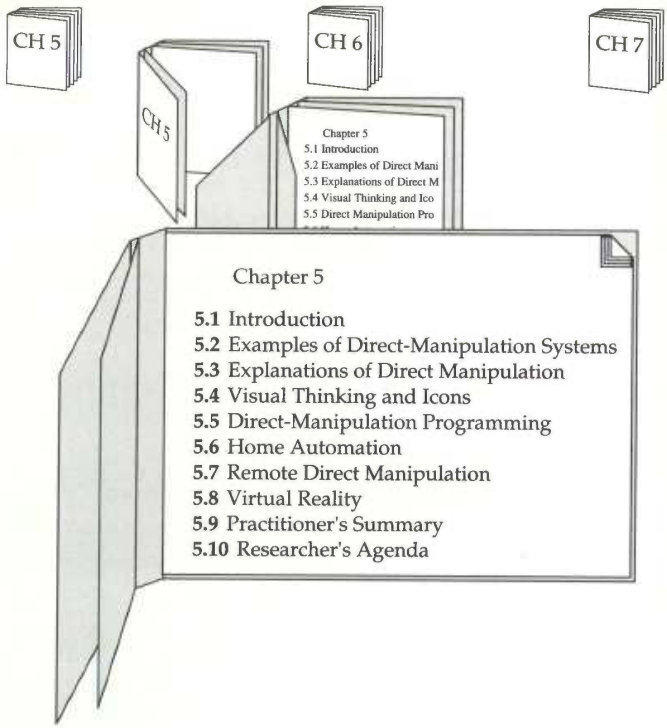
Tennant, Harry R., Ross, Kenneth M., and Thompson, Craig W., Usable natural language interfaces through menu-based natural language understanding, *Proc. CHI '83 Conference on Human Factors in Computing Systems*, ACM, New York (1983), 154–160.

CHAPTER 5

Direct Manipulation

Leibniz sought to make the form of a symbol reflect its content. "In signs," he wrote, "one sees an advantage for discovery that is greatest when they express the exact nature of a thing briefly and, as it were, picture it; then, indeed, the labor of thought is wonderfully diminished."

Frederick Kreiling, "Leibniz," ***Scientific American,*** **May 1968**



5.1 Introduction

Certain interactive systems generate a glowing enthusiasm among users that is in marked contrast with the more common reaction of grudging acceptance or outright hostility. The enthusiastic users' reports reflect the following positive feelings:

- Mastery of the system
- Competence in performing tasks
- Ease in learning the system originally and in assimilating advanced features
- Confidence in the capacity to retain mastery over time
- Enjoyment in using the system
- Eagerness to show the system off to novices
- Desire to explore more powerful aspects of the system

These feelings are not universal, but this amalgam is meant to convey an image of the truly pleased user. The central ideas seem to be visibility of the objects and actions of interest; rapid, reversible, incremental actions; and replacement of complex command-language syntax by direct manipulation of the object of interest—hence, the term *direct manipulation.* The SSOA model provides a sound foundation for understanding direct manipulation, since it steers the designer to represent the task domain objects and actions, while minimizing the computer concepts and the syntactic load.

5.2 Examples of Direct-Manipulation Systems

No single system has all the admirable attributes or design features—that may be impossible. Each of the following examples, however, has enough of them to win the enthusiastic support of many users.

My favorite example of direct manipulation is driving an automobile. The scene is directly visible through the front window, and performance of actions such as braking or steering has become common knowledge in our culture. To turn left, the driver simply rotates the steering wheel to the left. The response is immediate and the scene changes, providing feedback to refine the turn. Imagine trying to turn by issuing a command `LEFT 30 DEGREES` and then another command to see the new scene; but that is the level of operation of many office-automation tools of today! Another well-established example is air-traffic control in which users see a representation of the airspace with brief data blocks attached to each plane. Controllers move a trackball to point at specific planes and to perform actions.

5.2.1 Display editors and word processors

In the early 1980s, users of *full-page display editors* were great advocates of their systems, preferring these editors to the then-common *line-oriented text editors.* A typical comment was, "Once you've used a display editor, you will never want to go back to a line editor—you'll be spoiled." Similar comments came from users of stand-alone word processors such as the WANG system, early personal computer word processors such as WORDSTAR, FINALWORD, XYWRITE, and Microsoft WORD, or display editors such as EMACS on the MIT/Honeywell MULTICS system or `vi` (for visual editor) on the UNIX system. A beaming advocate called EMACS "the one true editor."

Roberts (1980) found overall performance times with line-oriented editors were twice as long as with display editors. Training time with display editors is also reduced, so there is evidence to support the enthusiasm of

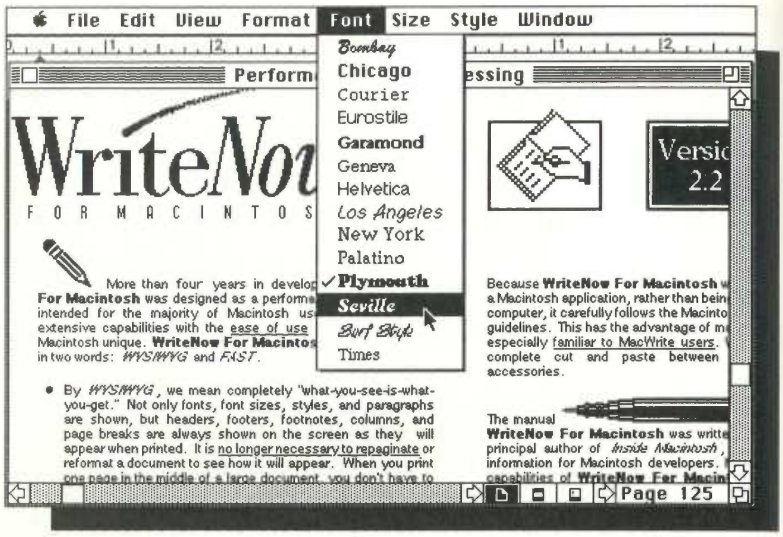


Figure 5.1

WriteNow, an example of a WYSIWYG ("What You See Is What You Get") editor. (Courtesy of T/Maker Company, Mountain View, CA.)

display-editor devotees. Furthermore, office-automation evaluations consistently favor full-page display editors for secretarial and executive use.

In the past decade, *what you see is what you get* (WYSIWYG) word processors have become standard. Claris MacWrite II and Microsoft Word 4.0 are popular on the Macintosh; Microsoft Word and Lotus Ami Pro are challenging WordPerfect's domination in the IBM PC and compatibles environment (Figure 5.1). An interesting combination of WYSIWYG display editors and command editing over structured documents shows two views simultaneously and permits editing on either view (Brooks, 1991). There are some advantages to command editing approaches, such as that history keeping is easier, more flexible markup languages are available (for example, SGML), macros tend to be more powerful, and some tasks are simpler to express (for example, `change all italics to bold`). The next generation of direct-manipulation display editors should accommodate many of these features. The advantages of display editors include

- *Display of a full page of text*: Showing 20 to 60 lines of text gives the reader a clearer sense of context for each sentence, while permitting simpler reading and scanning of the document. By contrast, working with the one-line-at-a-time view offered by some line editors is like seeing the

world through a narrow cardboard tube. Some large displays can support two full pages of text, set side by side.

- *Display of the document in the form that it will appear when the final printing is done*: Eliminating the clutter of formatting commands also simplifies reading and scanning of the document. Tables, lists, page breaks, skipped lines, section headings, centered text, and figures can be viewed in their final form. The annoyance and delay of debugging the format commands are almost eliminated because the errors are apparent immediately.
- *Show cursor action that is visible to the user*: Seeing an arrow, underscore, or blinking box on the screen gives the operator a clear sense of where to focus attention and to apply action.
- *Control cursor motion through physically obvious and intuitively natural means*: Arrow keys or cursor-motion devices—such as a mouse, joystick, or graphic tablet—provide natural physical mechanisms for moving the cursor. This setup is in marked contrast to commands such as `UP 6` that require an operator to convert the physical action into a correct syntactic form that may be difficult to learn and hard to recall, and thus may be a source of frustrating errors.
- *Use labeled buttons for actions*: Many workstations designed for use with display editors have buttons with actions etched onto them, such as `INSERT`, `DELETE`, `CENTER`, `UNDERLINE`, `SUPERSCRIPT`, `BOLD`, and `SEARCH`. These buttons act as a permanent menu-selection display to remind users of the features, so that users avoid memorization of a complex command-language syntax. On some editors, only 10 or 15 labeled buttons provide the basic functionality. A specially marked button may be the gateway to the world of advanced or infrequently used features that are offered on the screen in menu form.
- *Display of the results of an action immediately*: When a button is pressed to move the cursor or center text, the results are shown immediately on the screen. Deletions are immediately apparent; the character, word, or line is erased, and the remaining text is rearranged. Similarly, insertions or text movements are shown after each keystroke or function-key press. In contrast, with line editors, users must issue print or display commands to see the results of changes.
- *Provide rapid response and display*: Most display editors operate at high speed; a full page of text appears in a fraction of a second. This high display rate coupled with short response time produces a thrilling sense of power and speed. Cursors can be moved quickly, large amounts of text can be scanned rapidly, and the results of actions can be shown almost instantaneously. Rapid response also reduces the need for additional commands and thereby simplifies design and learning.

Line editors with slow display rates and long response times bog down the user. Speeding up line editors would add to their attractiveness, but they would still lack such features as direct overtyping, deletion, and insertion.

- *Offer easily reversible actions*: When users enter text, they repair an incorrect keystroke by merely backspacing and overstriking. They can make simple changes by moving the cursor to the problem area and overstriking, inserting, or deleting characters, words, or lines. A useful design strategy is to include natural inverse operations for each operation (Section 4.4.3). An alternative offered by many display editors is a simple UNDO command to return the text to its state before the previous action or action sequence. The easy reversibility reduces user anxiety about making a mistake or destroying the file.

Display editors are worth studying because the large market demand generates an active competition that propels the rapid evolutionary refinement of design. New directions for word processors include

- *Integration* of graphics, spreadsheets, animations, photographs, etc. in the body of a document. Advanced systems, such as Hewlett-Packard's NewWave, even permit "hot links" so that, if the graphic or spreadsheet is changed, the copy in the document also will be changed.
- *Desktop publication software* to produce sophisticated printed formats with multiple columns and output to high-resolution printers. Multiple fonts, gray scales, and color permit preparation of high-quality documents, newsletters, reports, newspapers, or books. Examples include Aldus PageMaker and Xerox Ventura.
- *Slide-presentation software* to produce text and color graphic slides for use as overhead transparencies or 35-millimeter slides, or directly from the computer with a large screen projector.
- *Hypermedia environments* with selectable buttons or embedded menu items to allow users to jump from one article to another. Links among documents, bookmarks, annotations, and tours can be added by readers.
- *Improved macro facilities* to enable users to construct, save, and edit sequences of frequently used actions. A related feature is a style sheet that allows users to specify and save a set of options for spacing, fonts, margins, etc.
- *Spelling checkers* have become standard on most full-feature word processors. Less common, but increasingly available, is an integrated thesaurus.
- *Grammar checkers*, such as RightWriter or Grammatik IV, offer users comments about potential problems in writing style, such as use of

passive voice, excessive use of certain words, or lack of parallel construction. Some writers, both novices and professionals, appreciate the comments and know they can decide whether to apply the suggestions. Critics point out, however, that the advice is often inappropriate and therefore wastes time.

- *Document assemblers* to compose complex documents such as contracts or wills, from standard paragraphs using appropriate language for males or females, citizens or foreigners, high, medium, or low income earners, renters or home owners, etc.

5.2.2 VISICALC and its descendents

The first electronic spreadsheet, VISICALC, was the product of a Harvard Business School student, Bob Frankston, who was frustrated when trying to carry out repetitious calculations in a graduate business course. With a friend, Dan Bricklin, they built an "instantly calculating electronic worksheet" (as the user manual described it) that permits computation and display of results across 254 rows and 63 columns. The worksheet can be programmed so that column 4 displays the sum of columns 1 through 3; then, every time a value in the first three columns changes, the fourth column changes as well. Complex dependencies among manufacturing costs, distribution costs, sales revenue, commissions, and profits can be stored for several sales districts and months so that the effects of changes on profits can be seen immediately.

By simulating an accountant's spreadsheet or worksheet, VISICALC made it easy for novices to comprehend the objects and permissible actions. The display of 20 rows and up to nine columns, with the provision for multiple windows, gave the user sufficient visible display for easy scanning of information and comprehension of relationships among entries. The command language for setting up the worksheet was tricky for novices to learn and for infrequent users to remember, but most users needed to learn only the basic commands. The distributor of VISICALC attributed the system's appeal to the fact that "it jumps," referring to the user's delight in watching the propagation of changes across the screen.

VISICALC users could try out many alternate plans easily, and could see the effects on sales or profit rapidly. Changes to commissions or economic slowdowns could be added quickly to the worksheet. The current status of the worksheet could be saved for later review.

Competitors to VISICALC emerged quickly, and they made attractive improvements to the user interface and expanded the tasks that were supported. Among these, LOTUS 1-2-3 has come to dominate the market (Figure 5.2a), although there are many successful competitors, such as Excel and Quattro. They offer integration with graphics, three-dimensional repre-

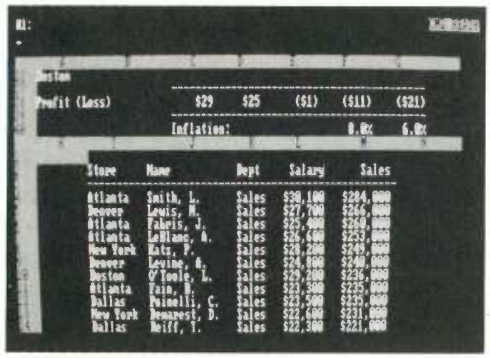


Figure 5.2(a)

Lotus spreadsheets. (Figure 5.2a and b: Printed with permission of Lotus Development Corporation, Cambridge, MA.) (a) Early version of Lotus 1-2-3, the dominant spreadsheet program.

sentations, multiple windows, and database features. The actions are invoked easily with command menus. Advanced systems such as Improv (Figure 5.2b) are attempting to win users with novel ways of showing and manipulating data items and graphs.

5.2.3 Spatial data management

In geographic applications, it seems natural to give a spatial representation in the form of a map that provides a familiar model of reality. The developers of the prototype spatial data-management system (Herot, 1980; 1984) attribute the basic idea to Nicholas Negroponte of MIT. In one early scenario, the user was seated before a color-graphics display of the world and could zoom in on the Pacific Ocean to see markers for convoys of military ships (Figure 5.3). By moving a joystick, the user caused the screen to be filled with silhouettes of individual ships that could be zoomed in on to display detailed data—such as, ultimately, a full-color picture of the captain.

In another scenario, icons representing such different aspects of a corporation as personnel, an organizational chart, travel information, production data, and schedules were shown on a screen. By moving the joystick and zooming in on objects of interest, the user was taken through complex "information spaces" or "I-spaces" to locate the item of interest. A building floorplan showing departments might be displayed; when a department was chosen, individual offices became visible. As the cursor was moved into a room, details of the occupant appeared on the screen. If users chose the wrong room, they merely backed out and tried another. The lost effort was

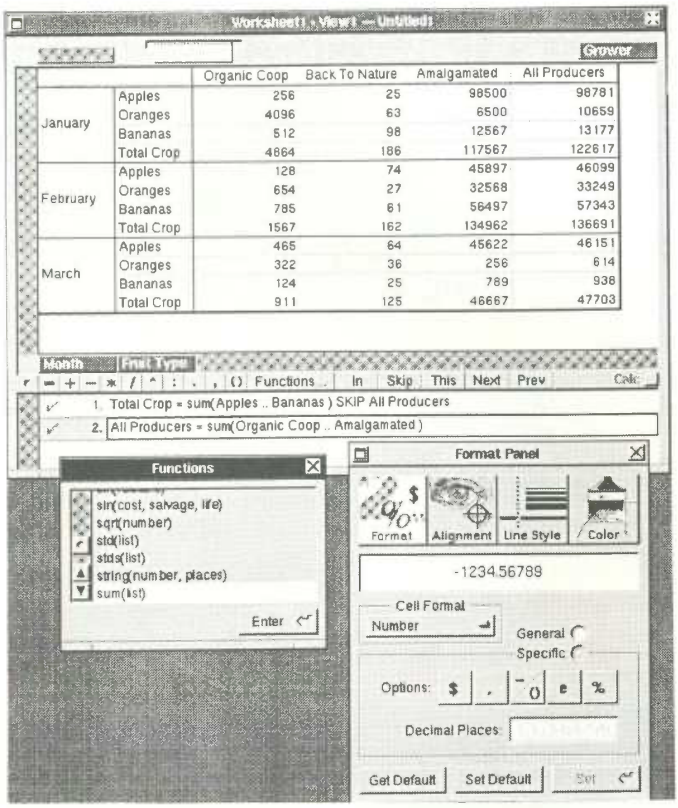


Figure 5.2(b)

Lotus's Improv, a spreadsheet program for the NeXT machine that has novel ways of showing and manipulating data. (Printed with permission of Lotus Development Corporation.)

minimal, and there was no stigma attached to error. The recent Xerox PARC Information Visualizer is an ensemble of tools that permit three-dimensional animated explorations of buildings, cone-shaped file directories, organization charts, a perspective wall that puts featured items up front and centered, and several two- and three-dimensional information layouts (Card et al., 1991).

The Voyager Data Exploration Software for Windows enables users to explore spatial and temporal databases visually. For example, if a map of the United States and an energy-use plot for the previous 30 years are shown on the screen, the user can select a year on the plot and can then see energy use

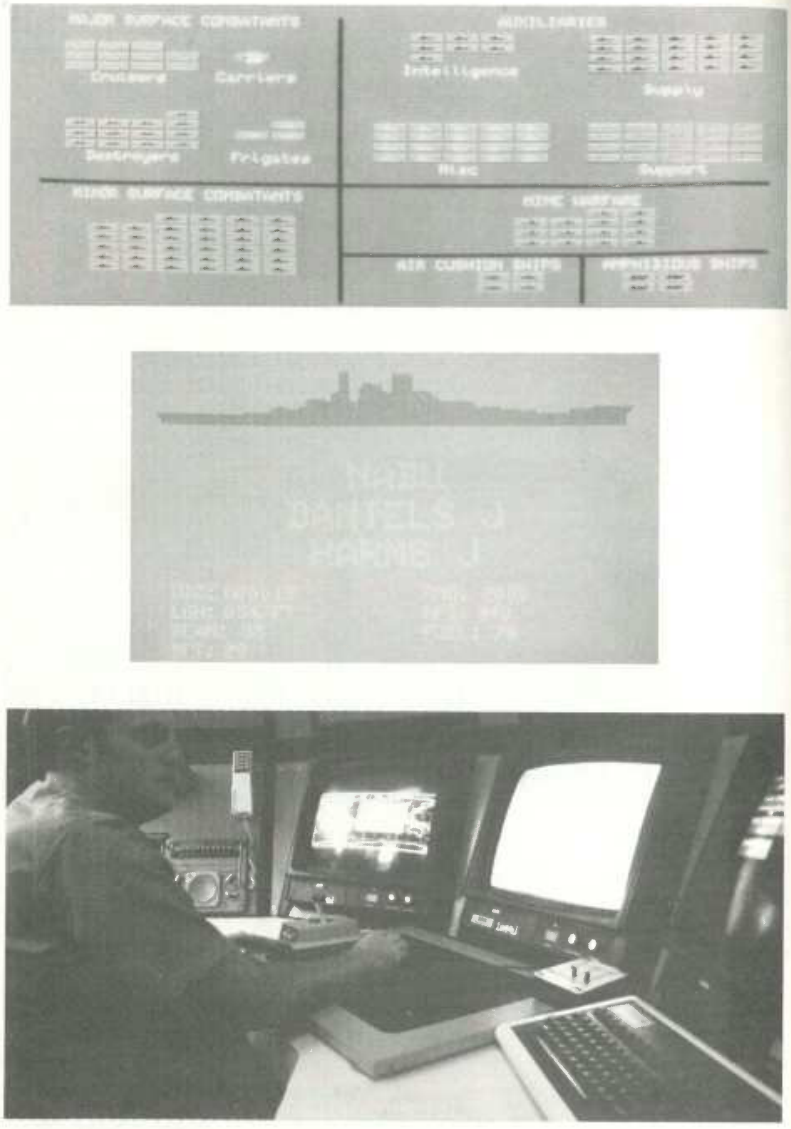


Figure 5.3

The Spatial Data Management System has three displays to show multiple levels of detail or related information. The user moves a joystick to traverse information spaces or to zoom in on a map to see more details about ship convoys. (Courtesy of the Computer Corporation of America, Cambridge, MA.)

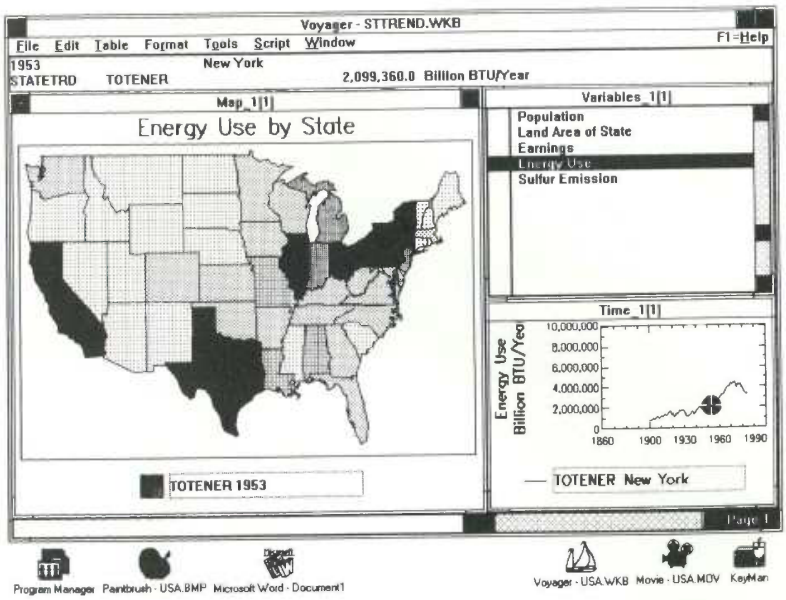


Figure 5.4

The Voyager Data Exploration Software with features for choosing a subset of data to show in detailed plots. Users can click on individual states to see energy use, or on the plot to select the year. (Courtesy of Husar, R. B., Oberman, T., and Hutchins, E. A., Environmental Informatics: Implementation Through the Voyager Data Exploration Software. *Air and Waste Management Association 83rd Annual Meeting*, June 24–29, 1990, Pittsburgh, PA.)

for every state in that year (Figure 5.4). Alternatively, users can retrieve facts about a specific state by pointing and clicking on the map.

The success of a spatial data-management system depends on the skill of the designers in choosing icons, graphical representations, and data layouts that are natural and comprehensible to the user. The joy of zooming in and out, or of gliding over data with a joystick, entices even anxious users, who quickly demand additional power and data.

5.2.4 Video games

For many people, the most exciting, well-engineered, and commercially successful application of these concepts lies in the world of video games (Crawford, 1984). The early but simple and popular game called PONG required the user to rotate a knob that moved a white rectangle on the screen. A white spot acted as a ping-pong ball that ricocheted off the wall

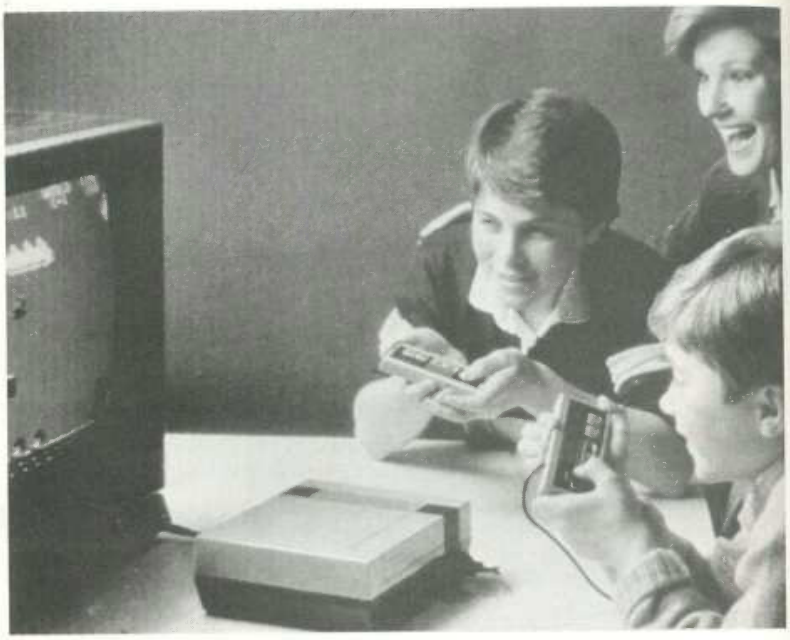


Figure 5.5(a)

Video games. (a) Home video games are enjoyable computer applications that have become extremely popular (©1991 Nintendo. Courtesy of Nintendo).

and had to be hit back by the movable white rectangle. Users developed speed and accuracy in placing the "paddle" to keep the increasingly speedy ball from getting by, while the speaker emitted a ponging sound when the ball bounced. Watching someone else play for 30 seconds is all the training needed to become a competent novice, but many hours of practice are required to become a skilled expert.

Later games, such as Missile Command, Donkey Kong, Pac Man, Tempest, TRON, Centipede, or Space Invaders, were much more sophisticated in their rules, color graphics, and sound effects. Recent games include videodisk images, two-person competition in tennis or karate, still higher resolution, and stereo sound (Figures 5.5a and b). The designers of these games provide stimulating entertainment, a challenge for novices and experts, and many intriguing lessons in the human factors of interface design—somehow, they have found a way to get people to put quarters in the sides of computers. Thirty-million Nintendo game players have penetrated to 70 percent of American households that include 8 to 12 year olds. Brisk sales of

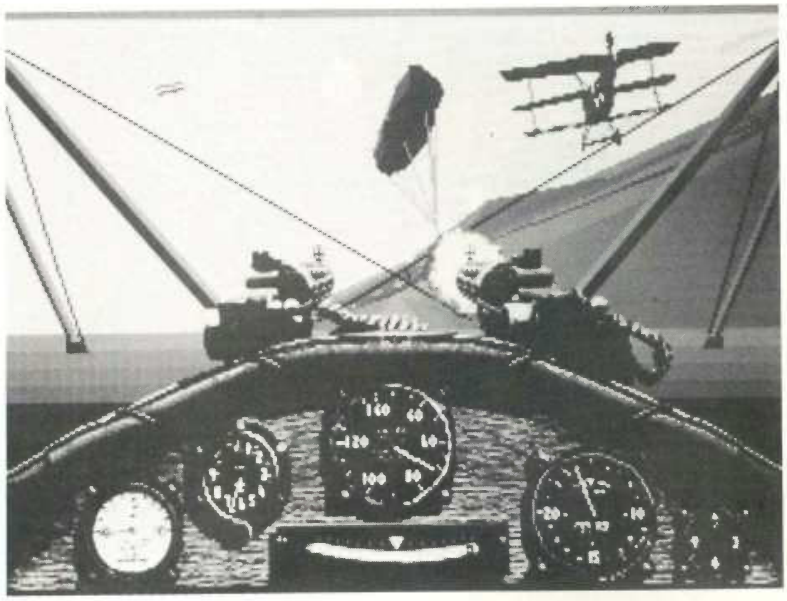


Figure 5.5(b)

Video games employ direct manipulation principles to create a world of action and fantasy, such as a flight simulation of an old plane, *Red Baron*. (©1992 Dynamix, Inc.)

Super Mario Brothers and variations testify to the games' strong attraction, in marked contrast to the anxiety and resistance many users have for office automation equipment.

These games provide a field of action that is simple to understand since it is an abstraction of reality—learning is by analogy. The commands are physical actions, such as button presses, joystick motions, or knob rotations, whose results are shown immediately on the screen. There is no syntax to remember, and therefore there are no syntax-error messages. If users move their spaceships too far left, then they merely use the natural inverse operation of moving back to the right. Error messages are unnecessary because the results of actions are obvious and can be reversed easily. These principles can be applied to office automation, personal computing, or other interactive environments.

Most games continuously display a numeric score so that users can measure their progress and compete with their previous performance, with friends, or with the highest scorers. Typically, the 10 highest scorers get to store their initials in the game for regular display. This strategy provides one

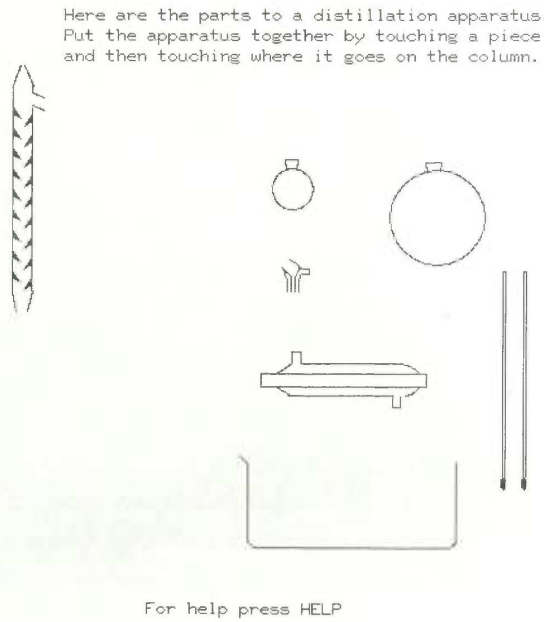


Figure 5.6(a)

Computer-based instruction can become more appealing with direct manipulation, instead of drill and practice. This early CDC PLATO lesson, written by Stan Smith of the Department of Chemistry at the University of Illinois, allows students to construct a distillation aparatus by using proper finger actions on a touch-sensitive screen. (Figure 5.6a and b: Courtesy of Stan Smith, University of Illinois.) (a) Once the student has assembled the apparatus and begun the experiment, the display shows an animation of the process with the graph of distillation temperature versus volume.

form of positive reinforcement that encourages mastery. Malone's (1981) and our studies with elementary-school children have shown that continuous display of scores is extremely valuable. Machine-generated feedback—such as "Very good" or "You're doing great!"—is not as effective, since the same score means different things to different people. Users prefer to make their own subjective judgments and perceive the machine-generated messages as an annoyance and a deception.

Many educational games use direct manipulation effectively. Elementary- or high-school students can learn about logic by using Rocky Boots, which shows logic circuits visually and lets students progress to more complex tasks by going through doors to enter a series of rooms.

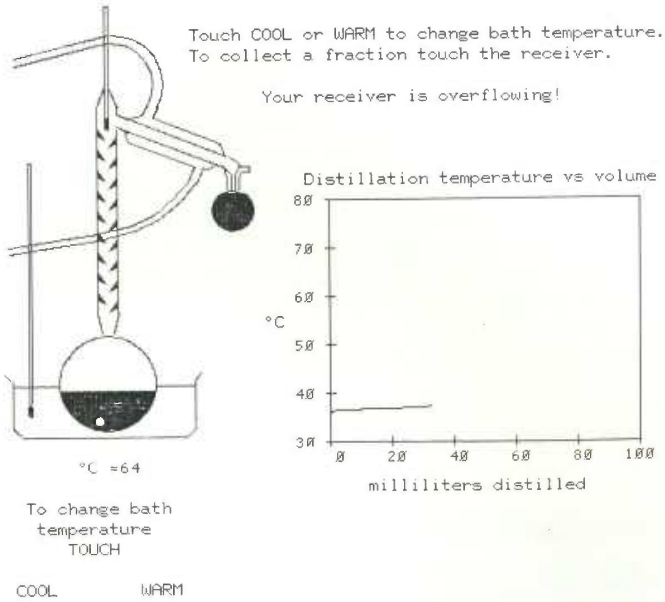


Figure 5.6(b)

The student experimenter has gotten into trouble.

Stan Smith's chemistry lessons on the PLATO system enabled college students to conduct laboratory experiments by touching beakers, pipettes, or burners to assemble and operate equipment (Figure 5.6). A Navy training simulator shows gauges, dials, and knobs that users can manipulate directly to gain experience with boilers, valves, and so on (Hollan et al., 1984). Several versions of the Music Construction Set offer users the possibility of constructing musical scores by selecting and moving notes onto a staff.

Carroll (1982) draws productive analogies between game-playing environments and applications-systems. However, game players are seeking entertainment and focus on the challenge of mastery, whereas applications-systems users focus on their task and may resent the intrusion of forced learning of system constraints. Furthermore, the random events that occur in most games are meant to challenge the user; in nongame designs, however, predictable system behavior is preferred. Game players are engaged in competition with the system, whereas applications-systems users apparently prefer a strong internal locus of control, which gives them the sense of being in charge.

5.2.5 Computer-aided design and manufacturing

Many *computer-aided design* (CAD) systems for automobiles, electronic circuitry, architecture, aircraft, or newspaper layout use principles of direct manipulation (Figure 5.7). The operator may see a circuit schematic on the screen and, with lightpen touches, be able to move resistors or capacitors into or out of the proposed circuit. When the design is complete, the computer can provide information about current, voltage drops, and fabrication costs, and warnings about inconsistencies or manufacturing problems. Similarly, newspaper-layout artists or automobile-body designers can easily try multiple designs in minutes, and can record promising approaches until they find a better one.

The pleasures in using these systems stem from the capacity to manipulate the object of interest directly and to generate multiple alternatives rapidly. Some systems have complex command languages; others have moved to using cursor action and graphics-oriented commands.

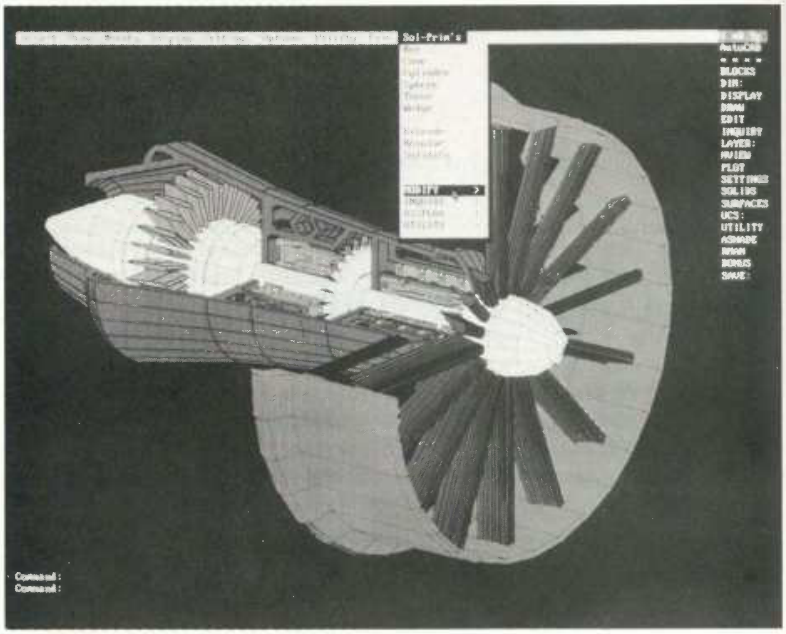


Figure 5.7

Many computer-aided design (CAD) systems use a direct-manipulation interaction style. This design tool, AutoCAD, supports three-dimensional designs. (Courtesy of AutoDesk, Inc., Sausalito, CA.)

Another related direction is the world of *computer-aided manufacturing* (CAM) and process control. Honeywell's process-control system provides the manager of an oil refinery, paper mill, or power-utility plant with a colored schematic view of the plant. The schematic may be on eight displays, with red lines indicating a sensor value that is out of normal range. By pressing a single, numbered button (there are no commands to learn or remember), the operator can get a more detailed view of the troubling component; with a second press, the operator can move down the tree structure to examine individual sensors or to reset valves and circuits.

A basic strategy for this design is to eliminate the need for complex commands that need to be recalled only in once-a-year emergency conditions. The schematic of the plant facilitates problem solving by analogy, since the linkage between real-world high temperatures or low pressures and screen representations is so close.

5.2.6 Office automation, databases, and directories

A large part of the success and appeal of the *Query-by-Example* (Zloof, 1975) approach to data manipulation is due to the direct representation of the relations on the screen (Figure 5.8). The user moves a cursor through the

```
Query:

SKI-RESORTS | NAME | CITY | STATE | LIFTS | VERTICAL
-------------------------------------------------------------------
            | P.   | P.   | NY    |       | P. >1200
            |      |      |       |       |

Response:

SKI-RESORTS | NAME         | CITY         | VERTICAL
-------------------------------------------------------------------
            | BELLEAYRE    | HIGHMOUNT    | 1340
            | GORE         | NORTH CREEK  | 2100
            | HUNTER       | HUNTER       | 1600
            | SKI WINDHAM  | WINDHAM      | 1550
            | WHITEFACE    | WIMLINGTON   | 3216
```

Figure 5.8

Zloof's Query-by-Example system shows users a relational-table skeleton and enables them to fill in literals (such as `NY` or `1200`) and to specify fields to be printed (`P.`). Users can also specify variables to link between relations. In this example, the query produces the `NAME`s of ski resorts in `NY` state that have a vertical drop of more than `1200` feet.

columns of the relational table and enters examples of how the result should look. There are just a few single-letter keywords to supplement the direct-manipulation style. Of course, expressing complex Booleans or mathematical operations requires knowledge of syntactic forms. Still, the basic ideas and facilities in this language can be learned within 1/2 hour by many nonprogrammers. Query-by-Example succeeds because novices can begin working with just a little training, yet there is ample power for the expert. Directly manipulating the cursor across the relation skeleton is simple, and showing the linking variable by giving an example is intuitively clear to someone who understands tabular data. Zloof (1982) expands his ideas into Office-by-Example, which smoothly integrates database search with word processing, electronic mail, business graphics, and menu creation. The OfficeVision line of products from IBM enables integration of OS/2 applications with networks of mainframe machines to support office work.

Other designers of advanced office-automation systems have made use of direct-manipulation principles. The pioneering Xerox Star (Smith et al., 1982) offered sophisticated text-formatting options, graphics, multiple fonts, and a high-resolution, cursor-based user interface (Figure 5.9). Users could move a document icon to a printer icon to generate a hard-copy printout. The Apple Lisa system elegantly applied many of the principles of direct manipulation and, although it was not a commercial success, it laid the groundwork for the successful Macintosh. The Macintosh designers drew from the Star and Lisa experience, but made many simplifying decisions while preserving adequate power for users (Figure 5.10). The hardware and software designs supported rapid and continuous graphical interaction for pull-down menus, window manipulation, editing of graphics and text, and dragging of icons. Variations on the Macintosh appeared soon afterward for other popular personal computers, such as the IBM PS/2 (Figure 5.11).

Studies of users of direct-manipulation interfaces have confirmed the advantages for at least some users and tasks. In a study of 30 novices, MS-DOS commands for creating, copying, renaming, and erasing files were contrasted with Macintosh direct-manipulation actions. After training and practice, average task times were 5.8 minutes versus 4.8 minutes, and average errors were 2.0 versus 0.8 (Margono and Shneiderman, 1987). Subjective preference also favored the direct-manipulation interface. In a study of a command-line versus a direct-manipulation database interface, 55 "computer naive but keyboard literate" users made more than twice as many errors with the command line format. No significant differences in time were found (Morgan et al., 1991). These users preferred the direct-manipulation interface overall, and rated it as more stimulating, easier, and more adequately powerful. Both reports caution about generalizing the results to more experienced users. A study with novices and experienced users was cosponsored by Microsoft and Zenith Data Systems (Temple,

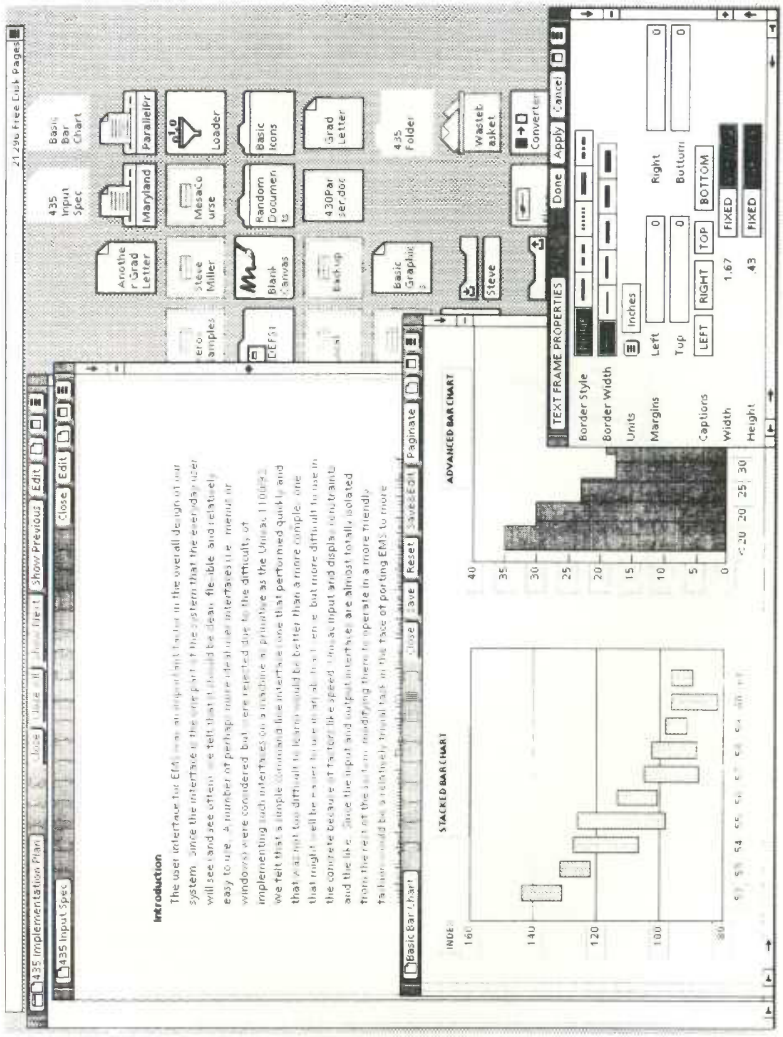


Figure 5.9

The Xerox Star 8010 with the ViewPoint system enables users to create documents with multiple fonts and graphics. This session shows the `Text Frame Properties` sheet over sample bar charts, with a document in the background and many desktop icons available for selection. (Prepared by Steve Miller, University of Maryland.)

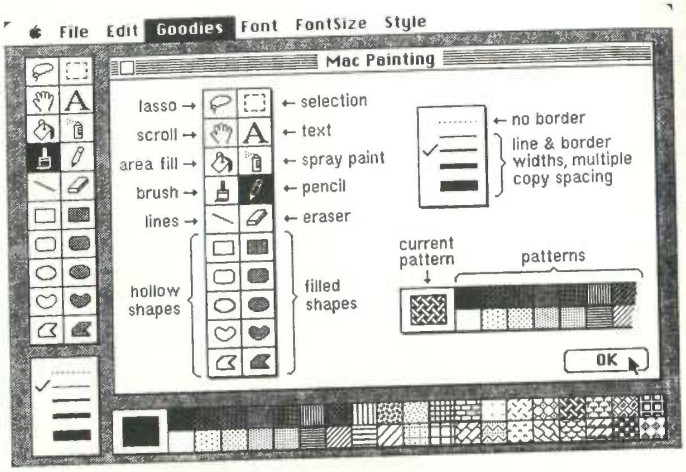


Figure 5.10

The original Apple Macintosh MacPaint program offers a command menu on the top, a menu of action icons on the left, a choice of line thicknesses on the lower left, and a palette of texture on the bottom. All actions can be accomplished with only the mouse. (Photo courtesy of Apple Computer, Inc.)

Barker, and Sloane, Inc., 1990). Although details about subjects, interfaces, and tasks were not reported, the results showed improved productivity and reduced fatigue for experienced users with a graphical user interface, as compared with a character-based user interface.

5.2.7 Further examples of direct manipulation

The trick in creating a direct-manipulation system is to come up with an appropriate representation or model of reality. Some designers may find it difficult to think about information problems in a visual form; with practice, however, they may find it more natural. With many applications, the jump to visual language may be difficult; later, however, users and designers can hardly imagine why anyone would want to use a complex syntactic notation to describe an essentially visual process.

Several designers applied direct manipulation to a stack of cards portraying a set of addresses, telephone numbers, events, and so on. Clicking on a card brings it to the front, and the stack of cards moves to preserve alphabetic ordering (see Figure 9.13). This simple card-deck metaphor, combined with other notions (Heckel, 1991) led to Bill Atkinson's innovative development of HyperCard stacks in 1987 (Section 11.4). Billed as a way to "create your own applications for gathering, organizing, presenting, search-