Exhibit 1015 – Part 4

```
X#    detailed information on a listed flight
S#    schedule information for the listed fare
R#    return flight information for this route
M     main menu
RF    return fares
```

Experienced users come to know the commands and do not need to read the prompt or the help screens. Intermittent users know the concepts and refer to the prompt to jog their memory and help them retain the syntax for future uses. Novice users do not benefit as much from the prompt and must take a training course or consult the online help.

The prompting approach emphasizes syntax and serves more frequent users. It is closer to but more compact than a standard numbered menu and preserves screen space for task-related information. WORDSTAR offers the novice and intermittent user help menus containing commands with one or two word descriptions (Figure 4.7). Frequent users can turn off the display of help menus, thereby gaining screen space for additional text.

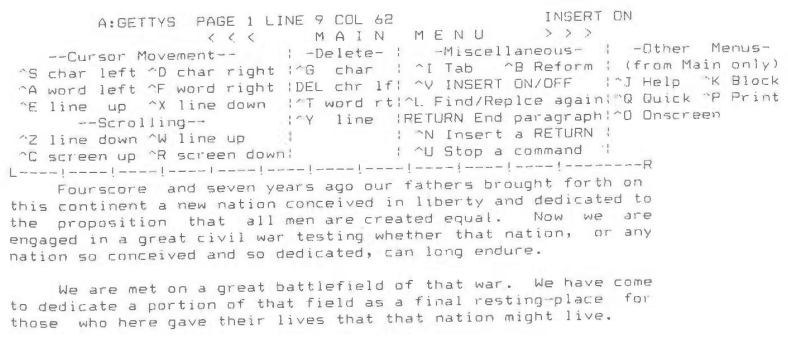


Figure 4.7: WordStar offers the user the option of bringing a help menu to a portion of the screen while the task is in process.

Several interactive systems on personal computers have another still more attractive form of prompts called command menus. Users are shown a list of descriptive words and make a selection by pressing the left and right arrow keys to move a light bar. When the desired command word is highlighted, the user presses the return key to carry out the command. Often, the command menu is a hierarchical structure that branches to a second- or third-level menu.

Even though arrow key movement is relatively slow and less preferred by frequent users, command menu items can be selected by single-letter keypresses. This strategy becomes a hierarchical command language, but it is identical to the typeahead (BLT) approach of menu selection. Novice users can use the arrow keys to highlight their choice or type single letter choices, but frequent users don't even look at the menus as they type 2, 3, 4, or longer sequences of single letters that come to be thought of as a command (see FinalWord commands in Figure 4.5).

The Lotus 1–2–3 (Figure 4.8) implementation is especially fast and elegant. As command words are selected, a brief description appears on the line below, providing further assistance for novice users without distracting experts from their concentration on the task. Experienced users appear to work as fast as touch typists, making three to six keystrokes per second.

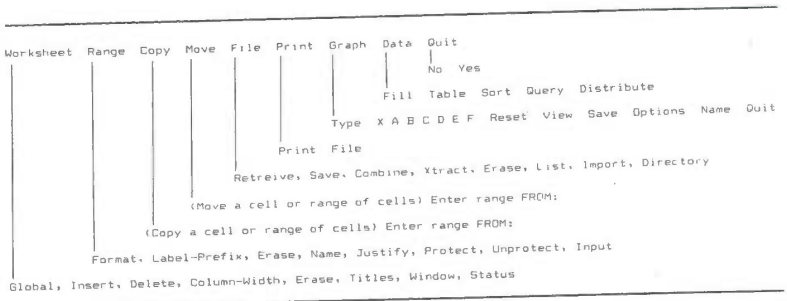


Figure 4.8: The first two levels of command menus from LOTUS 1–2–3 reveal the rich function available to users. At the third level, users may receive another menu or enter values.

Pop-up or pull-down menus that use mouse selection are another form of command menu. Frequent users can be extremely fast, and novices can take the time to read the choices before selecting a command. With a fast display, command menus blur the boundaries between what is thought of as commands and menus.

4.7 NATURAL LANGUAGE INTERACTION

Even before there were computers, people dreamed about creating machines that would accept natural language. It is a wonderful fantasy, and the success of word manipulation devices such as tape recorders, word processors, printing presses, and telephones may give encouragement to some people. A recurring hope is that computers will respond to commands issued by typing or speaking in natural language.

Natural language interaction (NLI) might be defined as the operation of computers by people using a familiar natural langauge (such as English) to give instructions. They do not have to learn a command syntax nor select from menus.

The problem with NLI is not only implementation on the computer, but also desirability for large numbers of users for a wide variety of tasks. People are different from computers, and human-human interaction is not necessarily an appropriate model for human operation of computers. Since computers can display information 1,000 times faster than people can enter commands, it seems advantageous to use the computer to display large amounts of information and allow novice and intermittent users simply to choose among the items. Selection helps guide the user by making clear what functions are available. For knowledgeable and frequent users, who are thoroughly aware of the available functions, a concise command language is usually preferred.

The syntactic/semantic model helps sort out the issues. NLI does not provide information about actions and objects in the task domain; users are usually presented with a simple prompt that invites a natural language query. But assume that the user is knowledgeable about the task domain; for example, the meaning of database objects and permissible actions.

Neither does NLI necessarily convey knowledge of the computer concepts; for example, tree-structuring of information, implications of a deletion, boolean operations, or query strategies. NLI does relieve the user of learning new syntactic rules, since it presumably will accept familiar English language requests. Therefore, NLI can be effective for the user who is knowledgeable about some task domain and computer concepts but who is an intermittent user who cannot retain the syntactic details.

NLI might apply to checkbook maintenance (Shneiderman, 1980) where the users recognize that there is an ascending sequence of integer numbered checks, and that each check has a single payee field, single amount, single date, and one or more signatures. Checks can be issued, voided, searched, and printed. In fact, following this suggestion, Ford (1981) created and tested an NLI system for this purpose. Subjects were paid to maintain their checkbook registers by computer using an APL-based program that was incrementally refined to account for unanticipated entries. The final system successfully handled 91 percent of users' requests, such as:

```
Pay to Safeway on 3/24/86 $29.77.
June 10 $33.00 to Cindy Lauper.
Show me all the checks paid to Ronald Reagan.
Which checks were written on October 29?
```

Users reported satisfaction with the system and were eager to use the system even when the several months of experimentation were completed. This can be seen as a success for NLI, but alternatives might be even more attractive. Showing a full screen of checkbook entries with a blank line for new entries might accomplish most tasks without any commands and minimal typing. Searches could be accomplished by entering partial information (for example, Ronald Reagan in the payee field) and then pressing a query key.

There have been numerous informal tests of NLI systems, but only a few have been experimental comparisons against some other design. A simulated query system was used to compare a subset of the structured SQL database facility to a natural language system (Small & Weldon, 1983). The SQL simulation resulted in faster performance on a

benchmark set of tasks. Similarly, a field trial with a real system, users, and queries pointed to the advantages of SQL over the natural language alternative (Jarke et al., 1985). Researchers seeking to demonstrate the advantage of NLI over command language and menu approaches for creating business graphics were surprised to find no significant differences for time, errors, or attitude (Hauptmann & Green, 1983).

Believers in NLI may claim that more research and system development is needed before excluding NLI, but improvements in menus, command languages, and direct manipulation seem equally likely. Supporters of NLI can point with some pride at the modest success of the commercially available INTELLECT system that has approximately 300 installations on large mainframe computers (Figure 4.9).

Business executives, salespeople, and others use INTELLECT to search databases on a regular basis. Several innovative implementation ideas help to make INTELLECT successful. The parser uses the contents of the database to parse queries; for example, the parser can determine that a query containing Cleveland refers to city locations because Cleveland is an instance in the database. Next, the system administrator can conveniently include guidance for handling domain-specific requests, by indicating fields related to who, what, where, when, how, etc. queries. Third, INTELLECT rephrases the user's query and displays a response such as: `PRINT THE CHECK NUMBERS WITH PAYEE = RONALD REAGAN`. This structured response serves as an educational aid, and users gravitate toward expressions that mimic the style. Eventually, as users become more frequent, they often use concise commandlike expressions that they believe will be successfully parsed. Even the promoters of INTELLECT recognize that novice users who are unfamiliar with the task domain will have a difficult time, and that the ideal user is a knowledgeable intermittent user.

There is some portion of the user spectrum that can benefit from NLI, but it may not be as large as promoters believe. Computer users usually seek predictable responses and are discouraged if they must frequently engage in clarification dialog. Since NLI has such varied forms, the users must constantly be aware of what the computer's response is to verify that the users' intentions were recognized. Finally, visually oriented interactions, embracing the notions of direct manipulation (see Chapter 5), make more effective use of the computer's capacity for rapid

```
PLEASE ENTER YOUR FIRST REQUEST

WHAT'S IN THE DATABASE?
FIELDS IN THE FILE OF EMPLOYEES

NAME                    JOB          SALARY          SEX
AGE                     FAMILY       CITY            STATE
EMPLOYEE NUMBER

WHO WORKS IN NEW YORK CITY
PRINT THE JOB AND NAME OF ALL EMPLOYEES WITH CITY = NEW YORK

PROFESSIONAL                        NAME
OCCUPATION

MACHINIST                           ANGELIN
PHYSICIAN                           ANGUS

WHAT IS THE NUMBER OF ANGUS
COUNT THE EMPLOYEES WITH NAME = ANGUS

ANSWER:  1

WHAT IS THE EMPLOYEE NUMBER OF ALL EMPLOYEES WITH NAME = ANGUS
EMPLOYEE
NUMBER

0034

WHICH WOMEN WORK IN NEW YORK CITY
PRINT THE JOB OF ALL EMPLOYEES WITH SEX = FEMALE & CITY = NEW YORK

PROFESSIONAL
OCCUPATION

MACHINIST
ADMINISTRATOR
ELECTRICIAN

FEMALE EMPLOYEES IN NEW YORK CITY
PRINT THE NAME OF ALL EMPLOYEES WITH SEX = FEMALE & CITY = NEW YORK

NAME

ANGELIN
CANTEBURY
EDWARDS

HOW MANY NEW YORK CITY MEN ARE OVER 30 YEARS OLD
COUNT THE EMPLOYEES WITH CITY = NEW YORK & SEX = MALE & AGE>30

ANSWER:  9

PRINT THEM
PRINT THE NAME AND AGE OF ALL EMPLOYEES WITH CITY = NEW YORK &
     SEX = MALE & AGE>30

                YEARS
NAME             OF
                AGE

ANGUS            43
ELEY             69
HILTON           44
```

Figure 4.9: Demonstration session with Intellect from Artificial Intelligence Corp., Cambridge, MA. User input is underscored. Intellect rephrases user input into a structured query language, which users often mimic as they become more frequent users.

display. In short, pointing and selecting in context is often more attractive than typing or even speaking an English sentence.

It is surprising that designers of expert systems have attempted to embed NLI. Expert systems already tax the user with complexity, lack

of visibility of the underlying processes, and confusion about what functions the system can and cannot handle. A precise, concise notation or selection in context from a list of alternatives seems far more suitable in providing users with predictable and comprehensible behavior (Hayes-Roth, 1984) (Figure 4.10).

An innovative blend of NLI and menus was developed under the name NLMENU (Tennant et al., 1983) and is now distributed by Texas

SAMPLE SESSION FROM AN EXPERT SYSTEM FOR OIL DRILLING ADVISOR

What is the name of WELL–159?
AGF7–93E

What is the profile of AGF7–93E?
DEVIATED

Please enter information about FORMATION–1:

upper-limit meters	lower-limit meters	main-rock-type	homogeneous/ interbedded
747	806	SHALE	HOMOGENEOUS

Please enter information on PROBLEM–1:

problem-type	prior-action	total-depth	casing-shoe depth
STICKING	REAMING	1111 METERS	747 METERS

Please enter the composition of the drill-string starting from the bit (type ? for assistance):
BIT 9″5/8 STAPB″5/8 SHORTDC7″3/4STAB9″5/8...NDP5

What was the drilling method employed when the problem occurred:
ROTARY

What is the depth of the freepoint?
UNKNOWN

Figure 4.10: This extract demonstrates one designers attempt at an expert system dialog. User input is shown in all upper case letters. Users must type in values even when selection from a menu would be more meaningful, rapid, and error-free. Furthermore, there does not appear to be any way to go back and change values, view values, or reuse values from previous sessions (F. Hayes-Roth, The knowledge-based expert system: A tutorial, *IEEE Computer 17*, 9 (Sept. 1984), 11–28. © 1984 IEEE)

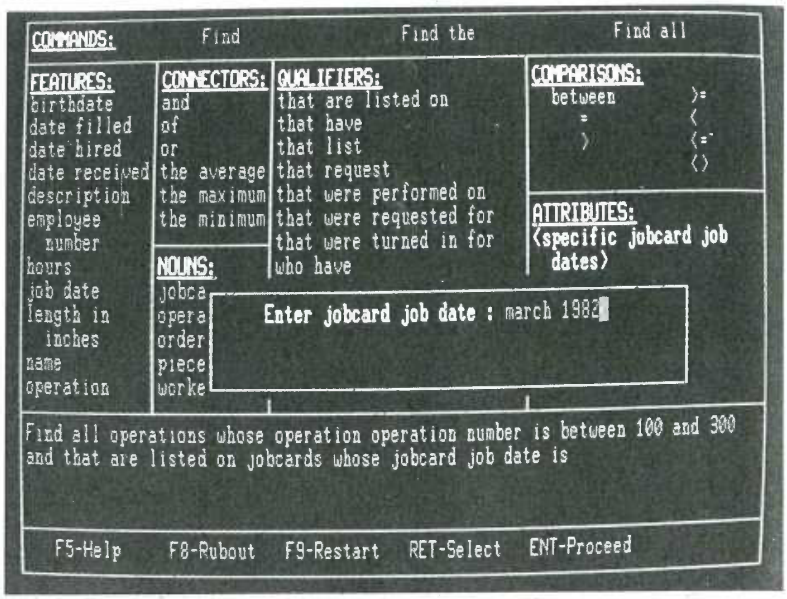


Figure 4.11: The NaturalLink (TM) allows users to specify natural langauge English queries against a database by choosing phrases from a set of menus. The content of the menus is determined by the contents of the database. (Courtesy of Texas Instruments, Dallas, TX)

Instruments under the name NaturalLink (Figure 4.11). Natural language phrases are shown as a series of menus. As phrases (for example, FIND / COLOR / AND / NAME OF PARTS / WHOSE COLOR IS) are chosen by a pointing strategy, a query is formed in a command window. Users receive information from the menus, obviating the need for a query. For example, if the parts and suppliers database contains only red, green, and blue parts, only these choices appear in the window containing the PART COLOR menu. Users can see the full range of possible queries and thereby avoid the frustration of probing the boundaries of system functionality. With this strategy, typing is eliminated and the user is guaranteed a semantically and syntactically correct query.

A notable and widespread success of NLI techniques is in the variety of adventure games (Figure 4.12). Users may indicate directions of

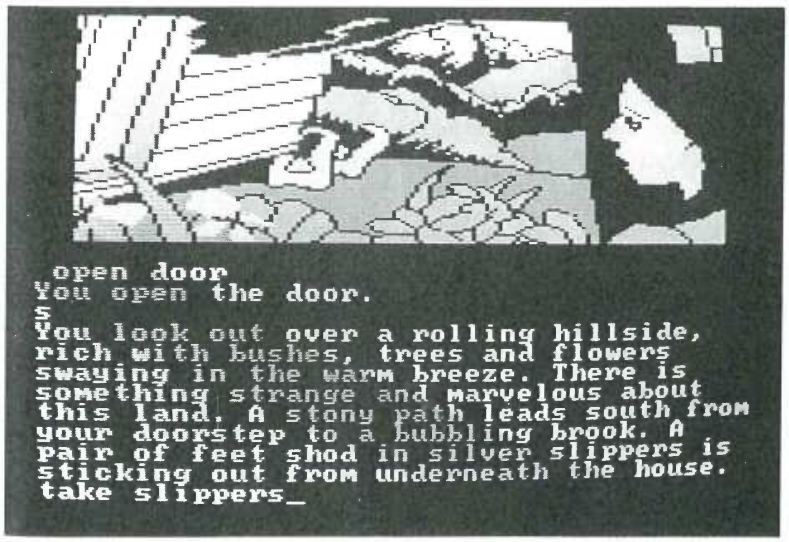


Figure 4.12: This adventure game is modelled on the Wizard of Oz story. The user types phrases such as "open the door" or "take slippers" or abbreviations such as "s" to move south. More complex phrases such as "put the hat on the scarecrow" are possible. (Courtesy of Spinnaker Software, Cambridge, MA)

movement or type commands, such as TAKE ALL OF THE KEYS, OPEN THE GATE, or DROP THE CAGE AND PICK UP THE SWORD. Part of the attraction of NLI in this situation is that the system is unpredictable and some exploration is necessary to discover the proper incantation.

So much research has been invested in NLI systems that undoubtedly some successes will emerge, but widespread use may not develop because the alternatives may be more appealing. More rapid progress can be made if carefully controlled experimental tests are used to discover the designs, users, and tasks for which NLI is most beneficial.

4.8 PRACTITIONER'S SUMMARY

Command languages can be attractive when frequent use of a system is anticipated, users are knowledgeable about the task domain and computer concepts, screen space is at a premium, response time and display rates are slow, and numerous functions that can be combined in many ways are supported. Users will have to learn the semantics and syntax, but they can initiate rather than respond, rapidly specifying actions involving several objects and options. Finally, complex sequences of commands can be easily specified and stored for future use as a macro.

Designers should begin with a careful task analysis to determine what functions should be provided. Hierarchical strategies and congruent structures facilitate learning, problem solving, and human retention over time. Laying out the full set of commands on a single sheet of paper helps show the structure to the designer and to the learner. Meaningful specific names aid learning and retention. Compact abbreviations constructed according to a consistent rule facilitate retention and rapid performance for frequent users.

Innovative strategies, such as command menus, can be effective if rapid response to screen actions can be provided. Natural language interaction can be implemented, but its advantage for widespread application is yet to be demonstrated.

4.9 RESEARCHER'S AGENDA

Designers could be helped by development of strategies for task analysis, taxonomies of command language designs, and criteria for using commands or other techniques. The benefits of structuring such concepts as hierarchicalness, congruence, consistency, and mnemonicity have been demonstrated in specific cases, but replication in varied situations is important. Experimental testing should lead to a more comprehensive cognitive model of command language learning and use. (See Table 4.3.)

A command language system generator would be a useful tool for research and development of new command languages. The designer

COMMAND LANGUAGE GUIDELINES
• Create explicit model of objects and actions
• Choose meaningful, specific, distinctive names
• Try for hierarichical structure
• Provide consistent structure (hierarchy, argument order, action-object)
• Support consistent abbreviation rules (prefer truncation to one letter)
• Offer frequent users the capability to create macros
• Consider command menus on high-speed displays
• Limit number of commands and ways of accomplishing a task

Table 4.3: High-level design guidelines based on empirical studies and practical experience.

could provide a formal specification of the command language, and the system would generate an interpreter. With experience in using such a tool, design analyzers might be built to critique the design, detect ambiguity, check for consistency, verify completeness, predict error rates, or suggest improvements. Even a simple but thorough checklist for command language designers would be a useful contribution.

Novel input devices and high-speed, high-resolution displays offer new opportunities, such as command and pop-up menus, for breaking free from the traditional syntax of command languages. Natural language interaction still holds promise in certain applications, and empirical tests offer a good chance to identify rapidly the appropriate niches and design strategies.

REFERENCES

Barnard, P. J., and Hammond, N. V., Cognitive contexts and interactive communication, IBM Hursley (U.K.) Human Factors Laboratory Report HF070, (December 1982), 18 pages.

Barnard, P., Hammond, N., MacLean, A., and Morton, J., Learning and remembering interactive commands, *Proc. Conference on Human Factors in Computer Systems*, Available from ACM DC., (1982), 2–7.

Barnard, P. J., Hammond, N. V., Morton, J., Long, J. B., and Clark, I. A., Consistency and compatibility in human-computer dialogue, *International Journal of Man-Machine Studies 15*, (1981), 87–134.

Benbasat, Izak, and Wand, Yair, Command abbreviation behavior in human-computer interaction, *Communications of the ACM 27*, 4, (April 1984), 376–383.

Black, J., and Moran, T., Learning and remembering command names, *Proc. Conference on Human Factors in Computer Systems*, Available from ACM DC., (1982), 8–11.

Carroll, John M., Learning, using and designing command paradigms, *Human Learning 1*, 1, (1982), 31–62.

Carroll, J. M., and Thomas, J., Metaphor and the cognitive representation of computing systems, *IEEE Transactions on Systems, Man, and Cybernetics, SMC–12*, 2, (March/April 1982), 107–115.

Ehrenreich, S. L., and Porcu, Theodora, Abbreivations for automated systems: Teaching operators and rules, In Badre, Al, and Shneiderman, Ben, (Editors), *Directions in Human-Computer Interaction*, Ablex Publishers, Norwood, NJ, (1982), 111–136.

Ford, W. Randolph, Natural Language Processing by Computer —A New Approach, Ph. D. Dissertation, The Johns Hopkins University Department of Psychology, Baltimore, MD, (1981), 88 pages.

Green, T. R. G., and Payne, S. J., Organization and learnability in computer languages, *International Journal of Man-Machine Studies 21*, (1984), 7–18.

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*, Available from ACM Order Dept., P. O. Box 64145, Baltimore, MD 21264, (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, (January 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*, Available from ACM Order Dept., P. O. Box 64145, Baltimore, MD 21264, (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.

Norman, Donald, The trouble with UNIX, *Datamation 27*, (1981), 556–563.

Roberts, Terry, Evaluation of computer text editors, Ph. D. dissertation, Stanford University. Available from University Microfilms, Ann Arbor, MI, order number AAD 80–11699, (1980).

Rosenberg, Jarrett, Evaluating the suggestiveness of command names, *Behaviour and Information Technology 1*, (19822), 371–400.

Rosson, Mary Beth, Patterns of experience in text editing, *Proc. CHI '83 Conference on Human Factors in Computing Systems*, Available from ACM Order Dept., P. O. Box 64145, Baltimore, MD 21264, (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 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 Publishers, 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 and Co., Boston, MA, (1980), 320 pages.

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*, available from ACM Order Dept., P. O. Box 64145, Baltimore, MD 21264 (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.

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 are filled with the positive feelings of:

- mastery of the system
- competence in performance of their task
- ease in learning the system originally and in assimilating advanced features
- confidence in their capacity to retain mastery over time
- enjoyment in using the system
- eagerness to show it 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*.

5.2 EXAMPLES OF DIRECT MANIPULATION SYSTEMS

No single system has all the admirable attributes or design features—that may be impossible; but each of the following examples has enough 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 actions such as braking or steering have become common knowledge in our culture. To turn left, the driver simply rotates the steering wheel to the left. The response is immmediate and the scene changes, providing feedback to

refine the turn. Imagine trying to turn by issuing a command LEFT 30 DEGREES and then having to issue another command to see the new scene; but this is the level of operation of many office automation tools of today.

5.2.1 Display editors

Users of full-page display editors are great advocates of their systems as compared with line-oriented text editors (Figure 5.1). 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, personal computer word processors such as WORDSTAR 2000,

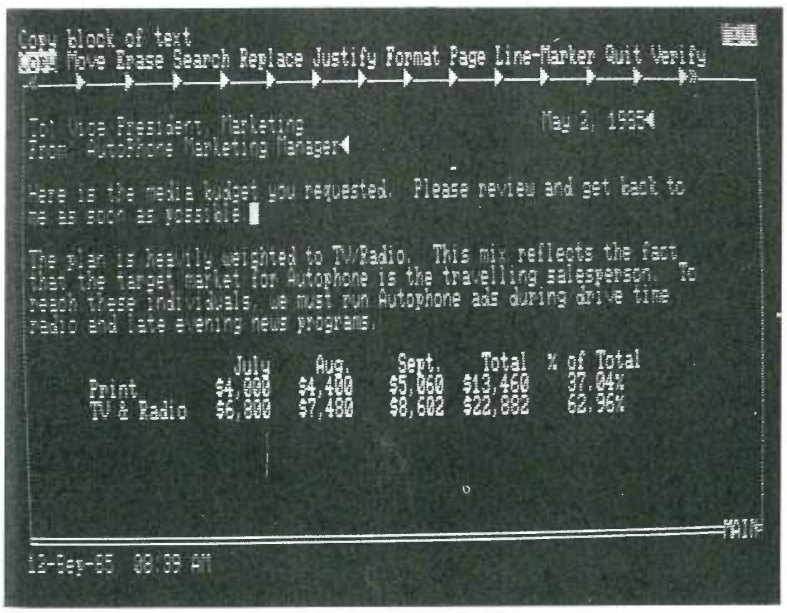


Figure 5.1: Word processor from Symphony. (Courtesy of © Lotus Development Corporation 1985. Used with permission.)

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

The advantages of display editors include:

- *display of a full 24 to 66 lines of text.*
  This gives the reader a clearer sense of context for each sentence while permitting simpler reading and scanning of the document. By contrast, the one-line-at-a-time view offered by some line editors is like seeing the world through a narrow cardboard tube.
- *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. This style has come to be known as WYSIWYG (what you see is what you get). The annoyance and delay of debugging the format commands are eliminated because the errors are immediately apparent.
- *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 apply action.
- *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 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, hard to recall, and a source of frustrating errors.

- *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, or LOCATE. These buttons act as a permanent menu selection display to remind the operator of the features and to avoid the need to memorize a complex command language syntax. On some editors, only ten or fifteen 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.

- *immediate display of the results of an action.*
  When a button is pressed to move the cursor or center text, the results are shown immediately on the screen. Deletions are immediately apparent since 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 button press. This is in contrast to line editors in which print or display commands must be issued to see the results of changes.

- *rapid action 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 commands can be shown almost instantaneously. Rapid action also reduces the need for additional commands and thereby simplifies design and learning. Line editors operating at thirty characters per second with three to eight

second response times seem bogged down in the mud. Speeding up line editors adds to their attractiveness, but they would still lack such features as direct overtyping, deletion, and insertion.

- *easily reversible actions*. When entering text, an incorrect keystroke is repaired by merely backspacing and overstriking. Simple changes can be made 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. Carroll (1982a) has shown that congruent pairs of operations are easy to learn (see 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 fear of 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.

5.2.2 VISICALC and its descendents

The first electronic spreadsheet, VISICALC, was the product of a Harvard MBA student who was frustrated when trying to carry out the multiple calculations in a graduate business course. He built an "instantly calculating electronic worksheet" (as the user manual describes 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 impact of changes on profits can be immediately seen.

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

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

Competitors to VISICALC emerged quickly and 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). It offers integration with graphics and database

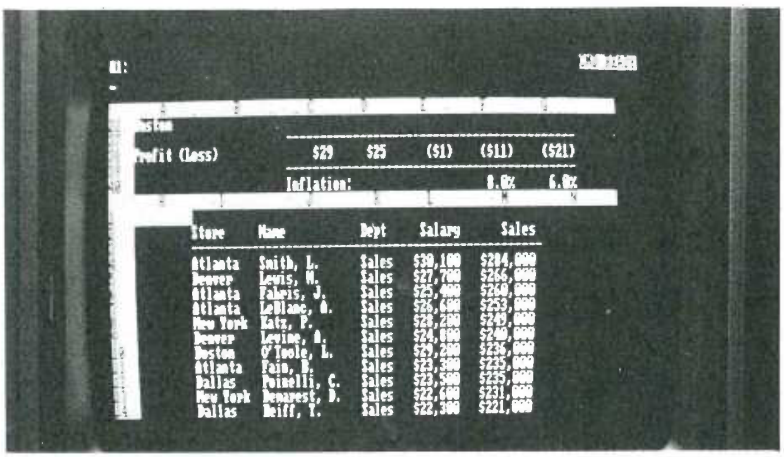


Figure 5.2a: Spreadsheet showing split window in the LOTUS 1–2–3 package. (Courtesy of © Lotus Development Corporation 1985. Used with permission.)

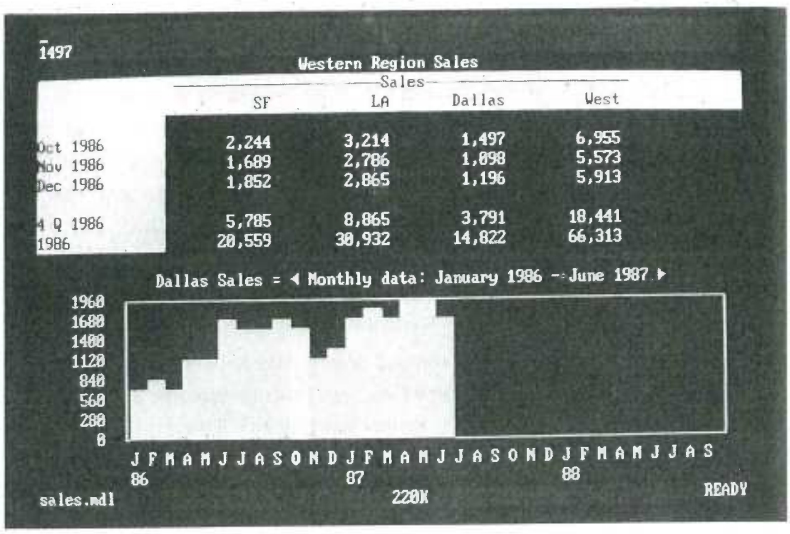


Figure 5.2b: The worksheet in Javelin has row and column labels created by the user. The top half of the screen shows sales data in a tabular format; the bottom half shows a bar chart. Changes to the data are immediately reflected in the bar chart and vice versa. (Used with permission of Javelin Software Corporation, Cambridge, MA)

features. The actions are easily invoked with command menus. Advanced systems such as Javelin (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; Herot, 1984), attribute the basic idea to Nicholas Negroponte of MIT. In one scenario, the user is seated before a color graphics display of the world and can zoom in on the Pacific Ocean to see markers for military ship convoys (Figure 5.3). By moving a

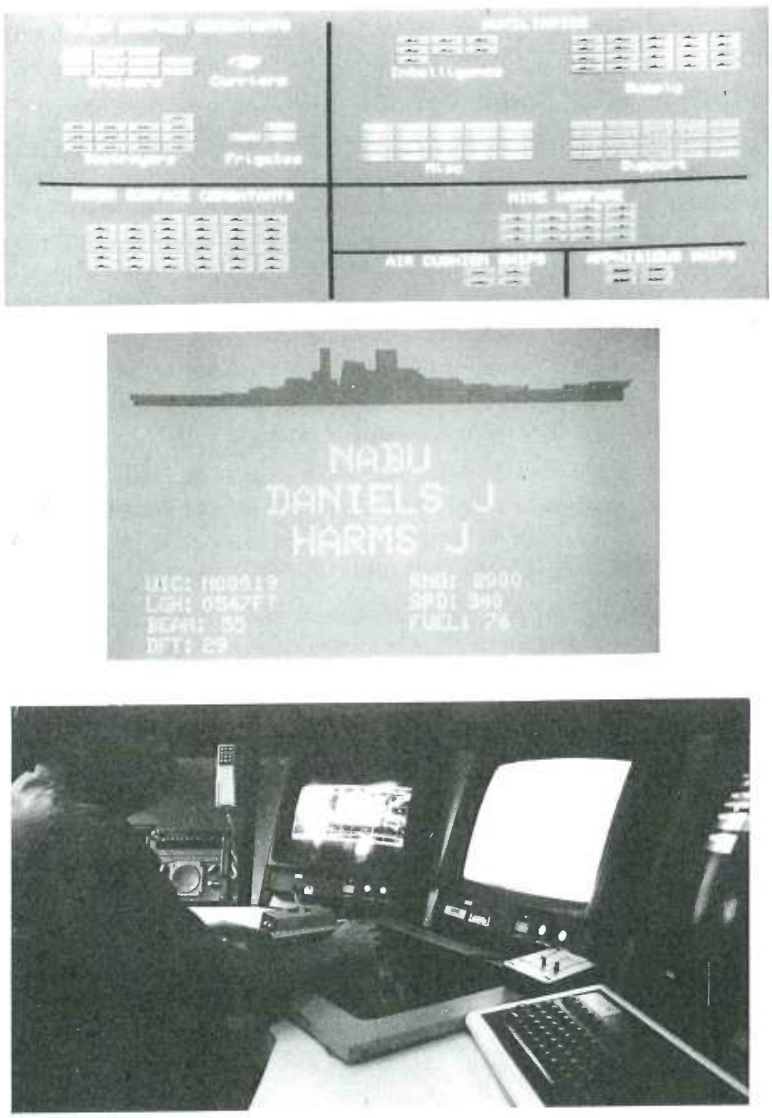


Figure 5.3: The Spatial Data Management System has three displays to show multiple levels of detail or related information. The user moves a joystick to traverse information spaces or zoom in on a map and see more details about ship convoys. (Courtesy of the Computer Corporation of America, Cambridge, MA)

joystick, the screen becomes filled with silhouettes of individual ships that can be zoomed in on to display detailed data or 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, or schedules are shown on a screen. By moving the joystick and zooming in on objects of interest, the user is taken through complex "information spaces" or "I-spaces" to locate the item of interest. A building floor plan showing departments might be shown, and when a department is chosen, individual offices become visible. On moving the cursor into a room, details about its occupant appear on the screen. If you choose the wrong room, merely back out and try another. The lost effort is minimal and there is no stigma of error.

The Filevision software for the Macintosh enables designers to perform database retrievals visually. For example, if a map of the United States is shown on the screen, the user can retrieve facts about each state by pointing and clicking.

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 is in the world of video games. 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 and had to be hit back by the movable white rectangle. The user developed skill involving 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 thirty seconds is all the training needed to become a competent

novice, but many hours of practice are required to become a skilled expert.

Contemporary games, such as Missile Command, Donkey Kong, Pac Man, Tempest, TRON, Centipede, or Space Invaders, are much more sophisticated in their rules, color graphics, and sound effects. 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 (Figure 5.4). The strong attraction of these games is 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 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

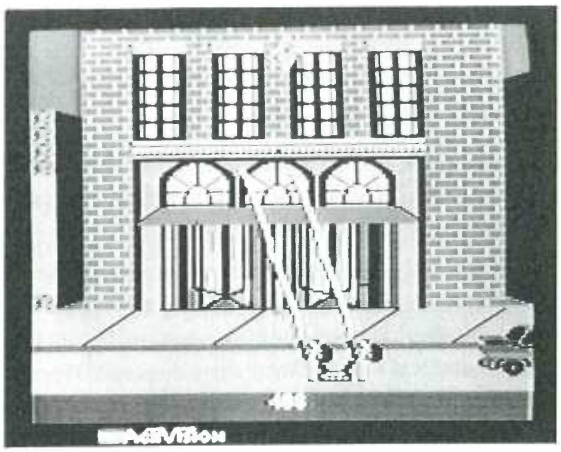


Figure 5.4: Videogames employ direct manipulation principles to create a world of action and fantasy. (*Ghostbusters: The Computer Game* art work courtesy of Activision, Inc.)

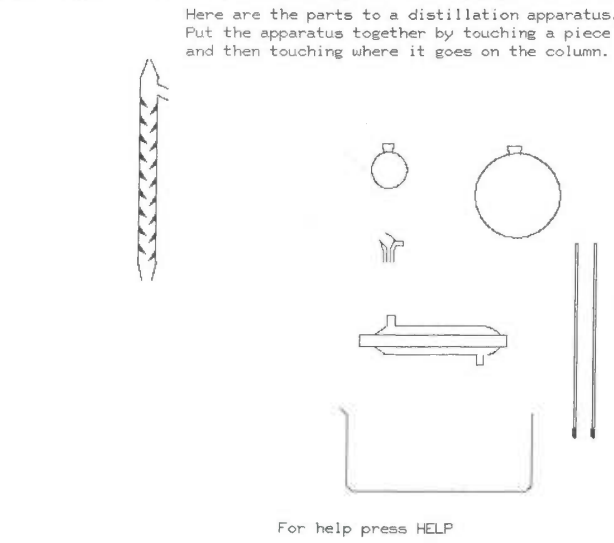
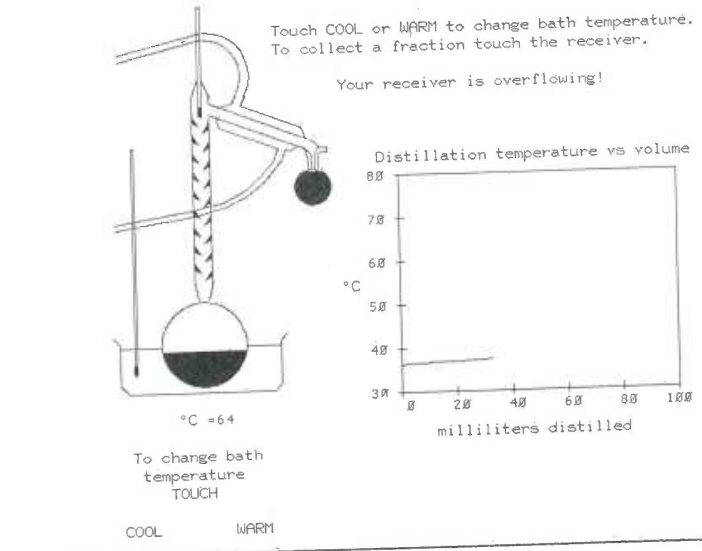


Figure 5.5: Computer-based instruction can become more appealing with direct manipulation, instead of drill and practice. This CDC PLATO lesson, written by Stan Smith of the Department of Chemistry at the University of Illinois, allows students to construct a distillation aparatus by proper finger actions on a touch-sensitive screen. Once the student has assembled the apparatus and begun

unnecessary because the results of actions are obvious and can be easily reversed. 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 ten highest scorers get to store their initials in the game for regular display. This is one form of positive reinforcement that encourages mastery. Malone (1981) and our studies with elementary school children have shown that continuous display of scores is extremely valuable. Machine-generated value judgments, such as "Very Good" or "You're doing great!" are not as effective, since the same score means different things to different people.



the experiment, the display shows an animation of the process with the graph of distillation temperature versus volume. The second figure shows that the student experimenter has gotten into trouble. (Courtesy of Stan Smith, University of Illinois.)

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. Stan Smith's chemistry lessons on the PLATO system often enabled college students to conduct lab experiments by touching beakers, pipettes, or burners in order to assemble and operate equipment (Figure 5.5). A Navy training simulator shows gauges, dials, and knobs that can be directly manipulated to gain experience with boilers, valves, and so on (Hollan, Hutchins, & Weitzman, 1984). Several versions of the Music

Construction Set offer the users the possibility of constructing musical scores by selecting and moving notes onto a staff.

Carroll (1982b) 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; but in nongame designs, however, predictable system behavior is preferred. Game players are engaged in competition with the system, whereas applications systems users apparently prefer strong internal locus of control that gives them the sense of being in charge.

5.2.5 Computer-aided design/manufacturing

Many computer-aided design systems for automobiles, electronic circuitry, architecture, aircraft, or newspaper layout use principles of direct manipulation. The operator may see a circuit schematic on the screen and with lightpen touches can 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, 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 record promising approaches until a better one is found. A playful application is Bill Budge's Pinball Construction Set that allows users to select bumpers, flippers, or flashers, drag them onto a pinball table, and then shoot the ball to see how the game plays (Figure 5.6).

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, but others have moved to using cursor action and graphics-oriented commands.

Another related direction is the world of computer-aided manufacturing and process control. Honeywell's process control system provides an oil

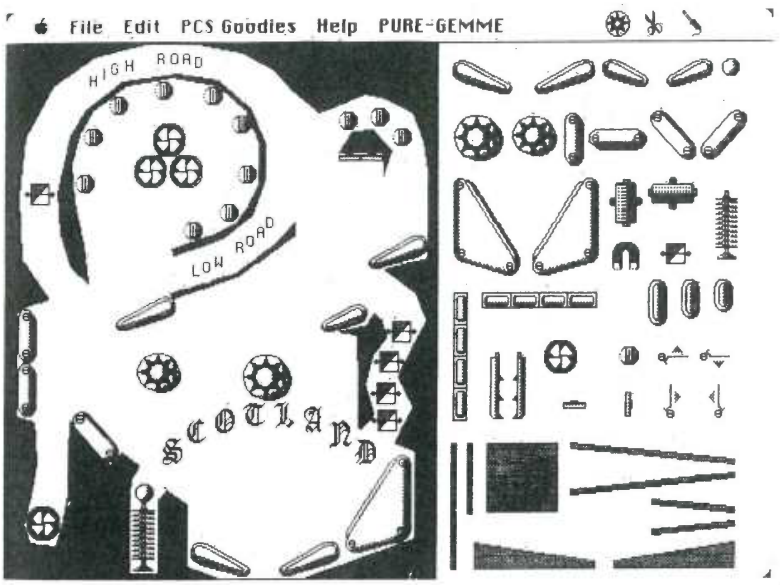


Figure 5.6: Pinball Construction Set (Electronic Arts) allows users to point at pinball components on the right and move them to the board on the left. When the user is satisfied, he or she can shoot the ball and work the flippers. (Courtesy of Electronic Arts, San Mateo, CA)

refinery, paper mill, or power utility plant manager 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; and with a second press, the operator moves 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 only be recalled 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 Further examples

The term *direct manipulation* is most 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 whenever necessary.

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 5.7). The user moves a cursor through the columns of the relational table and enters examples of what the result should look like. There are just a few single letter keywords to supplement the direct manipulation style. Of course, complex booleans or mathematical operations require knowledge of syntactic forms. Still, the basic ideas and facilities in this language can be learned within a half hour by many nonprogrammers.

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	WILMINGTON	3216

Figure 5.7: The Query-by-Example facility shows users a relational table skeleton and enables users to fill in literals (such as NY or 1200) and specify fields to be printed (P.). Users can also specify variables to link between relations. In this example, the query produces the NAMEs of ski resorts in NY state that have a vertical drop of more than 1200 feet.

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 elegantly integrates database search with word processing, electronic mail, business graphics, and menu creation.

Designers of advanced office automation systems have made use of direct manipulation principles. The Xerox Star (Smith et al., 1982) offers sophisticated text formatting options, graphics, multiple fonts, and a high resolution, cursor-based user interface (Figure 5.8). Users can move a document icon to a printer icon to generate a hardcopy 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 5.9). The hardware and software designs supported rapid graphical interaction for pull-down menus, window manipulation, graphics and text editing, and dragging of icons. Imitations of the Macintosh appeared soon afterward for popular personal computers, such as the IBM PC (Figure 5.10 and Color Plate 2).

Researchers at IBM's Yorktown Heights Labs (Schild et al., 1980) propose a future office system, called PICTUREWORLD, in which graphic icons represent file cabinets, mailboxes, notebooks, phone messages, and so on. The user could compose a memo with a display editor and then indicate distribution and filing operations by selecting from the menu of icons. Yedwab et al. (1981) describe a generalized office system with a visual representation under the term *automated desk*.

5.3 EXPLANATIONS OF DIRECT MANIPULATION

Several authors have attempted to describe the component principles of direct manipulation. Don Hatfield (1981), who is applying many of these principles in an advanced office automation system, describes the general

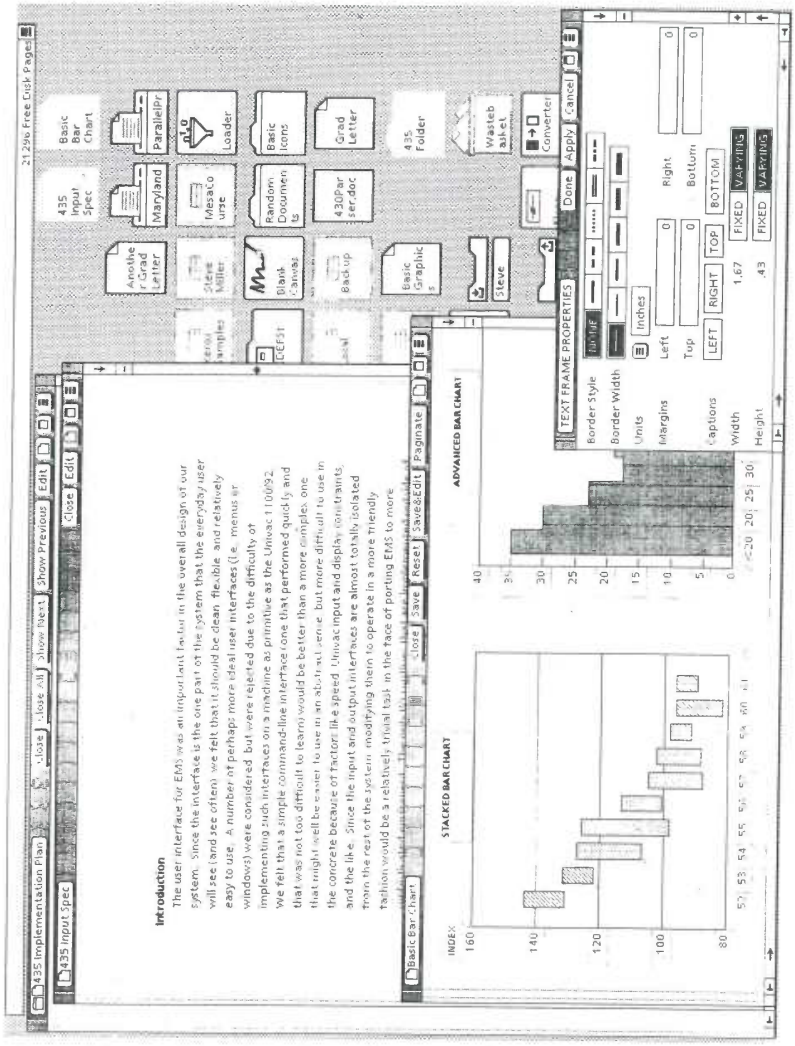


Figure 5.8: The Xerox Star 8010 with the ViewPoint system enables users to create documents with multiple fonts and graphics. This session shows the Text Frame Properties sheet over sample bar charts, with a document in the background and many desktop icons available for selection. (Prepared by Steve Miller, University of Maryland)

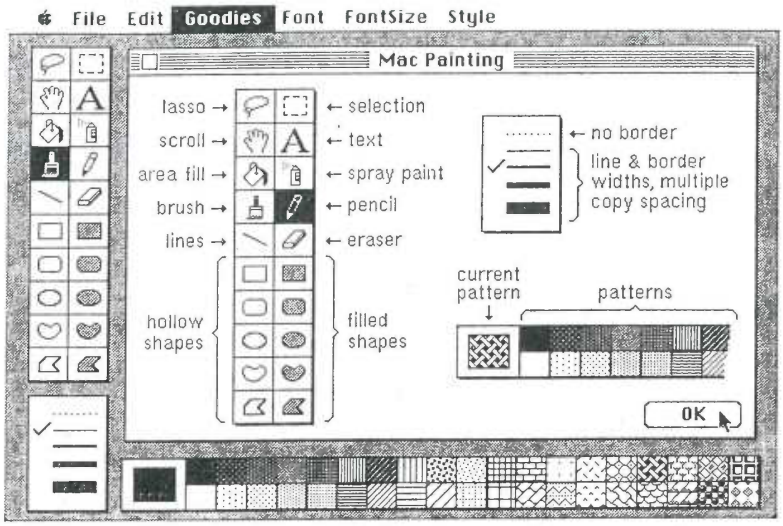


Figure 5.9: The Apple Macintosh Macpaint program offers a command menu on the top, a menu of action icons on the left, a choice of line thicknesses on the lower left, and a palette of texture on the bottom. All actions can be accomplished with only the mouse. (Photo courtesy of Apple Computer, Inc.)

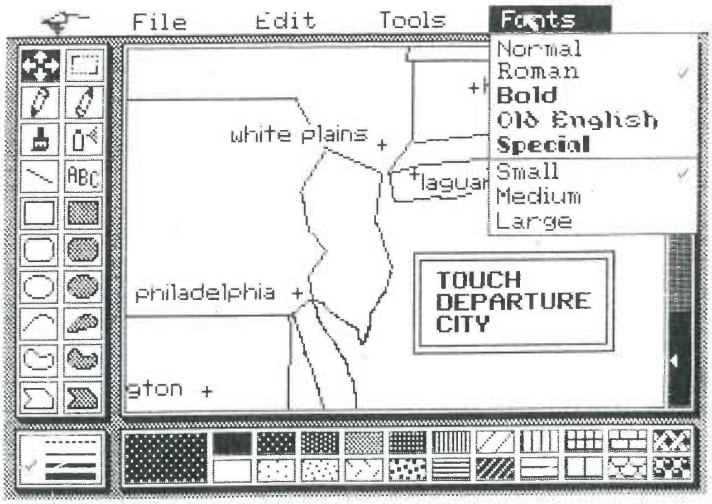


Figure 5.10: PCPAINT, a descendent of Macpaint, runs on IBM PCs. It offers similar features, but adds color. (Courtesy of Mouse Systems Corporation, Santa Clara, CA)

approach as "What you see is what you get." Harold Thimbleby (1982) expands in this direction by suggesting that "What you see is what you have got." He suggests that the display should indicate a more complete image of what the current status is, what errors have occurred, and what actions are appropriate.

Another 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." MacDonald (1982) emphasizes the term *visual programming* as a solution to the shortage of application programmers. He feels that visual programming speeds system construction and allows end users to generate or modify applications systems to suit their needs.

Heckel (1984) 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."

Each of these writers supports the growing recognition that a new form of interactive systems is emerging. Much credit also goes to the individual designers who have created systems that exemplify aspects of direct manipulation.

Problem-solving and learning research. Another perspective on direct manipulation comes from the problem-solving psychology literature. 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 is in harmony with Maria Montessori's teaching methods for children (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 subjects given an isomorphic problem with a temporal representation. 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 x 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 of algebra word problems are often encouraged to draw a picture to represent the 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 easily determine what has happened, can quickly spot and repair errors, and gain satisfaction from creative production of drawings. These features are all characteristic of a direct manipulation environment.

5.3.1 Problems with direct manipulation

In professional programming, use of high-level flowcharts, record structures, and database schema diagrams can be helpful for some tasks,

but there is an additional effort in absorbing the rules of the representation. Visual representations can be helpful when there are multiple relationships among objects and when the representation is more compact than the detailed object. Selectively screening out detail and presenting an abstraction suitable for a given task can facilitate performance.

Use of spatial or visual representations is not necessarily an improvement. In one study, subjects given a detailed flowchart did no better in comprehension, debugging, or modification than subjects given the code only (Shneiderman et al., 1977). In another study, subjects given a graphic representation of control flow or data structure did no better than subjects given textual descriptions of control flow or data structure in a program comprehension task (Shneiderman, 1982). On the other hand, subjects given the data structure documentation consistently did better than subjects given the control flow documentation. This study suggests that the content of graphic representations is a critical determinant of utility. The wrong information, or a too cluttered presentation, can lead to greater confusion.

A second problem is that users must learn the meaning of components of the graphic 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 their meaning 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 graphic representation may be misleading. The user may rapidly grasp the analogical representation but then make incorrect conclusions about permissible actions. Ample testing must be carried out to refine the displayed objects and actions and minimize negative side effects.

A fourth problem is that graphic representations may take excessive screen display space. For experienced users, a tabular textual display of fifty document names may be more appealing than only ten document graphic icons with the names abbreviated to fit the icon size.

A fifth problem is that for experienced typists, moving a mouse or raising a finger to point may sometimes be slower than typing. This is

true especially 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 screen 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 & 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 syntactic/semantic model

The attraction of systems that use principles 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:

- continuous representation of the objects and actions of interest
- physical actions or labeled button presses instead of complex syntax
- rapid incremental reversible operations whose impact 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 not, they can simply change the direction of their activity
- users experience less anxiety because the system is comprehensible and because actions are so easily reversible
- users gain confidence and mastery because they are the initiators of action, they feel in control, and the system responses are predictable.

The success of direct manipulation is understandable in the context of the syntactic/semantic 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 immediately visible. 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 5.11).

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 growth: sensorimotor (from birth to approximately two years), preoperational (two

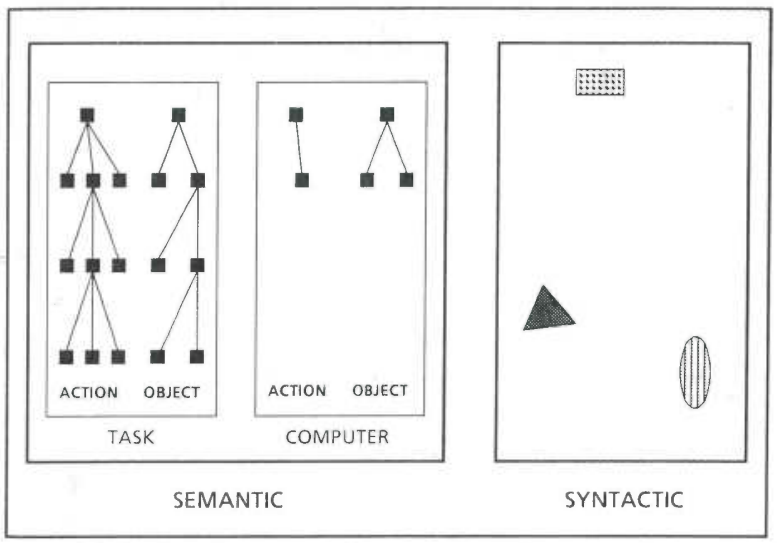


Figure 5.11: For direct manipulation systems, there may be substantial task-domain semantic knowledge. However, users must acquire only a modest amount of computer-related semantic knowledge and syntacctc knowledge.

to seven years), concrete operational (seven to eleven 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 around age eleven, children enter the formal operations stage of 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 is an attempt to bring activity to the concrete operational stage, thus making some tasks easier for children and adults.

It is easy to envision 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 approach. 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 in the early stages of learning to use the system; more specific abstract models may be more useful during regular use.

The syntactic/semantic model provides a simple model of human cognitive activity. It must be refined and extended to enhance its explanatory and predictive power. Empirical tests and careful measurements of human performance with a variety of systems are needed to develop and validate an improved model. Cognitive models of user behavior and mental models or system images of computer supplied functions are rapidly expanding areas of research in computer science and psychology.

5.4 POTENTIAL APPLICATIONS 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, but with practice it can become more natural. With many applications, the jump to visual language may be difficult, but later users and designers can hardly imagine why anyone would want to use a complex syntactic notation to describe an essentially visual process.

One application that we explored was a personal address list program that displays a Rolodex-like device (Figure 5.12). The most recently retrieved address card appears on the screen and the top line of the next two appear behind, followed by the image of a pack of remaining cards. As the joystick is pushed forward the Rolodex appears to rotate and successive cards appear in front. As the joystick is pushed further, the cards pass by more quickly; as the joystick is reversed, the direction of