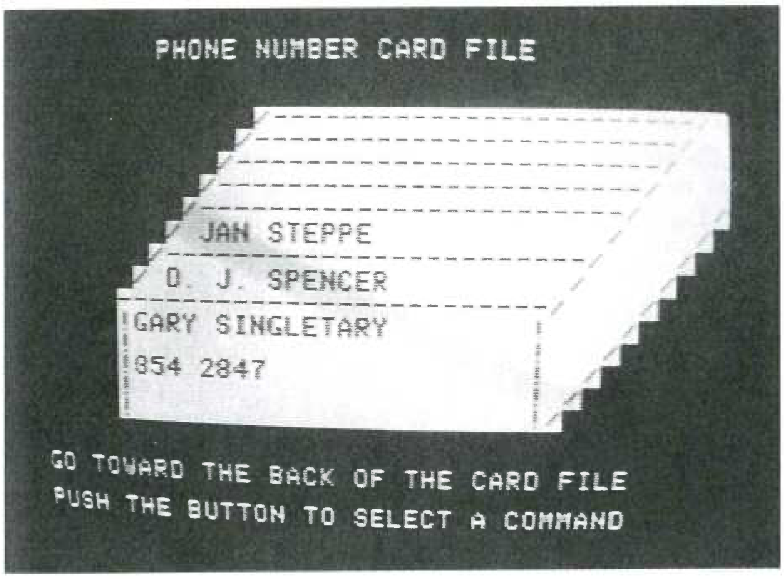
Exhibit 1015 – Part 5



PHONE NUMBER CARD FILE

JAN STEPPE

D. J. SPENCER

GARY SINGLETARY

854 2847

GO TOWARD THE BACK OF THE CARD FILE

PUSH THE BUTTON TO SELECT A COMMAND

Figure 5.12: This electronic Rolodex or phone number card file gives users rapid control over the card motion by a forward or backward joystick press. Commands are displayed by moving the joystick left or right. The lively motion of the cards appeals to many users.

movement reverses. To change an entry, the user need only move the cursor over the field to be updated and type the correction. To delete an entry, the user just blanks out the fields. Blank cards might be left at the top of the file, but when the fields are filled in, proper alphabetic placement is provided. To find all entries with a specific zip code, the user types the zip code in the proper field and enters a question mark.

Checkbook maintenance and searching might be done in a similar fashion. Display a checkbook register with labeled columns for check number, date, payee, and amount. The joystick might be used to scan earlier entries, changes could be made in place, new entries would be made at the first blank line, and a check mark could be made to indicate verification against a monthly report or bank statement. Searches for a particular payee could be made by filling in a blank payee field and then typing a question mark.

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

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

Why not do inventory by showing the aisles of the warehouse with the appropriate number of boxes on each shelf? McDonald (1983) deals with medical supply inventory with a visual warehouse display by combining videodisc and computer graphics technology.

Why not teach students about polynomial equations by letting them bend the curves and watch how the coefficients change or where the X-axis intercept occurs or how the derivative equation reacts (Shneiderman, 1974)?

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

5.5 DIRECT MANIPULATION DISK OPERATING SYSTEM (DMDOS)

This section gives a detailed description of one design project whose goal was to develop a direct manipulation user interface for a widely used command language. The difficulty of designing a direct manipulation system was explored in a visual design for the commands in the Microsoft Disk Operating System (MS-DOS) for IBM and compatible computers (detailed design and implemention by Osamu Iseki). The motivations were to avoid the:

1. Error-prone, difficult-to-remember, and difficult-to-type commands, such as;

   ```
   dir/w c:level2
   copy a:filel.pas b:filel.bak
   erase c:level2 filel
   ```

2. Need for many commands, such as: `VERsion`, `VOLume` `DATE`, `TIME`, `CD`, `MKDIR`, `RMDIR`, `CHDIR`, `TYPE`, `PRINT`, `DELETE`, and `RENAME`.
3. Frustration of watching the directory listings scroll off of the screen too quickly.
4. Uncertainty of not seeing the source and destination directories while copying, comparing, or deleting files. After issuing a file command, many users issue a directory display command to verify that the command was carried out correctly.
5. Need to type file and directory names, except when they are created. Once created, file and directory names can be selected from the display. When the number of files is in

the hundreds, it may be more convenient to type the file name, but with only tens of files, selection by pointing is often more rapid and accurate.

5.5.1 Design goals

In a positive way, we sought to provide a world in which the:

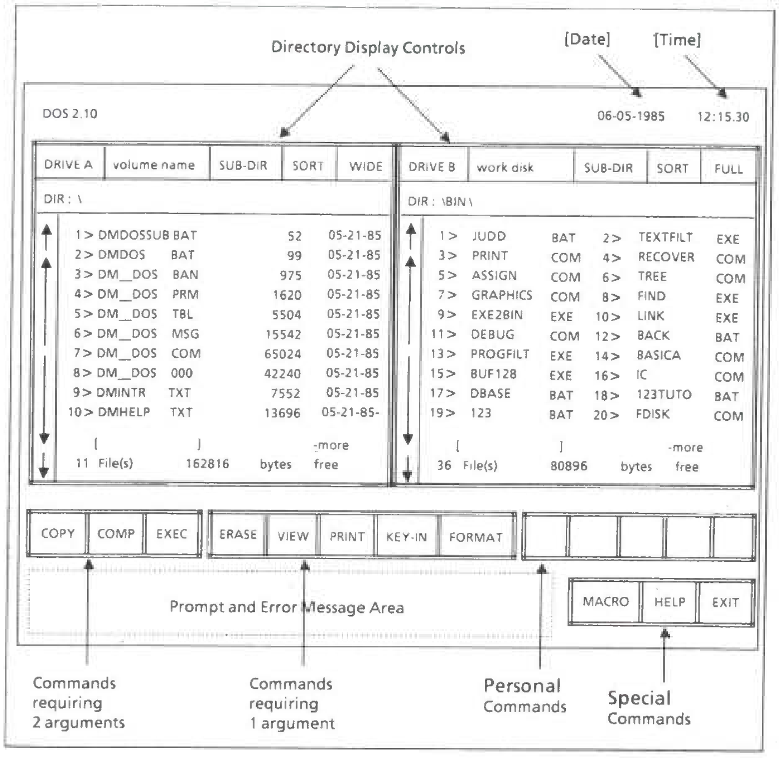
1. Task-related objects (files) and the actions (commands) were always visible.
2. Users could select objects and actions by pointing instead of typing.
3. Results of actions were immediately visible.

We hoped that such a design would be easy to learn and retain, rapid in performing tasks, low in errors, and high in user satisfaction. After many revisions and tests with hundreds of knowledgeable observers and novices, the screen layout was determined (Figure 5.13).

5.5.2 Screen organization

The largest parts of the screen are the right and the left drive information areas. Each area has the:

1. Drive name (A, B, or C).
2. Volume name of the disk.
3. [SUB-DIR/FILE], the toggle switch for changing the listing from/to file names or subdirectory names.
4. [SORT] switch that allows users to set the sort condition of the file listing to sort by file name, extension, size, or date.
5. [WIDE/FULL], toggle switch for changing the format of the file listing from a single to a double column listing.



```
DIR : \
 1> DMDOSSUB BAT      52   05-21-85
 2> DMDOS    BAT      99   05-21-85
 3> DM__DOS  BAN     975   05-21-85
 4> DM__DOS  PRM    1620   05-21-85
 5> DM__DOS  TBL    5504   05-21-85
 6> DM__DOS  MSG   15542   05-21-85
 7> DM__DOS  COM   65024   05-21-85
 8> DM__DOS  000   42240   05-21-85
 9> DMINTR   TXT    7552   05-21-85
10> DMHELP   TXT   13696   05-21-85-
    [               ]           -more
 11 File(s)     162816   bytes   free
```

```
DIR : \BIN\
 1>  JUDD      BAT    2>  TEXTFILT  EXE
 3>  PRINT     COM    4>  RECOVER   COM
 5>  ASSIGN    COM    6>  TREE      COM
 7>  GRAPHICS  COM    8>  FIND      EXE
 9>  EXE2BIN   EXE   10>  LINK      EXE
11>  DEBUG     COM   12>  BACK      BAT
13>  PROGFILT  EXE   14>  BASICA    COM
15>  BUF128    EXE   16>  IC        COM
17>  DBASE     BAT   18>  123TUTO   BAT
19>  123       BAT   20>  FDISK     COM
    [               ]           -more
 36  File(s)     80896   bytes   free
```

Figure 5.13: DMDOS (Iseki & Shneiderman, 1986) performs MS-DOS functions by arrow key or mouse movements instead of typing commands. The display shows the source and destination directory displays that are scrollable under user control.

6. Current directory name.
7. The listing of file names or subdirectory names in the current directory. Each listing has a maximum of twenty file names or ten subdirectory names.
8. Icons for scrolling up and down the listing. The short arrows serve for one-line scrolling and the long arrows for five-line.

These two large areas are independent from each other.

Below these drive information areas are command areas that are categorized into four groups:

1. Commands requiring two arguments: COPY, COMPare, EXECute.
2. Commands requiring one argument: ERASE, VIEW, PRINT, KEY-IN, and FORMAT.
3. Definable personal commands (space is provided for only five).
4. Special commands: MACRO, HELP, and EXIT.

The screen format contains most items users need, and it remains visible unless a program file is executed or the contents of a file are seen by using the [VIEW] command. Therefore, users may have less anxiety about the correct execution of a command than working in a line-by-line mode.

5.5.3 Functions

DMDOS functions are categorized into seven groups:

1. Functions automatically executed. For example, the directory listing is always displayed on the screen. This eliminates the need for the DIR command.
2. Functions executed by single object or switch selection. Selecting an object causes some functions.
3. Functions executed by overtyping the selected object. These are mainly for renaming files and changing the date or time.
4. Functions executed by commands with one argument. These commands need only a source object.
5. Functions executed by commands with two arguments. These commands need source and destination objects.

6. Functions executed by personal commands. These commands can be defined by the user and include a sequence of operations.
7. Functions executed by special commands.

Table 5.1 shows functions in each category. These are described with the equivalent MS-DOS commands for comparison. Since we tried to

DMDOS command category	PC-DOS internal commands	PC-DOS external commands
Automatically Executed or Executed by Single Selection	DIR DATE(display) TIME (display) VERsion VOLume name (CHDIR)	SORT(only for filename)
Executed by Overtyping the Object	RENAME MKDIR CHDIR DATE(set) TIME (set)	
1-object Command	ERASE [ERASE] RMDIR [ERASE] TYPE [VIEW]	FORMAT [FORMAT] PRINT [PRINT]
2-object Command	COPY [COPY]	COMP [COMP] DISKCOMP [COMP] DISKCOPY [COPY]
Additional Command	[EXEC] [KEY-IN] [HELP] [MACRO]	
not necessary in DMDOS	CLS	
not implemented in DMDOS	(Batch command), BREAK, PATH , VERIFY	ASSIGN, BACKUP, CHKDISK, GRAPHICS, MODE, RECOVER, RESTORE, SYS

Table 5.1: A comparison of DMDOS (Iseki & Shneiderman, 1986) and MS-DOS commands showing how selection of visible objects and actions replaces typed commands.

to reduce the number of commands to the minimum, some MS-DOS functions used infrequently have not been implemented. The most important commands not implemented are batch processing commands that allow users to make their own command sequences in an executable batch file. However, DMDOS has a special command [MACRO] to make user-definable commands (see Section 5.5.5).

5.5.4 Operational principles

Operation in DMDOS is mainly based on movement of the cursor and selection of an object or a command of interest. Cursor movement and selection can be done by using either keyboard or mouse.

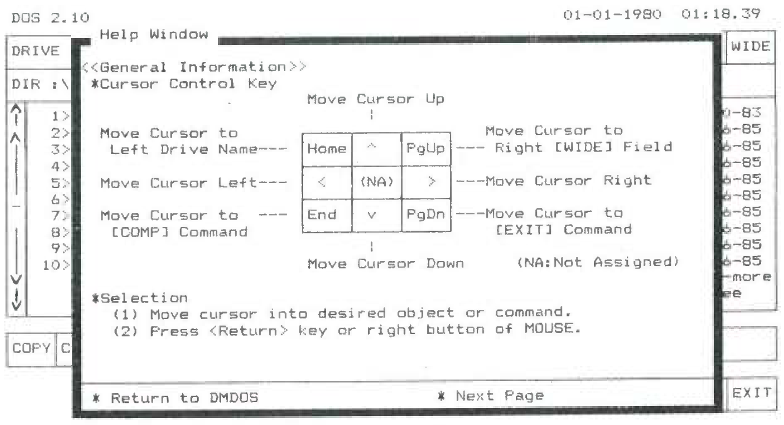
Using a keyboard. Users can use four arrow keys to move the cursor. The cursor is moved from item to item instead of from character to character except if it is in a selected object. Selection is done by placing the cursor in the object of interest and pressing the RETURN key.

Using a mouse. The cursor moves character by character according to the mouse movement. Selection is made by pressing one mouse button. The keyboard can also be used at the same time.

The mouse is a more 'direct' tool to locate the cursor on the item of interest quickly. However, two sets of special keys may help users move the cursor effectively by the keyboard. First, four corner keys adjacent to four arrow keys allow users to make a shortcut by jumping the cursor to the four corners of the screen. Second, the ten function keys (F1-F10) are dedicated to moving the cursor onto the drive field or scrolling icons of each drive and making a selection. Figure 5.14 shows the keys used for cursor movement.

The item visited by the cursor is reverse-displayed partly or fully. Once the item is selected, it remains highlighted until it is unselected or a function is executed.

As a special case of selection, multiselection of file names is permitted. The operation of multiselection is to hold down the CTRL key while pressing the RETURN key when selecting some file names. The



```
DOS 2.10                                          01-01-1980  01:18.39
DRIVE    Help Window                                               WIDE
DIR :\ <<General Information>>
       *Cursor Control Key
                            Move Cursor Up
                                  |
        Move Cursor to                            Move Cursor to
         Left Drive Name---  | Home |  ^   | PgUp | --- Right [WIDE] Field
        Move Cursor Left---  |  <   | (NA) |  >   | ---Move Cursor Right
        Move Cursor to   --- | End  |  v   | PgDn | ---Move Cursor to
         [COMP] Command                                [EXIT] Command
                                  |
                            Move Cursor Down      (NA:Not Assigned)

       *Selection
         (1) Move cursor into desired object or command.
         (2) Press <Return> key or right button of MOUSE.

COPY   * Return to DMDOS                    * Next Page              EXIT
```

Figure 5.14: This help screen from DMDOS shows cursor key movement possibilities.

maximum number of multiselection files is ten. This function may be used for copying several files to another drive or erasing several files at one time. In MS-DOS, it is necessary to use such wild-card characters as "*" and "?" to indicate the multiple files. Though this mechanism is also allowed in DMDOS, multiselection is sometimes an easier way to indicate a set of files.

The general principle of operation is to select an object first, called the source object. After that, overtyping or selecting a command is effective. Three typical operational sequences are:

1. Overtyping
   a. Select an object of interest
   b. Overtype the new name
   c. Select the object again to install the new name
2. Command with one argument
   a. Select an object of interest
   b. Select a command

   c. Users might be prompted by some messages.
      Answer these prompts, if any.
      Then the command will be executed.
3. Command with two arguments
   a. Select a source object
   b. Select a command
   c. Select a destination object
   d. Users might be prompted by some messages.
      Answer these prompts, if any.
      Then the command function will be executed.

This object-first strategy is different from the order of MS-DOS's command syntax. After executing a command function, the most recently selected object is still selected. Therefore, if users need to execute any other command on this object, they can simply select a new command just after the execution of the previous command.

5.5.5 Programming with the macro command

One major objective of DMDOS development was to try to implement a programming capability based on direct manipulation. This macro definition function corresponds to the batch command in MS-DOS. A command defined by this function is called a personal command and can be regarded as a small program. To maintain consistency, a personal command can be programmed by showing an example of a sequence of ordinary operations. Users can enter the macrodefining stage by selecting the MACRO special command. A sequence of selections will be recorded as a replayable operational sequence during this stage.

Each personal command can have one or two arguments like other commands for source and destination objects. By modifying the example of the operational sequence, these arguments can be implemented in the personal command.

For example, making a personal command called "MOVE," whose function is to move a file (source argument) from one drive to another (destination argument), requires these steps:

1. Select [MACRO] special command.
2. Select [Define...] macro subcommand.
3. Type the name of MOVE and press RETURN. The macrodefining stage begins.
4. Show an operational example.
   a. Select a file named FILE1.
   b. Select [COPY] command.
   c. Select a drive name B:.
   d. Select FILE1 again.
   e. Select [ERASE] command.
   f. Select [Yes] for confirmation.
5. Select [Macro] command again to indicate the end of the macrodefining stage. Then the recorded operational sequence and edit-parameter will be displayed.
6. Select [Source Argument] parameter for the first object FILE1. Then the FILE1 selected at 4.d will also be changed to the source argument automatically.
7. Select [Destination Argument] parameter for the second object B:.
8. Select [Return to DMDOS] to indicate the end of the modification.

This MOVE command will copy a source file to the destination drive and erase the source file.

5.6 CONCLUSION

Novice and knowledgeable users have expressed a strong interest in DMDOS, but field trials with real users are just beginning. There are some performance problems that may limit use of the current version. DMDOS takes almost 20 seconds to load on a standard PC, requires use of one of the floppy disk drives, and consumes more than 52,000 bytes. These problems are reduced when a hard drive is available. Since DMDOS invokes MS-DOS functions, hardware failures produce error messages that sometimes destroy the DMDOS screen.

On the user interface side, there is some annoyance in having to do so much arrow key pressing, and therefore some further shortcuts should be explored. The mouse version reduces this problem. Single-letter alternates for some commands might further speed the work of some frequent users. The KEY-IN feature that uses the MS-DOS console input mechanism might be replaced by a small text editor.

Knowledgeable users often remark that they would prefer to type commands and believe that they can work more rapidly by just typing the commands. A field trial over several weeks would be helpful in ascertaining actual performance.

We feel that we succeeded in our goal to create a direct manipulation interface for MS-DOS commands. Whether these ideas can become successfully integrated into commercial software remains to be seen. The 6,000 lines of Turbo Pascal were written in a seven-month period by one person (Osamu Iseki). Re-implementation should be easier and would permit inclusion of additional features.

Several competing software products accomplish some of the goals of DMDOS. File Command from IBM reduces the memorization and keystroking burden by showing commands on the screen and allowing users to press function keys to produce commands. The 1DIR package from Bourbaki provides some features of DMDOS but shows only one directory at a time. The Apple Macintosh desktop and the GEM Desktop for IBM PCs offer visual representations of directories and convenient icon movement or copying.

5.7 DIRECT MANIPULATION PROGRAMMING

Performing tasks by direct manipulation is not the only goal. It should be possible to do programming by direct manipulation as well, for at least some problems. The MACRO command of DMDOS supports a limited form of programming, but more complex forms of programming seem possible with direct manipulation ideas.

Robot programming is sometimes done by moving the robot arm through a sequence of steps that are later replayed, possibly at higher speed. This example seems to be a good candidate for generalization. How about moving a drill press or a surgical tool through a complex series of motions that are then repeated exactly? How about programming a car by driving it once through a maze and then having the car repeat the path? In fact, these direct manipulation programming ideas are implemented in modest ways with automobile radios that are preset by turning the frequency control knob and then pulling out a button. When the button is depressed, the radio tunes to the frequency. Some professional television camera supports allow the operator to program a sequence of pans or zooms and then replay them smoothly when required.

Programming of physical devices seems quite natural by direct manipulation, but an adequate visual representation of information may make direct manipulation programming possible in other domains. Several word processors allow creation of macros by simply performing a sequence of commands that are stored away for later reuse. The Wang Decision Processing system enables the creation of "glossary" items that can be lengthy sequences of text, special function keys such as TAB, and control structures (Figure 5.15). The control structures can test for user input and cursor location. Glossary items can invoke each other, leading to complex programming possibilities. LOTUS 1–2–3, Symphony, and Framework have rich programming languages and allow portions of programs to be created by carrying out standard spreadsheet operations. The result of the operations is stored in another part of the spreadsheet and can be edited, printed, and stored in a textual form.

A delightful children's program, Delta Drawing from Spinnaker, enables children to move a cursor and draw on the screen by typing D to draw one unit, R to rotate right 30 degrees, and so on. The forty commands provide rich possibilities for drawing some kinds of screen images. In addition, Delta Drawing allows users to save, edit, and then invoke programs. For example, a circle can be drawn by saving the

(g)

```
(-TAB-) (-TAB-) Very truly yours, (-RETURN-)
(-RETURN-) (-RETURN-)
(-TAB-) (-TAB-) J. S. Bach
(-RETURN-) (-TAB-) (-TAB-)
President (-RETURN-)

(-PROMPT-)
   Printout? y or n
(-EXECUTIVE-)
(–1-KEY-)

(-BACKSPACE-)

(-IF-) "y"
   (-GO-TO-GL-)y
(-END-)
(-GO-TO-GL-)n
```

Figure 5.15: Direct manipulation programming in the Wang Decision Processing System. Users press labelled function keys for each token to create the program which is stored in the glossary.

program consisting of a `D` and a `R`. Invoking the program with the argument, `12`, then produces a rough twelve-sided circle.

A number of research projects have attempted to create direct manipulation programming systems. Halbert's Smallstar (1984) was a programming-by-example system to enable programming of Xerox Star actions. PICT-D (Glinert & Tanimoto, 1984) and ThinkPad (Rubin, Golin, & Reiss, 1985) both tried to make graphical icons into a programming language. This strategy has many of the elements of direct manipulation.

5.8 PRACTITIONER'S SUMMARY

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

Just because direct manipulation principles have been used in a system does not ensure its success. A poor design, slow implementation, or inadequate functionality can undermine acceptance. For some applications, menu selection, form fill-in, or command languages may be more appropriate. Iterative design (see Section 10.2) is especially important in testing direct manipulation systems because the novelty of this approach may lead to problems for designers and users.

5.9 RESEARCHER'S AGENDA

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

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

REFERENCES

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

Bewley, William L., Roberts, Teresa L., Schroit, David, and Verplank, William L., Human factors testing in the design of Xerox's 8010 "Star" Office Workstation, *Proc. CHI '83 Conference - Human Factors in Computing Systems*, Available from ACM Order Dept., P. O. Box 64145, Baltimore, MD 21264, (1983), 72–77.

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

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

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

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

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

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

Glinert, Emphraim, and Tanimoto, Steven L., Pict: An interactive graphical programming environment, *IEEE Computer 17*, 11, (November 1984), 7–25.

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

Hatfield, Don, Personal communication and lecture at Conference on Easier and More Productive Use of Computer Systems, Ann Arbor, MI (1981).

Heckel, Paul, *The Elements of Friendly Software Design*, Warner Books, New York, NY, (1984), 205 pages.

Herot, Christopher F., Spatial management of data, *ACM Transactions on Database Systems*, Vol. 5, No. 4, (December 1980), 493–513.

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

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

Hutchins, Edwin L., Hollan, James D., and Norman, Don A., Direct manipulation interfaces, In Norman, Don A., and Draper, Stephen W. (Editors), *User Centered System Design: New Perspectives on Human-Computer Interaction*, Lawrence Erlbaum Associates, Hillsdale, NJ, (1986).

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

MacDonald, Alan, Visual Programming, *Datamation 28*, 11, (October 1982), 132–140.

McDonald, Nancy, Multi-media approach to user interface, In Vassiliou, Yannis (Editor), *Human Factors in Interactive Computer Systems*, Ablex Publishing Co., Norwood, NJ, (1983).

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

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

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

Nelson, Ted, Interactive systems and the design of virtuality, *Creative Computing* Vol. 6, No. 11, (November 1980), 56 ff., and Vol. 6, No. 12, (December 1980), 94 ff.

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

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

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

Rubin, Robert V., Golin, Eric J., and Reiss, Steven P., Thinkpad: a graphics system for programming by demonstrations, *IEEE Software* Vol. 2, No. 2, (March 1985), 73–79.

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

Schild, W., Power, L. R., and Karnaugh, M., PICTUREWORLD: A concept for future office systems, IBM Research Report RC 8384, Yorktown Heights, NY, (July 30, 1980).

Shneiderman, Ben, A computer graphics system for polynomials, *The Mathematics Teacher*, Vol. 67, No. 2, (1974), 111–113.

Shneiderman, Ben, Control flow and data structure documentation: Two Experiments, *Communications of the ACM*, Vol. 25, No. 1, (January 1982), 55–63.

Shneiderman, Ben, Mayer, R., McKay, D., and Heller, P., Experimental investigations of the utility of detailed flowcharts in programming, *Communications of the ACM*, Vol. 20, (1977), 373–381.

Smith, Cranfield, Irby, Charles, Kimball, Ralph, Verplank, Bill, and Harslem, Eric, Designing the Star user interface, *BYTE*, Vol. 7, No. 4, (April 1982), 242–282.

Thimbleby, Harold, What you see is what you have got? Unpublished paper, University of York, England, (1982).

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

Yedwab, Laura, Herot, Christopher F., and Rosenberg, Ronni L., The automated desk, *Sigsmall Newsletter 7*, 2, (October 1981), 102–108.

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

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