Applying Direct Manipulation Concepts:

Direct Manipulation Disk Operating System (DMDOS)

Osamu Iseki and Ben Shneiderman
Human-Computer Interaction Laboratory
Center for Automation Research
Department of Computer Science
University of Maryland
College Park, MD 20742

Abstract: Software engineers are often called upon to design user interfaces, but strategies and guidelines are only beginning to emerge. Shneiderman (1983) introduced the term "Direct Manipulation" to describe user interfaces which have:

1) continuous representation of the objects of interest.

2) physical actions (movement and selection by mouse, joystick, touch screen, etc.) or labeled button presses instead of complex syntax.

3) rapid, incremental, reversible operations whose impact on the object of interest is immediately visible.

4) layered or spiral approach to learning that permits usage with minimal knowledge.

The concepts of direct manipulation has been applied in some distinctive systems such as XEROX STAR and APPLE Macintosh, and many application software products such as spread sheets, word processors, drawing tools, desk-top managers, etc.

However, the basic software of personal computers, the operating system, is still often based on command language concepts. This paper describes DMDOS (Direct Manipulation Disk Operating System), that we designed by applying the concepts of direct manipulation to MS-DOS on the IBM PC.

*Address correspondence to Ben Shneiderman

1. INTRODUCTION

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 Oseki). 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:file1.pas b:file1.bak
erase c:\level2\file1
```

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.

2. 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) user could select objects and actions by pointing instead of typing, and

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 1).

3. 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 which allows users to set the sort condition of the file listing: 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.

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 1-line scrolling and the long arrows for 5-line.

These two large areas are independent from each other.

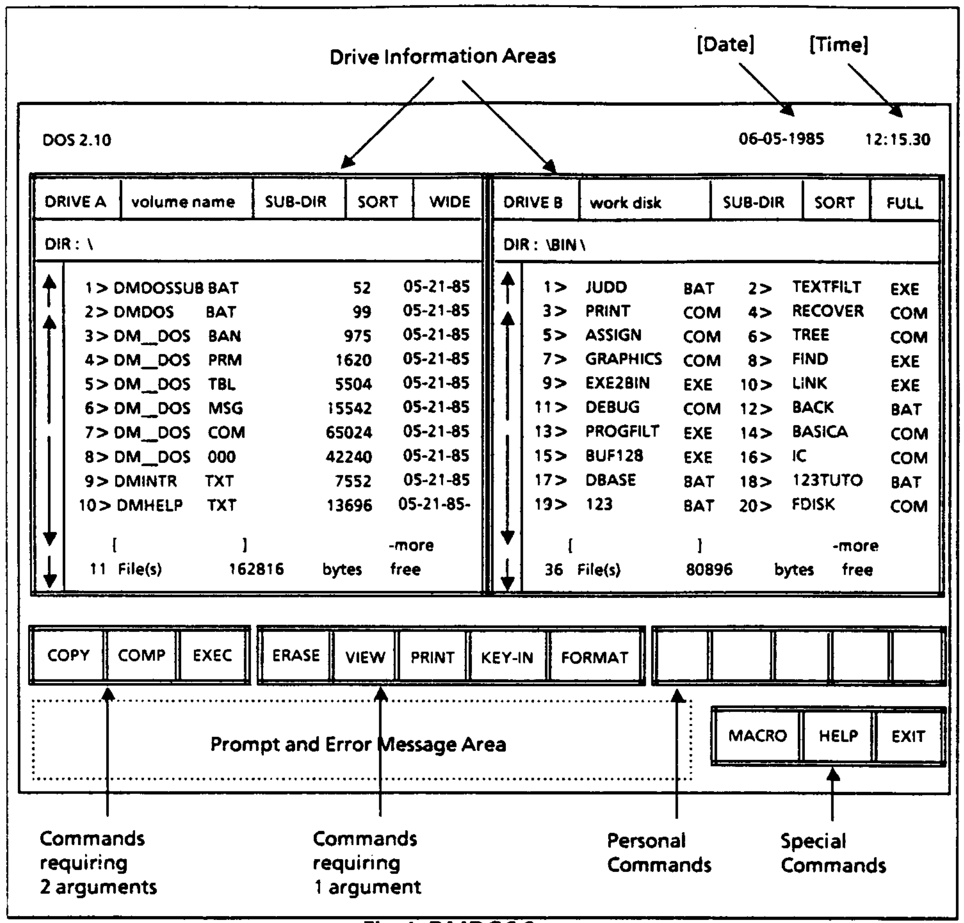
Below these drive information areas, there are command areas, which are categorized into four groups.

1) Commands requiring 2 arguments: COPY, COMPare, EXECute.

2) Commands requiring 1 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.



```
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
```

Fig.1 DMDOS Screen

The screen format contains most items users need and it remains visible unless a program file is executed or the contents of a file is 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.

Ten screens of online help are available to briefly describe the screen layout and the commands.

4. 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.

DMDOS command category	Functions	PC-DOS commands
Automatically Executed	*directory listing with sort *display DOS version *display volume name	DIR and SORT VERsion VOLume
Executed by Single Selection	*display current date and time	DATE and TIME (display)
Executed by Overtyping the Object	*rename a file *make a subdirectory *change current directory *set date and time	RENAME MKDIR CHDIR DATE and TIME (set)
Executed by Command with 1-argument	*[ERASE]:erase file or subdirectory *[VIEW]:display contents of file *[PRINT]:print out contents of file *[FORMAT]:format a disk *[KEY-IN]:make a text file	ERASE and RMDIR TYPE PRINT FORMAT COPY-CON
Executed by Command with 2-argument	*[COPY]:copy file *[COMP]:compare files *[EXEC]:execute file	COPY COMP (type filename)
Executed by Special Command	*[HELP]:display brief manual *[MACRO]:define personal command *[EXIT]:exit to PC-DOS	
not implemented in DMDOS		(Batch command), BREAK, PATH, VERIFY, ASSIGN, BACKUP, CHKDISK, GRAPHICS, MODE RECOVER, RESTORE, SYS

Table 1. Functions of DMDOS

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 1 shows functions in each categories. These are described with the equivalent MS-DOS commands for comparison. Since we tried 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 which 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. This feature will be discussed later.

5. 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.

<u>Using a keyboard</u>: 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.

<u>Using a mouse</u>: 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 than the keyboard to locate the cursor on the item of interest quickly. However, two sets of special keys may help users to move the cursor effectively by the key board. First, four corner keys adjacent to four arrow keys allow users to take 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.

The item visited by the cursor will be 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, multi-selection of a file name is permitted. The operation of multi-selection is to hold down the CTRL key while pressing the RETURN key when selecting file names. The maximum number of multi-selection 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 wild-card characters such as "*" and "?" to indicate the multiple files. Though this mechanism is also allowed in DMDOS, multi-selection 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:

i) Overtyping
   (1) Select an object of interest.
   (2) Overtype the new name.
   (3) Select the object again to install the new name.

ii) Command with one argument
   (1) Select an object of interest.
   (2) Select a command.
   (3) Users might be prompted by some messages. Answer these prompts, if any. Then the command will be executed.

iii) Command with two arguments
   (1) Select a source object.
   (2) Select a command.
   (3) Select a destination object.
   (4) Users might be prompted by some messages. Answer these prompts, if any. Then the command will be executed.

This object-first strategy is different from the order of PC-DOS' 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.

6. PROGRAMMING THE MACRO COMMAND

One of the major objectives of DMDOS development was to try to implement the programming capability based on direct manipulation, which we call the macro definition function. This 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 keep consistency, a personal command can be programmed by showing an example of a sequence of ordinary operations. Users can enter the macro-defining 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 sub-command.

3) Type the name of "MOVE" and <Return>. The macro-defining stage begins.

4) Show an operational example.
   4.1) Select a file named "FILE1".
   4.2) Select [COPY] command.
   4.3) Select a drive name "B:".
   4.4) Select "FILE1" again.
   4.5) Select [ERASE] command.
   4.6) Select [Yes] for confirmation.

5) Select [Macro] command again to indicate the end of the macro-defining 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 4.4 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.

7. 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 over 52,000 bytes. These problems are reduced when a hard disk drive is available. Since DMDOS invokes MS-DOS functions, hardware failures produce error messages which 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 of the 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 are effective and 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. A fifty-page user's manual with complete illustrations is available. As a courtesy to permit review by researchers and developers, Cognetics Corporation, 55 Princeton-Hightstown Road, Princeton Junction, NJ 08550 will distribute the software plus users manual for $20 (this is meant to cover costs of diskette, manual reproduction, binding, and mailing). No guarantee or support is provided.

REFERENCE

Shneiderman, Ben, Direct Manipulation: A Step Beyond Programming Languages, IEEE COMPUTER, 16, 8, (August 1983), 57-69. .