A Window Manager for the UCSD p-System

Austin Tate
Microcomputer Support Unit
Edinburgh Regional Computing Centre

1. Introduction

There has been a great deal of interest recently in the use of computer displays which allow a screen to be divided into a number of separate areas or 'windows' such as are used in the Xerox environment (Alto, 1979). Examples of software environments based on the Alto using windows include Smalltalk (a summary of several environments is given in Goldberg, 1979) and the Lisp Programmer's Assistant (Teitelman, 1977). In these systems, separate windows are often 'attached' to separate, possibly concurrent, processes.

As part of the development of course materials for a series of seminars on Electronic Office Systems, the author has developed a demonstration Window Manager package for the portable UCSD p-System for microcomputers (Softech). Although the hardware on which this system is being used is too slow for this software to be used in a production environment it does provide a realistic demonstration of the types of facilities we can expect to be available on future personal computers. For example, the Three Rivers Computer Corporation's 'PERQ' computer (Three Rivers) includes a 'Raster Op' microcoded instruction to facilitate various windows on the screen.

The window manager package has been used in a demonstration program which provides some of the facilities of the UCSD Filer utility.

2. Structure of the Package

The Window Manager is written as a UCSD Pascal Unit. A Unit provides a publically available set of constants, variables, procedures, etc. and gives implementation details of how these 'interfaces' are provided. The listing of the window manager interface section is provided in Appendix I.

The structure of the implementation relies heavily on an internal data structure for individual windows not visible to the user. This includes the following items:

a) position of top left corner of window in screen coordinates

b) size of window in horizontal and vertical directions

c) position of current input or output portion in the window in window coordinates (i.e. position of <u>this</u> windows' cursor).

d) a data structure for lines of text in the window. These are allocated from a heap of lines.

e) a list structure pointer to show order in which windows were displayed to ensure they currectly overlap if moved, killed, etc.

f) flags to show various characteristics of the window. E.g. if horizontal scrolling is allowed, whether the window has a ´heading´ line, etc.

Given this data structure and a carefully chosen interface, the package itself is straightforward to write. The actual implementation and testing took less than 40 man/hours. The Window Manager is about 1000 lines of commented Pascal code. It is straightforward to build the implementation gradually providing the more estoteric procedures later.

3. Window Frames

Due to the different capabilities of video displays, it was found desirable to write the Window Manager in such a way that window frames were handled in a separately implemented Unit. The interface section for this FRAMES unit is provided in Appendix II. A straightforward, but rather unimpressive looking, character framing capability is possible on normal character displays. However, where graphics facilities can be used, e.g. on the Terak 8510a microcomputer (Terak), these can be exploited to provide quite impressive looking frames and header backgrounds. The Terak frames package was generated in a few hours by Ken Currie of ERCC. The frame unit for the Terak is approximately 120 lines of commented Pascal code.

4. The ´Window´ Filer

The Window Filer is a demonstration utility which provides a subset of the facilities available in the UCSD p-System Filer component.

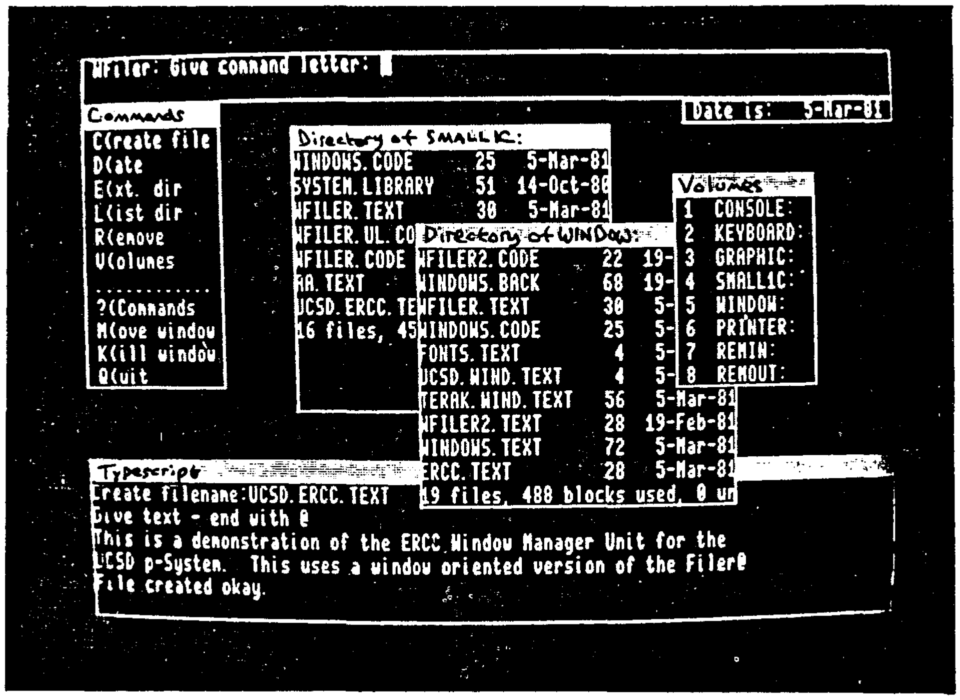
It employs the window manager package for all input and output. It was generated in a few hours by adding simple window initialisation code and altering the (already localised) input and output instruction in on existing program which provided the subset of UCSD filer facilities.

There are facilities to view the volumes on-line or the directories of floppy disk volumes in brief or detailed forms, to remove files from disk, to create text files on disk, to view the system date, etc. Displays are created in separate areas of the screen known as ´windows´.

The window filer is entered by X(executing) WFILER at the UCSD command (outer) level.

Several standard windows will appear. A ´message´ window at the top of the screen where prompts to a user will normally appear, a list of commands and a ´typescript´ window in which user input is normally solicited. There is a ´window cursor´ (@ sign) on the screen (initially in the centre of the screen.

The system expects a single character command as detailed in the commands list (e.g. "V" to get a list of volumes on-line).



```
WFiler: Give command letter: █

Commands                                                  Date is:   5-Mar-81
C(reate file   Directory of SMALL1C:
D(ate          WINDOWS.CODE      25   5-Mar-81
E(xt. dir      SYSTEM.LIBRARY    51  14-Oct-8            Volumes
L(ist dir      WFILER.TEXT       30   5-Mar-81           1  CONSOLE:
R(emove        WFILER.UL.CO Directory of WINDOW:         2  KEYBOARD:
V(olumes       WFILER.CODE  WFILER2.CODE      22  19-    3  GRAPHIC:
               AA.TEXT      WINDOWS.BACK      68  19-    4  SMALL1C:
............   UCSD.ERCC.TE WFILER.TEXT       30   5-    5  WINDOW:
?(Commands     16 files, 45 WINDOWS.CODE      25   5-    6  PRINTER:
M(ove window                FONTS.TEXT         4   5-    7  REMIN:
K(ill window                UCSD.WIND.TEXT     4   5-    8  REMOUT:
Q(uit                       TERAK.WIND.TEXT   56   5-Mar-81
                            WFILER2.TEXT      28  19-Feb-81
                            WINDOWS.TEXT      72   5-Mar-81
Typescript                  ERCC.TEXT         28   5-Mar-81
Create filename:UCSD.ERCC.TEXT 19 files, 488 blocks used, 0 un
Give text - end with @
This is a demonstration of the ERCC Window Manager Unit for the
UCSD p-System.  This uses a window oriented version of the Filer@
File created okay.
```

At any time, the four cursor control (arrowed) keyed can be used to move the window cursor to a new position. In addition, CTRL/O (for "Over") can be used to bring the window area in which the window cursor currently resides into full view. Windows may be M(oved) and K(illed) within WFILER.

The use of screen windows to simulate moving sheets of paper on a desk to access information needed for some work can be shown by using a (create file) function. While input is being typed to a file, the window manager can be directed to bring windows of interest on top.

5. Availability

The window manager unit is not considered to be a potential tool at present. However, for demonstration purposes it is being submitted to the UCSD p-System Users Society Software Library. If it passes the relevant review procedure it will subsequently become available to members of USUS.

6. References

Alto	Thacker, C.P., McGeight, E.M., Lampson, B.W., Sproull, R.F. and Boggs, D.R. (1979) Alto: a personal computer. In Siewiorek, D., Bell, C.G., and Newell, A., Computer Structures, Readings and Examples. Second Edition, McGraw-Hill.
Goldberg	Goldberg, A. (1979) Educational uses of a Dynabook. Computers and Education Vol.8 pp 247-266.
Softech	UCSD p-System User´s Manual. Softech Microsystems Inc. 9494, Black Mountain Road, San Diego, California 92126.
Teitelman	Teitelman, W. (1977) A display oriented programmer´s assistant. Proceedings of the 5th International Joint Conference on Artificial Intelligence, pp 905-915.
Terak	Terak 8510/a brochure. Terak Corporation, 14151, North 76th Street, Scottsdale, Arizona 85260.
Three Rivers	PERQ brochure. Three Rivers Computer Corporation. 160, North Craig Street, Pittsburg, Pennsylvania 15213.

Appendix I Interface section of Window Manager Unit for UCSD p-System

```
UNIT WindowManager;  {revision 23-Mar-81 Austin Tate}

INTERFACE

USES Frames;

  {Window Manager for the UCSD p-System}
  {Windows are displayed as rectangular areas on the screen, optionally
   bordered by a frame and headed by a heading.  Each window has its own
   size, screen location, text area, cursor and status information.
   Each window may be written into and will scroll independently, they
   may be cleared, moved, changed in size, etc.
  }

  CONST NoWindow=0;
        MaxWindow=10;

  TYPE  Window=NoWindow..MaxWindow;

  PROCEDURE WInit(AllowHides:BOOLEAN); {Initialise Window Manager System}
            {state whether Hides are allowed during Window Read functions}
            {depends whether application software keeps track of windows}

  PROCEDURE WSetControls(WCursorCode,    {symbol to be used for window}
                                         {managers cursor}
                         WUpKeyCode,     {Symbols to be intercepted by }
                         WDownKeyCode,   {Window Manager in WReadCh    }
                         WRightKeyCode,  {procedure. These cause action}
                         WLeftKeyCode,   {on the screen by moving the  }
                         WHideKeyCode,   {window cursor, removing      }
                         WShowKeyCode,   {windows from the screen,     }
                         WPerformKeyCode {showing them, etc.           }
                                        :INTEGER);
            {any -ve code means do not alter} {no checks for duplicates}
            {All are defaulted in WInit to values for Terak 8510a keyboard}
            {all are rounded to 0.255 before use}
```

```
FUNCTION WNew(WatX,WatY,WSizeX,WSizeY:INTEGER;
              WShowFrame,WHorizScroll,WShowHeading:BOOLEAN;
              WHeading:STRING):Window; {Get new window}

PROCEDURE WAlter(W:Window;
                 WatX,WatY,WSizeX,WSizeY:INTEGER;
                 AlterFlags,WShowFrame,WHorizScroll,WShowHeading:BOOLEAN;
                 WHeading:STRING); {Get new window}
          {WatX,WatY,SizeX,SizeY -ve means do not alter}
          {AlterFlags is false if falgs are not to be changed}
          {Window must not be in show when WAlter called}

PROCEDURE WDispose(W:Window); {Dispose of old window}
          {Window must not be in show when WDispose called}

PROCEDURE WShow(W:Window); {Display window and set it as "current" one}

PROCEDURE WClearAndShow(W:Window); {clear window, then "Show" it}

PROCEDURE WHide(W:Window); {remove window from screen - it is not disposed of}

{The following procedures all apply to the "current" last shown window}

PROCEDURE WClear; {Clear Window}

PROCEDURE WWriteCh(Ch:CHAR); {write Ch at cursor position in window}
          {non printable chs map to bell}

PROCEDURE WWriteStr(Str:STRING); {write Str at cursor position in window}
          {non printable chs map to bell}

PROCEDURE WWriteInt(Int,Width:INTEGER); {write Int at cursor posn in window}
          {equivalent to WRITE(Int:Width) in Pascal}
          {width may be 0 (or -ve) to mean as narrow as possible}

PROCEDURE WWriteLn; {write newline at cursor position in window}
          {If cursor goes below base of window, window is scrolled}
```

```
  PROCEDURE WReadCh(VAR Ch:CHAR;Echo:BOOLEAN); {get character from keyboard}
            {Window functions can only take place within WReadCh}
            {any non window function ch is returned to user     }
            {Echo is controlled by user - non printable chs echo as bell}

  PROCEDURE WReadLnStr(VAR Str:STRING); {get a string from keyboard - echoed}
            {string is ended by newline. Only edit ch allowed is backspace}
            {non printable chs are not returned - but echo as bell}

  PROCEDURE WReadLnInt(VAR Int:INTEGER); {Get an integer from keyboard - echoed}
            {integer is ended by newline. Only edit ch allowed is backspace}
            {non printable chs are not returned - but echo as bell}

  PROCEDURE WReadLn(Echo:BOOLEAN); {Read up to next newline from keyboard}
            {non printable chs echo as bell}

  {the following functions and procedures are utilities on windows}

  FUNCTION WInWindow(X,Y:INTEGER):Window;
            {returns window in which position X,Y occurs - NoWindow if none}
            {X,Y in screen coordinates}

  FUNCTION WChAtXY(X,Y:INTEGER; W:Window):CHAR;
            {return Ch under screen position X,Y in W}
            {space returned if X,Y not in Window, or NoWindow}
            {Ch need not be in view at time of call}

  PROCEDURE WXY(VAR X,Y:INTEGER); {Give Coordinates of window manager cursor}

  PROCEDURE WGotoXY(X,Y:INTEGER); {Set coordinates of window manager cursor}

  FUNCTION WCurrentWindow:Window;
            {return Current Window - one last shown - may be NoWindow}

IMPLEMENTATION

END.
```

Appendix II Interface section for frames unit for Window Manager in UCSD p-System.

```
UNIT Frames;     {Revision 2,16-Feb-81,K.W.Currie}

INTERFACE

   VAR FrameCh:CHAR;

   PROCEDURE Frame(AtX,AtY,X,Y:INTEGER;ShowHeading:BOOLEAN);
   PROCEDURE UnFrame(AtX,AtY,X,Y:INTEGER);
   PROCEDURE InitFrame;

IMPLEMENTATION

END.
```

Paper written 18July ´81