

# The 'Window' Terminal

J. PARKER\*, A. KENNARD AND D. KING

Centre for Computing and Computer Science, The University of Birmingham, Edgbaston, Birmingham B15 2TT

*A program ROM has been developed for the BBC microcomputer, which makes it function as a powerful window management terminal, with a high resolution display, mouse and programmable keyboard. The BBC Window terminal ROM provides a cheap alternative to the more expensive windowing workstations that provide the same resolution and facilities. It also enables windowing interfaces to be attached to existing computer hardware without having to port the software to the new workstation environment – a fast and cheap method for improved man-machine interface. All window management operations are handled locally, with control information being passed between the terminal and the host computer via an RS232 interface line.*

Received May 1986, revised September 1986

## 1. INTRODUCTION

The system of windows and menus displaying information on a screen as sheets of paper has been explored in various systems in recent years, e.g. Smalltalk,<sup>1</sup> Interlisp-D<sup>2</sup> and Mesa.<sup>3</sup> Such systems provide the graphical means of sectioning data and activities according to their subject matter. Their ability to stack screen areas like sheets of paper enables a screen to present more information (by implication) than can actually be displayed on the screen, and allows the user to view information from a number of different sources at the same time. However, these systems demand expensive raster-scan graphical displays to provide the high resolution required by windowing.

The family of home computers provides suitable graphical facilities for windowing at a fraction of the cost of the larger workstations which have been traditionally used for this application. However, microcomputer-based windowing systems have tended to be limited by their inferior processing power.

By using the microcomputer as a window-management terminal attached to a more powerful host, it can behave and look like the more expensive windowing workstations, and retain the original processing power of the host.

Such an approach has been taken by the Blit terminal,<sup>4</sup> which acts as a down-loadable window manager for UNIX systems. Motivated by these developments, a similar system has been designed for the widely used BBC microcomputer, which unlike the Blit can be used with several host machines (e.g. Multics, 4.2 Berkeley UNIX, UNIX version 7, and Tops 20). It supports many different types of user and application interfaces (e.g. tiled or overlapping windows), and has proved to be highly effective in applications such as representation of multiple processes in UNIX and an Interactive Programming Support Environment.<sup>5, 6</sup>

## 2. THE WINDOW TERMINAL AND UNIVERSE

The Window terminal was specifically developed for use with the Universe Programming Environment,<sup>5, 6</sup> with

the lesser aim of providing a cheap Windowing Workstation interface for UNIX.

Universe is an attempt at providing an ideal program-development environment, incorporating ergonomic aids such as structured editing, animated debugging and methodology support. Our main aim was to make Universe simple and easy to understand and use for both novice and expert programmers alike. To do this, we needed a good physical environment to provide the right kind of switching between programming tasks, without losing context – hence windows.

The system is organised into documents, such as a program, or output from a program, which the user may look at through windows. Each window has a title line that describes its contents, and a size icon, enabling windowing positioning to be change at will.

Universe does not have a command language, a programmer simply points at the operation that is required. Work is selected by pointing to a workspace document, which contains particular Universe language procedures and definitions for one program. Program code is simply typed in (being immediately parsed by the structured editor), or inspected/edited by simply pointing to the appropriate command and code.

## 3. FUNCTIONAL CHARACTERISTICS AND PRINCIPLES

The window-management terminal is fully supported, with a high-resolution display, mouse control device and keyboard. Window-management operations are handled locally, with control information being passed between the terminal and the host computer via a V24 (RS232)\* interface line (see Fig. 1). The user interacts with the system using either a mouse, which directs the cursor movements and selects items from the menus, or the keyboard, used for entering text.

### 3.1 The window display

The high-resolution display may be thought of as a desk on which document sheets (windows) are laid out. It is

\* The window terminal actually supports an RS432 interface, which is compatible with the RS232 protocol.

\* To whom correspondence should be addressed.

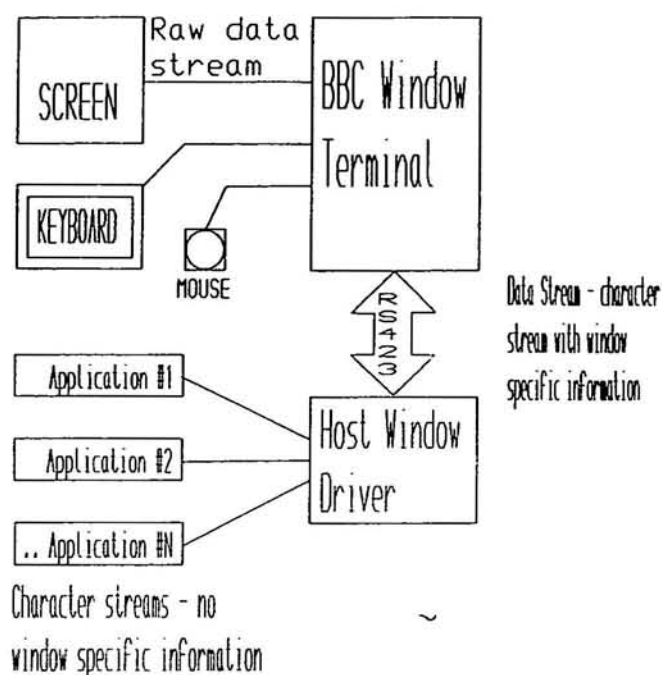


Figure 1. Communicating with the window terminal.

therefore important that corresponding windows can overlap and be active at the same time, and a specific background colour is used (black points on a white background) to strengthen the impression of a real three-dimensional desktop (Fig. 2).

Up to fifteen windows can be displayed concurrently, the most recently displayed overlaying earlier instances. Each window may be used to display output from a program running on the host. It is an important feature of the window-management system that these programs

do not need to be modified in any way to be compatible with the windowing terminal. This is because the system provides full terminal emulation of the ADM3A family of devices in each window.

Position and size of each individual window as well as overlapping hierarchy may be changed, either by program or mouse control.

### 3.2 Window types

The window terminal can be programmed to support the emulation of any windowing hierarchy, be it a tiled, overlapping or pull-down menu-type interface. Full control is obtained through the use of the window-type command (see Section 4.3).

### 3.3 The mouse

The movement of the mouse is accessible by program and may easily be converted to a corresponding movement of the cursor on the display. The mouse is therefore an ideal input device for randomly accessed data on the display.

One method of entering commands via the mouse is by positioning the cursor over an area of the screen defined, to the terminal, as a menu bar (Fig. 3.1) and pressing the mouse button. This pulls down a small rectangular area presenting a collection of commands (Fig. 3.2). If the menu button is released within the area of a command field, that command is selected and the corresponding action is started. Releasing the button outside the menu area makes the menu disappear without initiating any action.

Windows can also be defined under mouse control, by pinning one corner (pressing a mouse button) and dragging the mouse to the location of its diagonal corner (see Fig. 4).

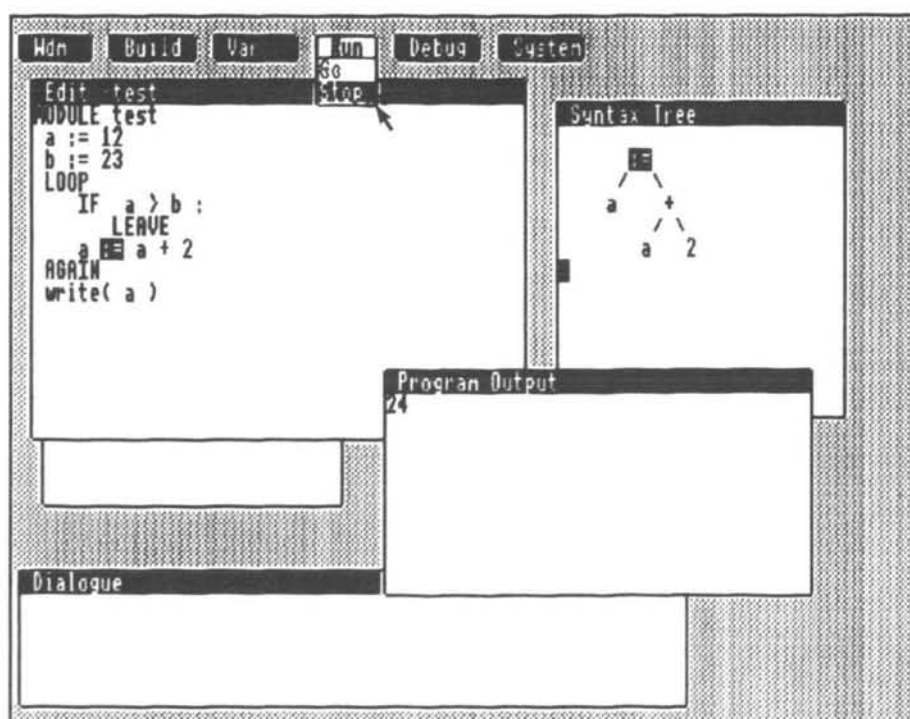


Figure 2. The desk top.

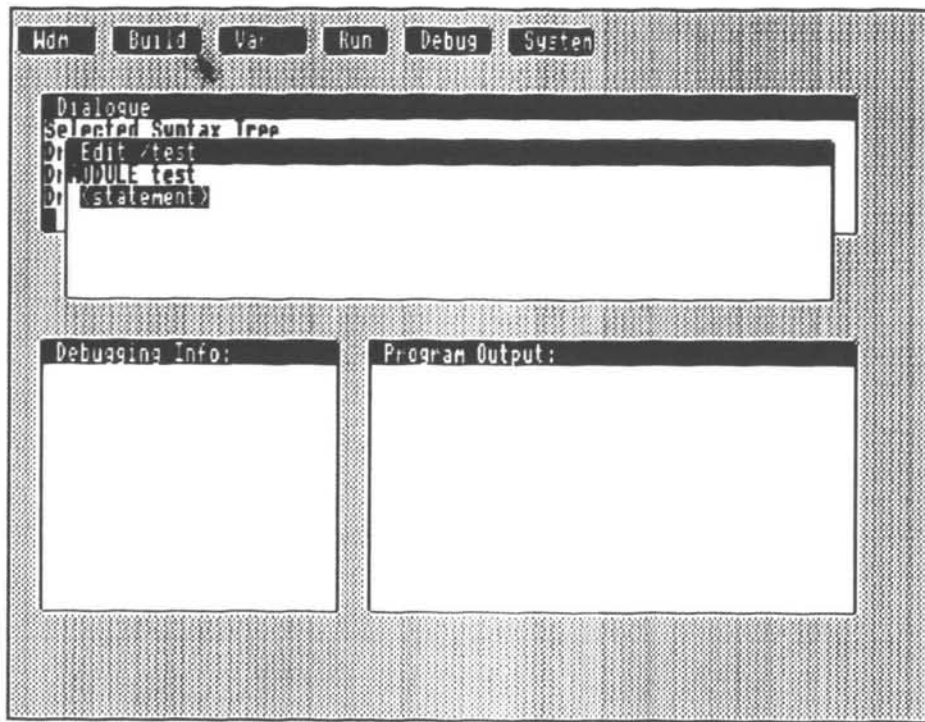


Figure 3.1. Mouse cursor positioned on pull-down menu bar.

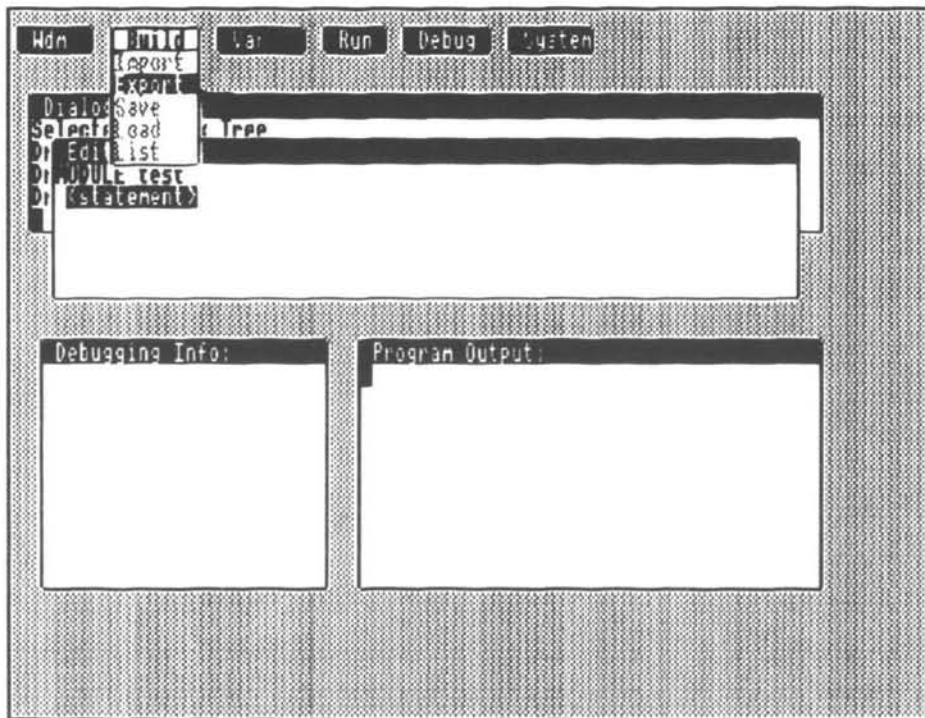


Figure 3.2. Mouse button press reveals contents of the menu.

### 3.4 Definable character fonts

Since windowing systems can simultaneously display a number of information sources, a range of character fonts may be required to represent them (mathematical equations, foreign languages, etc.). The terminal supports 96 user-definable characters, 15 user-definable display masks (reverse video, underlining, half intensity, etc.), and a definable cursor mask.

### 4. THE HOST CONTROL

The BBC microcomputer communicates, via a V24 (RS232) interface lead, with a host system which provides all the necessary computing power of a workstation. The connection provides fast two-way communication between the machines. Packets of control information synchronise the graphics display with the remote station, ensuring first that the information from the host is

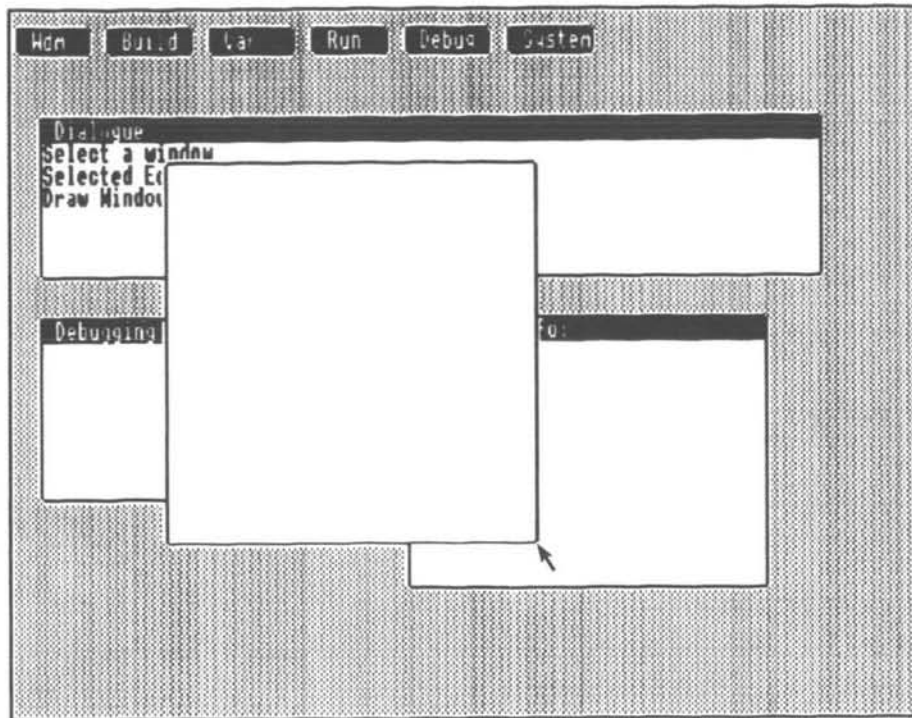


Figure 4. Dragging out a window with the mouse.

presented in the appropriate window, and secondly that the host is aware of important changes in the window display.

#### 4.1 Communicating with the terminal

The window terminal program resides in ROM located in the BBC microcomputer. The BBC can be configured to execute the program on power-up, or alternatively, the program can be invoked manually using a conventional command sequence. The program is responsible for processing the input from the keyboard, the mouse and the host processor via the V24 (RS232) interface. The program polls each device in turn.

Information flows between the host and the terminal as either bytes or packets of bytes. The packets are recognised by the host or the terminal by the first character of the packet which is the ASCII code DLE (Data Link Escape, Control-P). This is followed by the control information.

Full flow control is provided by the terminal, supporting both the XON-XOFF and CTS-RTS protocols.

Commands sent to the window terminal are automatically filtered out and executed whenever a command packet is identified in the input line stream. All remaining information is sent directly to the currently active window via the terminal emulator (which handles local control of each window, for example cursor positioning and underlining).

Host commands affect three different areas: the windows, the mouse and character font loading.

#### 4.2 Window control

Position, size and overlapping priority of each window can be defined under program control. These attributes

are set by the host sending the following command sequence down the line:

DLE <window-attribute> <window-no> <value>

This command sequence would then set one window attribute with a new value, such as the X and Y origins, window height and width, and window type (pull-down menu, top line protected, cursor on/off, etc.). Each of the fifteen windows is identified by a unique number enabling each window to have its own attribute setting.

Window-output selection control is provided by the command:

DLE N <window-no> <i>

which directs the next *i* characters to be sent to the identified window, with output returning to the default window, when complete. If <i> is zero (0) the default window is changed. For example, if window number 1 is selected as the default window (by the command 'DLE N 1 0'), and the command sequence 'DLE N 3 5' is sent. Then the next five bytes of information are sent to window number 3, with any remaining information being redirected to window 1 (the default).

#### 4.3 Setting window types

Selection of a window type is provided by the command:

DLE T <window-no> <type>

which directs the identified window to be of the defined type. <type> is made up of bit patterns, which if set defines the window to be combinations of the following types:

- Bit 0: Pull down menu
- Bit 1: Selection menu
- Bit 2: Top line protected

- Bit 3: Bottom line protected
- Bit 4: Cursor on (text cursor displayed in window)
- Bit 5: No scroll
- Bit 6: No cursor wrap round

Setting 'top/bottom line protected' (bit 2/3), enables titles or footings to be set in the defined window. When set these lines cannot be cleared or scrolled off the screen, nor can the cursor enter them. Setting 'pull down menu' (bit 0), hides the contents of the window behind the menu bar (window title line). Pressing the mouse over the menu bar reveals the contents of the window. Setting 'selection menu' (bit 1), displays the window as a pulled-down menu. In this way many different sorts of window type can be displayed, completely under host control.

#### 4.4 Mouse control

The mouse has been shown to be a more convenient input device than the keyboard for windowing systems where a high proportion of spatial inputs (for window manipulation) is required. It can be used to point to items on the screen or to pull down and select items from a menu. The terminal communicates with the host whenever a mouse event occurs using the following command format:

DLE <mouse-event> <window-no> <rel-X> <rel-Y>

where <mouse-event> is a letter defining the mouse event, such as 'M' for mouse move, 'P' for press, or 'S' for select; <window-no> is the window in which the event occurred (i.e. the window where the mouse cursor lies), and <rel-X>, <rel-Y> are the co-ordinates of the mouse cursor relative to the origin of the window.

The terminal recognises and supports up to eight different types of mouse event which can be defined by the following letters.

- A When the mouse is moved or even lightly touched after being inactive.
- M When the mouse has moved to a new character position on the display.
- P When the mouse button (any of them) is pressed.
- D As Move, except that the button is being held down.
- R When the mouse button is released.
- I When the mouse stops moving for some time.
- S When a window is selected.
- W When a window is user-defined.

Each mouse event can be enabled or disabled as required by the host control device, using the command sequence:

DLE M D <event>

to disable a specific event and

DLE M E <event>

to enable the event.

For example, the command sequence 'DLE M E P', from the host, enables mouse-button press event, which results in the command sequence being sent to the host whenever the mouse button is pressed. Pressing the mouse button in window 5, row 0, column 5, results in the command sequence 'DLE P 5 0 5' being sent to the host.

In order to limit the occurrence of too much information being generated by the mouse, with resulting

queue overflow, it is good practice for the host to enable and disable mouse events as they are required.

#### 4.5 Character font loading

The host can select an alternative character set to be displayed in each window by sending the command sequence:

ESC c

to select the alternative character set, and the sequence:

ESC d

to return to the standard character set.

Each character in the alternative set can be redefined by the host, enabling down-loadable character fonts. Individual characters in the set are specified by the command packet:

DLE D <character>

where <character> is an ASCII character, and set by defining the contents of the selected characters 8-bit by 8-bit character cell, using the following packet format:

DLE <line-no> <first-four-bits> <last-four-bits>

each packet defining a row in the 8 × 8 bit character cell, with the defined bits being sent in HEX. For example 'DLE D A' selects character-cell upper-case 'A':

	0	1	2	3	4	5	6	7
0	.	.	.	.	.	.	.	.
1	.	.	.	*	*	.	.	.
2	.	.	*	.	.	*	.	.
3	.	*	.	.	.	.	*	.
4	.	*	.	.	.	.	*	.
5	.	*	*	*	*	*	*	.
6	.	*	.	.	.	.	*	.
7	.	*	.	.	.	.	*	.

The command 'DLE 3 F F' sets line 3 to be all ones (defining a bar):

	0	1	2	3	4	5	6	7
0	.	.	.	.	.	.	.	.
1	.	.	.	*	*	.	.	.
2	.	.	*	.	.	*	.	.
3	*	*	*	*	*	*	*	*
4	.	*	.	.	.	.	*	.
5	.	*	*	*	*	*	*	.
6	.	*	.	.	.	.	*	.
7	.	*	.	.	.	.	*	.

In addition to alternative character set, the display masks for the windows (grey, inverse, underline, etc.), are programmable, allowing full user control of the display screen.

#### 5. INTERFACING THE TERMINAL WITH A HOST MACHINE

Universe interfaces to the Window terminal through a set of window primitives, which send/return the appropriate codes to the Window terminal. Mouse events are handled internally, via an event handler which simply updates global variables, sending a command to be picked up by the Universe code.

For example, window events are picked up by the input

routine 'input()', used by the universe code. This routine detects mouse events calling the mouse event handler.

```
FUNCTION input() : character;
BEGIN
  c : character;
  /* read raw input */
  read (input_line, c);
  /* filter out mouse events */
  If (c = MOUSE_EVENT) /* DLE */ THEN
    RETURN Wmgr_MouseEvent();
  ELSE
    RETURN c;
  END input();
```

The mouse event handler simply reads in the rest of the command packet and takes the appropriate action.

```
FUNCTION Wmgr_MouseEvent() : character;
BEGIN
  EVENT_TYPE, WINDOW : character;
  PosX, PosY : character;
  /* read raw input */
  read(input_line, EVENT_TYPE);
  read(input_line, WINDOW);
  read(input_line, PosX);
  read(input_line, PosY);
  /* decode event */
  CASE (EVENT_TYPE) OF
    MOUSE_BUTTON_PRESS:
      CursorX = PosX;
      CursorY = PosY;
      SelectedWindow = WINDOW;
      IF (PosX = WIconX AND PosY = WIconY)
        THEN /* cursor pressed on window
              icon */ RETURN Wmgr_DefineWindow
                (WINDOW);
      ELSE
        RETURN COMMAND_CURSOR_PRESS;
      DEF_WINDOW:
        Wmgr_Draw (WINDOW, Height (WINDOW) ,
          Width(WINDOW), PosX, PosY);
        RETURN COMMAND_WINDOW_RESIZED;
      MENU_SELECT:
        RETURN Wmgr_MenuManager(WINDOW,
          PosX);
      /*
       * other events are not enabled or
       required
       */
      OTHERS:
        RETURN input();
      END CASE;
  END Wmgr_MouseEvent;
```

## REFERENCES

1. A. Goldberg, *Smalltalk 80: The Language and Its Implementation*. Addison-Wesley, London (1980).
2. Richard R. Burton, R. M. Kaplan, L. M. Masinter, B. A. Shell, A. Bell, W. Melle, D. G. Bobrow, P. Deutsch and W. S. Haugeland, *Papers on Interlisp-D*. Technical Report CIS-5 revised, Xerox PARC (1981).
3. R. E. Sweet, The Mesa Programming Environment. *SIG-PLAN Notices* 20 (7) 1985.
4. R. Pike, *The Blit: A Multiplexed Graphics Terminal*, AT&T Bell labs. technical journal, vol. 63, no. 8, October 1984.

## 6. DISCUSSION

The Window Terminal attempts to improve upon man-machine dialogue by facilitating the display of material. Previous character-display windowing systems,<sup>7, 8, 9</sup> have relied upon heavy host control of the windows with resultant loss of performance. Our windowing system tries to do as much as possible, locally, before passing major changes to the host control device. In this way the speed of screen updating is greater than that of existing crude host-driven window systems. Furthermore, there is no need to hold copies of each window in memory on the host device.

An important feature of the system is full terminal emulation of a general family of terminals in each window (e.g. ADM3A,<sup>10</sup> VDT-1,<sup>11</sup> TV1912/920),<sup>12</sup> allowing existing software to run on the Window terminal without modification.

## 7. CONCLUSIONS

The BBC window-terminal ROM provides a cheap alternative to the more expensive windowing workstations that provide the same resolution and facilities. It also enables windowing interfaces to be attached to existing computer hardware without having to port the software to the new workstation environment – a fast and cheap method for an improved man-machine interface.

Work is currently being done to improve the UNIX interface to make use of the windowing terminal. It is hoped that the CWSH windowing system<sup>13</sup> and an interface to the SUN windowing system<sup>14</sup> can be adapted for use with this system.

The window-terminal ROM can be used on any BBC Model B, BBC+, Master series microcomputers, with 32K of memory. There is no requirement for a disc drive or other associated hardware, and in fact it makes an alternative cheap computer terminal.

Further information concerning the window terminal can be obtained from the authors at the above address.

## Acknowledgements

The authors would like to thank Mr R. J. Pannell and the technical staff in the Department of Computer Science at the University of Birmingham, for their helpful advice and assistance in the development of the window ROM.

5. J. M. Parker, A program development system for the casual user. *Proceedings of the 21st Annual ACM SIGCPR/SIGMOD Conference, Minneapolis, USA, May 1985*.
6. J. M. Parker, Human aspects of program development environments. Ph.D. thesis, University of Birmingham, UK (1986).
7. *The Maryland Window System*, Comp. Sci. Dept., Rep. TR-1271, University of Maryland, USA.
8. Ken Arnold, *Screen Updating and Cursor Movement*

- Optimisation: A Library Package*. The UNIX Programmer's Guide, volume 3, release 4.1bsd, University of California at Berkeley, October 1983.
9. R. Gammil and P. Ram, VT – a virtual terminal window package for UNIX. *Software Practice and Experience* **14**, 1197–1205 (1984).
  10. *ADM Terminal Family*, Lear Siegler Inc.
  11. TeleVideo Inc., 'Terminal 912/920 Operating Instructions', TeleVideo Inc., Sunnyvale, CA 94086, USA. TVI920.
  12. VDT-1, 106A Bedford Road, Wooton, Bedford, UK.
  13. Mark Weiser, CWSH: the windowing shell of the Maryland windowing system. *Software Practice and Experience* **15**, 515–522 (1985).
  14. Sun Microsystems Inc., *Programmer's Reference Manual for SunWindows* (1984).