SCREEN ROUTINES

Mode D'Emploi
In a BBC Micro program various 'trade-off' decisions must be made: hi-res modes use a lot of memory and support few colours; text modes use less memory, allow better colour ranges but support only medium or lo-res graphics. In this program, Mode 1 gives the necessary resolutions, but at the expense of a 20 Kbyte screen memory

Although most adventure games are text-based, some take advantage of the large memory and colourful graphics now available on home micros to create relevant screen displays. We present the first of three articles in which we design sample screens for our adventure game for the BBC Micro, Commodore 64 and the Spectrum.

In this instalment, we will consider how the graphics facilities of the BBC Micro can be used to create screen displays for adventure games. The game that we have been developing, which we have called Digitaya, is a text-based adventure game. That is to say, it uses words to describe the imaginary surroundings in which the player is placed. A text-based adventure, for example, would simply display the message 'You are in the throne room' to conjure up a setting, while a graphic adventure would attempt to draw a room with a throne.

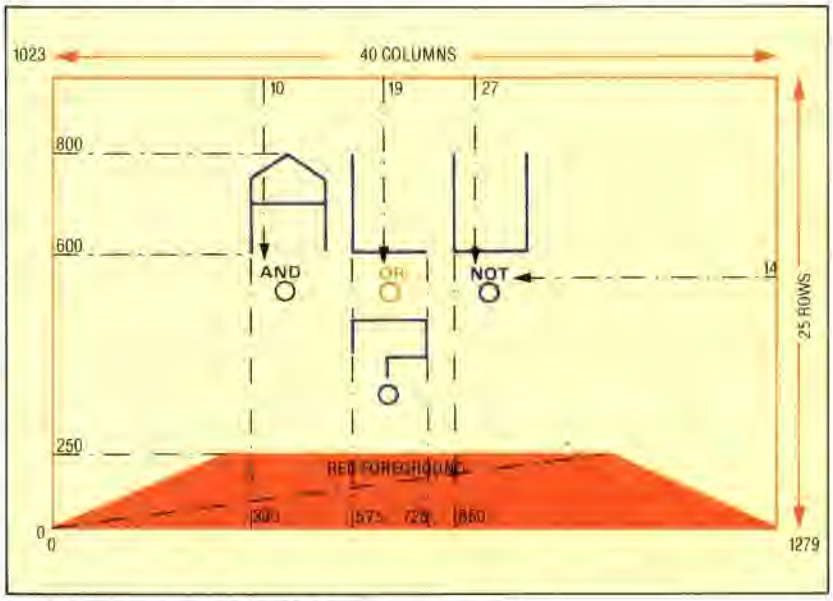
The screens that we will design here display two locations of particular interest in Digitaya: namely, the entrance to the joystick port and the Arithmetic and Logic Unit. The number of such screens is often limited by the amount of memory available; the commands required to produce each display take up memory space that would otherwise be available to increase the complexity of the plot.

ALU SCREEN DESIGN

Before we can start to design a screen for the BBC Micro, we must answer several questions:
1) How much memory do I have available?
2) How many colours do I need?
3) What standard of resolution is required?
All these questions can, in fact, be combined into one: 'What mode shall I use?'. Higher resolution and a wider range of colours mean that valuable RAM is taken up by the screen area. In our design, we shall use mode 1, which gives us four colours, a 40 by 25 screen and medium resolution. We should set the mode to be used by inserting the following line at the beginning of the program:

```
1095 MODE 1
```

Having decided on the mode, we can then sketch out what our screen is to look like, pencilling in suitable co-ordinates as we go. The design chosen here scrolls the upper-case letters A, L and U onto the screen. In the game, the player must press one of three buttons — marked AND, OR and NOT — and these must also be moved onto the display. Additional features include a thin border around the edge of the screen and a tapering foreground. Our rough design looks like this:

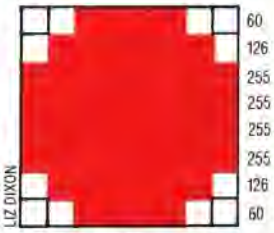


Each letter is formed by MOVEing to a start point and then using PLOT 1 to draw the shape of the letter as a series of lines relative to the start point. By designing the letters in this way they can be moved around the screen simply by changing the initial MOVE command. We can also rub out letters by redrawing the letter shape in the same position, but specifying Exclusive-OR plotting by using GCOL 3.

The buttons are formed by redefining a character. In this case, CHR$(240) is redefined by the procedure button to become the shape shown on the right. Notice that CHR$(240) is assigned to the variable button$ for use in the main part of the routine. The buttons and labels can be simply positioned by PRINTing them at co-ordinates specified by the TAB command.

The foreground is created using the triangular fill primitives provided by the PLOT 85 command. This command joins the point specified to the last two points previously plotted and then fills the resulting triangle with colour. The quadrilateral shape of the foreground can be drawn and filled by two such fill primitives.

The code for the screen display forms a subroutine of the special routine designed to deal with the ALU location in the game. The command A$=GET$, at line 7560, waits for a keypress before restoring the original foreground colour, clearing the screen and RETURNing to the main ALU routine to continue with the game. To call this graphic subroutine, the following line should also



On The Button
In the ALU picture for the BBC Micro a button shape is required to represent the three choices, AND, OR and NOT, available to the player. In the absence of a CIRCLE command or special PET-type graphics characters we must redefine an existing BBC character. Using an 8 by 8 grid we can design a shape and represent it using 8 decimal numbers. CHR$(240) can then be redefined using the VDU 23 command:

VDU 23, 240, 60, 126, 255, 255, 255, 255, 126, 60