SMOOTH MOVER

Many arcade-type computer games use a scrolling background to give a sense of rapid movement. The Commodore 64 supports 'smooth' scrolling (i.e. one pixel at a time) in both the vertical and horizontal directions. We create a routine to scroll a background design horizontally across the 64's screen.

The Commodore video controller (VIC) chip can displace the Commodore screen by up to eight pixels in either direction. Horizontal displacement is controlled by the lowest three bits of the VIC register at location 53270 ($D016). Setting these three bits to values from 7 to 0 in sequence progressively displaces the screen one pixel to the left. In BASIC, we would use the following POKE statement:

```
POKE 53270, (PEEK(53270) AND 248)+P
```

where P has a value from 0 to 7.

By combining this facility with a machine code routine that moves all screen data one cell to the left and introduces a new column of data at the right-hand edge, we can produce a smooth scrolling effect. So that data can appear to scroll smoothly into and out of view, the Commodore 64 screen width should be reduced to 38 columns, instead of the normal 40 columns. To change to 38-column mode, bit 3 of the horizontal-scrolling register should be set to zero. In BASIC, we make the following POKE:

```
POKE 53270,PEEK(53270)AND247
```

The screen can be reset to the normal 40 columns by setting bit 3 to one.

The flowchart details the various tasks that have to be carried out to produce smooth horizontal scrolling. It is important to note that if we move or insert screen data, we must also make corresponding changes to the colour data.

MOVING SCREEN DATA

In principle, this task is straightforward. The screen data is normally held in 1,000 consecutive bytes starting at location 1024 ($0400): the first 40 bytes making up the top row, the next 40 forming the second row, and so on. To make the data appear to move one place left, we simply have to move each byte of data into the byte below its original position. This section of the routine employs zero-page pointers and indirect addressing to move each byte of screen and colour data one byte lower in memory.

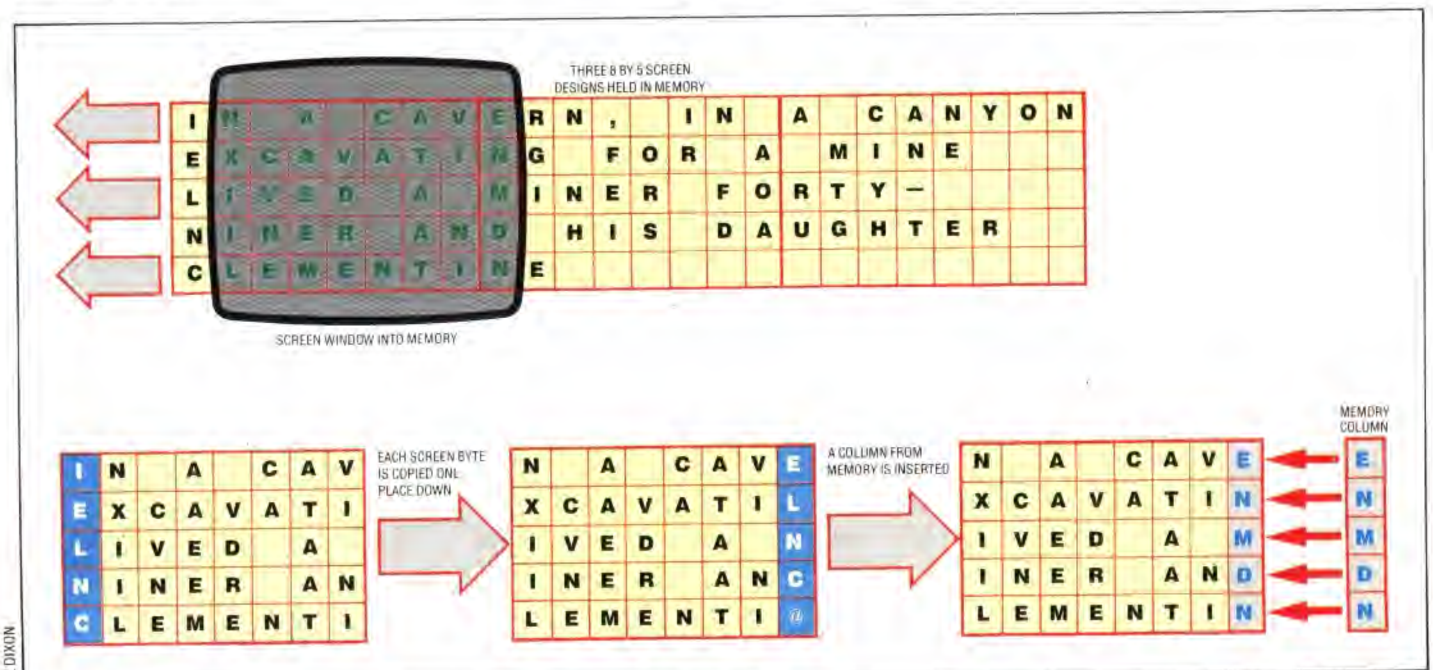
If we call the base address of the screen area SB, then the last cell in the top row will be SB+39, the last cell in the second row will be SB+79, and so on. So that the data to be scrolled onto the screen can be stored in memory in a similar way to the actual screen — that is, in packets of 1,000 bytes — the data for insertion to the right-hand edge of the screen will be the first, 41st, 81st (and so on) bytes of the area we have set side for that data. The following diagram clarifies this:

Scroll Your Own Screen

Scrolling screen designs onto the VDU from memory involves three main stages. First of all, each byte in the screen memory area is moved down one position. Because the screen is set out so that screen rows are held as sequential bytes, this has the effect of making each character on the screen appear to move one place to the left, with the exception of the characters appearing in the leftmost column of the screen. Each character in this column appears to 'wrap around' to the rightmost column. The top left screen character disappears during this process and a spurious character enters the screen area in the bottom left corner.

The second phase involves copying the relevant column from the screen memory into the rightmost column of the screen.

Having done this, the VIC chip scrolling registers can be manipulated repeatedly to make the screen appear to scroll a pixel at a time into the visible screen area



LIZ DIXON