BUGS IN THE WORKS

Having covered turtle geometry in some detail, the course moves on to look at LOGO's use of sprites. We start with a discussion of the basic principles behind turtle sprites, drawing our examples from Commodore LOGO, and show how the language can use sprites to create animation effects.

Using LOGO, sprites act in a similar way to turtles, obeying all the commands that a turtle obeys. Unlike turtles, however, we can define the shape of a sprite ourselves — although these shapes do not rotate on the screen as the sprite's heading changes, in the way that a turtle does.

In Commodore LOGO, the turtle is counted as sprite number 0, and there are seven other sprites — numbered 1 to 7. To begin with, sprite 0 is the 'current' sprite, and obeys all the sprite commands entered. To make sprite 1 the current sprite, you simply type in TELL 1. From then on, all sprite commands will be obeyed by sprite 1 until a different current sprite is specified.

After typing TELL 1, however, you won't see anything on the screen. This is because all the sprites, other than the turtle, begin as 'hidden' objects and have their pens up. In order to see sprite 1, and see where it moves, you have to type ST, and a vague square will appear on the screen. Experiment with this sprite grid, using the turtle commands FD, BK, RT, LT, PD, PU, ST, HT, and so on.

If you move sprite 1 to the same position as sprite 0 (the turtle) you will notice that it appears to be behind the turtle. In general, lower numbered sprites are shown 'in front of' higher numbered sprites. This is useful for three-dimensional effects.

There is a sprite editor on the Commodore LOGO utilities disk. Read this in by typing READ "SPRED. To edit the shape of sprite 1, first make it the current sprite with TELL 1 and then type EDSH. The display will show a much enlarged view of the sprite grid and we can now move the cursor around the screen. Pressing the asterisk key (*) will fill in a pixel, while pressing the space bar will blank it out.

Having designed your sprite, press Control-C to define the shape. If the sprite is not visible, try entering ST. This same shape can now be given to other sprites as well. SETSHAPE 1 will give the current sprite the same definition as sprite 1. Having defined a set of shapes, you can save the sprites to a file with SAVESHAPES "FILENAME, and read them back with READSHAPES "FILENAME.

There is a well-known mathematical problem in which four bugs are placed at the corners of a square. They are all set off at the same speed and each follows the bug to its right. The objective is to trace their paths. We give a LOGO program here that implements the problem using sprites.

The procedures that we give simply position a copy of the same sprite at each corner of the square, and then set them off following each other. The bug shape is defined as sprite 3, and the others are all given the same shape using SETSHAPE 3 in the position procedure.

The heart of our solution lies in the FOLLOW procedure. In this, X and Y are first set to the x and y co-ordinates of the sprite that is being followed (:B), and then the sprite that is doing the following (:A) has its heading set towards this point. To do this, we use the primitive TOWARDS. This takes two inputs, which represent the co-ordinates of the point to be headed towards, and outputs the heading from the current sprite to that point.

ANIMATION

One interesting use of sprite graphics is in the creation of animation effects. A series of sprite shapes representing the same object is defined. Each of these is slightly different from the one before, and when they are run together they give the effect of motion. Commodore LOGO gives three shapes that are a crude attempt at a man running. The following procedures set up the screen, and then set the three shapes in motion.

```
TO RUNN
   TELL 0
   DRAW
   PU
   BIGX BIGY
   SETH 90
   RUNNING 2
END

TO RUNNING :SHAPE
   FD 5
   SETSHAPE :SHAPE
   IF :SHAPE = 4 THEN MAKE "SHAPE 1
   RUNNING :SHAPE + 1
END
```

Before running these procedures, read in the file SPRITES from the utilities disk. This contains a number of useful procedures including BIGX and BIGY, which double the size of a sprite. SMALLX and SMALLY are the reverse procedures: these are used to return a sprite to its original size. Read in the three sprites by typing READSHAPES "RUNNER, and then run the procedures.

We also define four sprite shapes on the following page, which we will use in a game in the next instalment.