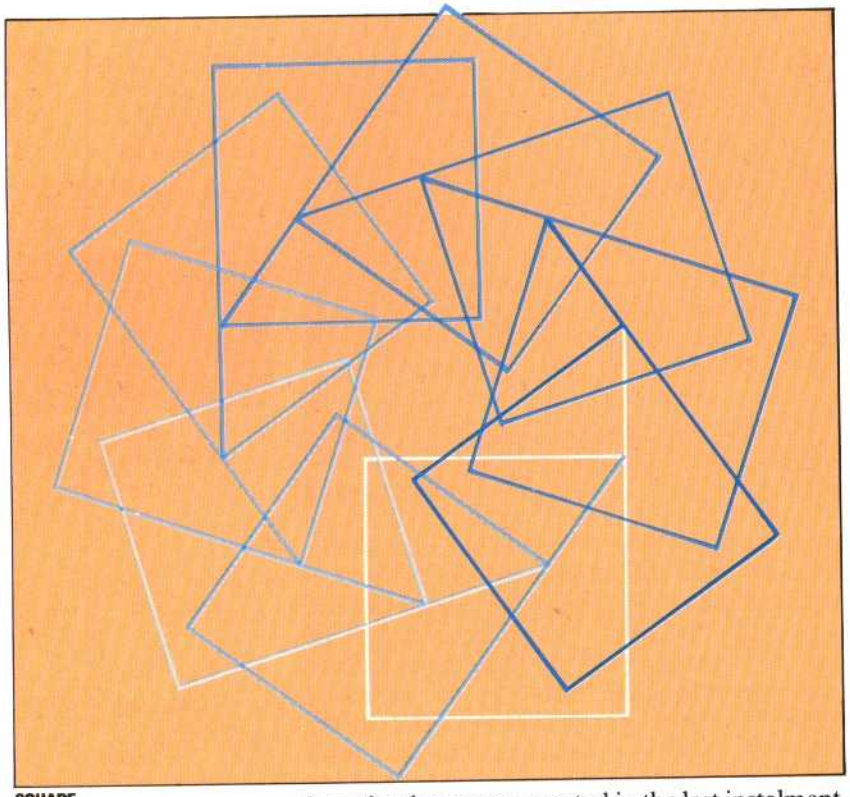


SQUARE

draw the shapes you created in the last instalment (see page 532): triangles, rectangles, pentagons, stars, etc.

Now try using your procedures with the REPEAT command. For example, define PENT as:

```
REPEAT 5 [FD 50 RT 72]
```

and then try:

```
REPEAT 10 [PENT RT 36]
```

You can experiment with other shapes in the same way, and you should soon discover that complex shapes can be built up quickly and easily by using REPEAT. Try this one:

```
REPEAT 10 [SQUARE RT 36 FD 25]
```

We show what results are given by the turtle in the diagrams on the following page. The examples we gave in the last instalment (see page 532) were designed for you to experiment with; your attempts at drawing the various shapes may well have led to some unexpected results. To draw a triangle, you may first have tried something like this:

```
REPEAT 3 [FD 50 RT 60]
```

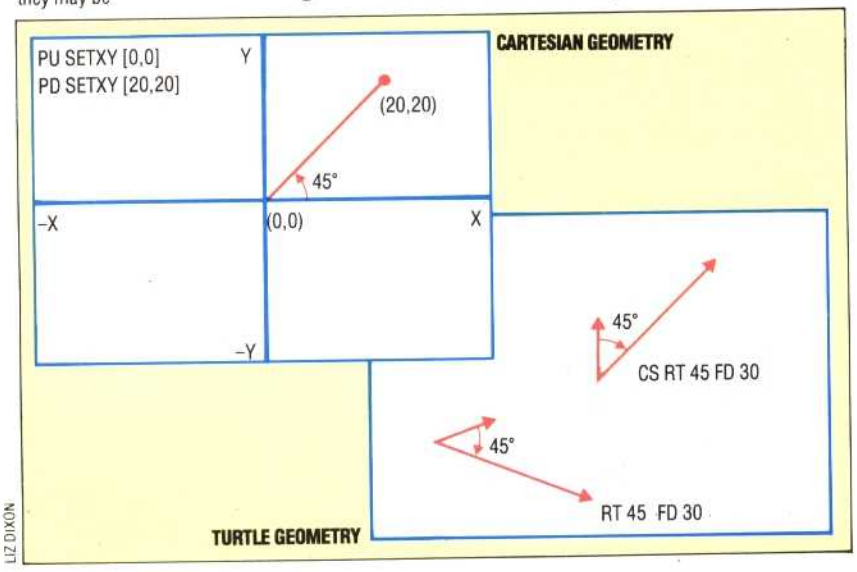
and then discovered that you have created half a hexagon! If you tried to draw the six-pointed star, you will certainly have found it much harder than you expected. When dealing with such problems, it is often helpful to 'play turtle' by imagining that *you* are the turtle moving around. Indeed, it is said that you can tell the difference between BASIC and LOGO programmers by watching their shoulders move while they program! Imagine drawing any closed shape from the turtle's point of view. The turtle has to go completely around the shape, and must end up back in the same place, facing in the original direction. So it must turn through a multiple of 360 degrees. If the shape is 'convex' (none of its internal angles is greater than 180 degrees), then the turtle will have turned through exactly 360 degrees. In the case of the triangle, there are three turns to be made — so each must be 360/3 = 120 degrees. From this, you should be able to deduce that the correct command for drawing a triangle is:

```
REPEAT 3 [FD 50 RT 120]
```

Even for non-convex polygons, such as the star shapes we have already investigated, the same principle applies — the only difference is that now the total angle turned is a whole-number multiple of 360 degrees (because you are creating more than one complete shape).

These principles are general (they don't apply simply to polygons) and they make up what is known as the 'total turtle trip theorem'. If you examine the six-pointed star from the turtle's point of view, you will see that the shape cannot be drawn by the turtle simply moving forward and turning through the same angle each time. Instead, a more complex procedure is required.

Cogito Ergo LOGO
The turtle can be addressed in terms of Cartesian or 'Turtle' geometry. In the former, turtle positions are expressed absolutely — they are measured from the imaginary X and Y axes whose origin [0,0] is the CS position; in turtle geometry, commands are expressed relative to the turtle's current position and heading, whatever they may be



LIZ DIXON

TURTLE CO-ORDINATE GEOMETRY

Turtle geometry is concerned with the properties of shapes themselves, rather than the relationship of shapes to an external point of reference, as is the case with co-ordinate geometry. Turtle geometry is relative — movements are made in a specified number of units from the turtle's current position on the screen. Co-ordinate geometry uses absolute values — the screen is imagined as a grid, with a defined number of units extending vertically and horizontally from the centre. Each point on the grid has a specific numeric value, and movement is defined in relation to these grid references. However, it is possible to use co-ordinate geometry with the turtle.

For example, the command SETXY 20 30 moves the turtle from its current position to the point (20,30). If PENDOWN has been specified, the turtle will draw a line as it moves; conversely, the PENUP