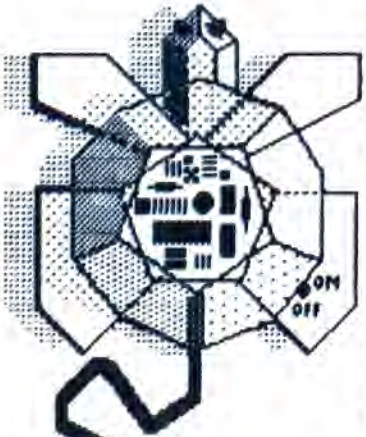
Draw Me A Turtle

One cannot progress far in LOGO without coming across recursion, something defined in terms of itself. We have seen such examples as procedures that call themselves, lists defined in terms of lists, and now procedures to write procedures. With a little imagination, it would be easy in LOGO to create a drawing in the style of MC Escher, using the turtle to generate a turtle that draws a turtle...



GROW first wipes out the original drawing (Commodore LOGO uses PENCOLOR-1 for erasing), then uses DEFINE to define the current procedure as a rewritten one. The pen colour is then returned to normal and the new shape drawn. Note that the input to GROW is stored in the variable OPLIST — which we will need to use later.

```
TO GROW :OPLIST
  PENCOLOR -1
  RUN ( LIST :CURRENT )
  DEFINE :CURRENT REWRITE.PROC TEXT
    :CURRENT
  PENCOLOR 1
  RUN ( LIST :CURRENT )
END
```

REWRITE.PROC splits the text up into lines and passes them one at a time to REWRITE.LINE:

```
TO REWRITE.PROC :TEXT
  IF EMPTY? :TEXT THEN OUTPUT []
  OUTPUT FPUT REWRITE.LINE FIRST :TEXT
    REWRITE.PROC BUTFIRST :TEXT
END
```

REWRITE.LINE checks along a line looking for a FD or FORWARD. If one is found, it passes the rest of the line over to CHANGE to deal with it.

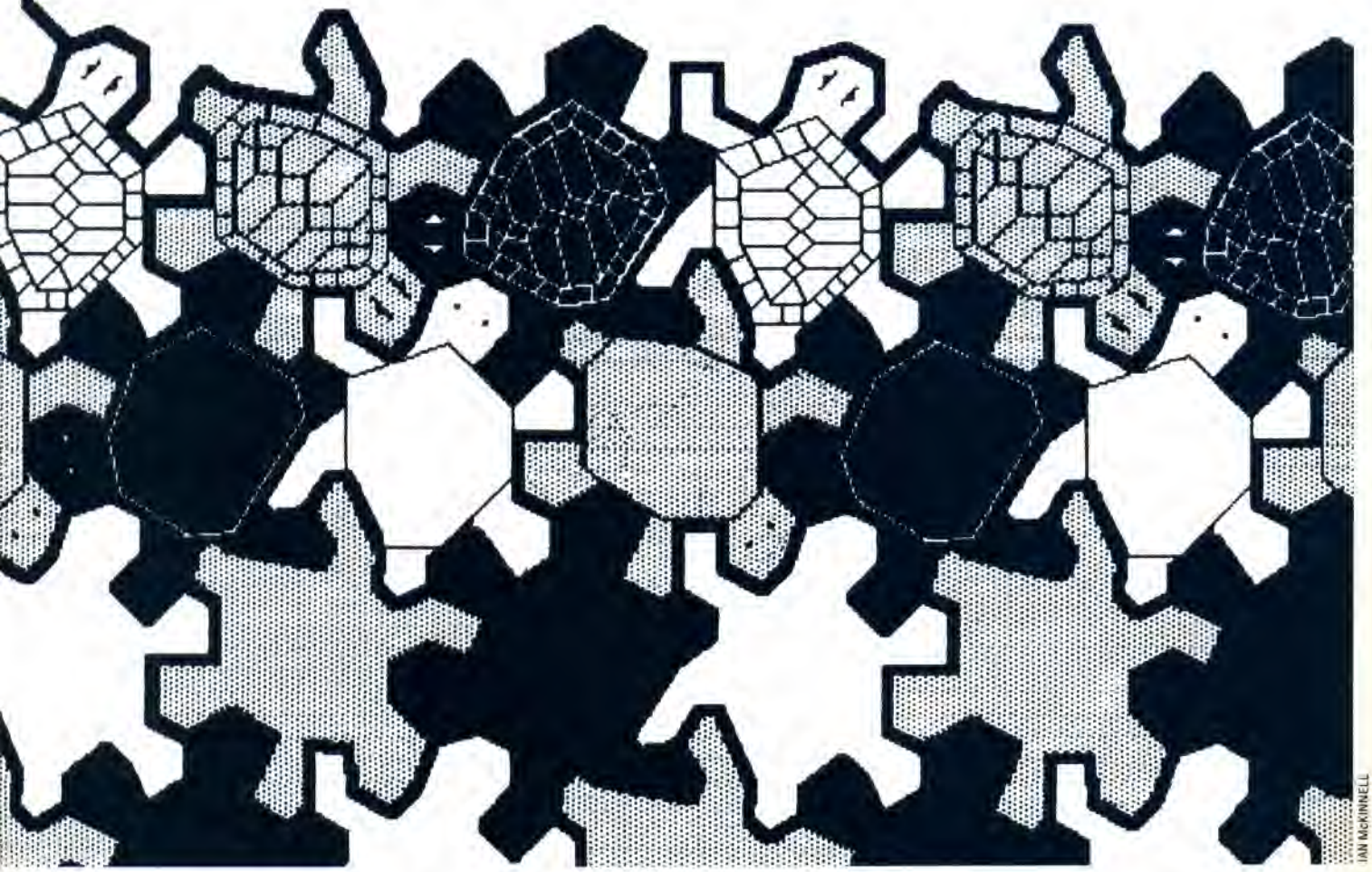
```
TO REWRITE.LINE :LINE
  IF EMPTY? :LINE THEN OUTPUT []
  IF ANYOF FIRST :LINE = "FD FIRST :LINE =
  "FORWARD THEN OUTPUT CHANGE BUTFIRST
    :LINE
  OUTPUT FPUT FIRST :LINE REWRITE.LINE
    BUTFIRST :LINE
END
```

CHANGE constructs the 'rewritten' line. The first item of LIST — the input to CHANGE — would have been the input to FORWARD in the original procedure. Say this is 50, and if OPLIST contains [* 2], then SENTENCE FIRST :LIST :OPLIST would be [50 * 2]. CHANGE now uses RUN to evaluate this list (obtaining a result of 100). Finally, a list is constructed consisting of FD, the newly evaluated quantity and then the rewrite of the rest of the line.

```
TO CHANGE :LIST
  OUTPUT ( SENTENCE "FD ( RUN SENTENCE
    FIRST :LIST :OPLIST ) REWRITE.LINE
    BUTFIRST :LIST )
END
```

COPYCAT

It is sometimes useful to be able to make a copy of a procedure. So let's define a procedure — COPYDEF — so that COPYDEF "NEWNAME "OLDNAME would define NEWNAME as a copy of OLDNAME (OLDNAME would itself remain unaffected). An



IAN McKINNELL