```
TO LOOK
   PRINTL :DESCRIPTION
   PRINT "
   PRINT [YOU CAN SEE:]
   IF EMPTY? :CONTENTS THEN PRINT [NOTHING
   SPECIAL] ELSE PRINT :CONTENTS
   PRINT "
   PRINT [YOU CAN GO:] PRINT.EXITS :EXIT.LIST
   PRINT "
   IF PERIL? THEN RUN :PERILS
END
```

RUN is a very powerful LOGO primitive. It takes a list as input and runs the procedures in that list. Here, [SNAKE] might be assigned to PERILS, so RUN :PERILS would run SNAKE.

```
TO PERIL?
   IF EMPTY? :PERILS THEN OUTPUT "FALSE
   OUTPUT "TRUE
END
```

A number of other procedures now need to be modified to take account of these perils:

```
TO ASSIGN.VARIABLES
   MAKE "ROOM.NAME WORD "ROOM. :HERE
   MAKE "ROOM THING :ROOM.NAME
   MAKE "DESCRIPTION DESCRIPTION :ROOM
   MAKE "CONTENTS CONTENTS :ROOM
   MAKE "EXIT.LIST EXIT.LIST :ROOM
   MAKE "PERILS PERILS :ROOM
END

TO PERILS :ROOM
   OUTPUT ITEM 4 :ROOM
END

TO HERE.DETAILS
   OUTPUT (LIST :DESCRIPTION :CONTENTS
   :EXIT.LIST :PERILS)
END

TO MOVE :DIR :LIST
   IF PERIL? THEN PRINT [YOU CAN'T GO THAT
   WAY] STOP
   IF EMPTY? :LIST THEN PRINT [YOU CAN'T GO
   THAT WAY] STOP
   MAKE "EXIT FIRST :LIST
   IF :DIR = FIRST :EXIT THEN MOVE1 LAST :EXIT
   STOP
   MOVE :DIR BUTFIRST :LIST
END
```

MOVE now prevents any movement until PERILS is set to [ ]. By setting up perils in this way we can use the same peril in a number of rooms, and move it from room to room by simply altering the room descriptions.

We can now use the procedures that we have developed here to build up a complete adventure game called The Shrine of Zoltoth. In this game, the adventurer is in search of the sceptre of Gilgesh, which has been stolen by the high priests of Zoltoth and taken to their temple underground. The adventurer begins the game standing at the entrance to the underground cave leading to the shrine of Zoltoth. When designing your own game, you could begin by writing out a scenario for a successful journey through the game, and structuring the game around that. We do not give the scenario for our game here, so that you can still attempt to play it if you choose.

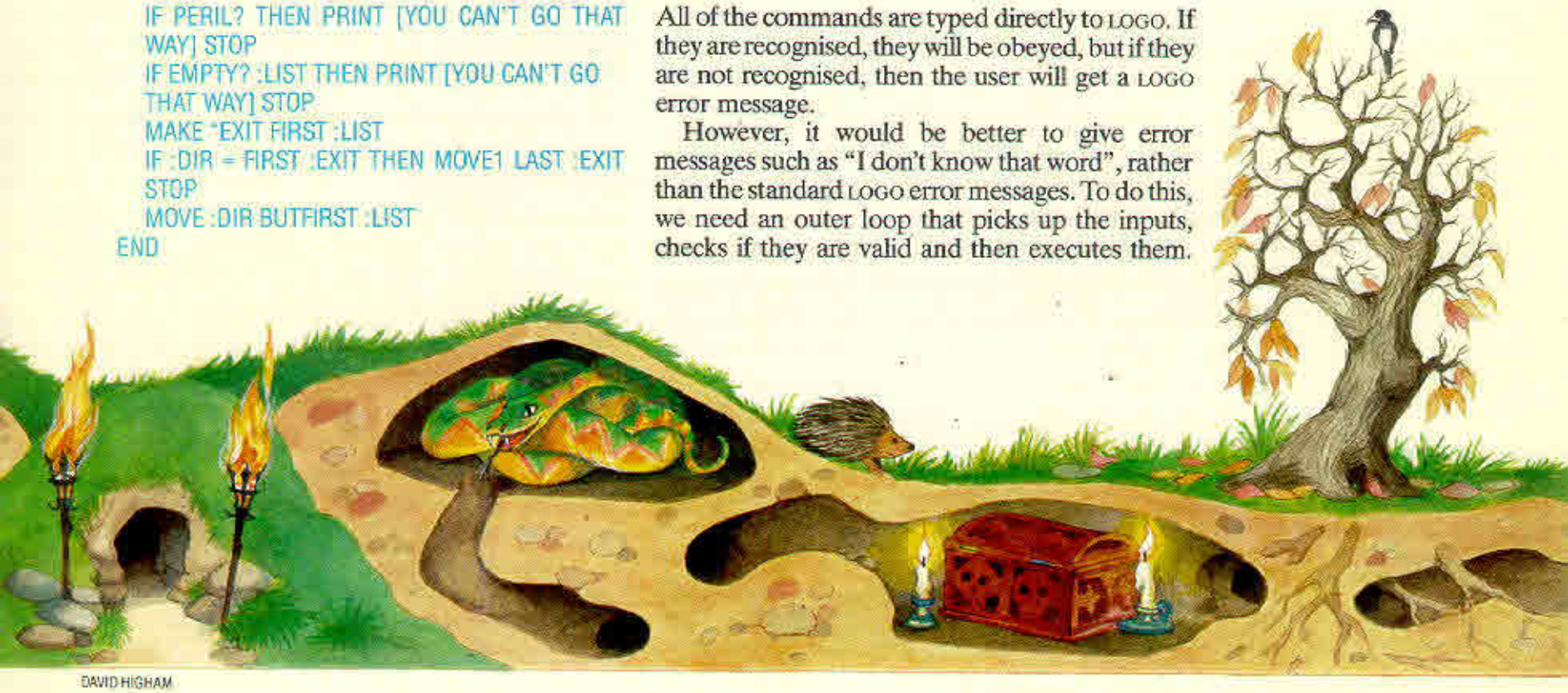
The next stage is to plan out the game in terms of 'rooms' — that is, locations within the game, their contents and positions relative to one another. This drawing of the fantasy world is then used to define the locations in the program, giving the exits allowed from each location. Adventurers will, in turn, have to build up a map as they go along.

We now need to decide on the vocabulary to be used by the game — what words from the adventurer will the program be able to understand? We will allow:

**1.** Seven single word commands: START, LOOK, N, S, E, W, and INVENTORY (these were all described in the last instalment).
**2.** Double word commands consist of a verb followed by a noun.
The verbs are: GET, DROP, EXAMINE, KILL, RUB and OPEN.
The nouns are: SWORD, CHEST, SCEPTRE, RING and SNAKE.

All of the commands are typed directly to LOGO. If they are recognised, they will be obeyed, but if they are not recognised, then the user will get a LOGO error message.

However, it would be better to give error messages such as "I don't know that word", rather than the standard LOGO error messages. To do this, we need an outer loop that picks up the inputs, checks if they are valid and then executes them.



DAVID HIGHAM