names of the objects in both the GET and DROP commands. Using quotes this way might be second nature to a LOGO programmer, but it is likely to be very confusing for an adventurer who knows nothing about the language. In order to allow the more natural GET SWORD to be used, we must define SWORD as follows:

```
TO SWORD
  OP "SWORD
END
```

Of course, we'll have to do this for each noun used in the game.

The command LOOK will print a description of the current room, a list of its contents, and the possible exit routes from the room. To do this we'll need two further lists — a description list and an exit list. In order to allow for fairly long descriptions taking up more than one line across the screen, the description list is defined as a list of lists. For example:

```
MAKE "DESCRIPTION [[YOU ARE STANDING AT THE
ENTRANCE][TO A CAVE]]
```

To keep a record of how the rooms join each other, every room is assigned a number. The exit list is simply a list of sublists, each consisting of a direction and a room number. Thus:

```
MAKE "EXIT.LIST [[N 4][E 6]]
```

We can now define LOOK:

```
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 "
END
```

Two special print routines have been used in this procedure to make the display easier to read. PRINTL is used to print several lines of text.

```
TO PRINTL :LIST
  IF EMPTY? :LIST THEN STOP
  PRINT FIRST :LIST
  PRINTL BUTFIRST :LIST
END
```

PRINT.EXITS prints the exits from the room without printing the room numbers.

```
TO PRINT.EXITS :LIST
  IF EMPTY? :LIST THEN PRINT " STOP
  MAKE "EXIT FIRST :LIST
  PRINT1 FIRST :EXIT
  PRINT1 "' '
  PRINT.EXITS BUTFIRST :LIST
END
```

We can describe everything that is known about a room in the game by putting together the three sublists: the description, the contents and the exits. For example:

```
MAKE "ROOM.1 [[[YOU ARE STANDING AT THE
ENTRANCE][TO A CAVE]] [SWORD][[N 4][E 6]]]
```

Given that ROOM.1 is defined in this way, we could split it into its individual components with the following procedure:

```
TO ASSIGN.VARIABLES
  MAKE "ROOM THING "ROOM.1
  MAKE "DESCRIPTION DESCRIPTION :ROOM
  MAKE "CONTENTS CONTENTS :ROOM
  MAKE "EXIT.LIST EXIT. LIST :ROOM
END
```

THING "ROOM.1 is an alternative to :ROOM.1; it means 'the contents of the variable ROOM.1'. We will discuss the reason for using this form shortly. The subprocedures are defined as follows:

```
TO DESCRIPTION :ROOM
  OUTPUT ITEM 1 :ROOM
END
TO CONTENTS :ROOM
  OUTPUT ITEM 2 :ROOM
END
TO EXIT.LIST :ROOM
  OUTPUT ITEM 3 :ROOM
END
```

As it stands, this procedure works for ROOM.1 only. We need to extend it so that it can be used more generally for any room. We do this by using a global variable, HERE, which contains the number of the current room. Let's say it is 2 at the moment. The LOGO primitive WORD outputs a word consisting of a combination of its two inputs (thus, WORD "ROOM. :HERE would output ROOM.1). We then assign this name to the variable ROOM.NAME — thus :ROOM.NAME is ROOM.2. We can now assign