SWORD PLAY

The list processing facilities of LOGO make it ideal for a variety of games applications. Here, we show you how the language can be used in the development of a text-based adventure game. We approach the implementation in a general way, to allow you to build up your own game by defining the locations, objects and perils yourself.

In this article, we'll restrict ourselves to looking at the more general aspects of programming an adventure game and deal with the specific details for a particular game in the next instalment.

In all adventure games there are five basic activities that the player should be able to perform: you need to pick up objects or drop them, to list the things you are carrying, to look at your surroundings and to move about the game from room to room (or location to location). So it is these basic commands that we will program first of all. For simplicity, we will restrict the form of the commands to one of two types: either single words (such as LOOK) or verb-noun pairs (such as DROP RING). The program will maintain two lists: one called INVENTORY, which will be a list of everything the player is currently carrying, and the other, called simply CONTENTS, will be a list of the objects in the current room.

The first command we will define is INVENTORY:

```
TO INV
  PRINT [YOU ARE CARRYING:]
  IF EMPTY? :INVENTORY THEN PRINT [NOTHING]
  ELSE PRINT :INVENTORY
END
```

Notice that this procedure uses the full form of the IF statement: IF ‹condition› THEN ‹action1› ELSE ‹action2›. The command for picking up an object will be GET:

```
TO GET :ITEM
  IF MEMBER? :ITEM :CONTENTS
  THEN GETIT :ITEM ELSE PRINT [I
  CAN'T IT'S NOT HERE]
END
```

MEMBER? is a primitive that tests to see if an element belongs to a list. To 'get' an item we need to do two things: add it to the inventory and remove it from the list of contents. These are the procedures that do these tasks:

```
TO GETIT :ITEM
  ADD.TO.INV :ITEM
  REMOVE.FROM.ROOM :ITEM
END

TO ADD.TO.INV :ITEM
  MAKE "INVENTORY SENTENCE :ITEM
  :INVENTORY
END

TO REMOVE.FROM.ROOM :ITEM
  MAKE "CONTENTS DELETE :ITEM
  :CONTENTS
END
```

The last of these procedures involves deleting an element from a list — which was one of the exercises given in the previous instalment.

```
TO DELETE :ITEM :LIST
  IF :ITEM = FIRST :LIST THEN OUTPUT BUTFIRST
  :LIST
  OUTPUT SENTENCE FIRST :LIST DELETE :ITEM
  BUTFIRST :LIST
END
```

The command for dropping an object is implemented in a similar way:

```
TO DROP :ITEM
  IF MEMBER? :ITEM :INVENTORY THEN DROPIT
  :ITEM ELSE PRINT [YOU DON'T HAVE IT TO
  DROP!]
END

TO DROPIT :ITEM
  REMOVE.FROM.INV :ITEM
  ADD.TO.ROOM :ITEM
END

TO REMOVE.FROM.INV :ITEM
  MAKE "INVENTORY DELETE :ITEM :INVENTORY
END

TO ADD.TO.ROOM :ITEM
  MAKE "CONTENTS FPUT :ITEM :CONTENTS
END
```

Having entered all the procedures we have given so far, it is now time to test their operation. First of all, we must define the two global variables — INVENTORY and CONTENTS — and then test for the following commands:

```
MAKE "CONTENTS [SWORD SPEAR TORCH]
MAKE "INVENTORY [LANTERN]
GET "SWORD
DROP "LANTERN
```

Now examine CONTENTS and INVENTORY using these statements:

```
PRINT :CONTENTS
PRINT :INVENTORY
```

and check that they are correct.

Notice that we used quotation marks before the