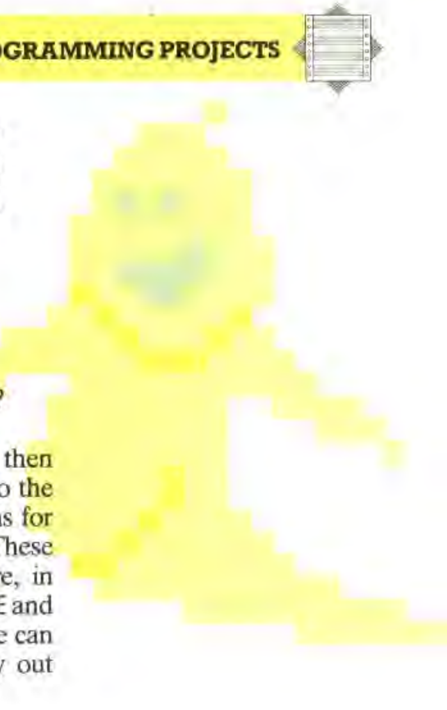
GHOST APPEARANCE



In the last part of our adventure game project, we started to look at the special locations used in the Haunted Forest game, concentrating on the decision presented to the player to enter the tunnel. Now we look at the rest of the tunnel routine and design a subroutine to produce random ghosts to haunt the forest.

In the last instalment, we discussed the special locations that have a tunnel entrance: at these locations the player is given the opportunity either to enter the tunnel or retreat back down the path that led to the entrance. If the player elects to enter the tunnel, then a new subroutine is called at line 4655. Let's now look at the subroutine that handles the option where the player goes into the tunnel. This subroutine is written according to certain rules laid down by the game's designer. To begin with, the player can pass through the tunnel only if he is carrying the lamp; and, in addition, the player must light the lamp to see the way forward.

As the player must be able to issue instructions while inside the tunnel, the subroutine should begin with a sequence that accepts an instruction input and splits this up for processing. We can allow the player to use some of the normal input instructions — such as TAKE, DROP, LIST or END — but here we must be careful. As far as the location pointer, P, is concerned, the player is still at the mouth of the tunnel and, therefore, able to GO in certain permitted directions. Consequently, we must suppress the GO instructions while we are inside the tunnel.

On returning from the 'normal commands' subroutine, if a GO command has been issued the 'move flag' (MF) will be set, and the value of P will have changed. This effect can be negated by simply restoring P to the value it had before the 'normal commands' subroutine was called.

Having handled normal commands satisfactorily, we can move on to deal with the specialised commands necessary for this particular situation. A RETREAT command can be used to allow the player to return to the tunnel entrance he came through. The only other command that we will allow is LIGHT, or a variation, USE. If the instruction issued is neither of these commands, the routine will output a catch-all I DON'T UNDERSTAND message before looping back for another instruction.

If the command is LIGHT or USE, then we have to make several checks before obeying the command:

1. Is the specified object a valid object?
2. Is the specified object held by the player?
3. Is the specified object the lamp?

If the answer to all these questions is 'yes', then the player will be allowed to pass through to the other end of the tunnel, as all the conditions for passing through the tunnel have been met. These object checks may seem familiar. They are, in fact, almost identical to those used in the TAKE and DROP routines (see page 846). Therefore, we can use previously written subroutines to carry out these checks.

```
4700 REM ** ENTER TUNNEL **
4705 SN$="YOU ENTER THE TUNNEL BUT IT IS TOO DARK
TO"
4710 SN$=SN$+" FIND YOUR WAY.":GOSUB5500
4725 PRINT:INPUT"INSTRUCTIONS";IS$
4730 GOSUB2500:REM SPLIT INSTRUCTION
4732 :
4735 IF F=0 THEN 4725:REM INVALID INSTRUCTION
4740 OP=P:GOSUB3000:REM NORMAL INSTRUCTIONS
4745 IF MF=1THEN SN$="IT IS SO DARK THAT YOU CAN O
NLY SEE":P=OP
4747 IF MF=1THENSN$=SN$+" THE TUNNEL ENTRANCE":GOS
UB5500:MF=0:GOTO4725
4750 IF VF=1 THEN 4725:REM INSTRUCTION OBEYED
4755 IF VB$="RETREAT" AND P=4 THEN MF=1:P=6:RETURN
4760 IF VB$="RETREAT" AND P=1 THEN MF=1:P=9:RETURN
4762 IFVB$<>"USE"ANDVB$<>"LIGHT"THEN SN$="I DON'T
UNDERSTAND"
4765 IFVB$<>"USE"ANDVB$<>"LIGHT"THEN GOSUB5500:GOT
O4725
4777 :
4780 REM ** SEARCH FOR LAMP **
4790 GOSUB5300:REM VALID OBJECT ?
4795 OV=F:GOSUB5450:REM IS OBJECT HELD ?
4797 IF F=0 THEN SN$="THERE IS NO "+W$:GOSUB5500:G
OTO4725
4800 IF HF=0 THEN SN$="YOU DO NOT HAVE THE "+IV$(F
,1):GOSUB5500:GOTO4725
4810 REM ** IS OBJECT LAMP ? **
4815 IF F<>2 THEN SN$="THE "+IV$(F,1)+" IS NO USE"
:GOSUB5500:GOTO4725
4835 REM ** SUCCESS **
4840 SN$="YOU USE THE LAMP TO LIGHT YOUR WAY THROU
GH THE TUNNEL"
4845 SN$=SN$+" AND EVENTUALLY EMERGE FROM THE EXIT
.":GOSUB5500
4850 IF P=1 THEN MF=1:P=4:RETURN
4855 IF P=4 THEN MF=1:P=1:RETURN
```

SUPERNATURAL EVENTS

In addition to having special locations, such as the tunnel entrances, we can also program random events or perils into our adventure game. Up to this point in the development of our Haunted Forest game we have not mentioned ghosts, nor do they appear on the adventure world map for the game (see page 766). Instead, the ghosts randomly appear to the player as he moves around the forest, and they can be fended off only by taking a bizarre form of action. Before we look in detail at the 'ghosts' routine, let's consider how we can incorporate the routines to generate random appearances into the main program structure. The main program loop calls a subroutine at line 2700 to test whether or not a