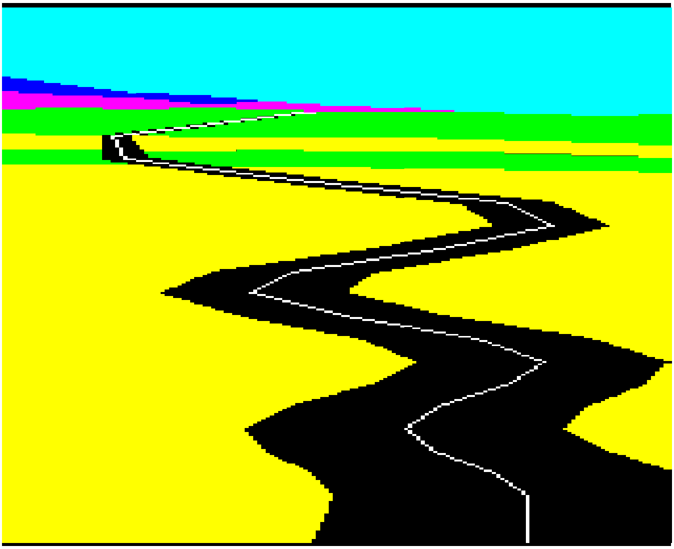
13

Get that bird!

If your kids are like the one that has the contract to hike mud into our house, they will love a certain cartoon character that travels at a great rate of knots along a road, always eluding the foxy type who is trying to catch him. In this computer game, they are trying to catch the bird. We suppose that a charge of dynamite has been placed in the road and as soon as the bird appears, the player has to slap any key to detonate the charge. The problem is of course that the bird travels very fast indeed. Actually, I have included skill levels from 0



to 9, and although anyone can win at the lowest level, quite frankly, I think that level 9 is completely impossible - which is to admit that I at least have never won at that level!

The background to the picture comprises sky and hills and contains a random element so that every game is slightly different. When you are tired of playing the game, you can try experimenting with the numeric values of line 90 in order to see the different types of terrain that will occur. What I have done is to divide the screen across into 12 sections and then, starting at the top of the picture, to create random values varying slightly as we move across. These are put into one row of the landscape array, L%(X,Y). Subsequent rows have a smaller deviation, so that the hills gently flatten into undulating land.

The road is always the same and is built around a sine wave decaying as it nears the bottom of the screen. In addition, choosing a step size of -43 (lines 210 and 250) make it impossible for the road to follow the wave closely, thus appearing asymmetrical.

Lines 280 to 310 define four separate characters. These are four round shapes of increasing size, to represent the fleeting bird as it rushes headlong down the picture. In run-time, there is no opportunity to study the bird, and the general impression of something rushing down the road towards the viewer is quite sufficient. If desired however, the last two shapes might be redesigned as a cartoon character.

The game loop is contained in the REPEAT-UNTIL loop of lines 350 to 470 and will repeat forever until the player is successful. There is an indefinite pause in line 360 and then line 370 starts a cackling sound that accompanies the bird. The *FX15 command of line 360 has flushed the input buffer so that the user cannot slap a key before this instant. Line 390 closely mirrors the road construction, so the path of the bird is down the road centre, with lines 400 to 430 choosing a character size to suit the position down the screen.

Line 440 looks for a user input (no RETURN necessary), while line 450 checks if the bird has reached the bottom of the screen. The difficulty level affects the height up the screen that represents safety for the bird, with zero equal to the very bottom. This is checked on line 460. If a key has been hit and the bird is still above the safety level, the program jumps to line 510, where an explosive sound is made and a flashing explosion appears at an appropriate spot in the road.

Of the two procedures, PROCW provides a delay of three one-hundredths of a second for the reason outlined in the Bicycle Wheel program - the image of the spot or bird must be allowed to form before being wiped. A value less than three may make the bird totally invisible, while a larger value will make the game a trifle too easy. Try it and see. PROCPAINT will draw and fill any four-sided figure by utilising two triangles. The colour is specified before the procedure is called. Notice that we have a number of variables declared as LOCAL; this is a useful provision, because not only may we use the same variable labels elsewhere, but also there is a saving in memory space, which sometimes is important.

Variables

L%(X,Y)	Lanscape heights
LVL%	Diffiuclty level
Z%	Counter for row in landscape array
X%	Position along the row; also horizontal position
F%	Field counter
C1%	Field colour
Y%	Road sections, or vertical position
K	Timer for starting run; also checking for key-pressed condition
Q%	Counter for decreasing amplitude of explosion (decay)

```
 10DIML%(7,11):*FX9,0
 20MODE7:PRINTTAB(5,5);"GET THAT BIRD!"''
 30PRINT"Difficulty level 0-9?"
 40REPEAT:LVL%=GET-48:UNTIL LVL%>=0 AND LVL%<=9
 50
 60REM - Draw background
 70
 80MODE2:FOR Z%=1TO6
 90Y%=RND(30)+(900-30*Z%):L%(Z%,1)=Y%
100FOR X%=2 TO 11:IF RND(1)>.5 GOTO120
110Y%=Y%+RND(20)/Z%
120Y%=Y%-RND(20)/Z%
130L%(Z%,X%)=Y%:NEXT:NEXT:GCOL0,134:CLG
140FORF%=1TO6:READC1%:GCOL0,C1%:PROCpaint(F%):NEXTF%
150ENVELOPE1,1,-26,-36,-45,255,255,255,127,0,0,-127,126,0
160DATA4,5,2,3,2,3
170
180REM - Draw road
190
200VDU29,600;0;18,0,5:MOVE0,820:PLOT0,0,0:GCOL0,0
210FOR Y%=820 TO 0 STEP-43
220X%=SINY%*Y%*.6
230PLOT85,(X%+820-Y%),Y%:PLOT81,-820+Y%,0
240NEXT:GCOL0,7:MOVE0,820
250FOR Y%=820 TO 0 STEP-43
260DRAWSINY%*Y%*.6+(820-Y%)*.5,Y%:NEXT
270GCOL4,0:*FX11,0
280VDU23,224,192,192,0,0,0,0,0,0
290VDU23,225,192,192,192,192,0,0,0,0
```

```
300VDU23,226,224,224,224,224,224,224,224,0
310VDU23,227,28,62,255,255,255,255,62,28
320
340
350REPEAT
360K=TIME:REPEAT:UNTIL TIME>=K+RND(1000)+300:*FX15,0
370SOUND1,1,255,255
380Y%=820:*FX12,0
390X%=SINY%*Y%*.6+(820-Y%)*.5:MOVEX%,Y%:*FX15,1
400IFY%<=200PRINTCHR$227:PROCW:PRINTCHR$227:GOTO440
410IFY%<=400PRINTCHR$226:PROCW:PRINTCHR$226:GOTO440
420IFY%<=600PRINTCHR$225:PROCW:PRINTCHR$225:GOTO440
430PRINTCHR$224:PROCW:PRINTCHR$224
440K=INKEY(0)
450Y%=Y%-43:IFY%>=0 AND K=-1 GOTO390
460IFLVL%*50<Y%GOTO510
465*FX15,0
470UNTIL 0
480
490REM - Got him!
500
510VDU29,X%+600;Y%+43;
520FOR Z%=1 TO 10:MOVE0,0
530MOVERND(200)-100,RND(200)
540PLOT81,RND(200)-100,RND(200)
550NEXT:*FX15,0
560FOR Q%=-160TO0:SOUND0,Q%/10,16,1:NEXT
570G$=INKEY$(400):MODE7
580PRINTTAB(5,5);"YOU GOT HIM!"
590G$=INKEY$(500):RESTORE:GOTO80
600
610DEFPROCW:K=INKEY(3):MOVEX%,Y%:ENDPROC
620
630DEFPROCpaint(S%):LOCALLZ%,X%,X1%,A%,B%,C%,D%
640FOR Z%=1 TO 10:X%=(Z%-1)*128:X1%=Z%*128
650A%=L%(S%+1,Z%):B%=L%(S%+1,Z%+1)
660C%=L%(S%,Z%):D%=L%(S%,Z%+1)
670MOVE X%,A%:MOVE X%,C%
680PLOT85,X1%,B%:PLOT85,X1%,D%
690NEXT
700ENDPROC
```