21 Games for the Electron

21 Games for the Electron

Mike James, S. M. Gee and Kay Ewbank

GRANADA

London Toronto Sydney New York

Granada Technical Books
Granada Publishing Limited
8 Grafton Street, London WIX 3 LA

First published in Great Britain by
Granada Publishing 1984
Reprinted 1984 (twice)

British Library Cataloguing in Publication Data
James, M.
21 games for the Electron.
1. Electronic games 2. Acorn Electron (Computers)—Programming
I. Title II. Gee, S. M. III. Ewbank, K.
794.8'028'404 GVI469.2

ISBN 0-246-12344-3

Typeset by V & M Graphics Ltd, Aylesbury, Bucks
Printed and bound in Great Britain by
Mackays of Chatham, Kent

DIGITALLY REMASTERED ON RISC OS COMPUTERS, MARCH 2012.

Contents

Introduction

This is a collection of twenty-one games written to be played on your Electron. Every game is complete in itself so you can turn to whichever one takes your fancy, type it in and play it. We've tried to include something for everyone and each one has its own detailed description so that you'll know what to expect before you embark on it. You also have a chance to see what to expect as there are samples of the displays produced on your TV screen. Of course these cannot really do justice to many of the programs which use colour graphics and we cannot find any way of letting you hear the accompanying sound effects.

All the games are written in BASIC and, presented in this form, they serve a dual purpose. Typing in games for yourself is a good way to absorb BASIC. If you are a beginner you will soon become familiar with its syntax and structure and, if you already have some experience, you will quickly pick up some handy techniques that you can incorporate into your own programs.

What's to follow

It's really impossible to indicate the range of programs included in this book as they do not fall into neat categories. Of the twenty-one programs about two-thirds can be described as moving graphics games. Some of these are variations on familiar favourites, for example Positron Invaders, Rainbow Squash and Bobsleigh. Others have titles that probably won't ring any bells – Sheepdog Trials, Commando Jump and Across the Ravine, but we hope they will soon become popular once you start to play them. Laser Attack and Mighty Missile are both very fast moving 'zap-the-enerny' type games with special features that make them very different from others we've played. Treasure Island in another program that is out of the ordinary – it is a game that tests your memory and relies on a

variety of interesting graphics techniques. Capture the Quark is aboard game in which you play against the computer on an eight-by-eight grid. There are also some programs for traditional pastimes – Magic Dice, Noughts and Crosses, Word Scramble and Mirror Tile all come into this category. Smalltalker is a rather unusual program that enables your Electron to hold a conversation with you. If you think we are joking you'll have to try it for yourself.

Improve your programming

This book isn't intended as just another collection of programs. As well as hoping to provide programs that you can have hours of fun with, we also hope to cater for all Electron owners by presenting a book that can be used in more than one way. True, you can use it simply as a source of exciting games programs, but on the otherhand you can use it to further improve your own knowledge of BBC BASIC programming. Each program is accompanied by an outline of its subroutine structure, details of special programming techniques and suggestions for further improvements. These sections are included for those of you who want to develop your ownprogramming skills. By giving away some of our trade secrets we hope that you'll be able to extend your range of techniques.

It is because we would like to be able to help you experiment with your own programming that all these games stick to BASIC. This means, of course, that the games cannot be as fast-moving, or as complicated, as the ones that are available pre-programmed on cassettes which are written in machine code. But if you want to learn to write your own programs it is far easier to start with BASIC than to attempt to come to terms with machine code.

Perfect programming

The programs included have all been extensively tested in the form in which they appear and then printed out directly from working versions. This means that if you type in exactly what is printed in this book every program should work for you every time you run it.

However, it's a well-known fact that bugs creep in whenever you

enter a program – so if a program won't work when you've typed it in, check it very carefully against the listing and if it still won't work, have a cup of tea and check again – it's all too easy to read what you think should be there rather than what is there! If there are any particular points to look out for when entering a program you'll be alerted to them in the section on *Typing Tips*. Common sources of possible errors are, confusing full stops and commas or semi-colons and colons, omitting brackets (or putting in extra ones), or misreading 'less than' and 'greater than' symbols (getting these round the wrong way will lead to chaotic results).

If you do get an error message when you try to RUN a program, don't just give up but use the information it gives you to trace your mistake. The error will not necessarily be in the line whose number is reported but that line will try to use some part of the program with the bug in it. If the line uses a variable or an array, check to see if it was defined properly. If the line identified goes to another part of the program or calls a procedure or a subroutine, make sure that section is complete. For more information about programming in BASIC, consult *The Electron Programmer*, also published by Granada.

The Electron has a number of features that you may find helpful when entering programs. The first is the automatic numbering facility that provides you with line numbers that go up in jumps of ten, which is the convention followed throughout this book. Type "AUTO" before you start typing in (or "AUTO" next line number if you want to continue a partly completed listing). Then the Electron allows you to enter keywords with a single keypress when you press the CAPS LK/FUNC key at the same time as the key with the required command on it. There is also the copying facility which makes use of the cursor control keys.

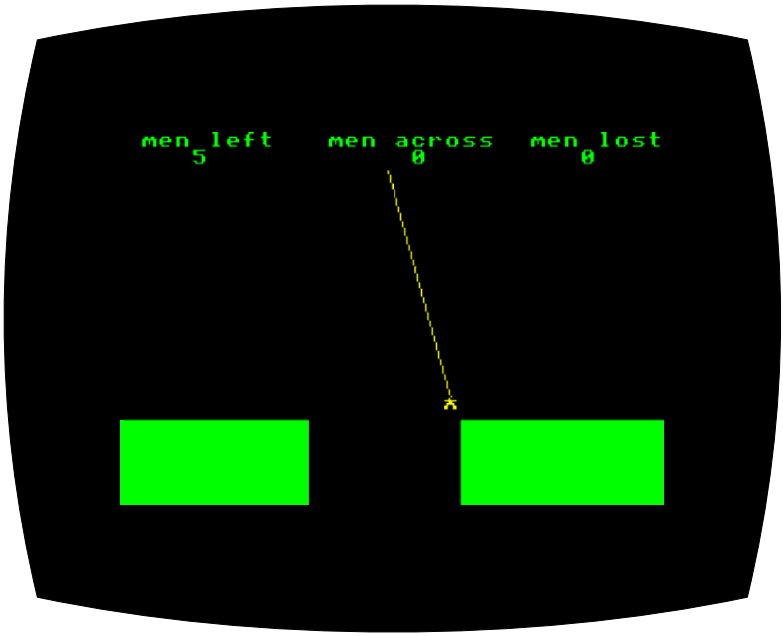
Cassette tapes

Typing in long programs can be a very frustrating business. It's noteasy to avoid typing mistakes altogether and there is always the risk that you'll lose your program before you've saved it after hours of careful effort – it's far too easy to disconnect your power supply and you can't guard against thunderstorms or other sources of voltage fluctuations without great expense! Many of the programs in this

book are indeed long. This is unavoidable as the games concerned include lots of features. However, you can avoid having to type them all in for yourselves. The programs, exactly as listed here, are available either on a pair of cassette tapes or on a 5 inch diskette. For full details and an order form send a stamped, self-addressed envelope to:

RAMSOFT
P.O. Box 6
Richmond
North Yorkshire
DL10 4HL

1
Across the Ravine



Have you got the skill and judgement needed to lead an expedition through dangerous terrain? This colour graphics game provides an easy way to discover your potential. The object of the game is to get your party of five intrepid explorers across a deep ravine with a fast flowing river at the bottom of steep cliffs. There is only one option, to swing across on a rope. The rope swings all the time so each man must run and leap to catch it − any one who mistimes his jump falls into the river and is lost. Listen out for the sound effects as a man falls towards the river!

How to play

This game is all a matter of timing − you have to judge when to start each run for the rope. It is vital to catch the rope on the downswing and each man can jump approximately his own width. You can wait as long as you like before making any run and when you are ready

press any key to jump. When your first little man has made his run and either swung to safety, or perished, the second explorer will appear in position. There are five of them in all and the length of the rope remains the same for all of them. In each new game the rope changes at random.

Typing tips

The program − like every other one in this book − has been numbered in multiples of l0. This means that rather than type in every line number you can get the Electron to provide them. Type “AUTO” before you start typing. If you do not want to start at 10, for example if you are adding to an existing listing, type “AUTO next line number.” Remember that you can save effort by using the Electron's keywords and by taking advantage of its copy facility when inputing lines that are similar to earlier ones. Use the cursor keys to move to the characters you want to copy and then press the COPY key.

In line 740 there are two sets of quotation marks with nothing in between. This is known as a null string. Notice that there is one space between the quotes in line 610.

Subroutine structure

20	Initialises arrays
40	Main play loop
110	Swings rope
400	Calculates positions of rope
470	Prints ravine
680	Plots man on end of rope
740	Man jumps routine
860	Gets next man ready
980	Man falls in water routine
1150	Sets up game
1240	End of game

Programming details

This program uses an interesting combination of low resolution and high resolution graphics. The river banks are made up of straightforward low resolution graphics − the solid blocks defined as CHR$(224). The rope is plotted in high resolution graphics and its old positions are blanked out by re-plotting them. The little man figure is a low resolution user defined graphic, CHR$(225), PRINTed in high resolution mode. This is made possible by using a VDU 5 command and means that he can appear anywhere on the screen. He therefore seems to move smoothly rather than in the jerky fashion that would result if he could only be printed at set character positions.

Another point to note is the way the co-ordinates of the positions of the swinging rope are first calculated by subroutine 400 and stored in two arrays, ‘X’ and ‘Y’. These positions are then used repeatedly in the plotting of the swinging rope. This saves having to recalculate them each time they are needed and so speeds the whole program up. This technique can be applied to any situation where anything is moving rhythmically or periodically.

Scope for alteration

You can make the game easier by increasng the value to the right of the < in line 830 or make it more difficult by decreasing this value.

Program

```
10 REM Across the Ravine

20 DIM X(32)
30 DIM Y(32)

40 GOSUB 470
50 GOSUB 400
60 GOSUB 1150
70 GOSUB 860
80 GOSUB 110

90 IF MEN=0 THEN GOTO 1240
```

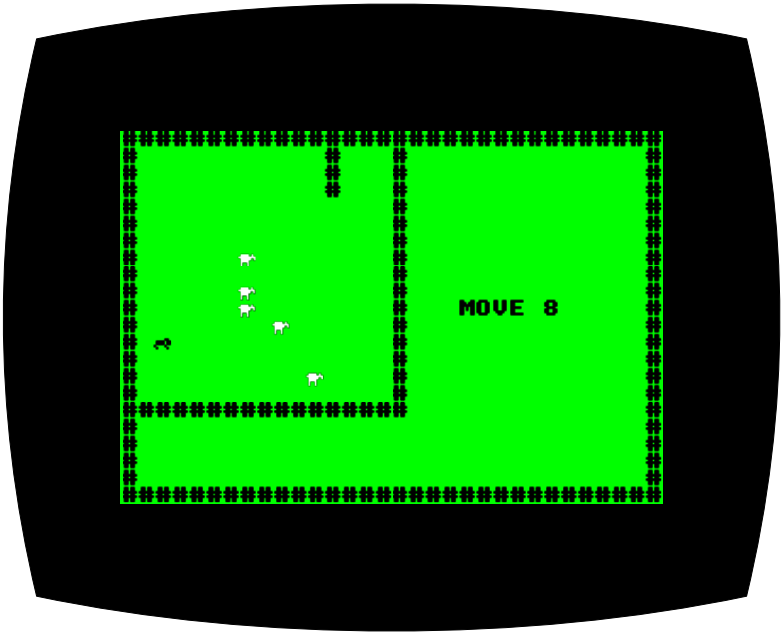
```
100 GOTO 80

110 FOR T=1+R TO N-R
120 GCOL 4,4
130 PLOT 4,500,950
140 PLOT 9,X(T),Y(T)
150 S=1:GOSUB 680
160 PLOT 4,500,950
170 PLOT 9,X(T),Y(T)
180 IF J=1 THEN S=2:GOSUB 680
190 NEXT T
200 IF C=1 THEN GOTO310
210 FOR T=N-R TO 1+R STEP -1
220 GCOL 4,4
230 PLOT 4,500,950
240 PLOT 9,X(T),Y(T)
250 S=3:GOSUB 680
260 PLOT 4,500,950
270 PLOT 9,X(T),Y(T)
280 IF J=1 THENS=4:GOSUB 680
290 NEXT T
300 RETURN
310 ACROSS=ACROSS+1
320 DX=(ACROSS-1)*50+25
330 DY=504
340 GOSUB 710
350 C=0
360 MEN=MEN-1
370 IF MEN=0 THEN GOTO 920
380 GOSUB 860
390 RETURN

400 N=0
410 FOR T=-PI/6 TO PI/6 STEP.05
420 N=N+1
430 X(N)=-450*SIN T
440 Y(N)=-450*COS T
450 NEXT T
460 RETURN

470 MODE 1
480 VDU 19,3,3,0,0,0
490 VDU 19,0,0,0,0,0
500 VDU 19,1,2,0,0,0
```

```
510 VDU 23,224,&FF,&FF,&FF,&FF,&FF,&FF,&FF,&FF
520 VDU 23,225,&18,&18,&7E,&18,&3C,&66,&66,&00
530 CLS
540 COLOUR 1
550 FOR I=17 TO 21
560 FOR J=0 TO 31
570 IF J>10 AND J<20 THEN GOTO 600
580 PRINT TAB(J,I);CHR$(224)
590 GOTO 620
600 IF I<19 THEN GOTO 620
610 PRINT TAB(J,I);" "
620 NEXT J
630 NEXT I
640 C=0
650 J=0
660 VDU 5
670 RETURN

680 IF C=0 THEN GOTO 740
690 DY=960+Y(T)
700 DX=485+X(T)
710 MOVE DX,DY
720 PRINT CHR$(225)
730 RETURN

740 IF INKEY$(0)="" AND J=0 THEN FOR Q=1 TO
    50:NEXT Q:RETURN
750 J=1
760 DY=MY
770 DX=MX
780 GOSUB 710
790 MX=MX-80
800 DY=MY
810 DX=MX
820 GOSUB710
830 IF ABS (MX-500-X(T))<50 AND S=2 THEN C=1:
    GOTO 710
840 IF MX<680 THEN GOTO 980
850 RETURN

860 MY=504
870 MX=1000
880 DY=MY
890 DX=MX
```

```
 900 J=0
 910 GOSUB 710
 920 VDU 4
 930 PRINT TAB(1,0);"men left   men across
     men lost"
 940 PRINT TAB(4,1);MEN;TAB(13);ACROSS;TAB(23);
     LOST
 950 IF MEN=0 THEN GOTO 1240
 960 VDU 5
 970 RETURN

 980 GOSUB 710
 990 MY=MY-20
1000 DY=MY
1010 GOSUB 710
1020 SOUND 1,-15,MY/8,4
1030 FOR Q=1 TO 500:NEXT Q
1040 GOSUB 710
1050 IF MY>350 THENGOTO 990
1060 SOUND 0,-15,2,6
1070 LOST=LOST+1
1080 C=0
1090 J=0
1100 MEN=MEN-1
1110 IF MEN=0 THEN GOTO 920
1120 FOR Q=1 TO 1000:NEXT Q
1130 GOSUB 860
1140 RETURN

1150 MEN=5
1160 LOST=0
1170 ACROSS=0
1180 J=0
1190 C=0
1200 R=RND(4)-1
1210 PRINT TAB(0,15);SPC(10)
1220 PRINT TAB(0,16);SPC(10)
1230 RETURN

1240 VDU 4
1250 PRINT TAB(0,25);"You lost ";LOST
1260 INPUT "Another game "A$
1270 IFLEFT$(A$,1)="Y" THEN RUN
1280 VDU 20
```

```
1290 CLS
```

2
Sheepdog Trials



If you've ever watched a shepherd and his dog coax a flock of sheep into a pen you're bound to agree that its a quite astounding feat. The experienced shepherd makes it all look so effortless as he shouts and whistles commands to his dog who obediently stands his ground, or edges up a few paces, or runs around the back of the flock to head off a straggler.

This game is an extremely realistic simulation. In fact its so true-to-life that the only person we know who's scored a 'Super Shepherd' rating was a real sheep farmer!! Five fluffy white sheep and a black dog appear in a green field surrounded by a picket fence − it creates just the right country atmosphere.

How to play

The object of the game is to herd all five sheep into the pen at the top right hand side of their field in the minimum number of moves. To do

this you have to control the dog using the arrow keys. If the dog approaches too close to the sheep they will scatter (they may also scatter randomly during the course of the game just to complicate matters). In normal play neither the dog, nor the sheep, are allowed to cross any fences, although when they scatter the sheep may jump out of the pen. There will always be a total of five sheep but if they crowd very close together they will appear to merge into one another.

Once you've played this game a few times you'll realise that some strategies for controlling the sheep work better than others. Beginners tend to waste moves trying to manoeuvre the dog around the back of the flock. However, to achieve the title of 'Super Shepherd' or 'Good Dog' you'll need to make every move count.

Typing tips

The hash character is used to print the fence in subroutine 120. It is produced by typing the 3 key with the SHIFT key held down. The only other printing feature to look out for is the single space enclosed in double quotes in lines 440 and 780. These are used to blank out the previous positions of the dog and the sheep respectively.

Subroutine structure

20	Sets up graphics characters and arrays
120	Prints fences
270	Prints sheep
330	Prints dog
370	Moves dog
520	Moves sheep and checks for end of game
550	Move logic for sheep
780	Prints sheep
840	Prints messages and end of game
990	Scatters sheep

Programming details

Line 20 changes the cursor keys so that they return ASCII codes, thereby allowing them to be used as arrow keys in this game. At the end of the game their normal function is restored in line 960. The other *FX command used in this program (*FX 15,1 in line 380) has the effect of clearing the Electron's type ahead buffer − in other words it gets rid of any accumulation of key presses that have not already been acted upon.

When you RUN this game you may imagine that there are some special techniques involved to govern the movement of the sheep and sheepdog. This is, however, not the case and the program depends entirely on calculating their positions relative to each other according to mathematical equations. For example, lines 600 and 610 check to see if the dog has approached too close to the sheep. If he has (or if the random number generated is less than .01, a one-in-a-hundred chance occurence) then the sheep scatter according to the equations in 1000 and 1030. IF statements are also used to make sure that the dog does not move into any of the picket fences or that the sheep do not move on to the dog or on to the fences.

Scope for improvement

If you get really proficient at this game you can try to make it more difficult. You might increase the chance of the sheep scattering at random, by altering the value of the cut-off point for the random number in line 610 or you could add some obstacles such as a pond or a river that the sheep had to avoid or cross. Another suggestion is to modify the game to employ a time criterion, using the Electron's timer, instead of counting the number of moves needed.

Program

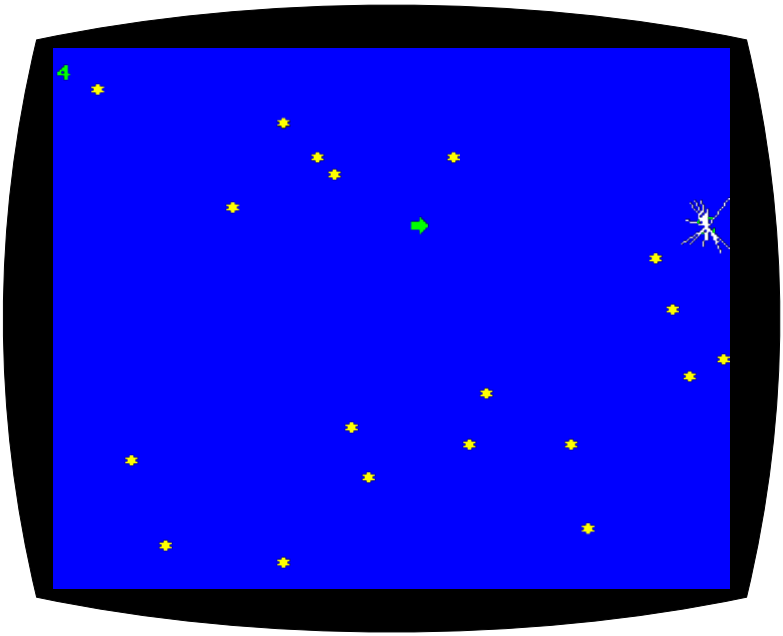
```
10 REM Sheepdog Trial

20 *FX 4,1
30 MODE 1
40 VDU 23,224,&00,&00,&7A,&FF,&7D,&78,&48,&48
```

```
 50 VDU 23,225,&00,&00,&06,&7B,&7B,&84,&42,&00
 60 VDU 19,0,2,0,0,0
 70 VDU 19,3,0,0,0,0
 80 VDU 19,1,7,0,0,0
 90 DIM Y(5)
100 DIM X(5)
110 M=0

120 FOR X=0 TO 15
130 PRINT TAB(X,16);"#"
140 NEXT
150 FOR Y=0 TO 16
160 PRINT TAB(16,Y);"#"
170 NEXT
180 FOR Y=0 TO 20
190 PRINT TAB(0,Y);"#";TAB(31,Y);"#"
200 NEXT
210 FOR X=0 TO 31
220 PRINT TAB(X,0);"#";TAB(X,21);"#"
230 NEXT
240 FOR Y=1 TO 3
250 PRINT TAB(12,Y);"#"
260 NEXT

270 COLOUR 1
280 FOR S=1 TO 5
290 Y(S)=5+RND(10)
300 X(S)=4+RND(6)
310 PRINT TAB(X(S),Y(S));CHR$(224)
320 NEXT S

330 COLOUR 3
340 YD=1+RND(3)
350 XD=1+RND(3)
360 PRINT TAB(XD,YD);CHR$(225)

370 COLOUR 3
380 *FX 15,1
390 D=INKEY(0)
400 IF D=-1 THEN GOTO 390
410 IF XD=12 AND YD=4 AND D=&8B THEN GOTO 390
420 IF XD=11 AND YD<4 AND D=&89 THEN GOTO 390
430 IF XD=13 AND YD<4 AND D=&88 THEN GOTO 390
440 PRINT TAB(XD,YD);" "
```

```
450 IF D=&88 AND XD>1 THEN XD=XD-1
460 IF D=&89 AND XD<15 THEN XD=XD+1
470 IF D=&8A AND YD<15 THEN YD=YD+1
480 IF D=&8B  AND YD>1 THEN YD=YD-1
490 PRINT TAB(XD,YD);CHR$(225)
500 M=M+1
510 PRINT TAB(20,10);"MOVE ";M

520 GOSUB 550
530 IF F=0 THEN GOTO 840
540 GOTO 370

550 F=0
560 COLOUR 1
570 FOR S=1 TO 5
580 Y=Y(S)
590 X=X(S)
600 IF (ABS(X(S)-XD)<2 AND ABS(Y(S)-YD)<2) THEN
    GOSUB 990
610 IF RND(1)<.01 THEN GOSUB 990
620 IF ABS(X(S)-XD)>2+RND(2) THEN GOTO 780
630 IF ABS(Y(S)-YD)>2+RND(2) THEN GOTO 780
640 X(S)=X(S)+SGN(X(S)-XD)
650 Y(S)=Y(S)+SGN(Y(S)-YD)
660 IF X(S)<13 AND X(S)>11 AND Y(S)<4
    THEN X(S)=X
670 FS=0
680 FOR Z=1 TO 5
690 IF Z=S THEN GOTO 710
700 IF (X(S)=X(Z)) AND (Y(S)=Y(Z)) THEN FS=1
710 NEXT Z
720 IF FS=1 THEN GOTO 640
730 IF X(S)=XD AND Y(S)=YD THEN GOTO 640
740 IF X(S)<1 THEN X(S)=1
750 IF X(S)>15 THEN X(S)=15
760 IF Y(S)<1 THEN Y(S)=1
770 IF Y(S)>15 THEN Y(S)=15
780 PRINT TAB(X,Y);" "
790 PRINT TAB(X(S),Y(S));CHR$(224)
800 IF X(S)>12 AND (Y(S)>0 AND Y(S)<4) THEN
    GOTO 820
810 F=1
820 NEXT S
830 RETURN
```

```
 840 REM ENDGAME
 850 COLOUR 1
 860 PRINT TAB(2,22);
 870 IF M<40 THEN PRINT "SUPER SHEPHERD!!":
     GOTO 920
 880 IF M<60 THEN PRINT "GOOD DOG!!":
     GOTO 920
 890 IF M<90 THEN PRINT "KEEP PRACTISING":
     GOTO 920
 900 IF M<120 THEN PRINT "BETTER LUCK NEXT TIME":
     GOTO 920
 910 PRINT "HAND IN YOUR CROOK !!"
 920 PRINT TAB(2,23);"YOU TOOK  ";M;" MOVES"
 930 INPUT "ANOTHER GAME Y/N ",A$
 940 A$=LEFT$(A$,1)
 950 IF A$="Y" THEN RUN
 960 *FX 4,0
 970 VDU 20
 980 END

 990 XT=X(S):YT=Y(S)
1000 X(S)=X(S)+(SGN(RND(1)-.5)*(2+RND(2)))
1010 IF X(S)<1 THEN X(S)=1
1020 IF X(S)>15 THEN X(S)=15
1030 Y(S)=Y(S)+(SGN(RND(1)-.5)*(2+RND(2)))
1040 IF Y(S)<1 THEN Y(S)=1
1050 IF Y(S)>15 THEN Y(S)=15
1060 IF X(S)=12 AND Y(S)<4 THEN GOTO 1000
1070 IF XT=X(S) AND YT=Y(S) THEN GOTO 1000
1080 RETURN
```

3
Laser Attack



This is a very exciting and fast moving space fight game that uses sound effects and some rather unusual graphics techniques to good effect. The screen is treated as if it were a spherical universe. So, if you go off the top of the screen you re-appear at the bottom, and if you go off at the right, you re-appear at the left. This game is a race against the clock. You have a hundred seconds in which to annihilate the enemy ship with your infallible laser weapon − the chase is on.

How to play

At the beginning of this game you have to select a difficulty factor. This governs the unpredictability of the enemy ship's course and the number of stars that appear. The stars act as obstacles in this game. If you hit one you will be deflected at random so the fewer there are the easier it is to steer your course. Your ship moves continuously and is shaped like an arrowhead and can point in any of eight directions.

Every time you press any key it turns clockwise through 45 degrees. The enemy is a revolving cartwheel-shaped disc that meanders through space. To fire your laser press the 'up arrow' key. Your weapon will fire in a straight line from the point of your arrow. If you hit the enemy ship it will disintegrate with appropriate sound and visual effects. The time taken is constantly displayed at the top left of the screen and when it reaches 100 your time is up.

Typing tips

You may find the copy key useful when entering some sections of this program where there are many similar lines.

Subroutine structure

20	Initialisation
50	Main play loop
120	Fires or rotates direction of arrow
200	Laser zap and hit logic
560	Moves arrow
700	Moves target
850	Title frame
950	Gets and prints time
990	Prints stars
1030	Initialises variables and defines graphics
1890	End of game

Programming details

This is a complicated program and one that illustrates a number of novel programming methods. Notice, for example, the way the revolving cartwheel is produced by using two user-defined graphics characters, one a version of the other rotated through 90 degrees,

which are printed alternately. The way that eight versions of the arrow graphic are used in order to allow it to be moved in any of eight different directions is another interesting technique. While the player's ship, the target and the stars (asterisks) are all PRINTed in low resolution graphics, the laser zap and the explosion are PLOTted in high resolution graphics. The way the direction of movement and velocity is set in PROCINIT using the arrays W and V is also worth attention.

Scope for improvement

If you feel very ambitious you could make this game even more exciting by enabling the enemy ship to fire at random − so that you have to dodge its fire at the same time as pursuing it.

Program

```
  10 REM Laser attack

  20 MODE 1
  30 PROCTITLE
  40 PROCINIT

  50 PROCTIME
  60 IF INT(T)>1000 THEN GOTO 1890
  70 PROCMOVE
  80 PROCSHIP
  90 IF H=1 THEN GOTO 1890
 100 PROCENEMY
 110 GOTO 50

 120 DEF PROCMOVE
 130 A$=INKEY$(0)
 140 *FX 15,1
 150 IF A$="" THEN ENDPROC
 160 IF A$="^" THEN GOTO 200
 170 K=K+1
 180 IF K>8 THEN K=1
 190 ENDPROC
```

```
200 XL=X*32+16
210 YL=1023-Y*32-16
220 GOSUB 490
230 MOVE XL,YL
240 DX=0
250 IF V(K)=1 THEN DX=1279-XL
260 IF V(K)=-1 THEN DX=-XL
270 DY=0
280 IF W(K)=1 THEN DY=-YL
290 IF W(K)=-1 THEN DY=1023-YL
300 IF V(K)*W(K)=0 THEN GOTO 330
310 IF ABS(DX)<ABS(DY) THEN DY=ABS(DX)*SGN(DY)
    :GOTO 330
320 DX=ABS(DY)*SGN(DX)
330 PLOT 1,DX,DY
340 MOVE XL,YL
350 SOUND &11,1,60,5
360 PLOT 2,DX,DY
370 IF H=0 THEN ENDPROC
380 MX=B*32+16: MY=1023-A*32-16
390 FOR I=1 TO RND(5)+20
400 DX=50-RND(100)
410 IF MX+DX>1279 OR MX+DX<0 THEN GOTO 470
420 DY=50-RND(100)
430 IF MY+DY>1023 OR MY+DY<0  THEN GOTO 470
440 MOVE MX,MY
450 PLOT 1,DX,DY
460 SOUND 0,-15,5,4
470 NEXT I
480 ENDPROC
490 H=0
500 DY=A-Y
510 DX=B-X
520 IF W(K)*DX<>V(K)*DY THEN RETURN
530 IF ABS(V(K))*SGN(DX)<>V(K) OR ABS(W(K))
    *SGN(DY)<>W(K) THEN RETURN
540 H=1
550 RETURN

560 DEF PROCSHIP
570 IF NB1=0 THEN PRINT TAB(X,Y);" "
580 X=X+V(K)
590 Y=Y+W(K)
600 IF X<0 THEN X=39
```

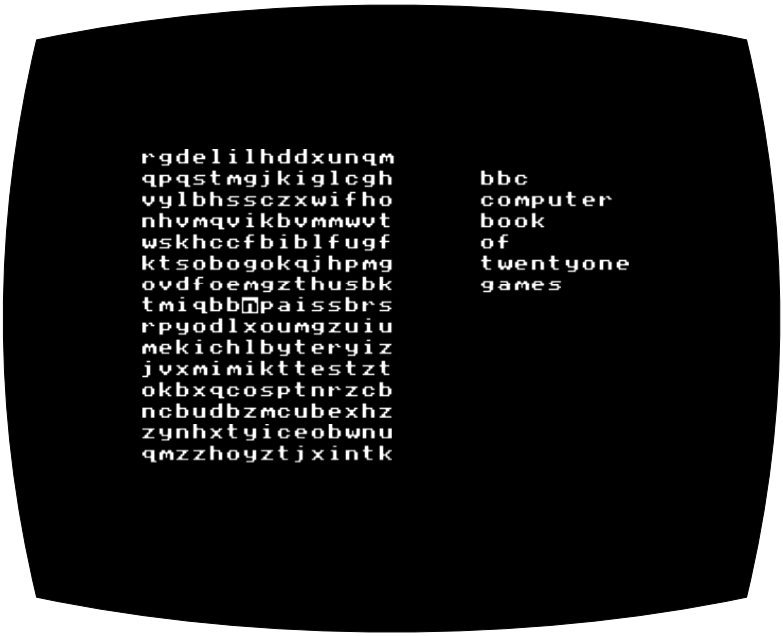
```
610 IF X>39 THEN X=0
620 IF Y<0 THEN Y=29
630 IF Y>29 THEN Y=0
640 IF POINT(X*32+16,1023-Y*32-16)<>0 THEN
    SOUND 1,-15,0,5:GOTO 680
650 PRINT TAB(X,Y);M$(K)
660 NB1=0
670 ENDPROC
680 NB1=1:K=K+1:IF K>8 THEN K=1
690 ENDPROC

700 DEF PROCENEMY
710 IF RND(1)>1.05-DF/20 THEN Z=Z+1
720 IF Z>8 THEN Z=1
730 IF NB2=0 THEN PRINT TAB(B,A);" "
740 A=A+V(Z)
750 B=B+W(Z)
760 IF B<0 THEN B=39
770 IF B>39 THEN B=0
780 IF A<0 THEN A=29
790 IF A>29 THEN A=0
800 IF POINT(B*32+16,1023-A*32-16)<>0 THEN
    SOUND 1,-15,0,5:NB2=1:Z=Z+1:GOTO 720
810 PRINT TAB(B,A);W$(R+1)
820 R=NOT R
830 NB2=0
840 ENDPROC

850 DEF PROCTITLE
860 PRINT TAB(8,2);"L a s e r   A t t a c k"
870 PRINT TAB(0,8);"You are in control of
    an advanced laser"
880 PRINT "attack ship in pursuit of an enemy
    craft"
890 PRINT '"Shoot it down before your time is
    up !!!"
900 PRINT TAB(0,20);
910 INPUT "Select the difficulty level
    1 (easy) "'"to 10 (difficult) ",DF
920 IF DF<1 OR DF>10 THEN GOTO 910
930 CLS
940 ENDPROC

950 DEF PROCTIME
```

```
 960 T=TIME/10
 970 PRINT TAB(0,1);INT(T/10);"  "
 980 ENDPROC

 990 DEF PROCSTAR
1000 IF RND(1)>.1+DF/50 THEN ENDPROC
1010 PRINT TAB(RND(39),RND(30));"*"
1020 ENDPROC

1030 DEF PROCINIT
1040 DIM W(8),V(8)
1050 DIM A(8),B(8),C(8),D(8),E(8),F(8)
1060 DIM W$(2),M$(8)
1070 DIM S$(8),R$(8)
1080 VDU 19,0,4,0,0,0
1090 VDU 19,1,3,0,0,0
1100 VDU 19,2,2,0,0,0
1110 VDU 19,3,7,0,0,0
1120 COLOUR 1:COLOUR 128
1130 DATA 0,1,1,1,1,0,1,-1,0,-1,-1,-1,-1,0,-1,1
1140 FOR I=1 TO 8
1150 READ W(I),V(I)
1160 NEXT I
1170 VDU 23;8202;0;0;0;
1180 K=1
1190 X=20
1200 Y=10
1210 V=V(K)
1220 W=W(K)
1230 S$(1)="00011000"
1240 S$(2)="00111100"
1250 S$(3)="01111110"
1260 S$(4)="11111111"
1270 S$(5)="00111100"
1280 S$(6)="00111100"
1290 S$(7)="00111100"
1300 S$(8)="00111100"
1310 R$(1)="11111000"
1320 R$(2)="11110000"
1330 R$(3)="11111000"
1340 R$(4)="11111100"
1350 R$(5)="10111110"
1360 R$(6)="00011111"
1370 R$(7)="00001110"
```

```
1380 R$(8)="00000100"
1390 FOR I=1 TO 8
1400 FOR J=1 TO 8
1410 A(I)=A(I)*2+VAL(MID$(S$(I),J,1))
1420 B(I)=B(I)*2+VAL(MID$(S$(9-J),I,1))
1430 C(I)=C(I)*2+VAL(MID$(S$(J),I,1))
1440 D(I)=D(I)*2+VAL(MID$(R$(I),J,1))
1450 E(I)=E(I)*2+VAL(MID$(R$(9-J),I,1))
1460 F(I)=F(I)*2+VAL(MID$(R$(9-I),9-J,1))
1470 PROCSTAR
1480 NEXT J
1490 NEXT I
1500 VDU 23,224
1510 FOR I=1 TO 8:VDU A(I):NEXT I
1520 VDU 23,225
1530 FOR I=1 TO 8:VDU A(9-I):NEXT I
1540 VDU 23,226
1550 FOR I=1 TO 8:VDU B(I):NEXT I
1560 VDU 23,227
1570 FOR I=1 TO 8:VDU C(I):NEXT I
1580 VDU 23,228
1590 FOR I=1 TO 8:VDU D(I):NEXT I
1600 VDU 23,229
1610 FOR I=1 TO 8:VDU D(9-I):NEXT I
1620 VDU 23,230
1630 FOR I=1 TO 8:VDU E(I):NEXT I
1640 VDU 23,231
1650 FOR I=1 TO 8:VDU F(I):NEXT I
1660 M$(1)=CHR$(226)
1670 M$(2)=CHR$(231)
1680 M$(3)=CHR$(225)
1690 M$(4)=CHR$(229)
1700 M$(5)=CHR$(227)
1710 M$(6)=CHR$(228)
1720 M$(7)=CHR$(224)
1730 M$(8)=CHR$(230)
1740 A=10:B=10
1750 Z=RND(8)
1760 VDU 23,232,&20,&42,&25,&18,&18,&A4,&42,&04
1770 VDU 23,233,&0E,&88,&88,&F8,&1F,&09,&09,&38
1780 W$(0)=CHR$(232)
1790 W$(1)=CHR$(233)
1800 R=0
1810 NB1=0
```

```
1820 NB2=0
1830 TIME=0
1840 H=0
1850 COLOUR 2
1860 GCOL 0,3
1870 ENVELOPE 1,1,100,0,-10,100,50,100,126,0,
     0,-126,126,126
1880 ENDPROC

1890 IF H<>1 THEN GOTO 1920
1900 PRINT TAB(0,30);"You did it !!!"
1910 GOTO 1930
1920 PRINT TAB(0,30);"Your time is up"
1930 *FX 15,1
1940 INPUT "Another game Y/N ",A$
1950 IF LEFT$(A$,1)="Y" THEN RUN
1960 CLS
```

4
Word Scramble



This is a game that all the family can play and it really presents quite a challenge even to the most sharp-eyed and keen-witted. Your Electron invites you to give it a list of up to ten words, each up to ten letters long. Once you have entered them it hides them within a fifteen-by-fifteen grid and fills up all the vacant spaces with random letters. All the words you've entered appear along straight lines vertically, horizontally or diagonally but they can be backwards, upside-down, or slanting from bottom to top. Once they have been camouflaged by all the random letters, spotting them is like looking for a needle in a haystack. If you want to make the task a little easier you can opt to preview the puzzle before any extra letters are added. This at least gives you a chance to unscramble the puzzle. There is yet another helpful hint you can opt for − you can have the list of hidden words displayed on the screen beside the puzzle − but you may be surprised how difficult to spot they still are.

The object of the game is to find all the hidden words. Your

position in the square is indicated by an inverted character which you can use as a pointer to identify the first letter of each word you find. Move this pointer using the cursor control keys. When you think you have found the first letter of a word type a lower case ‘w’. If you are correct you score a point, otherwise you’ll hear a dismal tone.

How to play

The Electron guides you through the early stages of this game, asking you first how many words you wish to supply and then prompting you for each. Then it has to create the puzzle — which takes a little time − longer the more long words you’ve included. It tells you that its ‘working’ on it so that you don’t think its forgotten about you. When it’s ready it asks if you want to preview the puzzle. If you prefer to play the game without any advantages you can skip the preview by answering “n”. Similarly, you can answer either “y” or “n” to the next question which gives the option of having the list of hidden words displayed on the screen beside the completed grid. When the grid appears, you’ll see that the top left hand position is displayed in inverted graphics. This is where the pointer starts. You have to use the arrow keys to move this cursor to a letter that you think is the first letter of one of the words in the list. Once the cursor is in position, type “w”. The Electron will then ask you for the word that you have identified − type this in. If you are correct you will score one point (and your score total will go up by one) but if you are wrong you will hear a tone. Once you’ve completed the puzzle you’ll see the message “You got them all”. If you want to give up during the game type “r” and you’ll be given the option to resign.

Typing tips

When playing this game remember to use lower case letters only. There is a single space between double quotes in lines 420, 600, 800, 810, 930, 1400 and 1420, so don’t type in a null string instead. A null string (two sets of double quotes without any space between) does, however, occur in line 1090.

Subroutine structure

20	Initialisation routine
80	Calls finding words routine
90	Asks for words to be input
280	Gives error messages if more than l0 letters input
330	Finds longest word left in list
380	Constructs puzzle
760	Prints puzzle
870	Allows preview and fills up puzzle with random letters
1010	Main play loop
1240	Asks for guess of word
1320	Word correct routine
1500	Checks for word
1620	Defines dot character and sets score to zero
1670	Resign routine
1720	End of game

Programming details

Some interesting techniques are used in this program. The words are fitted into the square by choosing starting points at random (lines 510-520) and also by choosing directions at random (lines 530-540). Each word is then tested against the square position-by-postion and if there is an empty space, or the identical letter is already present, for every letter of the word, then the word goes in. This allows for two or more words to cross over sharing a space at a common letter. If the word can't be fitted in at its first random spot and direction, the program jumps back and chooses another spot and direction. This procedure is repeated until all the words are fitted.

Scope for improvement

If you have a printer, you could add a routine to this program to enable the completed puzzle to be printed out so that you take it away to be solved. To make the game more difficult you could allow it to

accept more words. Notice, however, that the more words there are the longer it will take to be set up initially. To make the game easier you could remove words from the word list, or mark them in some way, once they had been found.

Program

```
  10 REM Word scramble
  20 MODE 6
  30 *FX 4,1
  40 PROCWORDS
  50 PROCFIT
  60 PROCDOT
  70 PROCPREVIEW

  80 GOSUB 1010

  90 DEF PROCWORDS
 100 DIM L$(10)
 110 DIM C(10)
 120 LST=0
 130 CLS
 140 PRINT TAB(5,2);
 150 INPUT "How many words ",W
 160 IF W<2 OR W>10 THEN SOUND 1,-10,50,4:GOTO
140
 170 PRINT '"Enter words (lower case only)"
 180 FOR I=1 TO W
 190 PRINT TAB(5,5+I);"Word Number ";I;"=";
 200 INPUT W$
 210 IF LEN(W$)>10 THEN PROCERROR
 220 IF LEN(W$)<1 THEN GOTO 200
 230 L$(I)=W$
 240 C(I)=LEN(W$)
 250 NEXT I
 260 VDU 23;8202;0;0;0;
 270 ENDPROC

 280 DEF PROCERROR
 290 PRINT TAB(5,5+I);"Maximum of ten letters"
 300 INPUT W$
  310  PRINT  TAB(5,5+I);"Word  Number  ";I;"=
```

```
";W$;SPC(10)
 320 ENDPROC

 330 M=0: J=0
 340 FOR Z=1 TO W
 350 IF M<C(Z) THEN M=C(Z): J=Z
 360 NEXT Z
 370 RETURN

 380 DEF PROCFIT
 390 DIM D$(15,15)
 400 FOR J=0 TO 14
 410 FOR I=0 TO 14
 420 D$(I,J)=" "
 430 NEXT I
 440 NEXT J
 450 CLS
 460 F=0
 470 FOR I=1 TO W
 480 GOSUB 330
 490 L=C(J)
 500 C(J)=0
 510 X=RND(15-L)
 520 Y=RND(15-L)
 530 V=RND(3)-2
 540 U=RND(3)-2
 550 IF U=0 AND V=0 THEN GOTO 530
 560 A=X: B=Y
 570 IF V<0 THEN A=A+L
 580 IF U<0 THEN B=B+L
 590 FOR K=1 TO L
 600 IF D$(A,B)<>" " AND D$(A,B)<>
     MID$((L$(J)),K,1) THEN F=1:K=L:GOTO 630
 610 A=A+V
 620 B=B+U
 630 NEXT K
 640 IF F=1 THEN F=0:GOTO 510
 650 PRINT "Working"
 660 A=X: B=Y
 670 IF V<0 THEN A=A+L
 680 IF U<0 THEN B=B+L
 690 FOR K=1 TO L
 700 D$(A,B)=MID$(L$(J),K,1)
 710 A=A+V
```

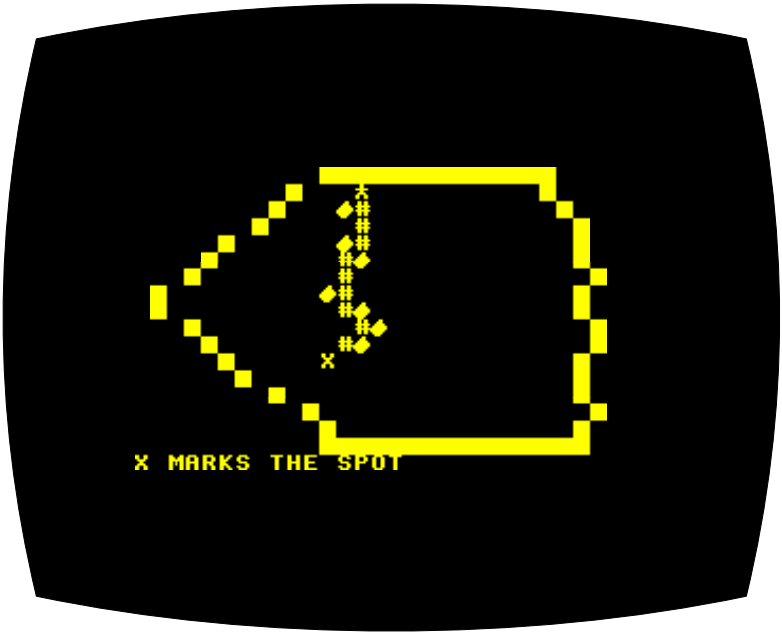
```
 720 B=B+U
 730 NEXT K
 740 NEXT I
 750 ENDPROC
 760 DEF PROCPRINT
 770 CLS
 780 FOR M=0 TO 14
 790 FOR N=0 TO 14
 800 IF D$(N,M)=" " THEN PRINT CHR$(224);
 810 IF D$(N,M)<>" " THEN PRINT D$(N,M);
 820 NEXT N
 830 IF LST=0 OR M>10 THEN PRINT
 840 IF LST=1 AND M<=10 THEN PRINT TAB(20);L$(M)
 850 NEXT M
 860 ENDPROC

 870 DEF PROCPREVIEW
 880 INPUT "Do you want to preview"'" the puzzle
y/n ",A$
 890 IF LEN(A$)=0 THEN GOTO 880
 900 IF LEFT$(A$,1)="y" THEN PROCPRINT
 910 FOR I=0 TO 14
 920 FOR J=0 TO 14
 930 IF D$(J,I)=" " THEN   D$(J,I)=CHR$(RND(25)+
97)
 940 NEXT J
 950 NEXT I
 960 INPUT "Do you want to display"'
     " the words beside the puzzle y/n ",A$
 970 IF LEN(A$)=0 THEN GOTO 960
 980 IF LEFT$(A$,1)="y" THEN LST=1
 990 PROCPRINT
1000 ENDPROC

1010 X=0
1020 Y=0
1030 COLOUR 0
1040 COLOUR 135
1050 PRINT TAB(X,Y);D$(X,Y);
1060 COLOUR 7
1070 COLOUR 128
1080 A$=INKEY$(0)
1090 IF A$="" THEN GOTO 1080
1100 IF A$="w" THEN GOTO 1230
```

```
1110 PRINT TAB(X,Y);D$(X,Y);
1120 IF ASC(A$)=&88 AND X>0 THEN X=X-1
1130 IF ASC(A$)=&89 AND X<14 THEN X=X+1
1140 IF ASC(A$)=&8B AND Y>0 THEN Y=Y-1
1150 IF ASC(A$)=&8A AND Y<14 THEN Y=Y+1
1160 IF A$="r" THEN GOSUB 1670
1170 COLOUR 0
1180 COLOUR 135
1190 PRINT TAB(X,Y);D$(X,Y);
1200 COLOUR 7
1210 COLOUR 128
1220 GOTO 1080
1230 PRINT TAB(0,20);

1240 INPUT "What is the word ",W$
1250 IF LEN(W$)=0 OR LEN(W$)>15 THEN
     SOUND 1,-10,50,2: GOTO 1230
1260 GOSUB 1450
1270 PRINT TAB(0,20);SPC(30)
1280 IF MATCH=0 THEN SOUND 1,-10,50,2:GOTO 1080
1290 FOR U=-1 TO 1
1300 FOR V=-1 TO 1
1310 IF U=0 AND V=0 THEN GOTO 1340
```

```
1320 PROCMATCH
1330 IF MATCH=1 THEN V=1:U=1:GOTO 1340
1340 NEXT V
1350 NEXT U
1360 IF MATCH=1 THEN GOTO 1390
1370 SOUND 1,-10,50,4
1380 GOTO 1080
1390 SCORE=SCORE+1
1400 PRINT TAB(10,21);"Score= ";SCORE;" "
1410 SOUND 1,-10,50,4
1420 L$(WORD)=" "
1430 IF SCORE=W THEN GOTO 1780
1440 GOTO 1080
1450 MATCH=0
1460 FOR I=1 TO W
1470 IF W$=L$(I) THEN MATCH=1: WORD=I:I=W
1480 NEXT I
1490 RETURN

1500 DEF PROCMATCH
1510 MATCH=1
1520 A=X
1530 B=Y
1540 FOR I=1 TO LEN(W$)
1550 IF MID$(W$,I,1)<>D$(A,B) THEN I=LEN(W$):
     MATCH=0:GOTO 1600
1560 A=A+U
1570 B=B+V
1580 IF A<1 OR A>15 THEN I=LEN(W$):
     MATCH=0:GOTO 1600
1590 IF B<1 OR B>15 THEN I=LEN(W$):
     MATCH=0:GOTO 1600
1600 NEXT I
1610 ENDPROC

1620 DEF PROCDOT
1630 REM dot character
1640 VDU 23,224,&00,&00,&00,&18,&18,&00,&00,&00
1650 SCORE=0
1660 ENDPROC
```

```
1670 PRINT TAB(0,22);
1680 INPUT "Are you sure that you"'
     "can find no more words y/n ",A$
1690 PRINT TAB(0,22);SPC(70)
1700 IF LEFT$(A$,1)<>"y" THEN RETURN
1710 CLS

1720 PRINT TAB(10,8);"Final score= ";SCORE
1730 PRINT TAB(8,10);
1740 INPUT "Another game y/n ",A$
1750 IF A$="y" THEN RUN
1760 IF A$<>"n" THEN GOTO 1730
1770 STOP
1780 PRINT TAB(0,22);"You got them all!"
1790 GOTO 1740
```

5
Treasure Island



Find the hidden treasure before the pirate ship reaches the island. This game has all the ingredients of high adventure. A desert island, peopled by natives both hostile and friendly, gold buried at the spot marked ‘X’ on the map, quicksands that spell danger to the unlucky treasure-seeker and even Long John Silver’s parrot. While you hunt for the booty the pirate ship is heading towards the island and once the pirates land you are certain to be captured. The game is displayed in colour graphics and has sound effects as well.

How to play

The treasure is buried at the spot marked ‘X’ on the map that is briefly flashed on to the screen at the start of the game. The path is also indicated − you have to follow it exactly. If you stray there are three possible outcomes. If you are lucky Long John Silver’s parrot will guide you back to the path − you will see the parrot hovering

over the next position on the path. If you are unlucky you will encounter hostile natives and will find yourself back on the path three paces back from where you left it, and if you are unluckier still, you will end up in a quicksand. This can be a final fate or you may be rescued by a friendly native. If you need to consult the map in order to follow the path you can type "h". When you do this you will be shown the map now also indicating the locations of the quicksands − for a short, random length of time. However, every time you ask to see the map the pirate ship comes nearer and if the ship arrives before you find the treasure you will be captured. The ship advances anyway once for every five moves you make so you need to be accurate. To move along the path use the right, left and forward arrow keys you cannot move backwards.

Subroutine structure

20	Sets up screen display
90	Defines graphics characters
230	Initialises variables and arrays
310	Main play loop
510	Logic for natives' attack
670	Logic for parrot's help
810	Prints map
1040	Moves man
1250	Prints quicksands on map
1330	Logic for quicksands
1460	Constructs island
1590	Constructs path
1680	Locates treasure
1710	Constructs quicksands
1800	Moves, prints pirate ship
1950	Prints island
2150	Help routine
2220	Treasure found routine
2420	End of game

Programming details

This is a very long program with lots of different ingredients. It therefore appears rather complicated whereas in fact it is quite straightforward. One technique to notice is the way the island is constructed at random by subroutine 1460. There the use of SGN function results in the contour of the island first going out and then coming in at random, giving an island-like shape.

Scope for alteration

If you want to alter the length of time the map is displayed for when you ask to see it, you can change the final value of the FOR loop in line 2170. You might also want to alter the amount that the pirate ship moves at each step by setting a different value for 'R' in line 1840. Currently it's set randomly to a value between 1 and 3.

Program

```
 10 REM Treasure island

 20 MODE 1
 30 VDU 23;8202;0;0;0;
 40 VDU 19,0,6,0,0,0
 50 VDU 19,1,2,0,0,0
 60 VDU 19,2,3,0,0,0
 70 VDU 19,3,0,0,0,0
 80 *FX 4,1

 90 REM parrot
100 VDU 23,224,&20,&36,&3E,&1C,&1E,&27,&40,&00
110 REM native
120 VDU 23,225,&01,&19,&D9,&FF,&D9,&19,&25,&25
130 REM ship sail
140 VDU 23,226,&FF,&7E,&D5,&C3,&C3,&D5,&7E,&FF
150 REM ship
160 VDU 23,227,&10,&FF,&FF,&7E,&7E,&73,&3C,&3C
170 REM quicksand
180 VDU 23,228,&0C,&1C,&3E,&7F,&FF,&FE,&7C,&38
190 REM man
```

```
200 VDU 23,229,&18,&18,&7E,&18,&3C,&66,&66,&00
210 REM island
220 VDU 23,230,&FF,&FF,&FF,&FF,&FF,&FF,&FF,&FF
230 F=0
240 XS=1:YS=1
250 MES=0
260 DIM V(10)
270 DIM U(10)
280 DIM L(20)
290 DIM R(20)
300 DIM X(20)

310 GOSUB 1460
320 XM=X(T+1)
330 YM=T+1
340 GOSUB 810
350 FOR Q=1 TO 5000+RND(2000):NEXT Q
360 GOSUB 1950
370 GOSUB 1160
380 GOSUB 1040
390 IF XM=XT AND YM=YT THEN GOTO 2220
400 F=F+1
410 IF INT(F/5)=F/5THEN GOSUB 1800
420 IF X(YM)=XM AND INT(F/5)=F/5 THEN
    GOTO 360
430 IF X(YM)=XM THEN GOTO 380
440 REM off path
450 SOUND 0,-10,10,2:GOSUB 1800
460 FOR Q=1 TO 10
470 REM test for quicksands
480 IF V(Q)=XM AND U(Q)=YM THEN GOSUB 1330
490 NEXT Q
500 IF RND(1)<=.4 THEN GOTO 670

510 REM natives logic
520 GOSUB 1950
530 PRINT TAB(1,25);"HOSTILE NATIVES AHEAD"
540 FOR N=1 TO 3
550 R=RND(3)
560 IF YM+R>=B THEN R=0
570 COLOUR 129
580 COLOUR 3
590 PRINT TAB(XM,YM+R);CHR$(225)
600 NEXT N
```

```
 610 YM=YM-3
 620 IF YM<=T+1 THEN YM=T+1
 630 XM=X(YM)
 640 PRINT TAB(XM,YM);CHR$(229)
 650 MES=1
 660 GOTO 380
 670 REM parrot logic
 680 GOSUB 1950
 690 GOSUB 1160
 700 COLOUR 128
 710 COLOUR 3
 720 PRINT TAB(1,25);"FOLLOW LONG JOHN SILVERS
     PARROT"
 730 YJ=YM+1
 740 IF YJ>P THEN YJ=P
 750 XJ=X(YJ)
 760 COLOUR 129
 770 COLOUR 3
 780 PRINT TAB(XJ,YJ);CHR$(224)
 790 MES=1
 800 GOTO 380

 810 REM print island
 820 COLOUR 130
 830 CLS
 840 COLOUR 3
 850 FOR X=L(T)TO R(T)
 860 PRINT TAB(X,T);CHR$(230)
 870 NEXT X
 880 FOR Y=T TO B
 890 PRINT TAB(L(Y),Y);CHR$(230);TAB(R(Y),Y);
     CHR$(230)
 900 NEXT Y
 910 FOR X=L(Y-1)TO R(Y-1)
 920 PRINT TAB(X,Y);CHR$(230)
 930 NEXT X
 940 REM print path
 950 FOR Y=T+1 TO P
 960 PRINT TAB(X(Y),Y);"#"
 970 NEXT Y
 980 PRINT TAB(1,20);"X MARKS THE SPOT"
 990 PRINT TAB(X(P),P);"X"
1000 PRINT TAB(XM,YM);CHR$(229)
1010 GOSUB 1250
```

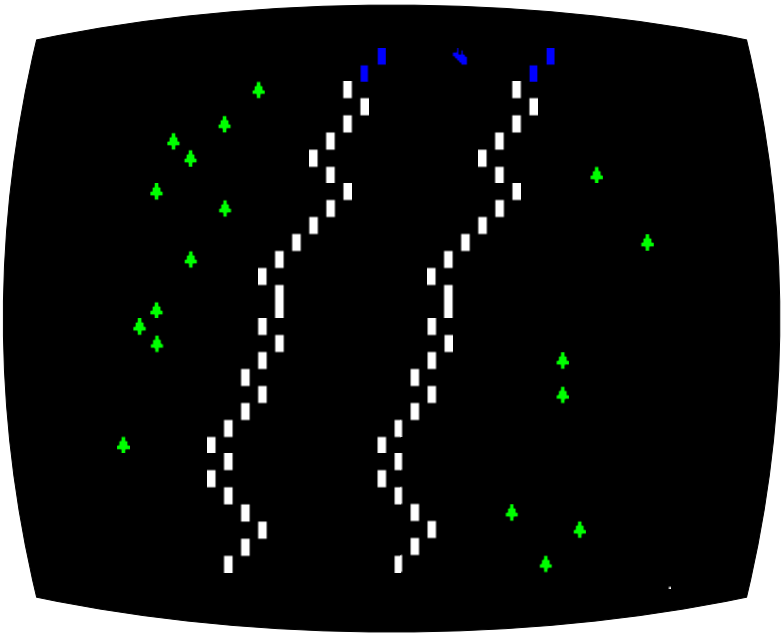
```
1020 RETURN
1030 GOTO 1050

1040 REM move man
1050 SOUND 1,-10,50,2
1060 A$=INKEY$(0)
1070 IF A$="" THEN GOTO 1060
1080 COLOUR 129
1090 COLOUR 3
1100 PRINT TAB(XM,YM);" "
1110 IF A$="H"THEN GOSUB 2150:RETURN
1120 IF ASC(A$)=&88 THEN XM=XM-1:GOTO 1150
1130 IF ASC(A$)=&89 THEN XM=XM+1:GOTO 1150
1140 IF ASC(A$)<>&8A THEN GOTO 1050
1150 YM=YM+1
1160 COLOUR 129
1170 COLOUR 3
1180 PRINT TAB(XM,YM);CHR$(229)
1190 IF MES=0THEN RETURN
1200 COLOUR 128
1210 FOR I=0TO 31
1220 PRINT TAB(I,25);" "
1230 NEXT I
1240 MES=0:RETURN

1250 REM print quicksands
1260 COLOUR 3
1270 FOR Q=1TO 10
1280 PRINT TAB(V(Q),U(Q));CHR$(228)
1290 NEXT Q
1300 XS=XS+1
1310 YS=YS+1
1320 RETURN

1330 REM quicksand
1340 GOSUB 1950
1350 COLOUR 2
1360 COLOUR 129
1370 PRINT TAB(XM,YM);CHR$(228)
1380 COLOUR 3
1390 PRINT TAB(XM+1,YM+1);"AARGH"
1400 FOR I=50 TO 10 STEP -5
1410 SOUND1,-10,I,2
1420 NEXT I
```

```
1430 PRINT TAB(1,25);"IN THE QUICKSAND"
1440 IF RND(1)>.5THEN PRINT;"A friendly native
     pulled you out":FOR Q=1 TO 2000:NEXT Q:
     RETURN
1450 GOTO 2420
1460 REM calculate island
1470 L=RND(3)+10
1480 T=RND(3)+2
1490 W=10+RND(3)
1500 B=RND(2)+17
1510 FOR Y=T TO B
1520 L(Y)=L
1530 R(Y)=L+W
1540 L=L-(SGN(10-Y)*RND(2))
1550 W=W+(SGN(10-Y)*RND(2))
1560 IF L(Y)<1 THEN L(Y)=1
1570 IF R(Y)>30 THEN R(Y)=30
1580 NEXT Y

1590 REM path logic
1600 X(T+1)=L(T+1)+RND(4)
1610 K=T+2
1620 FOR P=K TO B-1-RND(3)
1630 X(P)=X(P-1)+RND(3)-2
1640 IF X(P)>=R(P)THEN X(P)=R(P)-1
1650 IF X(P)<=L(P)THEN X(P)=L(P)+1
1660 NEXT P
1670 P=P-1

1680 REM treasure
1690 XT=X(P)
1700 YT=P

1710 REM quicksand logic
1720 FOR Q=1TO 10
1730 D=RND((P-T-2))+T+1
1740 U(Q)=D
1750 V(Q)=X(D)+SGN(RND(1)-.5)
1760 IF V(Q)<=L(D)THEN V(Q)=V(Q)+3
1770 IF V(Q)>=R(D)<THEN V(Q)=V(Q)-3
1780 NEXT Q
1790 RETURN

1800 REM pirate ship
```

```
1810 COLOUR 128
1820 COLOUR 3
1830 CLS
1840 R=RND(3)
1850 XS=XS+R
1860 YS=YS+R
1870 PRINT TAB(18,18);CHR$(228)
1880 PRINT TAB(XS,YS);CHR$(226)
1890 PRINT TAB(XS,YS+1);CHR$(227)
1900 FOR Q=1 TO 2000:NEXT Q
1910 IF YS<18 THEN RETURN
1920 PRINT TAB(0,25);"THE PIRATES HAVE LANDED"
1930 PRINT "YOU ARE CAPTURED"
1940 GOTO 2420

1950 REM reprint island
1960 COLOUR 128
1970 CLS
1980 COLOUR 2
1990 FOR X=L(T)TO R(T)
2000 PRINT TAB(X,T-1);CHR$(230)
2010 NEXT X
2020 FOR Y=T TO B
2030 PRINT TAB(L(Y)-1,Y);CHR$(230)
2040 COLOUR 1
2050 FOR X=L(Y)TO R(Y)
2060 PRINT TAB(X,Y);CHR$(230)
2070 NEXT X
2080 COLOUR 2
2090 PRINT TAB(X,Y);CHR$(230)
2100 NEXT Y
2110 FOR X=L(Y-1)TO R(Y-1)
2120 PRINT TAB(X,Y);CHR$(230)
2130 NEXT X
2140 RETURN

2150 GOSUB 810
2160 GOSUB 1250
2170 FOR Q=1 TO 5000+RND(1000):NEXT Q
2180 GOSUB 1800
2190 GOSUB 1950
2200 GOSUB 1160
2210 RETURN
```

```
2220 REM treasure found
2230 DATA 53,61,69,73,81,89,99,101,109,117,121,
     129,137
2240 DIM N(13)
2250 FOR Q=1 TO 13
2260 READ N(Q)
2270 NEXT Q
2280 MODE 2
2290 VDU 19,7,3,0,0,0
2300 VDU 19,3,7,0,0,0
2310 FOR C=128 TO 135
2320 COLOUR C
2330 CLS
2340 FOR Q=1 TO 500:NEXT Q
2350 SOUND 1,-15,N(C-127),C-124
2360 SOUND 2,-15,N(C-125),C-124
2370 SOUND 3,-15,N(C-123),C-124
2380 NEXT C
2390 COLOUR 0
2400 PRINT TAB(0,10);"YOU FOUND THE GOLD"
2410 FOR Q=1 TO 500:NEXT Q

2420 INPUT "Another game (Y/N)",A$
2430 IFLEFT$(A$,1)="Y"THEN RUN
2440 VDU 20
2450 MODE 7
2460 *FX 4,0
```

6
Bobsleigh



In this game you have to steer your blue bobsleigh down a random course that winds its way past fir trees. You can choose whether to try a course that is easy to manoeuvre or one that is difficult. (There are actually five levels of difficulty which govern the width of the course.) If you crash you'll hear a dismal tone and that round of the game is over. Play this game to see how adept you are at keeping on co urse.

How to play

The bobsleigh starts off at the top of the course and the course automatically moves past it. You have to steer the bobsleigh using the right and left arrow keys to ensure that you do not crash into the edges of the course. At the beginning of the game you have to select the difficulty level for the game. This governs the width of the course with 1 producing the widest, and therefore easiest, and 5 the

narrowest, and most difficult.

Subroutine structure

20	Start of game
30	Defines graphics characters, sets up keyboard and colours
150	Prints first part of track
260	Main play loop
380	Scrolls last part of track off screen
440	Win/lose messages
480	End of game
560	Title frame
670	Moves bobsleigh
770	Prints fir trees

Programming details

The impression of the bobsleigh moving down the course is actually achieved by the course scrolling up the screen past the bobsleigh which only moves to left and right and is at a fixed vertical position. In line 730 the POINT function is used to test whether or not the bobsleigh has hit the side wall. This is done simply by testing what colour is present at the next position that the bobsleigh will be printed at. If the colour is not white then you've crashed into the wall.

The *FX commands in lines 90 and 100 increase the rate at which the keyboard auto-repeats. Normal operation is restored in the end of game routine at line 480. If, for any reason you break out of this program prematurely you will find that the keyboard will still be repeating every key press. To overcome this you will have to press BREAK and type OLD.

It is also worth noting the use of *FX 15,1 in lines 500 and 690, which has the effect of clearing the Electron's type ahead buffer.

Scope for improvement

If you find this game too fast-moving you can slow it down by increasing the final value of the FOR loops in lines 360 and 420. You might like to add a tune-playing routine to this program like the one given in ‘Electron Epsom’.

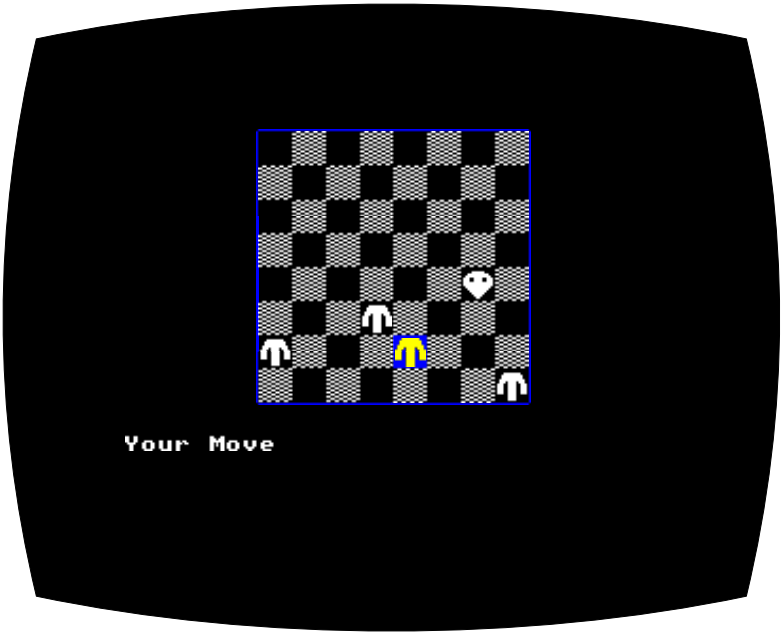
Program

```
 10 REM Bobsleigh

 20 PROCTITLE

 30 MODE 1
 40 *FX 4,1
 50 VDU 19,1,0,0,0,0
 60 VDU 19,0,7,0,0,0
 70 VDU 19,3,4,0,0,0
 80 VDU 19,2,2,0,0,0
 90 *FX 12,1
100 *FX 11,1
110 VDU 23;8202;0;0;0;
120 VDU 23,224,&20,&28,&E8,&FC,&7E,&3E,&1E,&0E
130 VDU 23,225,&0F,&0F,&0F,&0F,&0F,&0F,&0F,&0F
140 VDU 23,226,&18,&18,&3C,&3C,&7E,&7E,&18,&18

150 YB=0
160 X=RND(10)+5
170 XB=X+2
180 FOR Y=1 TO 30 STEP 2
190 PRINT TAB(X,31);CHR$(225);TAB(X+D,31);
    CHR$(225)
200 PROCTREE(X)
210 X=X+SGN(RND(1)-.5)
220 IF X>31 THEN X=31
230 IF X<1 THEN X=1
240 NEXT Y
250 PRINT TAB(XB,YB);CHR$(224)
260 FOR Z=1 TO 500:NEXT
270 F=0
```

```
280 FOR Z=1 TO 300
290 X=X+SGN(RND(1)-.5)
300 IF X>29 THEN X=29
310 IF X<1 THEN X=1
320 PRINT TAB(X,31);CHR$(225);TAB(X+D,31);
    CHR$(225)
330 PROCTREE(X)
340 PROCBOB
350 IF F=1 THEN GOTO 460
360 FOR Q=1 TO 100:NEXT Q
370 NEXT

380 FOR Z=1 TO 30
390 PRINT TAB(1,31)'
400 PROCBOB
410 IF F=1 THEN GOTO 460
420 FOR Q=1 TO 100:NEXT Q
430 NEXT Z

440 PRINT TAB(1,20);"CONGRATULATIONS YOU MADE
    IT"
450 GOTO 480
460 PRINT TAB(1,20);"YOU CRASHED"
470 SOUND 0,-10,2,20

480 *FX 12,0
490 *FX 4,0
500 *FX 15,1
510 INPUT "ANOTHER GAME (Y/N)",A$
520 A$=LEFT$(A$,1)
530 IF A$="Y" THEN RUN
540 CLS
550 STOP

560 DEF PROCTITLE
570 CLS
580 PRINT TAB(8,1);" B O B S L E I G H"
590 PRINT TAB(4,4);" You must steer your
    bobsleigh"
600 PRINT TAB(4,7);" down a dangerous course
    using"
610 PRINT TAB(4,10);" the left and right arrow
    keys."
620 PRINT TAB(4,14);" Select th difficulty
```

```
    level -"
630 INPUT '"     from 1 (easy) to 5
    (difficult) ",D
640 IF D<1 OR D>5 THEN GOTO 630
650 D=11-D
660 ENDPROC

670 DEF PROCBOB
680 A=INKEY(0)
690 *FX 15,1
700 COLOUR 3
710 IF A=&88 THEN XB=XB-1
720 IF A=&89 THEN XB=XB+1
730 IF POINT(XB*32+16,1024-32*YB-16)<>0 THEN
     F=1
740 PRINT TAB(XB,YB);CHR$(224)
750 COLOUR 1
760 ENDPROC

770 DEF PROCTREE(X)
780 IF RND(1)<.4 THEN GOTO 860
790 K=SGN(RND(1)-.5)
800 XT=X+K
810 XT=X+D/2+(K*(RND(5)+D))
820 IF XT<0 OR XT>39 THEN GOTO 780
830 COLOUR 2
840 PRINT TAB(XT,31);CHR$(226);
850 COLOUR 1
860 ENDPROC
```

7
Capture the Quark



What on earth is a ‘quark’? Well may you ask that question, but to find out you’ll have to play this game. Here are just a few clues. The game is played on an eight-by-eight checkered board and the object of the game is to trap the quark and prevent him from reaching the bottom of the board. To do this you have two, three or four (determined at random) pieces, or ‘quatlins’, which can be moved diagonally one square at a time and only up the board. The quark also moves diagonally but he can move both forwards and backwards.

How to play

At the beginning of the game your quatlins (two, three or four of them according to the luck of the draw) are ranged along the bottom line and the quark is at the top of the board. It is your move first.

You'll notice that one of your quatlins is displayed in different colours from the rest. This is the piece that is ready to move. If you want to move another piece hit any key and control will pass to the next piece in an anti-clockwise direction. Try pressing keys to see this in operation. When you are ready to move a piece press the left arrow key if you want to move diagonally forward left and the right arrow key if you want to move diagonally forward right. Once you have moved, the quark will make his move automatically and its your turn again. The Electron will display a message when the game is won, either by you or the quark, but if you want to resign before this, type "R". Just in case you hit this key by mistake you will be given the chance to reconsider and will have to answer "Y" or "N" to the question "Resign?".

Typing tips

The IF statement in line 140 looks as though its wrong as there are no relational signs. It is however correct − the values stored in the array are either '1' or '0' and the Electron treats these as being equivalent to 'true' or 'false'.

Subroutine structure

20	Initialises variables
60	Sets up game
90	Main play loop
120	Quark move logic
500	Moves quatlins and validates their moves
650	Selects which quatlin to move
830	Prints differently coloured quatlin
890	Prints quatlin
930	Draws board
1120	Draws initial positions of qualtins and quark and initialises board
1390	Defines graphics characters

1590 End of game

Programming details

Although this program runs in a four-colour mode, a hatched black and white character is used to give an extra tone (grey). This is defined at the beginning of subroutine 1390. Notice the way that subroutine 830 uses COLOUR to paint a quatlin in two different colours from those normally used for the qualtins.

Program

```
 10 REM Capture the quark

 20 MODE 1
 30 DIM D(10,10)
 40 DIM X(4),Y(4)
 50 DIM A(4),B(4)

 60 PROCINIT
 70 PROCBOARD
 80 H=RND(2)+2:PROCSTART

 90 PROCMOVE
100 PROCQUARK
110 GOTO 90

120 DEF PROCQUARK
130 PRINT TAB(0,21);"Quarks Move"
140 IF D(QI+2,QJ+2) AND D(QI+2,QJ) AND
    D(QI,QJ+2) AND D(QI,QJ) THEN GOTO 1590
150 M=1
160 GOSUB 400
170 IF QI+N<1 OR QI+N>8 THEN GOTO 200
180 IF D(QI+N+1,QJ+M+1) THEN GOSUB 330
190 IF D(QI+N+2,QJ+M+2)=1 AND D(QI+N,QJ+M+2)=1
    AND QJ<7 AND QJ>1 THEN M=-M
200 IF D(QI+N+1,QJ+M+1) THEN GOSUB 330
210 IF OI=QI AND OJ=QJ THEN M=-M:
    N=SGN (RND(1)-.5): GOTO 170
```

```
220 OI=QI:OJ=QJ
230 PRINT TAB(QX,QY);SPC(2);TAB(QX,QY+1);
    SPC(2);
240 QX=QX+N*2
250 QY=QY+M*2
260 PRINT TAB(QX,QY);CHR$(225);CHR$(226);
    TAB(QX,QY+1);CHR$(227);CHR$(228)
270 D(QI+N+1,QJ+M+1)=2
280 D(QI+1,QJ+1)=0
290 QI=QI+N
300 QJ=QJ+M
310 IF QJ=8 THEN GOTO 1610
320 ENDPROC
330 N=-N
340 IF D(QI+N+1,QJ+M+1)<>1 THEN RETURN
350 M=-M
360 IF QJ<4 THEN N=1
370 IF D(QI+N+1,QJ+M+1)<>1 THEN RETURN
380 N=-N
390 RETURN
400 N=(QI<5)-(QI>=5)
410 R=RND(1)
420 IF QJ>6 THEN RETURN
430 IF R>.5 AND QI>3 THEN GOTO 470
440 IF QI>7 THEN GOTO 460
450 IF (D(QI+3,QJ+3)=0 OR D(QI+2,QJ+3)=0)
    AND D(QI+2,QJ+2)=0 THEN N=-1:RETURN
460 IF QI<4 OR R>.5 THEN RETURN
470 IF (D(QI-3,QJ+3)=0 OR D(QI-2,QJ+3)=0)
    AND D(QI-2,QJ+2)=0 THEN N=1: RETURN
480 IF R>.5 THEN GOTO 440
490 RETURN
500 A(HM)=A(HM)+M
510 B(HM)=B(HM)-1
520 IF D(A(HM)+1,B(HM)+1)<>0 THEN GOTO 560
530 D(A(HM)-M+1,B(HM)+2)=0
540 D(A(HM)+1,B(HM)+1)=1
550 GOTO 600
560 A(HM)=A(HM)-M
570 B(HM)=B(HM)+1
580 SOUND 1,-10,80,5
590 GOTO 660
600 PRINT TAB(X(HM),Y(HM));SPC(2);
    TAB(X(HM),Y(HM)+1);SPC(2);
```

```
610 Y(HM)=Y(HM)-2
620 X(HM)=X(HM)+M*2
630 GOSUB 830
640 ENDPROC

650 DEF PROCMOVE
660 PRINT TAB(0,21);"Your Move   "
670 SOUND 1,-10,100,4
680 M$=INKEY$(0)
690 IF M$="" THEN GOTO 680
700 IF M$="R" THEN GOTO 750
710 IF ASC(M$)=&88 THEN M=-1: GOTO 500
720 IF ASC(M$)=&89 THEN M=+1: GOTO 500
730 GOSUB 780
740 GOTO 660
750 INPUT "Resign Y/N",A$
760 IF A$="Y" THEN GOTO 1610
770 GOTO 660
780 GOSUB 890
790 HM=HM+1
800 IF HM>H THEN HM=1
810 GOSUB 830
820 RETURN
830 COLOUR 2
840 COLOUR 128+3
850 PRINT TAB(X(HM),Y(HM));CHR$(229);
    CHR$(231);TAB(X(HM),Y(HM)+1);CHR$(230);
    CHR$(232);
860 COLOUR 1
870 COLOUR 128
880 RETURN

890 COLOUR 1
900 COLOUR 128
910 PRINT TAB(X(HM),Y(HM));CHR$(229);
    CHR$(231);TAB(X(HM),Y(HM)+1);CHR$(230);
    CHR$(232);
920 RETURN

930 DEF PROCBOARD
940 PRINT TAB(8,3);
950 FOR I=1 TO 4
960 FOR J=1 TO 8
970 PRINT  SPC(2);CHR$(224);CHR$(224);
```

```
 980 IF J/4=INT (J/4) THEN PRINT:PRINT TAB(8);
 990 NEXT J
1000 FOR J=1 TO 8
1010 PRINT CHR$(224);CHR$(224);SPC(2);
1020 IF J/4=INT (J/4) THEN PRINT:PRINT TAB(8);
1030 NEXT J
1040 NEXT I
1050 GCOL 0,1
1060 MOVE 255,415
1070 PLOT 2,515,0
1080 PLOT 2,0,515
1090 PLOT 2,-515,0
1100 PLOT 2,0,-515
1110 ENDPROC
```

```
1120 DEF PROCSTART
1130 FOR I=1 TO H
1140 X=I*4+6
1150 PRINT TAB(X,17);CHR$(229);CHR$(231);
     TAB(X,18);CHR$(230);CHR$(232)
1160 D(I*2+1,9)=1
1170 X(I)=X
1180 Y(I)=17
1190 HM=1
1200 A(I)=I*2
1210 B(I)=8
1220 NEXT I
1230 GOSUB 830
1240 QI=5
1250 QJ=1
1260 QX=QI*2+6
1270 QY=3
1280 PRINT TAB(QX,QY);CHR$(225);CHR$(226);
     TAB(QX,QY+1);CHR$(227);CHR$(228)
1290 FOR I=1 TO 10
1300 D(I,1)=1
1310 D(1,I)=1
1320 D(10,I)=1
1330 D(I,10)=1
1340 NEXT I
1350 D(QI+1,QJ+1)=2
1360 OI=0
1370 OJ=0
1380 ENDPROC

1390 DEF PROCINIT
1400 VDU 23,224,&33,&CC,&33,&CC,&33,&CC,&33,&CC
1410 VDU 23,225,&00,&00,&07,&1F,&3F,&73,&73,&7F
1420 VDU 23,226,&00,&00,&E0,&F8,&FC,&CE,&CE,&FE
1430 VDU 23,227,&7F,&3F,&1F,&0F,&07,&03,&01,&00
1440 VDU 23,228,&FE,&FC,&F8,&F0,&E0,&C0,&80,&00
1450 VDU 23,229,&00,&00,&0F,&1F,&1F,&3F,&3B,&73
1460 VDU 23,230,&73,&73,&73,&73,&03,&03,&03,&00
1470 VDU 23,231,&00,&00,&F0,&F8,&F8,&FC,&DC,&CE
1480 VDU 23,232,&CE,&CE,&CE,&CE,&C0,&C0,&C0,&00
1490 *FX 4,1
1500 VDU 23;8202;0;0;0;
1510 VDU 19,0,7,0,0,0
1520 VDU 19,1,0,0,0,0
```

```
1530 VDU 19,2,3,0,0,0
1540 VDU 19,3,4,0,0,0
1550 COLOUR 1
1560 COLOUR 128
1570 CLS
1580 ENDPROC

1590 PRINT TAB(0,21);"You Win    "
1600 GOTO 1620
1610 PRINT TAB(0,21);"Quark Wins "
1620 INPUT "Another Game",A$
1630 IF A$="Y" THEN RUN
1640 CLS
1650 PRINT "Bye"
1660 *FX 4,0
```

8
Commando Jump

This game is a real test of your reaction time and dexterity, and is quite compulsive to play. A bright red wall of varying height appears with a little man figure beside it. A countdown “Ready, Steady, GO” is flashed up on the left of the screen and on the word “GO” the man has to jump as high as possible and then scrabble up the remainder of the wall. Y our success in this game depends entirely on your quick wits and nimble fingers.

How to play

On the word “GO”, and no sooner, press any key to make the man jump. The height of the initial jump depends entirely on the delay between the signal appearing and your key press. The quicker you react, the higher the man will jump. The time left to scale the wall is displayed on the screen and while the rest of your five seconds tick away you must keep on pressing any key to get the man over the

wall. The man will climb one brick higher for every ten key presses so the more rapidly you press, the more quickly he will climb. If you keep your finger on a key the man will stay where he is this is because only complete key presses, i.e. press and release, count. If the man is not over within the time limit he will slither back down the wall and you have another try. In all you are given ten attempts. Even if you are very slow off the mark, do press a key − until you do so you cannot move on to the next try. If you hit a key just before the "GO" signal, the computer will accuse you of cheating and you will lose that turn.

Subroutine structure

20	Sets up mode and calls initialisation routine
40	Play loop
60	Countsdown
250	Cheat routine
320	Prints wall
420	Jump logic
560	Scramble over remainder of wall routine
680	Man falls back down wall
780	Prints man over wall
900	Zeroes time
920	Sets up screen and defines graphics characters
1000	End of game and messages

Programming details

This is a fairly straightforward application of low resolution dynamic graphics. It runs in Mode 2, a sixteen colour mode and takes advantage of the Electron's flashing colours. The words "Ready" and "Steady" are made to flash on and off by making them alternate between red and cyan which causes them to disappear because cyan is used as the background colour. In this game the Electron's internal

clock is used as a reaction timer and a countdown device. The timer counts in one hundreth of a second intervals and in this game the variable T is scaled to count in seconds. Another interesting point to note is that the repeat feature of the keyboard is disabled by *FX 11,0 in line 980. The repeat key feature is restored at the end game by *FX 12,0 in line 1130.

Program

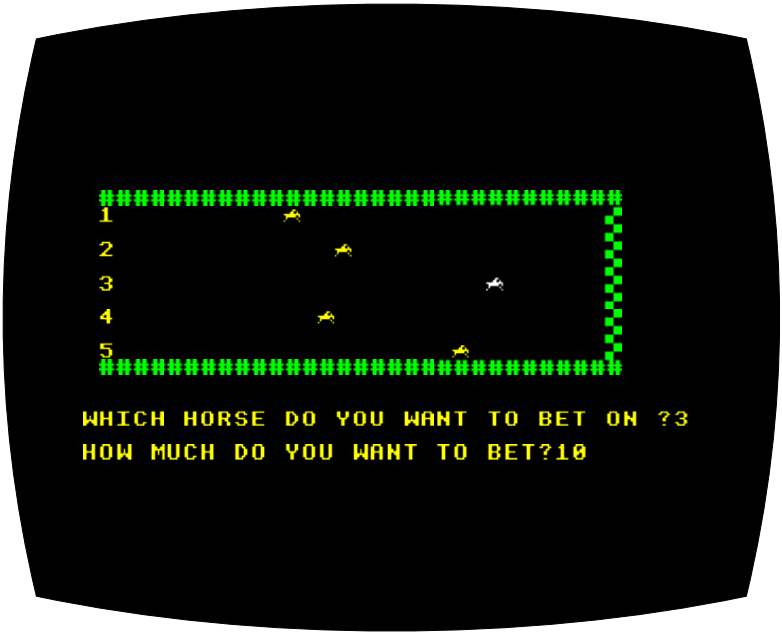
```
 10 REM Commando Jump

 20 MODE 2
 30 PROCINIT

 40 PROCWALL
 50 PROCFIN

 60 COLOUR 9
 70 PRINT TAB(0,20);SPC(5)
 80 PRINT TAB(0,20);"Ready";
 90 PRINT TAB(0,21);SPC(3)
100 FOR I=1 TO RND(2000)+2000
110 NEXT I
120 PRINT TAB(0,20);"Steady"
130 COLOUR 1
140 *FX 15,1
150 FOR I=1 TO RND(2000)+2000
160 NEXT I
170 PRINT TAB(0,20);SPC(6)
180 IF INKEY$(0)<>"" THEN GOTO 250
190 TIME=0
200 SOUND 1,-10,100,5
210 PRINT "GO"
220 IF INKEY$(0)="" THEN GOTO 220
230 T=TIME/100
240 RETURN

250 PRINT TAB(2,10);"Cheat"
260 SOUND 1,-15,3,30
270 FOR K=1 TO 5000:NEXT K
280 PRINT TAB(2,10);SPC(5)
290 TIME=5*100
```

```
300 T=TIME/100
310 RETURN

320 DEF PROCWALL
330 COLOUR 6+128
340 CLS
350 JUMP=1
360 H=10+RND(5)
370 FOR I=18 TO 19-H STEP -1
380 COLOUR 1
390 COLOUR 3+128
400 PRINT TAB(10,I);CHR$(224);
410 NEXT I

420 PRINT TAB(10,18-H);CHR$(226);
430 COLOUR 0
440 COLOUR 6+128
450 PRINT TAB(10,20);"Jump ";JUMP
460 PRINT TAB(13,18);CHR$(225);
470 GOSUB 60
480 COLOUR 0
490 IF T>=5 THEN GOTO 740
500 FOR I=18 TO 18-H+INT (T*28) STEP -1
510 PRINT TAB(13,I);" ";
520 PRINT TAB(13,I-1);CHR$(225);
530 SOUND 1,-10,150-I*8,1
540 FOR K=1 TO 100:NEXT K
550 NEXT I
```

```
560 J=I: L=INT I
570 T=TIME/100
580 IF T>5 THEN GOTO 680
590 PRINT TAB(5,25);"Time left ";
    INT ((5-T)*10)/10;"  "
600 IF INKEY$(0)="" THEN GOTO 570
610 PRINT TAB(13,INT L);" ";
620 J=J-0.2
630 L=INT J
640 PRINT TAB(13,L);CHR$(225);
650 IF L<=17-H THEN PRINT TAB(13,L+1);" ":
    GOTO 780
660 *FX 15,1
670 GOTO 570

680 FOR I=L TO 18
690 PRINT TAB(13,I-1);" "
700 PRINT TAB(13,I);CHR$(225);
710 SOUND 1,-10,200-I,1
720 FOR K=1 TO 100:NEXT K
730 NEXT I
740 JUMP=JUMP+1
750 PRINT TAB(5,25);SPC(15);
760 IF JUMP<=10 THEN GOTO 450
770 ENDPROC

780 FOR I=13 TO 5 STEP -1
790 PRINT TAB(I+1,L);" ";
800 PRINT TAB(I,L);CHR$(225);
810 FOR K=1 TO 200: NEXT K
820 NEXT I
830 FOR I=L TO 18
840 PRINT TAB(5,I-1);" ";
850 PRINT TAB(5,I);CHR$(225);
860 SOUND 1,-10,200-I,1
870 FOR K=1 TO 200:NEXT K
880 NEXT I
890 ENDPROC

900 TIME=0
910 RETURN

920 DEF PROCINIT
930 VDU 23,224,0,&FF,&FF,&FF,&FF,&FF,&FF,&FF
```

```
 940 VDU 23,225,&18,&18,&FF,&3C,&3C,&24,&24,&24
 950 VDU 23,226,&FF,&FF,&FF,&FF,&FF,&FF,&FF,&FF
 960 TIME=0
 970 VDU 23;8202;0;0;
 980 *FX 11,0
 990 ENDPROC

1000 DEF PROCFIN
1010 IF JUMP<=10 THEN GOTO 1050
1020 PRINT TAB(0,26);"You FAILD !!!"
1030 *FX 15,1
1040 GOTO 1090
1050 *FX 15,1
1060 PRINT TAB(0,27);"You took ";JUMP;
1070 IF JUMP>1 THEN PRINT " jumps" ELSE PRINT
     " jump"
1080 PRINT "to clear the wall"
1090 PRINT
1100 INPUT "Another game ",A$
1110 IF LEFT$(A$,1)="Y" THEN RUN
1120 CLS
1130 *FX12,0
```

9
Electron Epsom



This is a very simple betting game with an impressive and convincing horse race display plus an appropriate musical accompaniment. The tune is ‘Camptown Races’ and if you’ve ever heard it before, you’re sure to recognise it. If you want to show off the graphics and sound capabilities of your Electron this program provides a good demonstration.

How to play

At the beginning of the game you are allocated a hundred chips. You have to bet on which horse will come in first and must decide how much to stake (the odds are five to one, so if you win having placed 20 you will receive 100). The Electron keeps a tally of your winnings and losses and will tell you if you go broke. During the race, the horse you have backed stands out from the rest of the field − it is the white one. The race is run automatically and so there’s nothing you

can do to make your horse win except cheer it along.

Typing tips

Pay very careful attention to the details of the DATA statements in the function FNTUNE. If you make mistakes in typing in this function, the program may still run but the tune may be unrecognisable or sound discordant! The character between quotes in line 210 is the hash symbol (SHIFT and 3).

Subroutine structure

20	Sets up arrays and defines graphics characters
70	Displays title frame with music playing
110	Sets up second mode
200	Prints course and horses
320	Betting routine
430	Main play loop
530	Plays complete tune
640	Plays one note of tune
820	End of game

Programming details

This is the only program in the collection that plays a tune, so it is worth drawing your attention to the details of lines 650-720. Think of these DATA statements as being made up of a pair of values. The first in each pair relates to the pitch of the note and the second to the length of time it is sounded for. This second value is typically 40 or 20 or l0 representing a minim, a crochet or a semiquaver respectively. The number 999 crops up instead of a pitch value every so often. The effect of this is to cause a rest, or pause in the music. Line 730 signals the end of the tune. After detecting 999, 999 the program resets the DATA statements so that next time round the tune starts playing from

the beginning. At the beginning of the game the tune is produced by calling subroutine 530 which plays all the notes one after the other. While the race is being run, the notes of the tune are produced by a call directly to subroutine 640 which plays one note in the tune and then moves the pointer to the next mte so that the next time it is called the next note is played. To synchronise the sound and the movement, the horses are moved, then subroutine 640 is called to play a note, the horses are moved again, another note is played and so on. The result is a sequence of notes with longer than normal rests between them but, because it does not take much time to move the horses, you still get the impression of a tune being played. You can use this technique in other games but, if the amount of calculation that you have to do between calling each note becomes too long, you'll no longer be able to hear the tune.

Notice that this game uses two display modes. The title frame is in Mode 5 which gives it a bold look. This mode is, however, not suitable for the race itself which is run in Mode 1.

Scope for Improvement

If you get tired of the background music you can subsitute any tune you like. You could use the same graphics for a horse race program in which the player controlled one horse and tried to beat the rest of the field.

Program

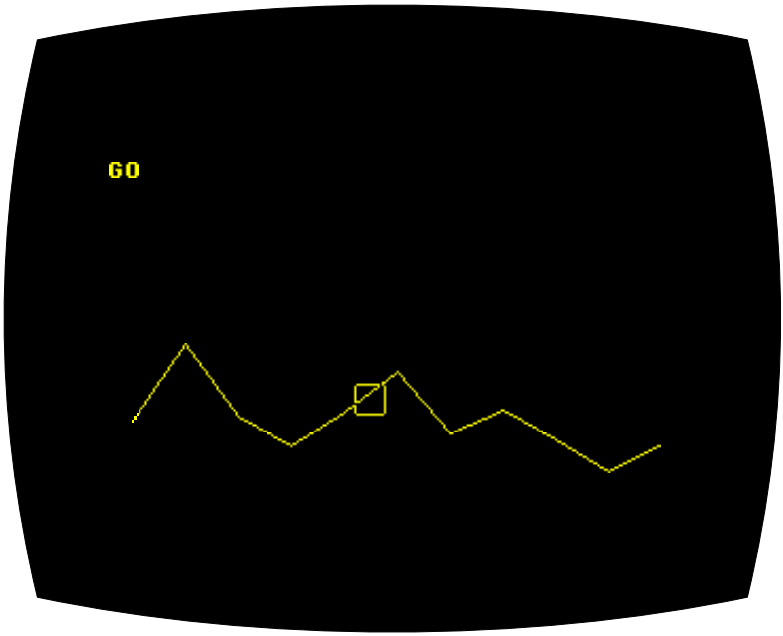
```
10 REM Electron Epsom

20 MODE 5
30 DIM X(5)
40 DIM Y(5)
50 VDU 23,224,&00,&0C,&1A,&FF,&7D,&42,&81,&00
60 VDU 23,226,&0F,&0F,&0F,&0F,&F0,&F0,&F0,&F0

70 CLS
80 PRINT TAB(0,3);"ELECTRON EPSOM"
90 VDU 23;8202;0;0;0;
```

```
100 GOSUB 530

110 MODE 1
120 VDU 19,0,2,0,0,0
130 VDU 19,1,0,0,0,0
140 VDU 19,2,3,0,0,0
150 VDU 19,3,7,0,0,0
160 TTAL=100
170 COLOUR 128
180 COLOUR 1
190 CLS

200 FORX=1 TO 31
210 PRINT TAB(X,1);"#";TAB(X,11);"#"
220 NEXT
230 FORY=2 TO 10
240 PRINT TAB(31,Y);CHR$(226)
250 NEXT
260 COLOUR 2
270 FORY=1 TO 5
280 X(Y)=2
290 Y(Y)=Y*2
300 PRINT TAB(X(Y)-1,Y(Y));Y;CHR$(224)
310 NEXT

320 PRINT TAB(0,14);
330 INPUT"WHICH HORSE DO YOU WANT TO BET
    ON ",B
340 IF B<1 OR B>5 THEN PRINT TAB(0,20);
    "NO SUCH HORSE":GOTO 320
350 PRINT TAB(0,16);
360 INPUT"HOW MUCH DO YOU WANT TO BET",M
370 IF TTAL-M<0 THEN PRINT TAB(0,20);"YOU
    DON'T HAVE ENOUGH MONEY!":SOUND 0,-5,50,20
    :GOTO 350
380 TTAL=TTAL-M
390 PRINT TAB(0,20);STRING$(80," ")
400 VDU 23;8202;0;0;0;
410 COLOUR 3
420 PRINT TAB(X(B),Y(B));CHR$(224)
```

```
430 COLOUR 2
440 TEMPO=5
450 Z=RND(5)
460 PRINT TAB(X(Z),Y(Z));" "
470 X(Z)=X(Z)+RND(2)
480 IFZ=B THEN COLOUR 3 ELSE COLOUR 2
490 PRINT TAB(X(Z),Y(Z));CHR$(224)
500 IF X(Z)>30 THEN GOTO 820
510 IF FNTUNE(P,T,TEMPO)=999 THEN RESTORE
520 GOTO 440

530 TEMPO=6
540 I=1
550 IF FNTUNE(P,T,TEMPO)=999 THEN GOTO 610
560 PRINT TAB(I,20);" ";CHR$(224)
570 FOR Z=1 TO 10
580 NEXT
590 I=I+.2
600 GOTO 550
610 PRINT TAB(I+1,15);" "
620 RESTORE
630 RETURN

640 DEF FNTUNE(P,T,TEMPO)
650 DATA 89,20,89,20,89,20,77,20,89,20,97,
    20,89,20,77,20,-999,20,77,20,69,60,77,
    20,69,40
660 DATA 89,20,89,20,77,20,89,20,97,20,89,20,
    77,20
670 DATA -999,20,77,10,69,10,61,10,69,10,77,
    20,69,20,61,60
680 DATA -999,20,-999,20,61,30,61,10,77,20,
    89,20,109,60
690 DATA -999,20,97,30,97,10,109,20,97,20,
    89,60
700 DATA 77,10,81,10,89,20,89,20,77,10,77,10
710 DATA 89,10,89,10,97,20,89,20,77,40
720 DATA 69,20,77,20,81,10,77,20,69,10,69,10,
    61,60
730 DATA 999,999
740 READ P,T
750 IFP=999 THEN GOTO 810
760 T=T/TEMPO
770 IF P=-999 THEN P=0:AM=0 ELSE AM=-10
```

```
780 SOUND 1,AM,P,T
790 IF ADVAL(-6)<>15 THEN GOTO 790
800 SOUND 1,0,0,1
810 =T
820 IF Z=B THEN PRINT TAB(0,18);" YOU WIN ";
    INT(5*M):TTAL=TTAL+INT(5*M)
830 IF Z<>B THEN PRINT TAB(0,18);
    " YOU LOSE ";M
840 IF TTAL<=0 THEN PRINT TAB(0,18);"YOU'RE
    BROKE   ":FOR Z=1 TO 5000:NEXT:CLS:
    GOTO 910
850 PRINT "YOU HAVE ";TTAL
860 INPUT "ANOTHER RACE Y/N",A$
870 A$=LEFT$(A$,1)
880 RESTORE
890 CLS
900 IF A$="Y"THEN GOTO 170
910 VDU 20
```

10
Guideline



This is a game of skill in which you have to guide a square along an irregular wavy wire. When you play this game with a real wire and a ring, it's a test of how steadily you can guide the ring along the wire. In this Electron version it's a matter of speed as well as hand and eye co-ordination. The square moves from left to right across the screen automatically but you have to keep it on course by pressing the up and down arrow keys. The object of the game is to be on target for as much as the length of the wire as possible and at the end of the game the percentage of time you were successful is displayed.

How to play

At the beginning of the game you have to select the level of difficulty of the game. This determines the size of the square that has to be guided along the wire. Once the square appears, a tone sounds to indicate the start of the game and then it automatically starts to move

right. Press the up and down arrow keys to change its direction. Keeping your finger down on an arrow key will keep the square moving in the same direction.

Typing tips

For this game the rate at which keys auto-repeat is altered to be extremely rapid. This can cause problems if you either break out a game while it is playing or if you try to RUN the program before it is working perfectly. One way to overcome the problem of having your keyboard run wild is to define one of the user-definable functions keys to restore it to normality. For example, try:

```
*KEY 1 *FX 12,0| M
```

Of course, you need to have done this before you run into trouble. The alternative method is to press BREAK and then type OLD.

Subroutine structure

20	Initialisation routine
140	Call to main play loop
150	End of game
250	Initialises variables and prints title frame
480	Plots wire
590	Draws and moves square and calculates amount on target

Programming details

High resolution graphics are used in this program to give the smooth movement needed to test the players' skill. Subroutine 480 is

responsible for plotting the line for the wire and the square is constructed by a collection of PLOT statements at the beginning of subroutine 590. The POINT function is used in this subroutine to test whether the square ring is actually on target around the wire.

Notice the use of *FX 15,1 in line 670. This call flushes the input buffer to ensure that the computer responds to the current keypress and not to one that was stored earlier. The same call is used in the end game routine to empty the buffer before the player responds to the "Another game" question.

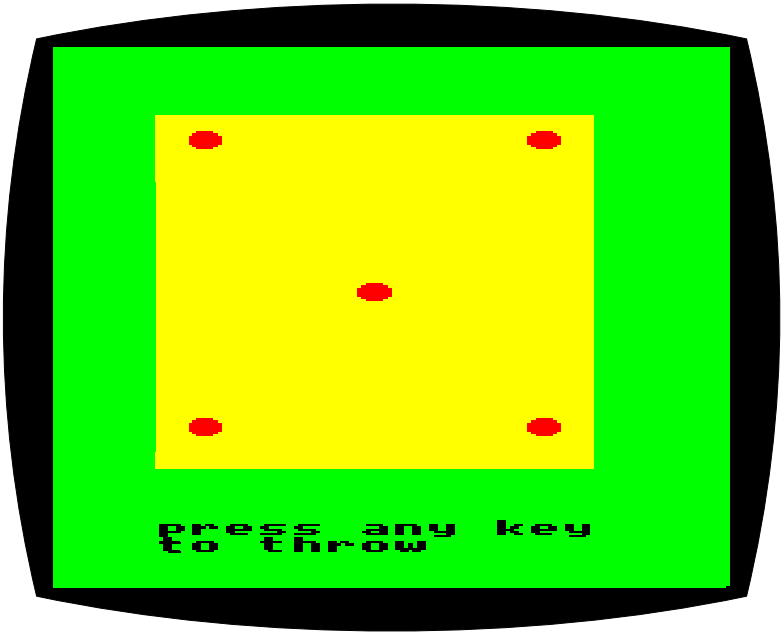
Program

```
 10 REM Guideline
 20 GOSUB 250
 30 *FX 4,1
 40 *FX 11,1
 50 *FX 12,1
 60 VDU 23;8202;0;0;0;
 70 GOSUB 480
 80 FOR Q=1 TO 1000:NEXT Q
 90 PRINT TAB(0,1);"GO"
100 SOUND 1,-5,50,10
110 Y=500:  X=48
120 B=Y:  R=60-DF
130 R2=R/2:  V=10

140 GOSUB 590

150 GCOL 4,4
160 *FX 12,0
170 *FX 4,0
180 *FX 15,1
190 PRINT TAB(0,1);"You were on target  ";
    INT(HIT/(HIT+MISS)*10000)/100;"% of
    the time"
200 INPUT "Another game ",A$
210 IF LEFT$(A$,1)="Y" THEN RUN
220 IF LEFT$(A$,1)<>"N" THEN GOTO 200
230 CLS

240 STOP
```

```
250 X=50
260 Y=500
270 D=148
280 P=0
290 MODE 1
300 VDU 19,0,3,0,0,0
310 VDU 19,3,0,0,0,0
320 CLS
330 PRINT TAB(10,2);"G U I D E L I N E"
340 PRINT TAB(4,10);"You must guide a ring
    along the"
350 PRINT TAB(4);"wavy 'wire' using the
    up and"
360 PRINT TAB(4);"down arrow keys"
370 PRINT 'TAB(4);"You will be marked on how"
380 PRINT TAB(4);"accurate you are"
390 PRINT TAB(4,25);
400 INPUT "Select the difficulty level"'
    "    1 (easy) to 5 (difficult) ",DF
410 IF DF<1 OR DF>5 THEN GOTO 390
420 COLOUR 3
430 COLOUR 132
440 DF=DF*4
450 CLS
460 GCOL 4,4
470 RETURN

480 PLOT 4,X,Y
490 FOR I=1 TO 10
500 R=D-RND(2*D)-1
510 Y=Y+R
520 IF Y>800 THEN  Y=Y-R:  R=-R
530 IF Y<50 THEN  Y=Y+R:  R=-R
540 PLOT 9,100,R
550 NEXT I
560 HIT=0
570 MISS=0
580 RETURN

590 PLOT 4,X-R2,Y-R2
600 PLOT 1,0,R
610 PLOT 1,R,0
620 PLOT 1,0,-R
630 PLOT 1,-R,0
```

```
640 A$=INKEY$(0)
645 FOR I=1 TO 100:NEXT
650 IF ASC(A$)=&8A AND Y-R2>2*V THEN B=B-2*V
660 IF ASC(A$)=&8B AND Y+R<900-2*V THEN
    B=B+2*V
670 *FX 15,1
680 PLOT 4,X-R2,Y-R2
690 PLOT 1,0,R
700 PLOT 1,R,0
710 PLOT 1,0,-R
720 PLOT 1,-R,0
730 X=X+V
740 IF X>1050 THEN RETURN
750 Y=B
760 F=0
770 FOR I=-R+4 TO R-4
780 IF POINT(X,Y+I)=3 THEN  HIT=HIT+1:
    FOR Q=1 TO 10:NEXT Q:I=R-4:F=1
790 NEXT I
800 IF F=1 THEN GOTO 590
810 MISS=MISS+1
820 SOUND 1,-2,10,5
830 GOTO 590
```

11
Magic Dice



Before the days of the micro, family games usually meant one of two things card games or board games that involved dice. In our family we often couldn't find the dice and we spent ages hunting through draws and cupboards before we could start our game. Equally often, in the excitement of the game, the dice would end up rolling over the floor and our game would be interrupted as we retrieved it from dark corners.

You may think there's no place for your old games of Ludo and Monopoly now you have a Electron to absorb you, but think again. They are actually very enjoyable games for lots of players especially if you don't have to spend too much time hunting for the dice, or worse still, arguing about which way up it actually fell! Such problems can be solved if you let your Electron join in the game and take over from the dice.

Of course, your Magic Dice can become the centre of a game. You can devise gambling games to play against the computer or against other people. After all, dice have been around for thousands of years

so there must be plenty of ideas about how to use them.

However you choose to use the program, it will give you a large clear display on the TV screen — colourful too if you run it on a colour set. Notice the realistic way the dice actually slows down before it comes to a final half and the use of sound effects to emphasize this feature.

How to use the program

Using this program is simplicity itself. Type RUN and, when your Electron prompts, just press any key in order to start the dice rolling and then it carries on rolling for a random number of turns and slows down and stops when it is ready. The tone that sounds when the dice has stopped is longer than the one that accompanies each turn. When you've finished with the program press the BREAK key to stop it running.

Subroutine structure

20	Selects mode and disables cursor
40	Set-up routine
100	Main play loop
260	Prints and unprints dots
430	Draws yellow square for dice
510	Defines graphics characters and sets colours
590	Emits beep sound

Programming details

The essence of a dice program is in generating random numbers. In fact, this program uses random numbers in two ways. Firstly, randomness is used in the conventional way, to determine which face of the dice will show at the next turn − this is done in line 190, which uses the RND function to select 'R', a number between one and six.

This information is then used in the printing subroutine (starting at line 260). The program goes to one of the six line numbers 280, 300, 330, 350, 380 or 400, according to the value of 'R', obeying a *computed* GOTO instruction which has the syntax:

ON (result of arithmetic expression) GOTO

At 280 one dot is printed, at 300 two dots are printed and so on.

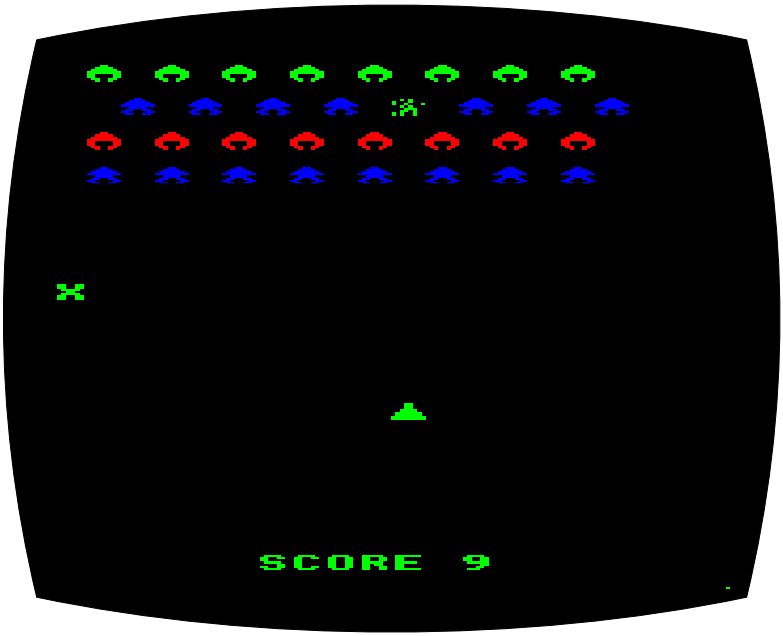
The other use of the random number generator is to give the dice realistic suspense. When you throw a dice it will turn just a few times or quite a number of times, and before it actually stops it will slow down. This program copies both these features by incorporating lines 130 and 150-160. Another random number, 'T', with a value between 5 and 12 is selected. This governs the number of turns the dice makes and the pause before the dots reappear lengthens each time it rolls over.

Program

```
  10 REM Magic Dice

  20 MODE 5
  30 VDU 23;8202;0;0;0;

  40 GOSUB 510
  50 GOSUB 430
  60 COLOUR 128
  70 COLOUR 3
  80 PRINT TAB(3,28);"press any key"
  90 PRINT TAB(3);"to throw"

 100 IF INKEY(0)=-1 THEN GOTO 100
 110 COLOUR 128+1
 120 COLOUR 2
 130 T=RND(7)+5
 140 FOR I=1 TO T
 150 TIME=0
 160 REPEAT:UNTIL TIME>2+I*8
 170 COLOUR 1
 180 GOSUB 260
 190 R=RND(6)
 200 COLOUR 2
 210 GOSUB 260
```

```
220 NEXT I
230 *FX 15,1
240 SOUND 1,-10,0,4
250 GOTO 100
260 GOSUB 590
270 ON R GOTO 280,300,330,350,380,400
280 PRINT TAB(9,14);CHR$(224);
290 RETURN
300 PRINT TAB(4,5);CHR$(224);
310 PRINT TAB(14,22);CHR$(224);
320 RETURN
330 GOSUB 280
340 GOTO 300
350 PRINT TAB(4,22);CHR$(224);
360 PRINT TAB(14,5);CHR$(224)
370 GOTO 300
380 GOSUB 350
390 GOTO 280
400 PRINT TAB(4,14);CHR$(224);
410 PRINT TAB(14,14);CHR$(224);
420 GOTO 350

430 COLOUR 1
440 FOR I=1 TO 21
450 PRINT TAB(3,3+I);STRING$(13,CHR$(225))
460 NEXT I
470 R=1
480 COLOUR 128+1
490 COLOUR 2
500 GOTO 260

510 VDU 23,224,&3C,&7E,&FF,&FF,&FF,&FF,&7E,&3C
520 VDU 23,225,&FF,&FF,&FF,&FF,&FF,&FF,&FF,&FF
530 VDU 19,0,2,0,0,0
540 VDU 19,1,3,0,0,0
550 VDU 19,2,1,0,0,0
560 VDU 19,3,0,0,0,0
570 CLS
580 RETURN

590 SOUND 1,-10,0,1
600 RETURN
```

12
Positron Invaders



Invaders from the war-like planet Positron are heading towards earth in two types of alien ships and you and your Electron have to defend civilisation as we know it. Your task is daunting you have to ensure that none of the aliens get within firing range of earth. The point of no return is marked with a ‘X’ on the left of the screen. Once any of the advancing ships pass this point the game is over − you will have lost. Your only hope is to wipe out all the aliens with your missiles. Every missile that hits its target increases your chance of saving the world.

This game is especially exciting to play because of the use of a two-tone throbbing sound that decreases in pitch as the invaders get closer. If you play this game on a colour TV you’ll see that the ships are red and yellow.

How to play

You can move your missile launcher to left and right using the apropriate arrow keys. Press the up arrow key to fire a missile. You score points for every alien you hit, the ones further away from you xmnting for more points than the nearer ones. The game is over when you have destroyed all the invaders or when the nearest remaining ones reach the point marked by the X.

Subroutine structure

20	Sets up screen and defines graphics characters
150	Initialises variable
220	Initialises strings
270	Main play loop
580	End of game
740	Moves and fires missile
880	Fire routine
1250	Tests whether alien hit
1360	Moves invaders to left and right

Programming details

The alien ships are stored in strings. The front row (line 220) consists of eight CHR$(224) with blanks after each of them. The second row (line 230) consists of eight CHR$(225) arranged, by the simple device of printing a blank before each graphics character, so that the ships are staggered in relation to those in the first row. The third row (line 240) repeats the first and the back row (line 250) repeats the second. Line 260 sets up a string of blank spaces equivalent to the length of each of the rows of invaders. This is used in lines 520 to 570 to detect whether any of the strings still contain aliens when they reach the critical screen location marked by the ‘X’.

The ways the four rows of ships move is interesting as it increases the difficulty of the game. The front row moves to the right, while the second row moves to the left and the back to rows oscillate from side to side. Line 1260 detects when an alien invader is hit. When this

happens its position in the string is replaced by a blank space. This manipulation is carried out in line 1270 which divides the string at the appropriate point, inserts a space in place of the destroyed ship and then re-joins the two halves of the string.

One other point to note in this program is the way in which the sound buffer is flushed in line 3 l0 by the presence of a I as the third parameter of the first section of the SOUND command. This has the effect of synchronising the noise of the ships approaching closer with the strings moving one position closer. The same technique is used in line 1320 so that a new explosion sound can commence each time an alien is hit.

Scope for improvement

You might like to add a routine to make the aliens shoot back at random so that the missile launcher faced the added problem of dodging enemy fire. If you want to make the game easier by lengthening the time before the alien ships move forward a row, increase the value to the right of the > sign in line 310.

Program

```
 10 REM Positron Invaders

 20 MODE 5
 30 VDU 19,0,3,0,0,0
 40 VDU 19,1,4,0,0,0
 50 VDU 19,2,1,0,0,0
 60 VDU 19,3,2,0,0,0
 70 *FX 4,1
 80 *FX 12,1
 90 *FX 11,1
100 VDU 23;8202;0;0;0;
110 VDU 23,224,&18,&3C,&7E,&FF,&C3,&C3,&66,&24
120 VDU 23,225,&18,&3C,&7E,&FF,&3C,&66,&C3,&66
130 VDU 23,226,&18,&18,&18,&3C,&7E,&7E,&FF,&FF
140 VDU 23,227,&28,&88,&91,&28,&1C,&34,&A4,&A4

150 Y=1
```

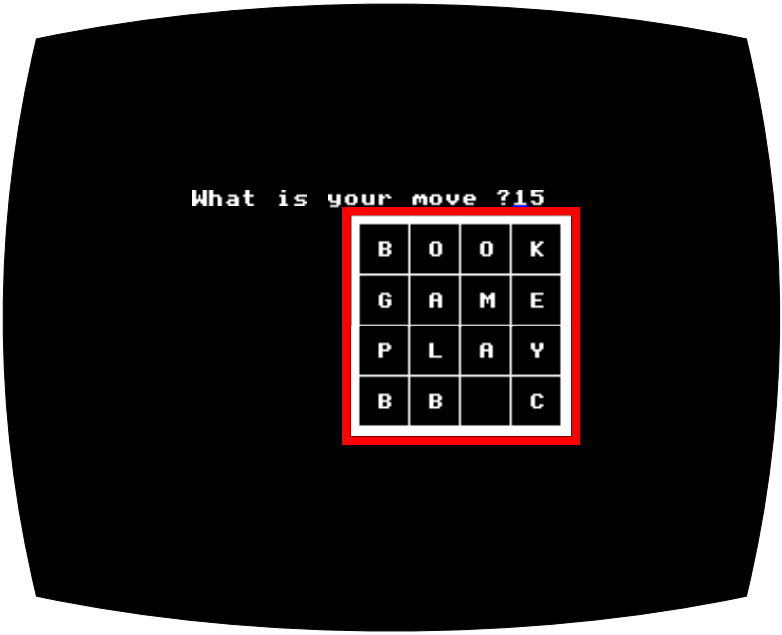
```
160 XL=10
170 YM=0
180 T=0
190 S=0
200 J=0
210 K=0

220 A$=STRING$(8,CHR$(224)+" ")
230 B$=STRING$(8," "+CHR$(225))
240 C$=A$
250 D$=B$

260 E$=STRING$(16," ")
270 COLOUR3
280 PRINT TAB(0,14);"X"
290 J=NOT J
300 IF K=1 GOTO600
310 IF T>30+RND(15) THEN PRINT TAB(1,Y);E$:
    Y=Y+2:T=0:SOUND &0011,0,0,1
320 IF J THEN A$=FNM(A$) ELSE A$=FNR(A$)
330 COLOUR 3
340 PRINT TAB(1,Y);A$
350 SOUND 1,-5,121-Y*8,5
360 PROCMOVE
370 IF J THEN B$=FNM(B$) ELSE B$=FNR(B$)
380 COLOUR 1
390 PRINT TAB(1,Y+2);B$
400 SOUND1,-5,129-Y*8,5
410 PROCMOVE
420 C$=FNM(C$)
430 COLOUR 2
440 PRINT TAB(1,Y+4);C$
450 SOUND1,-5,121-Y*8,5
460 PROCMOVE
470 D$=FNR(D$)
480 COLOUR 1
490 PRINT TAB(1,Y+6);D$
500 SOUND1,-5,129-Y*8,5
510 PROCMOVE
520 IF Y>8 AND D$<>E$ THEN GOTO 580
530 IFY>10 AND C$<>E$ THEN GOTO 580
540 IFY>12 AND B$<>E$ THEN GOTO 580
550 IF Y>14 AND A$<>E$ THEN GOTO 580
560 T=T+1
```

```
570 GOTO270

580 PRINT TAB(1,23);" THEY GOT YOU!!"
590 GOTO 610
600 PRINT TAB(1,23);"WELL DONE !"'
    "YOU SAVED THE WORLD!"
610 *FX 15,1
620 *FX 4,0
630 *FX 12,0
640 SOUND &0011,0,0,1
650 IF K=0 THEN SOUND &0010,-10,4,20
660 INPUT "ANOTHER GAME Y/N",A$
670 A$=LEFT$(A$,1)
680 IF A$="Y" THEN RUN
690 *FX 4,0
700 *FX 12,0
710 VDU 20
720 MODE 7
730 END

740 DEF PROCMOVE
750 A=INKEY(0)
760 *FX 15,1
770 T=T+1
780 COLOUR 3
790 PRINT TAB(XL,21);CHR$(226)
800 IF A=-1 THEN ENDPROC
810 PRINT TAB(XL,21);" "
820 IF A=&88 AND XL>1 THEN XL=XL-1
830 IF A=&89 AND XL<16 THEN XL=XL+1
840 COLOUR 3
850 PRINT TAB(XL,21);CHR$(226)
860 IF A=&8B THEN PROCFIRE
870 ENDPROC

880 DEF PROCFIRE
890 COLOUR 3
900 FOR M=19 TO Y+6 STEP -1
910 PRINT TAB(XL,M);":";
920 PRINT TAB(XL,M+1);" "
930 NEXT
940 PRINT TAB(XL,M+1);" "
950 F=0
960 Q$=D$
```

```
 970 R=6
 980 PROCHIT
 990 D$=Q$
1000 IF F=1 GOTO 1220
1010 COLOUR 3
1020 PRINT TAB(XL,Y+5);".";TAB(XL,Y+5);" ";
     TAB(XL,Y+4);".";TAB(XL,Y+4);" "
1030 Q$=C$
1040 R=4
1050 PROCHIT
1060 C$=Q$
1070 IF F=1 THEN GOTO 1220
1080 COLOUR 3
1090 PRINT TAB(XL,Y+3);".";TAB(XL,Y+3);" ";
     TAB(XL,Y+2);".";TAB(XL,Y+2);" "
1100 Q$=B$
1110 R=2
1120 COLOUR 3
1130 PROCHIT
1140 B$=Q$
1150 IF F=1 GOTO 1220
1160 PRINT TAB(XL,Y+1);".";TAB(XL,Y+1);" ";
     TAB(XL,Y);".";TAB(XL,Y);" "
1170 Q$=A$
1180 R=0
1190 COLOUR 3
1200 PROCHIT
1210 A$=Q$
1220 IF A$=E$ ANDB$=E$ AND C$=E$ AND D$=E$
     THEN K=1
1230 IF Q$=E$ THEN PRINT TAB(1,Y);E$:Y=Y+2
1240 ENDPROC

1250 DEF PROCHIT
1260 IF MID$(Q$,XL,1)=" " THEN ENDPROC
1270 Q$=MID$(Q$,1,XL-1)+" "+MID$(Q$,XL+1)
1280 F=1
1290 S=S+10-Y
1300 COLOUR 3
1310 PRINT TAB(XL,Y+R);CHR$(227)
1320 SOUND &0010,-15,4,3
1330 PRINT TAB(6,30);"SCORE ";S;"  ";
1340 T=T-RND(3)
1350 ENDPROC
```

```
1360 DEF FNM(Q$)=MID$(Q$,2)+LEFT$(Q$,1)
1370 DEF FNR(Q$)=RIGHT$(Q$,1)+
     LEFT$(Q$,LEN(Q$)-1)
```

13
Mirror Tile



Although your Electron opens up lots of new possibilities for games, it's good to know that it can also conjure up old favourites. Mirror Tile is a colourful and versatile version of a game that is conventionally played on a small board with the pieces slotted into one another and into their surrounding frame. This construction is vital to the game, the object of which is to rearrange the pieces to match a given pattern − hence it's title 'Mirror Tile'.

If you have never played with a tile puzzle, look at the illustration of the game's display. You'll see a four-by-four square of letters and you'll notice that one position is empty. In other words there are fifteen letters and one hole. The hole allows you to move the letters around the board.

How to play

Imagine for a moment that the board was really made of plastic

pieces. You could slide a piece that was either above or below, or to either side of the hole, into it, and the position of the piece you moved would then be empty − that is, it would become the hole. Notice that there are only a limited number of possible moves − two, three or four depending on the position of the hole, and that is it impossible to move on the diagonals.

These same rules apply to the Electron version of the game. You can move any letter that is directly to the left or right of the hole or just above or below it. To indicate your choice you type in the number (1 to l6) of the square containing the piece you want to move. If you try to make a wrong move the Electron won't let you. Instead it will be helpful and number each square for you, in case your mistake was due to typing in the wrong number for your choice. If you want to see these numbers displayed, type any letter key − in fact any key other than one for a legal move.

When you R UN this program the first thing it asks you is whether you wish to input your own set of words. If you reply "N" then the square will fill with the letters A to O. If you prefer to select your own starting arrangement you will be asked to type in three four-letter words, and one three-letter word. Of course, this means you can vary the difficulty of the puzzle. If you choose words that repeat some letters the game will actually be easier. For example, typing in ROOF, CATS, RENT and TENT would give a fairly simple game. The most difficult puzzle is one where every letter is different, e.g. HOME, CART, WING, SKY.

Once you've typed in your words, you'll see them being shuffled − the program will already have asked you how many shuffles it should perform and the more it shuffles the more difficult you will find it. Then it's your turn — to sort them out again into the initial arrangement. Your Electron will count your moves and let you know how many you took at the end of the game.

Subroutine structure

20	Defines arrays
50	Sets up game
140	Main play loop
180	End of game

230	Checks for end of game
320	Sets up default (alphabetic) board
460	Prints frame
750	Shuffle routine
900	Prints number overlay
990	Blanks out previous messages
1090	Move logic
1320	Locates empty space
1410	Prints title and initial questions
1550	Performs move
1610	Asks for input of words
1950	Updates position
2010	Defines graphics characters and sets up screen

Programming details

This program is complicated both because of its length and also because it involves a lot of logic. However, because of the way in which it uses BBC BASIC's procedures, its structure is very clear and if you follow it through you should be able to see what happens at every step.

Scope for improvement

Adding to a program that is already as long as this one may seem to be a tall order. However there is acually scope for improvement. A routine that reminded the player of the target arrangement might be a very useful extra.

Program

```
10 REM Mirror Tile

20 MODE 1
30 DIM B(4,4)
40 DIM B$(16),W$(16)
50 PROCTITLE
```

```
 60 PROCLET
 70 IF WR=1 THEN PROCWORDS
 80 MOV=0
 90 ER=0
100 PROCGRAPH
110 PROCFRAME
120 PROCHOLE
130 PROCSHUF

140 PROCMOVE
150 PROCFIN
160 MOV=MOV+1
170 IF FIN<>0 THEN GOTO 140
180 PRINT TAB(0,22);"You did it in ";MOV;
    " moves"
190 INPUT "Another game ",A$
200 IF LEFT$(A$,1)="Y" THEN RUN
210 CLS
220 STOP

230 DEF PROCFIN
240 FIN=0: K=0
250 FOR I=1 TO 4
260 FOR J=1 TO 4
270 K=K+1
280 IF B$(B(I,J))<>W$(K) THEN FIN=1
290 NEXT J
300 NEXT I
310 ENDPROC

320 DEF PROCLET
330 K=0
340 FOR I=1 TO 4
350 FOR J=1 TO 4
360 K=K+1
370 B$(K)=CHR$(64+K)
380 B(I,J)=K
390 NEXT J
400 NEXT I
410 B$(16)=" "
420 FOR I=1 TO 16
430 W$(I)=B$(I)
440 NEXT I
```

```
450 ENDPROC

460 DEF PROCFRAME
470 COLOUR 128+3
480 COLOUR 2
490 FOR I=0 TO 3
500 FOR J=0 TO 3
510 PRINT TAB(10+J*3,4+I*3);SPC(2);CHR$(224)
520 NEXT J
530 FOR J=0 TO 3
540 PRINT TAB(10+J*3,5+I*3);SPC(1);
    B$(B(I+1,J+1));CHR$(224)
550 NEXT J
560 FOR J=0 TO 3
570 PRINT TAB(10+J*3,6+I*3);CHR$(226);
    CHR$(226);CHR$(225)
580 NEXT J
590 NEXT I
600 FOR I=0 TO 11
610 COLOUR 2:COLOUR 128+1
620 PRINT TAB(10+I,3);CHR$(227)
630 COLOUR 1:COLOUR 128+2
640 PRINT TAB(10+I,16);CHR$(227)
650 PRINT TAB(9,4+I);CHR$(228)
660 COLOUR 2:COLOUR 128+1
670 PRINT TAB(22,4+I);CHR$(228)
680 NEXT I
690 PRINT TAB(9,3);CHR$(229)
700 PRINT TAB(22,16);CHR$(230)
710 PRINT TAB(22,3);CHR$(231)
720 PRINT TAB(9,16);CHR$(232)
730 COLOUR 128+3
740 ENDPROC

750 DEF PROCSHUF
760 IS=4: JS=4: IY=0:JY=0
770 FOR D=1 TO S
780 I=IS: J=JS
790 IF RND(1)>.5 THEN GOTO 830
800 I=IS+INT (RND(1)*2)*2-1
810 IF I>4 OR I<1 THEN I=IS:
820 GOTO 850
830 J=JS+INT (RND(1)*2)*2-1
840 IF J>4 OR J<1 THEN J=JS:GOTO 780
```

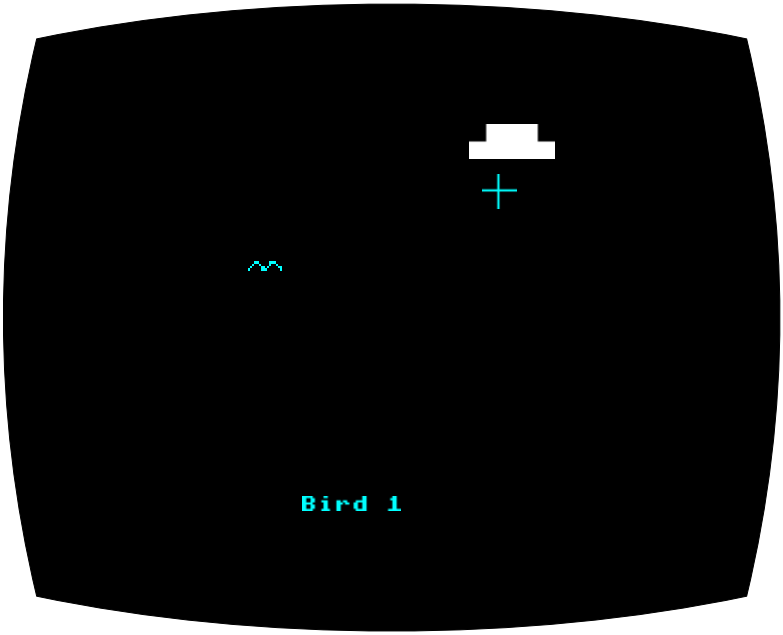
```
 850 IF I=IY AND J=JY THEN GOTO 780
 860 IY=IS: JY=JS
 870 PROCDOMOVE
 880 NEXT D
 890 ENDPROC
 900 DEF PROCOVER
 910 K=0
 920 FOR I=0 TO 3
 930 FOR J=0 TO 3
 940 K=K+1
 950 PRINT TAB(10+J*3,4+I*3);K
 960 NEXT J
 970 NEXT I
 980 ENDPROC

 990 DEF PROCBLANK
1000 FOR L=0 TO 3
1010 FOR P=0 TO 3
1020 PRINT TAB(10+P*3,4+L*3);SPC(2)
1030 NEXT P
1040 NEXT L
1050 IF ER=0 THEN ENDPROC
1060 PRINT TAB(0,20);SPC(64)
1070 ER=0
1080 ENDPROC

1090 DEF PROCMOVE
1100 PRINT TAB(0,2);"What is your move ";
1110 INPUT LINE M$
1120 IF M$="" THEN GOTO 1090
1130 IF LEN(M$)<2 THEN M$="0"+M$
1140 IF MID$(M$,1,1)<"0" OR MID$(M$,1,1)>"9" OR
     MID$(M$,2,1)<"0" OR MID$(M$,2,1)>"9" THEN
     PROCOVER:GOTO 1090
1150 M=VAL(M$)
1160 IF M>0 AND M<17 THEN GOTO 1220
1170 ER=1
1180 PRINT TAB(0,20); "A move must be a number
     between "
1190 PRINT TAB(0,21);"1 and 16 as shown"
1200 PROCOVER
1210 GOTO 1090
1220 I=INT ((M-1)/4)
1230 J=M-I*4
```

```
1240 I=I+1
1250 IF ABS (I-IS)+ABS (J-JS)=1 THEN
     PROCDOMOVE:ENDPROC
1260 SOUND 1,-10,0,3
1270 ER=1
1280 PRINT TAB(0,20);"You can only move a tile
     next to"
1290 PRINT TAB(0,21); "the space";SPC(10)
1300 PROCOVER
1310 GOTO 1090
1320 DEF PROCHOLE
1330 K=0
1340 FOR L=1 TO 4
1350 FOR P=1 TO 4
1360 K=K+1
1370 IF B(P,L)=16 THEN IS=L:JS=P:MS=K
1380 NEXT P
1390 NEXT L
1400 ENDPROC

1410 DEF PROCTITLE
1420 CLS
1430 PRINT TAB(5);"M i r r o r   T i l e"
1440 PRINT TAB(0,10);"Do you want to input
     your"
1450 PRINT "own set of words ";
1460 INPUT A$
1470 IF A$="" THEN GOTO 1460
1480 IF LEFT$(A$,1)="Y" THEN WR=1: GOTO 1510
1490 IF LEFT$(A$,1)="N" THEN WR=0: GOTO 1510
1500 GOTO 1460
1510 PRINT TAB(0,15);"How many shuffles ";
1520 INPUT S
1530 IF S<1 THEN GOTO 1510
1540 ENDPROC

1550 DEF PROCDOMOVE
1560 PROCBLANK
1570 PROCUPDATE
1580 PRINT TAB(11+(J-1)*3,5+(I-1)*3);B$(B(I,J))
1590        PRINT        TAB(11+(JS-1)*3,5+(IS-1)*
3);B$(B(IS,JS))
1600 ENDPROC
```

```
1610 DEF PROCWORDS
1620 CLS
1630 PRINT TAB(0,5);"Choose 3 four-letter words"
1640 PRINT "and 1 three-letter word."
1650 INPUT "Type the first four-letter word ",A$
1660 IF LEN(A$)<>4 THEN GOTO 1650
1670 FOR I=1 TO 4
1680 W$(I)=MID$(A$,I,1)
1690 NEXT I
1700 PRINT "First word= ";A$
1710 INPUT "Type the second four-letter word", A$
1720 IF LEN(A$)<>4 THEN GOTO 1710
1730 FOR I=5 TO 8
1740 W$(I)=MID$(A$,I-4,1)
1750 NEXT I
1760 PRINT "Second word= ";A$
1770 INPUT "Type the third four-letter word ",
     A$
1780 IF LEN(A$)<>4 THEN GOTO 1770
1790 FOR I=9 TO 12
1800 W$(I)=MID$(A$,I-8,1)
1810 NEXT I
1820 PRINT "Third word= ";A$
1830 INPUT "Type the three-letter word ",A$
1840 IF LEN(A$)<>3 THEN GOTO 1830
1850 FOR I=13 TO 15
1860 W$(I)=MID$(A$,I-12,1)
1870 NEXT I
1880 PRINT "Fourth word= ";A$
1890 FOR I=1 TO 1000:NEXT I
1900 W$(16)=" "
1910 FOR I=1 TO 16
1920 B$(I)=W$(I)
1930 NEXT I
1940 ENDPROC

1950 DEF PROCUPDATE
1960 B(IS,JS)=B(I,J)
1970 B(I,J)=16
1980 T=IS: IS=I: I=T
1990 T=J: J=JS: JS=T
2000 ENDPROC

2010 DEF PROCGRAPH
```

```
2020 VDU 23,224,1,1,1,1,1,1,1,1
2030 VDU 23,225,1,1,1,1,1,1,1,&FF
2040 VDU 23,226,0,0,0,0,0,0,0,&FF
2050 VDU 23,227,0,0,0,0,&FF,&FF,&FF,&FF
2060 VDU 23,228,&F0,&F0,&F0,&F0,&F0,&F0,&F0,&F0
2070 VDU 23,229,0,0,0,0,&0F,&0F,&0F,&0F
2080 VDU 23,230,&F0,&F0,&F0,&F0,0,0,0,0
2090 VDU 23,231,0,0,0,0,&F0,&F0,&F0,&F0
2100 VDU 23,232,&0F,&0F,&0F,&0F,0,0,0,0
2110 VDU 19,0,7,0,0,0
2120 VDU 19,1,1,0,0,0
2130 VDU 19,2,4,0,0,0
2140 VDU 19,3,3,0,0,0
2150 COLOUR 128+3
2160 COLOUR 2
2170 CLS
2180 ENDPROC
```

14
Pot Shot



This game provides the ideal type of target practice. However many times you score a direct hit, the magic bird continues to fly on, allowing you to take aim and fire again and again. The elements of this game are simple − a blue sky with a single white cloud, a bird winging its way from left to right and your rifle sight. The object is straightforward − to line up the sight with the bird and shoot it − but in practice its not that simple. Every time you fire, your rifle ‘kicks’ to one side or the other so you have to re-align your sight for your next shot and the bird (and your sight too) disappears behind the cloud when they reach it. A total of five birds fly across the sky and there is no limit to the number of hits you can score − your total for each bird and the whole game are displayed at the end. There is a convincing sound every time you fire your rifle and a distinctive bell-like sound when you hit the bird.

How to play

To hit the target you have to line the cross point of your sight up with the centre of the bird. Use all four arrow keys to move your sight and press ‘F’ to fire. There are five birds in all and you may hit each one as often as you can.

Subroutine structure

20	Sets mode and call initialisation routine
40	Main play loop
110	Plots cross
180	Moves sight, calls firing routine and moves bird
390	Draws cloud and sets up display
580	Draws bird
650	Shoots, tests for hit and recoils sight
720	Sets up screen, defines graphics characters and envelope
940	End of game

Programming details

The very realistic rifle shot sound is produced using the ENVELOPE command defined in line 920. It is used in the SOUND statement in line 660 where it modifies a noise produced on channel 0 (the noise channel). The same envelope is then used in line 690 where it modifies a tone produced on channel I to give the bell sound that indicates a hit. The use of high resolution graphics in this program means that the range and smoothness of both the bird and the rifle sight is better than could be achieved with low resolution graphics. The function POINT is used in line 670 to discover if the target has been hit. Notice the extensive use of GCOL 4,4 to plot and unplot high resolution shapes. It is also interesting to note the way the bird’s flight path is calculated and stored in the array B in line 840 for use later in the program.

Scope for improvement

To make this program even more of a challenge you could add more than one cloud and allow the bird to have more than one path across the sky.

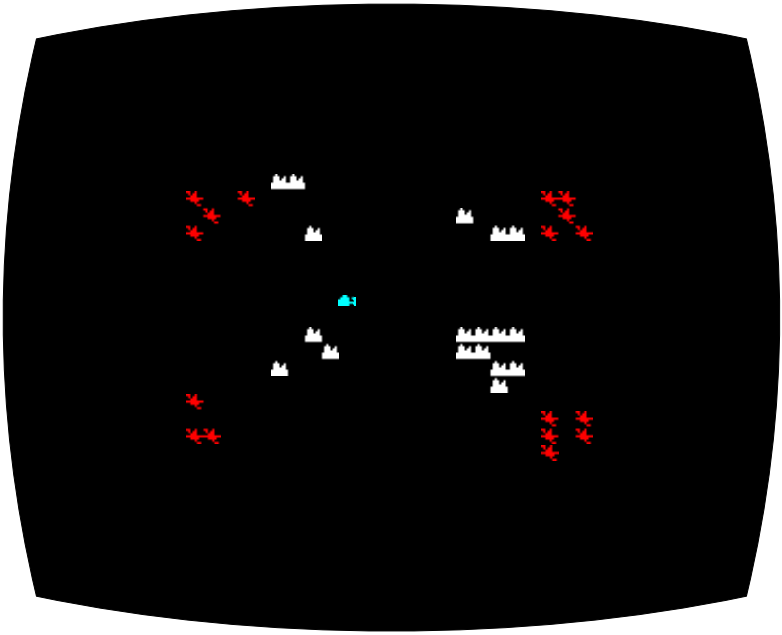
Program

```
 10 REM Pot Shot

 20 MODE 1
 30 PROCINIT

 40 FOR G=1 TO 5
 50 PROCSTART
 60 PRINT TAB(5,25); "Bird ";G;
 70 PROCMOVE
 80 NEXT G
 90 GOTO 940
100 STOP

110 DEF PROCCROSS
120 PLOT 4,X-30,Y
130 PLOT 1,60,0
140 PLOT 4,X,Y-30
150 PLOT 1,0,60
160 PLOT 64+5,X,Y
170 ENDPROC

180 DEF PROCMOVE
190 A$=INKEY$(0)
200 *FX 15,1
210 GCOL 4,4
220 PROCCROSS
230 J=B(I)
240 PROCBIRD
250 IF ASC(A$)=&88 AND X>38 THEN X=X-8
260 IF ASC(A$)=&8A AND Y>250 THEN Y=Y-8
270 IF ASC(A$)=&8B AND Y<1000 THEN Y=Y+8
280 IF ASC(A$)=&89 AND X<1268 THEN X=X+8
290 I=I+.75
300 J=B(I)
```

```
310 GCOL 4,4
320 PROCBIRD
330 IF A$="F" THEN PROCFIRE
340 *FX 15,1
350 PROCCROSS
360 IF FIN=1 THEN FIN=0:ENDPROC
370 FOR Z=1 TO 100:NEXT Z
380 GOTO190
390 DEF PROCSTART
400 COLOUR 128+1
410 CLS
420 COLOUR 128
430 J=RND(5)
440 R=RND(10)-1
450 PRINT TAB(R+9,J);SPC(3);
460 PRINT TAB(R+8,J+1);SPC(5)
470 COLOUR 128+1
480 FIN=0
490 X=RND(300)+200
500 Y=800
510 I=3
520 COLOUR 2
530 GCOL 4,4
540 PROCCROSS
550 J=B(I)
560 PROCBIRD
570 ENDPROC

580 DEF PROCBIRD
590 VDU 5
600 MOVE I*5,J
610 PRINT CHR$(224);CHR$(225);
620 VDU 4
630 IF I>245 THEN LET FIN=1
640 ENDPROC

650 DEF PROCFIRE
660 SOUND &10,1,5,2
670 IF POINT(X,Y)<>2 THEN X=X+50-RND(100):
    ENDPROC
680 X=X+50-RND(100)
690 SOUND &11,1,100,3
700 H(G)=H(G)+1
710 ENDPROC
```

```
 720 DEF PROCINIT
 730 HIT=0
 740 COLOUR 2
 750 VDU 23;8202;0;0;0;
 760 CLS
 770 PRINT TAB(11,10);"P O T  S H O T"
 780 VDU 19,0,7,0,0,0
 790 VDU 19,1,6,0,0,0
 800 VDU 19,2,0,0,0,0
 810 VDU 19,3,7,0,0,0
 820 DIM B(254)
 830 FOR I=1 TO 250
 840 B(I)=1000-(125-I)*(125-I)/40
 850 NEXT I
 860 DIM H(5)
 870 VDU 23,224,&00,&00,&00,&00,&18,&24,&43,&83
 880 VDU 23,225,&00,&00,&00,&00,&38,&24,&43,&81
 890 *FX 4,1
 900 *FX 11,1
 910 *FX 12,1
 920 ENVELOPE 1,1,0,0,0,0,0,0,126,-1,0,-3,
     126,126
 930 ENDPROC

 940 CLS
 950 T=0
 960 FOR I=1 TO 5
 970 PRINT TAB(10,I+5);"Bird ";I;" HIT ";H(I)
 980 T=T+H(I)
 990 NEXT I
1000 *FX 12,0
1010 *FX 15,1
1020 PRINT TAB(10,12);"Total hits= ";T
1030 PRINT TAB(10,14);
1040 INPUT " Another game Y/N ",A$
1050 IF A$="Y" THEN RUN
1060 *FX 4,0
```

15
Save The Whale



This is a moving graphics game for conservationists! The object of the game is to ensure that the whale survives to swim on in arctic seas. You have to outwit the eskimos who are hunting the whale in their kayaks. If they run into the icebergs they will have to abandon their hunt so you must lure them towards these obstacles by moving the whale in such a way that, in approaching it, the eskimos crash.

How to play

At the beginning of the game you can select the difficulty level for your turn. Y our selection governs the starting positions the icebergs and so makes the game easier or harder to play. To move the whale you press any of the arrow keys. If any kayak runs into an iceberg the kayak vanishes, if the whale runs into an iceberg the iceberg vanishes this of course reduces his protection so its not advisable except in extreme circumstances − and if an eskimo reaches the whale he

harpoons the whale and kills him. The game is over when all the eskimos have been removed from play or when the whale is dead.

Subroutine structure

20	Sets up graphics characters and arrays
150	Title frame
250	Sets up screen and turns graphics cursor off
280	Prints kayaks
350	Prints icebergs
410	Prints whale
450	Main play loop
520	Checks for game over
540	Move whale routine
680	Move kayaks routine
790	End of game

Programming details

The initial positions of the kayaks are set at random within a band at the edges of the screen (lines 290 and 300). The initial positions of the icebergs are set in a similar fashion (fines 370 and 380) but account is also taken of the difficulty factor, ‘D’ input at 230. The POINT function is used in line 740 to detect whether a kayak has landed on an iceberg in which case that is the end of the kayak. The alternative method of simply comparing co-ordinates is used in line 750 to detect whether a kayak has landed on the whale − in which case that is the end of the whale (and the game).

Program

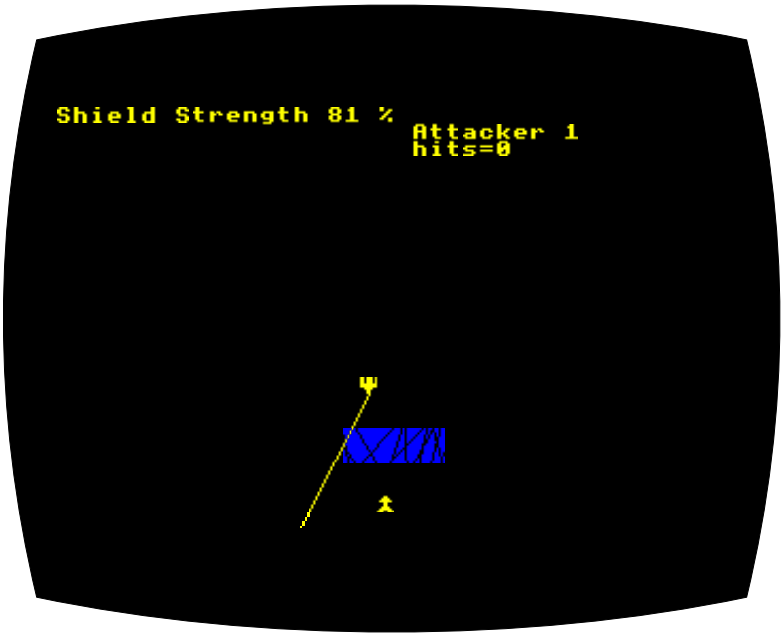
```
10 REM Save the Whale

20 MODE 1
30 *FX 4,1
40 DIM X(20)
50 DIM Y(20)
```

```
 60 DIM U(20)
 70 DIM V(20)
 80 VDU 23,224,&00,&00,&30,&7B,&F9,&FF,&F9,&00
 90 VDU 23,225,&00,&20,&72,&76,&7E,&FF,&FF,&FF
100 VDU 23,226,&00,&C8,&58,&38,&FF,&3C,&08,&06
110 VDU 19,0,6,0,0,0
120 VDU 19,1,7,0,0,0
130 VDU 19,2,1,0,0,0
140 VDU 19,3,0,0,0,0

150 CLS
160 PRINT TAB(2,2);"S A V E  T H E  W H A L E"
170 PRINT ''''"In this game, you, the
    whale    ";CHR$(224)
180 PRINT '"must outwit the eskimoes hunting"
190 PRINT '"you in their kayaks    ";CHR$(226)
200 PRINT '"by luring them onto the
    icebergs   ";CHR$(225)
210 PRINT TAB(0,20);"WHICH DIFFICULTY LEVEL
    DO YOU WISH TO"
220 PRINT '"PLAY AT"
230 INPUT '"(1) EXPERT,(2) INTERMEDIATE,
    (3) NOVICE ",D
240 IF D<1 OR D>3 THEN GOTO 230

250 CLS
260 VDU 23;8202;0;0;0;
270 COLOUR 2

280 FOR C=1 TO 20
290 X=SGN(RND(1)-.5)*(RND(4)+10)+15
300 Y=SGN(RND(1)-.5)*(RND(4)+6)+11
310 PRINT TAB(X,Y);CHR$(226)
320 X(C)=X
330 Y(C)=Y
340 NEXT C

350 COLOUR 1
360 FOR C=1 TO 20
370 U(C)=SGN(RND(1)-.5)*(RND(4)+4-D)+15
380 V(C)=SGN(RND(1)-.5)*(RND(4)+3-D)+9
390 PRINT TAB(U(C),V(C));CHR$(225)
400 NEXT C
```

```
410 COLOUR 3
420 X=INT(RND(2)+10)
430 Y=INT(RND(2)+10)
440 PRINT TAB(X,Y);CHR$(224)

450 GOSUB 540
460 F=0
470 FOR C=1 TO 20
480 IF X(C)=0 THEN GOTO 510
490 F=1
500 GOSUB 680
510 NEXT C

520 IF F=0 THEN GOTO 830
530 GOTO 450

540 SOUND 0,-15,3,10
550 Z=X:V=Y
560 *FX 15,1
570 A=INKEY(0)
580 IF A=-1 THEN GOTO 570
590 IF A=&88 AND X>1 THEN X=X-1
600 IF A=&89 AND X<31 THEN X=X+1
610 IF A=&8A AND Y<21 THEN Y=Y+1
620 IF A=&8B AND Y>1 THEN Y=Y-1
630 COLOUR 4
640 PRINT TAB(Z,V);" "
650 COLOUR 3
660 PRINT TAB(X,Y);CHR$(224)
670 RETURN

680 PRINT TAB(X(C),Y(C));" "
690 E=0
700 E=SGN(X(C)-X)
710 X(C)=INT(X(C)-E)
720 E=SGN(Y(C)-Y)
730 Y(C)=INT(Y(C)-E)
740 IF POINT(X(C)*32+16,1023-32*Y(C)-18)=1
    THEN X(C)=0:SOUND 2,-15,200,1:GOTO 780
750 IF X(C)=X AND Y(C)=Y THEN GOTO 790
760 COLOUR 2
770 PRINT TAB(X(C),Y(C));CHR$(226)
780 RETURN
```

```
790 COLOUR 3
800 PRINT TAB(X(C),Y(C));" "
810 PRINT TAB(1,20);" YOU WERE KILLED"
820 GOTO 840
830 PRINT TAB(1,20);"YOU ESCAPED THIS TIME"
840 INPUT "ANOTHER GAME (Y/N)",A$
850 A$=LEFT$(A$,1)
860 IF A$="Y" THEN RUN
870 *FX 4,0
880 MODE 7
890 CLS
900 STOP
```

16
Mighty Missile



Your weapon can destroy anything anything that it actually hits. So the only problem in this game is to ensure that the missile finds its target, quickly and accurately. The enemy ships sweep in from the left and the right firing relentlessly. Your missile is only vulnerable if its protective shield is eroded away and then it can be easily blasted in its home base. Otherwise, it is impervious to enemy fire, even outside its base. If it hits an enemy it will explode on contact but if it fails to find its target it will disintegrate as it reaches the upper atmosphere. You can launch ten missiles and there are ten enemy ships. Each ship maintains a stable orbit until you actually take a shot at it so you can wait in the base while deciding which side to fire from, and when to fire − except that with every orbit more of your shield is blasted away by enemy fire and once there is only 20 per cent of it left you will no longer have any protection from the enemies' lasers. This fast moving graphics game is enhanced by sound effects and is quite compulsive to play.

How to play

In this game it is important to notice how much 'Shield strength' you have left since when the figure displayed drops to 20% you will be vunerable to attack. Once the shield is so eroded the enemy laser is able to home in and destroy you wherever they hit you, including inside the missile base. The object of the game is to score as many hits as possible so it is worth watching each new enemy ship's orbit at least once or twice before you try to shoot it down. To fire you have to leave your base. Do this by pressing the right or left arrow key. This will take you to a fixed position on the right or left of the screen and launch the missile. Remember to take into account the time it will take for your missile to reach the enemy ship which will continue on its path! At the end of the game your score is displayed and you are given the option of another game.

Typing tips

In line 690, notice the space after the percentage sign and before the double quotes. This serves the important function of blanking out previous figures as the number displayed gets smaller and so occupies fewer positions on the screen.

Subroutine structure

20	Set-up
70	Main play loop
310	End of game
380	Prints attacker and fires laser
500	Checks to see if player has activated missile
620	Calculates shield strength
710	Moves and fires missile and tests for hit or miss
780	Explosion graphics and sound
940	Sets up attack orbit
1070	Prints shield
1200	Defines graphics characters and sets up screen display

Programming details

As this program has a clear structure and uses separate procedures for most of its major elements, you should find it relatively easy to follow. One interesting point to note is the way in which the path of the attacking ship is calculated in procedure PROCPATH and stored in a pair of arrays X and Y to be used repeatedly for the various orbits used during the game. A second point of interest is that although all the graphics used are low resolution graphics the laser zap from the attacking spaceship is a high resolution graphics command which will blank out any black points that it passes through. So although the shield is initially printed using low resolution blocks it is whittled away by the laser beam passing through it. The strength of the shield is estimated, in subroutine 620, by the number of black points left in the shield, using the POINT function. POINT is 1 if the point at the x,y co-ordinate is black and 0 if it is blue.

Program

```
10 REM Mighty missile

20 MODE 1
30 PROCINIT
40 PROCPATH
50 PROCBLOCK
60 PROCSTREN
```

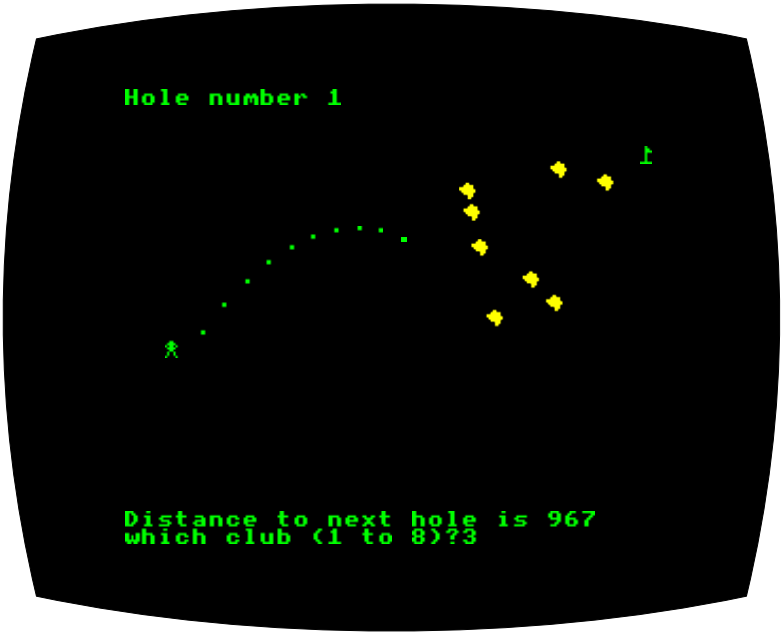
```
 70 FOR A=1 TO 10
 80 DIR=SGN (RND-.5)
 90 PRINT TAB(21,1);"Attacker ";A
100 PRINT TAB(21,2);"hits=";HIT
110 R=7-RND(14)
120 IF R<0 THEN S=3-R: E=38
130 IF R>=0 THEN S=3: E=38-R
140 IF DIR=-1 THEN T=S: S=E: E=T
150 FOR I=S TO E STEP DIR
160 PROCENEMY
170 PROCFIRE
180 IF F=1 THEN PROCGUIDE
190 NEXT I
200 PRINT TAB(X(I),Y(I+R));" ";
210 IF F=1 THEN FIN=1
220 IF FIN=2 THEN A=11: GOTO 300
230 PROCSTREN
240 IF F=1 THEN F=0: PRINT TAB(MX,MY);
    CHR$(226):SOUND 0,-15,5,5:FOR Q=1 TO 1000:
    NEXT Q
250 PRINT TAB(MX,MY);" ";
260 MX=19:MY=23
270 PRINT TAB(MX,MY);CHR$(225)
280 IF FIN=0 THEN GOTO 150
290 FIN=0
300 NEXT A

310 IF FIN=2 THEN PRINT TAB(0,29);"They got
    you"
320 PRINT TAB(21,2);"hits=";HIT
330 PRINT TAB(0,30);"You hit ";HIT
340 INPUT "Another game",A$
350 IF A$="Y" THEN RUN
360 *FX 4,0
370 STOP

380 DEF PROCENEMY
390 PRINT TAB(X(I),Y(I+R));" ";
400 PRINT TAB(X(I+DIR),Y(I+R+DIR));CHR$(224);
410 IF RND(1)<.5 THEN ENDPROC
420 IF C<=20 AND MX-X(I+DIR)=0 THEN FIN=2:
    GOTO 840
430 MOVE X(I+DIR)*32+16,1023-Y(I+R+DIR)*32-32
440 D=300-RND(600)
```

```
450 PLOT 1,D,-250
460 SOUND 1,-10,100,1
470 MOVE X(I+DIR)*32+16,1023-Y(I+R+DIR)*32-32
480 PLOT 2,D,-250
490 ENDPROC

500 DEF PROCFIRE
510 IF F=1 THEN ENDPROC
520 A$=INKEY$(0)
530 *FX 15,1
540 IF A$="" THEN ENDPROC
550 PRINT TAB(MX,MY);" ";
560 IF ASC(A$)=&88 THEN MX=MX-8: GOTO 590
570 IF ASC(A$)=&89 THEN MX=MX+8: GOTO 590
580 ENDPROC
590 PRINT TAB(MX,MY);CHR$(225);
600 F=1
610 ENDPROC

620 DEF PROCSTREN
630 C=0
640 J=1023-19*32-16
650 FOR I=17*32 TO 24*32  STEP 4
660 C=C+POINT(I,J)
670 NEXT I
680 C=INT(C/48*100)
690 PRINT TAB(0,0);"Shield Strength ";C; " % "
700 ENDPROC

710 DEF PROCGUIDE
720 PRINT TAB(MX,MY);" ";
730 MY=MY-1
740 IF MY<3 THEN F=0: FIN=1: GOTO 860
750 PRINT TAB(MX,MY);CHR$(225);
760 IF MX<>X(I+DIR) THEN ENDPROC
770 IF MY-Y(I+R+DIR)>2 OR MY-Y(I+R+DIR)<0 THEN
    ENDPROC
780 FIN=1
790 PRINT TAB(MX,MY);" ";
800 MX=X(I+DIR)
810 MY=Y(I+R+DIR)
820 HIT=HIT+1
830 F=0
840 MOVE X(I+DIR)*32+16,1023-Y(I+R+DIR)*32-32
```

```
 850 PLOT 1,0,Y(I+R+DIR)*32-MY*32
 860 PRINT TAB(MX,MY);CHR$(226);
 870 SOUND 0,-10,5,5
 880 FOR Q=1 TO 1000:NEXT Q
 890 IF FIN=2 THEN GOTO 920
 900 PRINT TAB(X(I+DIR),Y(I+R+DIR));" ";
 910 PRINT TAB(X(I+DIR),Y(I+R+DIR)+1);" ";
 920 I=E+DIR
 930 ENDPROC

 940 DEF PROCPATH
 950 DIM X(50),Y(50)
 960 X=0: Y=0
 970 N=39
 980 FOR I=1 TO N
 990 X=X+1
1000 Y=16-INT (((20-X)*(20-X))/25)
1010 X(I)=X
1020 Y(I)=Y
1030 NEXT I
1040 I=1
1050 HIT=0
1060 ENDPROC

1070 DEF PROCBLOCK
1080 COLOUR 129
1090 FOR I=0 TO 1
1100 PRINT TAB(17,20-I);SPC(6);
1110 NEXT I
1120 MX=19
1130 MY=23
1140 COLOUR 128
1150 COLOUR 3
1160 PRINT TAB(MX,MY);CHR$(225)
1170 F=0
1180 FIN=0
1190 ENDPROC

1200 DEF PROCINIT
1210 VDU 23,224,&DD,&DD,&DD,&FF,&FF,&3C,&18,&18
1220 VDU 23,225,&18,&3C,&7E,&18,&18,&3C,&7E,&E7
1230 VDU 23,226,&24,&24,&4F,&4A,&34,&70,&4A,&D4
1240 VDU 19,0,4,0,0,0
1250 VDU 19,1,0,0,0,0
```

```
1260 VDU 19,2,7,0,0,0
1270 VDU 19,3,3,0,0,0
1280 VDU 23;8202;0;0;0;
1290 COLOUR 128
1300 CLS
1310 *FX 4,1
1320 ENDPROC
```

17
Nine Hole Golf



This is a colour graphics game that combines both driving and putting and even includes the hazard of bunkers. You play around a nine hole course with two stages at each hole − the fairway and the green. When you RUN it notice how, in the first stage, the golfer makes his swing and how the ball flies through the air.

How to play

At the start of each hole you are told the distance to the hole marked on the screen by a flag and asked to select which club you wish to use. If you've ever played golf, or watched it on TV, you'll know that the lower the number of the club the further it will drive the ball. In other words, select the 1 iron to drive a long way and 8 iron for the really close shots. If you overshoot the green you'll get a new go at the hole and if you drive the ball off the screen you forfeit the hole and move on to the next one. Otherwise, once you get close enough

to the hole you'll find yourself on the green. A message will tell you how far you have to putt to the hole and will ask you to select the appropriate club. If you overshoot while putting you will find yourself still at some distance from the hole and will have to carry on putting until your ball drops in to the hole. Your score for each hole is displayed at the end of each hole and a score card for all nine holes is displayed at the end of each round.

Typing tips

As well as the three user-defined graphics characters, you will also find a capital 'O' used in this program in line 1400. It marks the hole on the putting green.

Subroutine structure

20	Defines graphics characters
80	Initialises variables
180	Sets up and plays each hole
670	Reports score for each hole
750	End of game
790	Plots balls' flight
1100	Lost ball routine
1160	Displays swinging club
1320	Putting routine
1600	Ball in bunker routine

Programming details

An interesting feature of this game is the use of high resolution graphics to make the player appear to swing his club. This is done in PROCSWING which draws a line which is the continually shifting radius of a circle. The flight of the ball is also plotted using high resolution graphics. The path that the ball appears to follow (subroutine 790) is a distorted parabola that always carries the ball in the direction of the flag.

Scope for improvement

You may have noticed that the score card at the end of the game does not total your score nor compare it with any ideal par for the course. You might like to add both these features. You will need to play the game a few times to discover what figure to set as the par.

Program

```
 10 REM Golf

 20 REM flag
 30 VDU 23,224,&08,&0C,&0E,&08,&08,&08,&08,&3E
 40 REM golfer
 50 VDU 23,225,&18,&3C,&5A,&3C,&18,&24,&24,&42
 60 REM bunker
 70 VDU 23,226,&18,&3E,&FE,&FF,&7F,&3E,&0E,&0C

 80 DIM T(9)
 90 B=1: XH=0:  XC=0
100 YH=0:  HT=0:  YC=0
110 MODE 1
120 VDU 19,0,2,0,0,0
130 VDU 19,1,0,0,0,0
140 VDU 19,2,3,0,0,0
150 VDU 19,3,0,0,0,0
160 VDU 23;8202;0;0;0;
170 VDU 5
```

```
180 FOR H=1 TO 9
190 CLS
200 PRINT TAB(10,0);"Hole number ";H
210 VDU 5
220 FOR Z=1 TO 8
230 XB=RND(300)+600
240 YB=RND(300)+600
250 GCOL 0,2
260 MOVE XB,YB
270 PRINT CHR$(226)
280 NEXT Z
290 REM Drive section
300 X=RND(100)
310 Y=RND(200)+350
320 XT=RND(300)+800
330 YT=RND(200)+800
340 D=SGN (XT-X)*SQR ((XT-X)*(XT-X)+(YT-Y)*
    (YT-Y))
350 GCOL 0,1
360 MOVE XT,YT
370 PRINT CHR$(224);
380 GCOL 0,3
390 MOVE X,Y
400 PRINT CHR$(225)
410 VDU 4
420 PRINT TAB(0,25);"Distance to next hole
    is ";INT(D);SPC(3)
430 INPUT"which club (1 to 8)",C
440 IF C<1 OR C>8 THEN GOTO 430
450 C=INT ((9-C)/B)+1
460 VDU 5
470 PROCSWING
480 GOSUB 790
490 B=1
500 D=SGN (XT-XHT)*SQR ((XT-XHT)*(XT-XHT)+
    (YT-YHT)*(YT-YHT))
510 VDU 4
520 IF D<-50 THEN PRINT TAB(1,20);"You
    overshot-try another hole":FOR Q=1 TO
    3000:NEXT Q:GOTO 190
530 IF D<50 THEN PRINT TAB(1,20);"on the
    green": GOTO 1320
540 FOR Q=1 TO 1000:NEXT Q
550 VDU 5
```

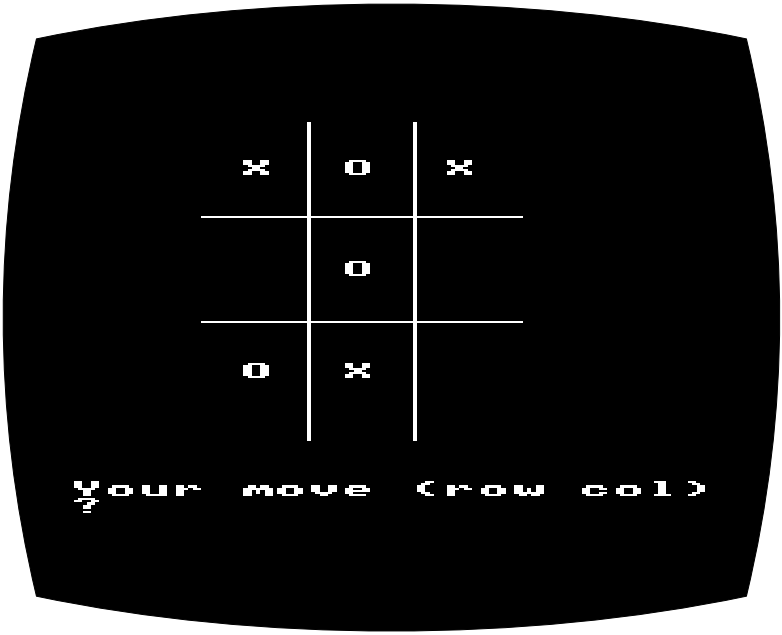
```
560 GCOL 4,4
570 MOVE XHT,YHT
580 PRINT "."
590 MOVE X,Y
600 PRINT CHR$(225)
610 X=XHT
620 Y=YHT
630 GOTO 350
640 PRINT TAB(10,2);"You took ";T(H);" strokes"
650 FOR Q=1 TO 1000:NEXT Q
660 NEXT H
670 CLS
680 PRINT TAB(10,2);"This round"
690 PRINT
700 FOR I=1 TO 9
710 PRINT TAB(8);"Hole";I;
720 IF T(I)=-1 THEN PRINT " lost ball":
    GOTO 740
730 PRINT " ";T(I);" strokes"
740 NEXT I
750 PRINT TAB(0,25);
760 INPUT "Another round ",A$
770 IF LEFT$(A$,1)="Y" THEN RUN
780 VDU 20:CLS:END
```

```
 790 REM HIT ROUTINE
 800 VT=C*(8+RND(1))
 810 HT=0
 820 XH=0
 830 REM plot ball
 840 Q=((YT-Y)/(XT-X))
 850 VV=VT*(SIN (45*PI/180))
 860 XC=(X+16)
 870 YC=Y-16
 880 VH=VT*(COS (45*PI/180))
 890 HT=HT+VV
 900 YH=Q*XH
 910 VV=VV-8
 920 XH=XH+VH
 930 YH=Q*XH
 940 IF XH+XC>1280 THEN  YH=0: YT=0: YC=0:
     XH=0: XC=0: GOTO 1100
 950 IF YH+HT+YC>1024 THEN YH=0: YT=0: YC=0:
     XH=0: XC=0: GOTO 1100
 960 IF HT<=0 THEN GOTO 1040
 970 GCOL 4,4
 980 MOVE XH+XC,YH+HT+YC
 990 PRINT "."
1000 FOR Z=1 TO 100:NEXT Z
1010 MOVE XH+XC,YH+HT+YC
1020 PRINT "."
1030 GOTO 890
1040 XHT=XH+XC
1050 YHT=YH+HT+YC
1060 IF POINT(XH+XC+16,YH+HT+YC)=2 THEN
     GOTO 1600
1070 MOVE XH+XC,YH+HT+YC
1080 PRINT "."
1090 RETURN

1100 VDU 4
1110 PRINT TAB(6,1);"You've lost your ball !!"
1120 SOUND 1,-10,50,4
1130 T(H)=-1
1140 FOR Q=1 TO 2000:NEXT Q
1150 GOTO 660

1160 DEF PROCSWING
1170 T(H)=T(H)+1
```

```
1180 XS=X+16
1190 YS=Y-10
1200 FOR S=-5 TO -40 STEP -5
1210 A=S/30*PI
1220 SX=30*SIN A:  SY=30*COS A
1230 GCOL 4,4
1240 PLOT 4,XS,YS
1250 PLOT 1,SX,SY
1260 IF S<>-30 THEN FOR Q=1 TO 50:NEXT Q
1270 IF S=-30 THEN SOUND 1,-5,50,2
1280 PLOT 4,XS,YS
1290 PLOT 1,SX,SY
1300 NEXT S
1310 ENDPROC

1320 REM putting
1330 CLS
1340 XG=RND(5)
1350 YG=15
1360 XH=RND(15)+10
1370 YH=15
1380 D=XH-XG
1390 IF D<0 THEN  D=ABS (D)
1400 PRINT TAB(XH,YH);"O";
1410 PRINT TAB(XG,YG); CHR$(225);
1420 PRINT TAB(1,18); "DISTANCE TO HOLE IS ";
     D;SPC(3)
1430 INPUT "WHICH CLUB (1 TO 8)",C
1440 IF C<1 OR C>8 THEN GOTO 1430
1450 T(H)=T(H)+1
1460 H1=8-C+RND(2)
1470 D=D-H1
1480 FOR Z=XG+1 TO XG+H1
1490 PRINT TAB(Z,YH);".";
1500 FOR Q=1 TO 500:NEXT Q
1510 PRINT TAB(Z,YH);" ";
1520 NEXT Z
1530 PRINT TAB(XG,YG);" ";
1540 XG=XG+H1
1550 IF XG=XH THEN GOTO 1580
1560 IF D<0 THEN CLS:  XG=XH+D: GOTO 1380
1570 GOTO 1400
1580 REM in the hole
1590 GOTO 640
```

```
1600 REM in the bunker
1610 VDU 4
1620 GCOL 0,3
1630 PRINT TAB(1,28);"in the bunker";
1640 SOUND 1,-15,50,5
1650 FOR Q=1 TO 5000:NEXT Q
1660 B=2
1670 PRINT TAB(1,28);SPC(20)
1680 RETURN
```

18
Noughts and Crosses



Noughts and crosses is a perennial favourite because it is a simple game of strategy. The problem with playing it against a computer is that the computer can be programmed so that the person challenging it can never win. However, this program makes your Electron an opponent who can be beaten. The Electron will make sensible moves but is not infallible so it is worth playing on until you beat it. It's actually a very good way of learning about game-playing strategy.

How to play

This game is played on a simple three-by-three grid in the traditional way. You have the 'X' and play first. To make your move you have to specify which square to place your mark on. Type in the row number first, then the column number. For example, type 11 to place your cross in the top, lefthand corner. If you type a number in the wrong format (for example 1,1) or a number that does not correspond

to a position on the grid, for example 41, the Electron won't accept it and will beep at you. If you type the number of a position that is already occupied a message to that effect will be displayed. Once you've made your move the computer replies with its 'O' and you make your next move. At the end the Electron will display "I WIN" if it has been successful, "YOU WIN" if you've been successful and "DRAW" if it's stalemate. The board has to be completely filled for the game to be over.

Typing Tips

A capital O is used in this program look out for it in the print statement in line 960.

Subroutine structure

20	Main play loop
120	Evaluates computers move
630	Tries each move
770	Gets player's move
920	Displays moves
1020	Sets up screen and plots frame
1220	End of game

Programming details

The method used for the computer to play noughts and crosses is based on an advanced technique from artificial intelligence. The program only looks one move ahead with deciding its move in other words it does not try to take account of the next move you'll make − which is why it slips up sometimes and allows you to win!

Program

```
10 REM Noughts and Crosses
```

```
 20 MODE 5
 30 PROCINIT
 40 PROCMOVE
 50 PROCBOARD
 60 PROCREPLY
 70 IF FIN=1 THEN GOTO 1260
 80 IF FIN=2 THEN PROCBOARD :GOTO 1220
 90 IF DR=1 THEN GOTO 1240
100 PROCBOARD
110 GOTO 40

120 DEF PROCVAL
130 FOR Z=1 TO 4
140 X(Z)=0
150 Y(Z)=0
160 NEXT Z
170 FOR L=1 TO 3
180 S=0
190 T=0
200 FOR K=1 TO 3
210 IF A(L,K)=1 THEN S=S+1
220 IF B(L,K)=1 THEN T=T+1
230 NEXT K
240 IF S=0 THEN Y(T+1)=Y(T+1)+1
250 IF T=0 THEN X(S+1)=X(S+1)+1
260 NEXT L
270 FOR L=1 TO 3
280 T=0
290 S=0
300 FOR K=1 TO 3
310  IF A(K,L)=1 THEN S=S+1
320 IF B(K,L)=1 THEN T=T+1
330 NEXT K
340 IF S=0 THEN Y(T+1)=Y(T+1)+1
350 IF T=0 THEN X(S+1)=X(S+1)+1
360 NEXT L
370 PROCDIA1
380 PROCDIA2
390 IF X(4)=1 THEN FIN=1:ENDPROC
400 IFY(4)=1 THEN FIN=2
410 E=128*Y(4)-63*X(3)+31*Y(3)-15*X(2)+7*Y(2)
420 ENDPROC
430 DEF PROCDIA1
440 T=0
```

```
450 S=0
460 FOR K=1 TO 3
470 T=T+A(K,K)
480 S=S+B(K,K)
490 NEXT K
500 IF S=0 THEN X(T+1)=X(T+1)+1
510 IF T=0 THEN Y(S+1)=Y(S+1)+1
520 ENDPROC
530 DEF PROCDIA2
540 T=0
550 S=0
560 FOR K=1 TO 3
570 T=T+A(4-K,K)
580 S=S+B(4-K,K)
590 NEXT K
600 IF S=0 THEN X(T+1)=X(T+1)+1
610 IF T=0 THEN Y(S+1)=Y(S+1)+1
620 ENDPROC

630 DEF PROCREPLY
640 M=-256: DR=1
650 FOR J=1 TO 3
660 FOR I=1 TO 3
670 IF A(I,J)=1 OR B(I,J)=1 THEN GOTO 730
680 DR=0:B(I,J)=1
690 PROCVAL
700 IF FIN=1 THEN ENDPROC
710 IF E>M THEN M=E: A=I:B=J
720 B(I,J)=0
730 NEXT I
740 NEXT J
750 B(A,B)=1
760 ENDPROC

770 DEF PROCMOVE
780 PRINT TAB(0,25);
790 INPUT "Your move (row col) ",A$
800 IF LEN(A$)<>2 THEN SOUND 1,-10,100,2:
    GOTO 780
810 J=VAL(MID$(A$,1,1)):I=VAL(MID$(A$,2,1))
820 IF I<1 OR I>3 THEN SOUND 1,-10,100,2:
    GOTO 780
830 IF J<1 OR J>3 THEN SOUND 1,-10,100,2:
    GOTO 780
```

```
 840 IF A(I,J)=1 THEN GOTO 890
 850 IF B(I,J)=1 THEN GOTO 890
 860 A(I,J)=1
 870 PRINT TAB(0,25);SPC(100)
 880 ENDPROC
 890 PRINT TAB(0,28);"Position already"'
     "occupied"
 900 SOUND 1,-10,100,2
 910 GOTO 780
 920 DEF PROCBOARD
 930 FOR J=1 TO 3
 940 FOR I=1 TO 3
 950 IF A(I,J)=1 THEN PRINT TAB(I*3+2,J*6);"X";
 960 IF B(I,J)=1 THEN PRINT TAB(I*3+2,J*6);"O";
 970 IF A(I,J)+B(I,J)=0 THEN PRINT
     TAB(I*3+2,J*6);" ";
 980 NEXT I
 990 PRINT
1000 NEXT J
1010 ENDPROC

1020 DEF PROCINIT
1030 DIM A(3,3)
1040 DIM B(3,3)
1050 DIM X(4),Y(4)
1060 PROCBOARD
1070 VDU 23;8202;0;0;0;
1080 GCOL 0,3
1090 PLOT 4,450,900
1100 PLOT 1,0,-600
1110 PLOT 4,650,900
1120 PLOT 1,0,-600
1130 PLOT 4,250,525
1140 PLOT 1,600,0
1150 PLOT 4,250,725
1160 PLOT 1,600,0
1170 FIN=0
1180 DR=0
1190 COLOUR 3
1200 COLOUR 128
1210 ENDPROC

1220 PRINT TAB(0,30);"I WIN"
1230 GOTO 1270
```

```
1240 PRINT TAB(0,30);"DRAW"
1250 GOTO 1270
1260 PRINT TAB(0,30);"YOU WIN"
1270 INPUT "Another game Y/N",A$
1280 IF A$="Y" THEN RUN
1290 CLS
```

19
Fruit Machine

Here's a way of playing the fruit machine without spending a penny your Electron gives you 100 pence to start with, takes l0 pence for every go, and awards you a sum between 5 pence and 50 pence every time you come up with a winning combination. You can give up while you are winning or carry on playing until you are broke.

Although short to type in, the program includes some really clever graphics techniques so that you see the drum of the fruit machine rotate smoothly, using only BASIC. In addition, there are sound effects that signal winning combinations. So listen out for the jackpot!

How to play

There are four symbols in the display − cherries, banana, apple and bell. All the winning combinations are displayed on the screen while you play. These combinations are winners wherever they occur on

the line and not just as in the pattern suggested by the screen display, but notice that where blanks occur you need some symbol *other* than the same type. To play just RUN and then answer "Y" every time you want another spin. If you do not answer "Y" then the computer will tell you how much money you are taking home with you. Once you run out of money the game is over.

Subroutine structure

20	Initialisation
50	Main play loop
170	Sets starting points of drum
220	Spins drum
400	Pay out routine
470	Defines graphics characters and displays title frame
740	Jackpot routine
800	Signals when broke

Programming details

This program uses some very tricky programming techniques which is why it achieves its effect in so short a length of BASIC. The patterns for each of the shapes are stored in an array, one line of dots to each array element. Each time the fruit machines drum is printed a different section of the array is used to load the user-defined graphics. You can think of the section of the array that is used as being defined by a window which moves down by one row of dots each time the characters are printed. This produces the visual illusion of a smoothly rotating drum.

Scope for improvement

If you like adding graphics and sound effects to programs there is scope in this game. For example, you could include a surround that looks like a one-armed-bandit and sounds of cascading coins and the drum rotating.

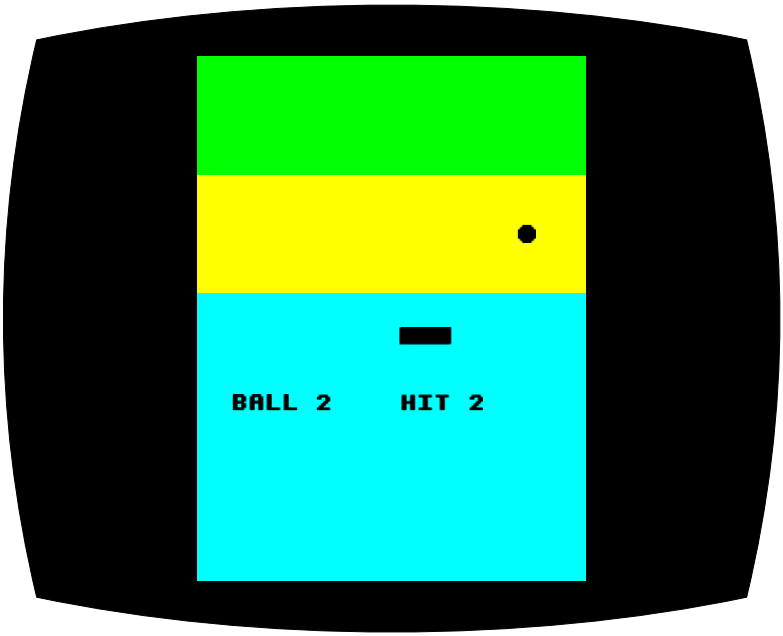
Program

```
 10 REM Fruit machine

 20 MODE 5
 30 PROCINIT
 40 M=100

 50 PROCRES
 60 M=M-10
 70 PROCSPIN
 80 PROCPAY
 90 IF M<=0 THEN PROCBUST
100 PRINT TAB(0,29);"You have ";M;" p";SPC(3)
110 PRINT "Another spin ?"
120 A$=INKEY$(0)
130 IF A$="" THEN GOTO 120
140 IF A$="Y" THEN  GOTO 50
150 PRINT TAB(0,31);"You take home ";M;" p"
160 STOP

170 DEF PROCRES
180 X=(RND(4)-1)*10+1
190 Y=(RND(4)-1)*10+1
200 Z=(RND(4)-1)*10+1
210 ENDPROC

220 DEF PROCSPIN
230 S=RND(2)+2
240 FOR I=0 TO S*10
250 VDU 23,229,C(X),C(X),C(X+1),C(X+1),C(X+2),
    C(X+2),C(X+3),C(X+3)
260 VDU 23,232,C(X+4),C(X+4),C(X+5),C(X+5),
    C(X+6),C(X+6),C(X+7),C(X+7)
270 VDU 23,230,C(Y),C(Y),C(Y+1),C(Y+1),C(Y+2),
    C(Y+2),C(Y+3),C(Y+3)
280 VDU 23,233,C(Y+4),C(Y+4),C(Y+5),C(Y+5),
    C(Y+6),C(Y+6),C(Y+7),C(Y+7)
290 VDU 23,231,C(Z),C(Z),C(Z+1),C(Z+1),C(Z+2),
    C(Z+2),C(Z+3),C(Z+3)
300 VDU 23,234,C(Z+4),C(Z+4),C(Z+5),C(Z+5),
    C(Z+6),C(Z+6),C(Z+7),C(Z+7)
310 PRINT TAB(7,20);CHR$(229);" ";CHR$(230);
    " ";CHR$(231)
```

```
320 PRINT TAB(7,21);CHR$(232);" ";CHR$(233);
    " ";CHR$(234)
330 IF X=40 THEN X=0
340 IF Y=40 THEN Y=0
350 IF Z=40 THEN Z=0
360 X=X+1:Y=Y+1:Z=Z+1
370 NEXT I
380 X=X-1:Y=Y-1:Z=Z-1
390 ENDPROC

400 DEF PROCPAY
410 REM Calculates winnings
420 IF X=1 AND Y=1 AND Z=1 THEN PROCJACK:
    ENDPROC
430 IF (X=31)+(Y=31)+(Z=31)=-2 THEN M=M+10:
    SOUND 1,-15,200,5
440 IF (X=21)+(Y=21)+(Z=21)=-2 THEN M=M+25:
    SOUND 1,-15,200,5
450 IF(X=1)+(Y=1)+(Z=1)=-1 THEN M=M+5:
    SOUND 1,-15,200,5:ENDPROC
460 ENDPROC

470 DEF PROCINIT
480 VDU 23,224,&06,&0A,&14,&24,&44,&CF,&EF,&E6
490 VDU 23,225,&02,&0C,&1C,&38,&38,&1C,&0C,&02
500 VDU 23,226,&18,&3C,&3C,&3C,&7E,&FF,&18,&18
510 VDU 23,227,&0C,&18,&7A,&FF,&FF,&FF,&7E,&3C
520 VDU 23,228,&00,&00,&00,&7E,&7E,&00,&00,&00
530 PRINT TAB(0,5);CHR$(224),CHR$(225),
    CHR$(226),CHR$(227),CHR$(228)
540 CLS
550 PRINT TAB(5);"F R U I T"
560 PRINT TAB(4);"M A C H I N E"
570 PRINT '"YOU HAVE `1.00"'"TO GAMBLE"
580 PRINT '"EACH SPIN COSTS 10p"
590 PRINT 'CHR$(224);" ";CHR$(224);" ";
    CHR$(224);" WINS 50p"
600 PRINT 'CHR$(226);" ";CHR$(226);" - WINS
    25p"
610 PRINT '"- ";CHR$(227);" ";CHR$(227);
    " WINS 10p"
620 PRINT'CHR$(224);" - - WINS 5p"
630 DATA &06,&0A,&14,&24,&44,&CF,&EF,&E6,0,0
640 DATA &02,&0C,&1C,&38,&38,&1C,&0C,&02,0,0
```

```
650 DATA &18,&3C,&3C,&3C,&7E,&FF,&18,&18,0,0
660 DATA &0C,&18,&7A,&FF,&FF,&FF,&7E,&3C,0,0
670 DATA &06,&0A,&14,&24,&44,&CF,&EF,&E6,0,0
680 DIM C(48)
690 FOR I=1 TO 48
700 READ C(I)
710 NEXT I
720 VDU 23;8202;0;0;0;
730 ENDPROC

740 DEF PROCJACK
750 FOR I=50 TO 200 STEP 8
760 SOUND 1,-15,I,10
770 NEXT I
780 M=M+50
790 ENDPROC

800 DEF PROCBUST
810 PRINT TAB(0,30);"YOU ARE BROKE !!!!!!!!"
820 STOP
```

20
Rainbow Squash



This is a colourful version of the popular computer squash game which is also enhanced by the addition of sound − a cheery beep every time the ball bounces either on the sides of the court or against the bat, and a dismal tone every time a ball goes out of play. Its other feature is that as you improve in skill the game gets more difficult and if you then start to get worse it gets easier. This means that your Electron will always give you a challenge that is suited to your ability − which makes it the perfect partner.

How to play

At the start of the game the bat is at the bottom of the screen and in the centre. You control the left and right movement of the bat by pressing the appropriate arrow keys. Every time you make two hits in succession the position of the bat changes − it moves nearer to the top of the screen − which makes returning the ball more difficult. If you

then miss a shot the ball will move back one position, making it easier. You score a point for every hit and you will be served a total of 10 balls. Information about the number of balls played and hits scored is displayed on the screen continuously.

Subroutine structure

20	Sets up screen and colours and initialise variables
150	Defines graphics characters
180	Sets up initial screen display
200	Main play loop
450	Draws court
620	Bounce routine
770	Moves bat up screen
890	End of game − prints final score, offers another game and re-runs or restores screen display
1010	Function to move bat
1070	Determines colour of bat

Programming details

Although this program is fairly short it uses some clever tricks to ensure that the old positions of the bat and ball are blanked out in the correct colours. Writing a program of this complexity is much easier using procedure calls and function definitions. Notice the use of PROCCOL which changes the background colour and the use of FNBAT which controls the bat's movement. The definition of this function includes *FX 15,1 − which flushes the buffer after every key press. This means that the bat responds instantaneously to the player's change of direction.

Program

```
  10 REM Rainbow Squash

  20 MODE 1
  30 VDU 19,1,0,0,0,0
  40 VDU 19,0,6,0,0,0
  50 VDU 19,2,2,0,0,0
  60 VDU 19,3,3,0,0,0
  70 *FX 4,1
  80 *FX 11,1
  90 *FX 12,1
 100 H=0
 110 HT=0
 120 D=19
 130 BALL=0
 140 C=2

 150 VDU 23,224,&FF,&FF,&FF,&FF,&FF,&FF,&FF,&FF
 160 VDU 23,225,&3C,&7E,&FF,&FF,&FF,&FF,&7E,&3C
 170 VDU 23;8202;0;0;0;

 180 PROCCOURT
 190 X=10
```

```
200 BALL=BALL+1
210 IF BALL>10 THEN GOTO 890
220 A=10+RND(6)
230 B=1
240 V=1
250 W=1
260 Y=D
270 COLOUR 128:COLOUR 1
280  PRINT TAB(10,21);"BALL ";BALL;
290 X=FNBAT(X):PROCCOL
300 PRINT TAB(X,Y);CHR$(E);CHR$(224)CHR$(224)
    CHR$(224);CHR$(F);
310 COLOUR 128:COLOUR 1
320 PRINT TAB(20,21);"HIT ";HT
330 FOR Z=1 TO 100:NEXT Z
340 PROCBOUNCE
350 COLOUR 1:COLOUR 128
360 IF B+W<>Y THEN Y=D:GOTO 290
370 SOUND 2,-5,50,10
380 PROCCOL
390 PRINT TAB(X+1,Y);SPC(3);
400 PROCBOUNCE
410 PRINT TAB(A,B);" "
420 IF D<19 THEN D=D+1
430 H=0
440 GOTO 200

450 DEF PROCCOURT
460 CLS
470 COLOUR 1
480 FOR I=0 TO 39
490 PRINT CHR$(224);
500 NEXT
510 COLOUR 130
520 PRINT STRING$(200," ");
530 PRINT STRING$(80," ");
540 COLOUR 131
550 PRINT STRING$(200," ");
560 PRINT STRING$(80," ");
570 COLOUR 1
580 FOR I=0 TO 30
590 PRINT TAB(0,I);STRING$(8,CHR$(224));
    TAB(31,I);STRING$(9,CHR$(224));
600 NEXT I
```

```
 610 ENDPROC

 620 DEF PROCBOUNCE
 630 COLOUR 128
 640 IF B<15  THEN COLOUR 131
 650 IF B<8  THEN COLOUR 130
 660 PRINT TAB(A,B);" "
 670 A=A+V
 680 B=B+W
 690 IF A=30 OR A=8 THEN V=-V:SOUND 1,-4,89,5
 700 IF B=1 THEN W=-W:SOUND 1,-4,89,5
 710 IF B+W=Y THEN GOTO 770
 720 COLOUR 128
 730 IF B<15  THEN COLOUR 131
 740 IF B<8  THEN COLOUR 130
 750 PRINT TAB(A,B);CHR$(225)
 760 ENDPROC

 770 R=A-X
 780 IF R<1 OR R>3 THEN GOTO 750
 790 W=-W
 800 SOUND 1,-4,89,5
 810 H=H+1
 820 HT=HT+1
 830 IF H<>1 THEN GOTO 720
 840 H=0
 850 D=D-1
 860 PROCCOL
 870 PRINT TAB(X+1,Y);SPC(3);
 880 GOTO 720

 890 FOR Z=1 TO 1000:NEXT Z
 900 CLS
 910 PRINT TAB(10,10);"You Scored ";HT
 920 PRINT TAB(10,15);
 930 *FX 15,1
 940 *FX 4,0
 950 *FX 12,0
 960 INPUT "ANOTHER GAME Y/N ",A$
 970 A$=LEFT$(A$,1)
 980 IF A$="Y" THEN RUN
 990 CLS
1000 END
```

```
1010 DEF FNBAT(Q)
1020 K=INKEY(0)
1030 *FX 15,1
1040 IF K=&89  AND Q<27 THEN Q=Q+1
1050 IF K=&88 AND Q>7 THEN Q=Q-1
1060 =Q

1070 DEF PROCCOL
1080 COLOUR 128
1090 IF Y<15 THEN COLOUR 131
1100 IF Y<8 THEN COLOUR 130
1110 IF X=27 THEN F=224 ELSE F=32
1120 IF X=7 THEN E=224 ELSE E=32
1130 ENDPROC
```

21
Smalltalker

Do you ever find yourself talking to your Electron? Well, if you do, you may be disappointed that it never answers back. This program, however, changes all that and gives your Electron the chance to start a conversation with you. Although it may not be able to rival the agony aunts of the glossy magazines, your Electron is anxious to hear about your problems and has some comments to offer.

Coping with the syntax of the English language is one of the very complicated problems with which this program has to contend. Programs like this one have been developed in order to extend our knowledge of how language works and how humans identify the key components of conversations. While these serious purposes are usually the province of artificial intelligence it is possible to have a good deal of fun trying to conduct a dialogue with your Electron.

How to use this program

The computer opens each conversation in the same way − by inviting you to tell it your problems. You can give any reply that you wish to and after a few moments delay your Electron will respond. Try to say more than just “YES” or “NO” when you make further responses but equally, don’t say too much at a time. If you type about a lineful each time you ought to be able to keep a reasonable conversation going.

Typing tips

Do remember to use upper case only when entering this program and when using it. As the computer has to match your sentences against its vocabulary it is also very important to the careful about your spelling. If you type in either the initial program, or subsequent responses, with mispellings the computer won’t recognise your messages and you won’t receive any sensible answers. The apostrophe is the only punctuation mark that should be used in dialogues with the Electron.

Subroutine structure

20	Main program loop
100	Initial message and set-up
400	Input human’s sentence
560	Divides sentences into words
650	Changes tense/pronouns
820	Tense/pronouns data
920	Finds keywords in sentence
1070	Keywords data
1250	Keyword responses
2210	Prints computer’s response
2270	Requests sensible input

Programming details

This program works by taking a sentence and splitting it down into individual words and then responding according to a list of keywords that it searches for in each sentence. So if, for example, your sentence contains the word 'why', the response 'Some questions are difficult to answer' will always be given by the computer. When the computer fails to find a specific reply to a sentence one of a number of responses is selected at random.

Although this technique sounds simple, the actual details of the program are really quite tricky as, amongst other things, the computer has to deal with tense changes and with the syntax of pronouns. It is therefore quite a difficult program to write or to modify extensively. Equally, despite the apparent simplicity of its underlying technique, it succeeds in making plausible responses on a surprising number of occasions.

Scope for improvement

If you wish to add to the list of keywords that the computer recognises, you need to notice how, in subroutine 1070, the keywords are paired with the line number of the subroutine that responds to them. It is also important to be aware of the priorities assigned to each keyword. If two keywords are present in a sentence then the one first in the list will be acted upon.

Program

```
10 REM Smalltalker

20 MODE 4:DIM W(20,2)
30 GOSUB 100
40 GOSUB 400
50 GOSUB 560
60 GOSUB 920
70 IF NM<>0 THEN ON NM GOSUB 1250,1270,1290,
   1310,1730,1780,1690,1710,1610,1380,2140,
   1330,1470,1900,2040,1800,1820,1840,1330,
   1880,1860
```

```
 80 GOSUB 2210
 90 GOTO 40

100 CLS
110 PRINT TAB(10);"HI"
120 PRINT
130 PRINT "I WOULD LIKE YOU TO TALK TO ME"
140 PRINT "BUT I DON'T HAVE EARS SO WILL"
150 PRINT "YOU TYPE SENTENCES ON MY KEYBOARD"
160 PRINT "IN UPPER CASE"
170 PRINT
180 PRINT "DON'T USE ANY PUNCTUATION APART"
190 PRINT "FROM APOSTROPHIES WHICH ARE
    IMPORTANT"
200 PRINT
210 PRINT
220 PRINT "WHEN YOU HAVE FINISHED TYPING"
230 PRINT "PRESS ENTER"
240 PRINT '' "TELL ME YOUR PROBLEMS"
250 R$=""
260 M$=""
270 D$=""
280 DIM N$(3)
290 N$(1)="PLEASE GO ON"
300 N$(2)="I'M NOT SURE I UNDERSTAND YOU"
310 N$(3)="TELL ME MORE"
320 DIM I$(3)
330 I$(1)="LET'S TALK SOME MORE ABOUT YOUR"
340 I$(2)="EARLIER YOU SPOKE OF YOUR"
350 I$(3)="DOES THAT HAVE ANYTHING TO DO
    WITH YOUR"
360 DIM J$(2)
370 J$(1)="ARE YOU JUST BEING NEGATIVE"
380 J$(2)="I SEE"
390 RETURN

400 A$=""
410 B$=INKEY$(0)
420 REM Input section
430 IF B$="" THEN GOTO 410
440 IF INKEY$(0)<>"" THEN GOTO 440
450 IF ASC(B$)=13 THEN GOTO 510
460 IF ASC(B$)=&7F AND A$<>"" THEN
    A$=LEFT$(A$,LEN(A$)-1):GOTO 490
```

```
470 IF ASC(B$)<32 OR ASC(B$)>126 THEN GOTO 400
480 A$=A$+B$
490 PRINT TAB(0,30);A$;" "
500 GOTO 410
510 IF A$=R$ THEN PRINT TAB(0,31);"YOU'RE
    REPEATING YOURSELF":PRNT:PRINT:GOTO 400
520 R$=A$
530 IF A$="" THEN GOTO 400
540 A$=" "+A$
550 RETURN

560 REM Divides sentences into words
570 N=1
580 B=0
590 FOR I=1 TO LEN(A$)
600 IF ((MID$(A$,I,1)=" " OR MID$(A$,I,1)=",")
    AND B=0) THEN B=1
610 IF ((MID$(A$,I,1)<>" " AND MID$(A$,I,1)
    <>",") AND B<=1) THEN W(N,1)=I:B=2
620 IF ((MID$(A$,I,1)=" " OR MID$(A$,I,1)=",")
    AND B=2) THEN W(N,2)=I-1:N=N+1:B=0
630 NEXT I
640 W(N,2)=LEN(A$)

650 FOR I=1 TO N
660 RESTORE
670 READ B$
680 IF B$="S" THEN GOTO 800
690 IF B$<>MID$(A$,(W(I,1)),(W(I,2)-W(I,1)+1))
    THEN GOTO 780
700 READ C$
710 A$=LEFT$(A$,(W(I,1)-1))+C$+
    RIGHT$(A$,(LEN(A$)-W(I,2)))
720 W(I,2)=W(I,2)+LEN(C$)-LEN(B$)
730 FOR J=I+1 TO N
740 W(J,2)=W(J,2)+LEN(C$)-LEN(B$)
750 W(J,1)=W(J,1)+LEN(C$)-LEN(B$)
760 NEXT J
770 GOTO 800
780 READ B$
790 GOTO 670
800 NEXT I
810 RETURN
```

```
 820 DATA "MY","YOUR*","I","YOU*"
 830 DATA "MUM","MOTHER","DAD","FATHER"
 840 DATA"DREAMS","DREAM","YOU","I*","ME",
     "YOU*"
 850 DATA "YOUR","MY*","MYSELF","YOURSELF*"
 860 DATA "YOURSELF","MYSELF*","I'M","YOU'RE*"
 870 DATA "YOU'RE","I'M","AM","ARE*"
 880 DATA "I'M","YOU'RE*"
 890 DATA "WERE","WAS"
 900 DATA "ARE","AM"
 910 DATA "S","S"

 920 RESTORE:FOR I=1 TO 34:READ Q$:NEXT I
 930 READ B$,NM
 940 IF B$="S" THEN GOTO 1010
 950 I=1
 960 IF MID$(A$,W(I,1),(W(I,2)-W(I,1)+1))<>B$
     THEN GOTO 990
 970 T$=RIGHT$(A$,LEN(A$)-(W(I,2)))
 980 RETURN
 990 I=I+1:IF I<=N THEN GOTO 960
1000 GOTO 930
1010 NM=0
1020 IF M$<>"" THEN GOTO 1050
1030 P$=N$(RND(3))
1040 RETURN
1050 P$=I$(RND(3))+M$
1060 RETURN
1070 DATA "COMPUTER",1,"MACHINE",1,"PROGRAM",1
1080 DATA "LIKE",2,"SAME",2,"ALIKE",2
1090 DATA "IF",3,"EVERYBODY",4
1100 DATA "CAN",5,"CERTAINLY",6
1110 DATA "HOW",7,"BECAUSE",8
1120 DATA "ALWAYS",9
1130 DATA "EVERYONE",4,"NOBODY",4
1140 DATA "WAS",10
1150 DATA "I*",11
1160 DATA "NO",12
1170 DATA "YOUR*",13
1180 DATA "YOU'RE*",14,"YOU*",15
1190 DATA "HELLO",16,"MAYBE",17
1200 DATA "MY*",18,"NO",19
1210 DATA "YES",6,"WHY",20
1220 DATA "PERHAPS",17,"SORRY",21
```

```
1230 DATA "WHAT",20
1240 DATA "S",0

1250 P$="DO COMPUTERS WORRY YOU ?"
1260 RETURN
1270 P$="IN WHAT WAY ?"
1280 RETURN
1290 P$="WHY TALK OF POSSIBILITIES"
1300 RETURN
1310 P$="REALLY "+B$+" ?"
1320 RETURN
1330 IF I=N THEN GOTO 1360
1340 I=I+1
1350 IF MID$(A$,W(I,1),(W(I,2)-W(I,1)+1))="ONE"
     THEN B$=B$+" ONE":GOTO 1310
1360 P$=J$(RND(2))
1370 RETURN
1380 IF I=N THEN GOTO 2270
1390 I=I+1
1400 IF I>N THEN GOTO 1020
1410 IF MID$(A$,W(I,1),(W(I,2)-W(I,)+1))<>
     "YOU*" THEN GOTO 1440
1420 P$="WHAT IF YOU WERE "+
     RIGHT$(A$,(LEN(A$)-W(I,2)))+" ?"
1430 RETURN
1440 IF MID$(A$,W(I,1),(W(I,2)-W(I,1)+1))<>"I*"
     THEN GOTO 1020

1450 P$="WOULD YOU LIKE TO BELIEVE I WAS "+
     RIGHT$(A$,(LEN(A$)-(W(I,1)+1)))
1460 RETURN
1470 I=I+1
1480 IF I>N THEN GOTO 1360
1490 IF MID$(A$,W(I,1),(W(I,2)-W(I,1)+1))=
     "MOTHER" THEN GOTO 1590
1500 IF MID$(A$,W(I,1),(W(I,2)-W(I,1)+1))=
     "FATHER" THEN GOTO 1590
1510 IF MID$(A$,W(I,1),(W(I,2)-W(I,1)+1))=
     "SISTER" THEN GOTO 1590
1520 IF MID$(A$,W(I,1),(W(I,2)-W(I,1)+1))=
     "BROTHER" THEN GOTO 1590
1530 IF MID$(A$,W(I,1),(W(I,2)-W(I,1)+1))=
     "WIFE" THEN GOTO 1590
1540 IF MID$(A$,W(I,1),(W(I,2)-W(I,1)+1))=
```

```
     "HUSBAND" THEN GOTO 1590
1550 IF MID$(A$,W(I,1),(W(I,2)-W(I,1)+1))=
     "CHILDREN" THEN GOTO 1590
1560 IF LEN(T$)>10 THEN M$=T$
1570 P$="YOUR "+T$+" ?"
1580 RETURN
1590 P$="TELL ME MORE ABOUT YOUR FAMILY"
1600 RETURN
1610 P$="GIVE ME A PARTICULAR EXAMPLE"
1620 RETURN
1630 I=I+1
1640 IF I>N THEN P$="AM I WHAT ?":RETURN
1650 P$="WHY ARE YOU INTERESTED IN WHETHER I
     AM "+RIGHT$(A$,W(I,1)+" OR NOT ?"
1660 RETURN
1670 P$="DO YOU THINK YOU ARE "+
     RIGHT$(S$,LEN(A$)-W(I,2))
1680 RETURN
1690 P$="WHY DO YOU ASK ?"
1700 RETURN
1710 P$="TELL ME ABOUT ANY OTHER REASONS"
1720 RETURN
1730 I=I+1
1740 IF I>N THEN P$="WHAT ?":RETURN
1750 IF MID$(A$,W(I,1),(W(I,2)-W(I,1))+1)="I*"
     THEN P$="DO YOU BELIEVE I CAN "+RIGHT$
     (A$,(LEN(A$)-W(I,2))+" ?":RETURN
1760 IF MID$(A$,W(I,1),(W(I,2)-W(I,1))+1)=
     "YOU*" THEN P$="DO YOU BELIEVE YOU CAN "+
     RIGHT$A$,LEN(A$)-W(I,2))+" ?":RETURN
1770 GOTO 1010
1780 P$="YOU SEEM VERY POSITIVE"
1790 RETURN
1800 P$="PLEASED TO MEET YOU - LET'S TALK ABOUT
     YOUR PROBLEMS"
1810 RETURN
1820 P$="COULD YOU TRY TO BE MORE POSITIVE"
1830 RETURN
1840 P$="WHY ARE YOU CONCERNED ABOUT MY "+T$
1850 RETURN
1860 P$="YOU DON'T HAVE TO APOLOGISE TO  ME"
1870 RETURN
1880 P$="SOME QUESTIONS ARE DIFFICULT TO
     ANSWER..."
```

```
1890 RETURN
1900 I=I+1
1910 IF I>N THEN GOTO 1020
1920 P$="I AM SORRY TO HEAR THAT YOU ARE "+
     MID$(A$,W(I,1),(W(I,2)-W(I,1)+1))
1930 IF MID$(A$,W(I,1),(W(I,2)-W(I,1)+1))=
     "SAD" THEN RETURN
1940 IF MID$(A$,W(I,1),(W(I,2)-W(I,1)+1))=
     "UNHAPPY" THEN RETURN
1950 IF MID$(A$,W(I,1),(W(I,2)-W(I,1)+1))=
     "DEPRESSED" THEN RETURN
1960 IF MID$(A$,W(I,1),(W(I,2)-W(I,1)+1))=
     "SICK" THEN RETURN
1970 P$="HOW HAVE I HELPED YOU TO BE "+
     MID$(A$,W(I,1),(W(I,2)-W(I,1)+1))
1980 IF MID$(A$,W(I,1),(W(I,)-W(I,1)+1))=
     "HAPPY" THEN RETURN
1990 IF MID$(A$,W(I,1),(W(I,)-W(I,1)+1))=
     "ELATED" THEN RETURN
2000 IF MID$(A$,W(I,1),(W(I,)-W(I,1)+1))=
     "GLAD" THEN RETURN
2010 IF MID$(A$,W(I,1),(W(I,)-W(I,1)+1))=
     "BETTER" THEN RETURN
2020 P$="IS IT BECAUSE YOU ARE "+
     MID$(A$,W(I,1),W(I,2)-W(I,1)+1)+
     " YOU WOULD LIKE TO TALK TO ME ?"
2030 RETURN
2040 IF I=1 THEN GOTO 2060
2050 IF MID$(A$,W(I-1,1),(W(I-1,2)-W(I-1,1)+1))
     ="ARE*" THEN GOTO 1670
2060 I=I+1
2070 IF I>N THEN GOTO 2270
2080 IF MID$(A$,W(I,1),(W(I,2)-W(I,1)+1))
     ="ARE*" THEN GOTO 1900
2090 IF MID$(A$,W(I,1),(W(I,2)-W(I,1)+1))="WANT"
     OR MID$(A$,W(I,1),(W(I,2)-W(I,1)+1))="NEED"
     THEN P$="WHAT WOULD IT MEAN IF YOU GOT "
     +RIGHT$(A$,LEN(A$)-W(I,2)):RETURN
2100 IF MID$(A$,W(I,1),(W(I,2)-W(I,1)+1))=
     "THINK" THEN P$="DO YOU REALLY THINK SO":
     RETURN
2110 IF MID$(A$,W(I,1),(W(I,2)-W(I,1)+1))=
     "CAN'T" OR MID$(A$,W(I,1),(W(I,2)-W(I,1)+1))
     ="CANNOT" THEN P$="HOW DO YOU KNOW YOU
```

```
     CAN'T"+RIGHT$(A$,LEN(A$)-W(I,2)):RETURN
2120 IF MID$(A$,W(I,1),(W(I,2)-W(I,1)+1))="FEEL"
     THEN P$="TELL ME MORE ABOUT HOW YOU FEEL":
     RETURN
2130 GOTO 1020
2140 IF I-1<1 THEN GOTO 2160
2150 IF MID$(A$,W(I,1),(W(I,2)-W(I,1)+1))="AM"
     THEN GOTO 1630
2160 I=I+1
2170 IF I>=N THEN P$="WHAT AM I ?":RETURN
2180 IF MID$(A$,W(I,1),(W(I,2)-W(I,1)+1))="AM"
     THEN P$="WHY DO YOU THINK SO ?":RETURN
2190 P$="IS THAT WHAT YOU THINK OF ME ?":RETURN
2200 RETURN

2210 FOR J=1 TO LEN(P$)
2220 IF MID$(P$,J,1)="*" THEN GOTO 2240
2230 COLOUR 0:COLOUR 129:PRINT MID$(P$,J,1);:
     COLOUR 1:COLOUR 128
2240 NEXT J
2250 PRINT '''
2260 RETURN

2270 P$="PLEASE TALK SENSIBLY !"
2280 RETURN
```