THE SPECTRUM BOOK OF GAMES

MIKE JAMES,
S.M.GEE AND KAY EWBANK

The Spectrum Book of Games

M. James, S. M. Gee and K. Ewbank

GRANADA
London Toronto Sydney New York

Granada Publishing Limited - Technical Books Division
Frogmore, St Albans, Herts AL2 2NF
and
36 Golden Square, London W1R 4AH
515 Madison Avenue, New York, NY 10022, USA
117 York Street, Sydney, NSW 2000, Australia
100 Skyway Avenue, Rexdale, Ontario, Canada M9W 3A6
61 Beach Road, Auckland, New Zealand

British Library Cataloguing in Publication Data
James, M.
The Spectrum book of games
1. Sinclair ZX Spectrum (Computer)—Programming
I. Title II. Gee, S. M. Ewbank, K.
001.64'2 QA76.8.S62/

ISBN 0-246-12047-9

First published in Great Britain 1983 by Granada Publishing Ltd
Reprinted 1983 (twice)

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

The Spectrum Book of Games

Contents

Introduction

This collection of twenty-one games has been written specially for the Spectrum. 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 our exciting sound effects.

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 Spectrum Invaders, Rainbow Squash and Bobsleigh. Others will have titles that don't ring any bells – Sheepdog Trials, Commando Jump and Across the Ravine. Laser Attack and Mighty Missile are both 'zap-the-enemy' type games with special features that make them very different from others we've played and we're sure you'll like the difference. Games that we don't call moving graphics games still use graphics that move. Capture the Quark is a board game in which you play against the computer but it makes very striking use of graphics and also uses sound effects to advantage. Spectrum Dice also uses graphics, sound and colour as does Mirror Tile. Indeed there is only one program in the entire collection that does not use graphics at all and that is Spectrum Smalltalk which is a program that enables your Spectrum to hold a

conversation with you. If you think we are joking you'll have to try it for yourself.

Improve your programming

As well as hoping to provide programs that you can have hours of fun with, we also hope to cater for all Spectrum owners by presenting a book that can be used in more than one way. You can use it simply as a source of exciting games programs, or you can use it to further improve your own knowledge of Spectrum 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 own programming skills.

Listing conventions

The programs included have all been extensively tested in the form in which they appear. As a deliberate policy we decided not to renumber the programs once they were in their final, fully-functioning form. By doing this we hoped to eliminate a source of possible errors. On the other hand we decided against reproducing programs listings from the ZX Printer. This decision meant that we could lay out the programs in a more easy-to-read form, with lines split, where necessary, at points that preserved their syntax and blank lines inserted between subroutines. Additionally we have produced listings that indicate how you should type the programs in, rather than how the programs will look once typed in. For example, the keywords GOSUB and GOTO appear as single words, as they do on the Spectrum keyboard, rather than as two words, as they do on the screen once entered. When you type the programs into your Spectrum however, you will not need to leave spaces after keywords, because the Spectrum will do this for you. Indeed, the only place you will ever need to use the SPACE key is when you want to use a space within a print statement. All characters that need to be entered in graphics mode have been enclosed in square brackets. So, if you need to type the letter 'a' in graphics mode you will see [a] in the program listing. Sometimes you will need to hold the CAPS SHIFT down while in graphics mode in order to produce graphics characters. (This applies to the characters that are printed on the

number keys.) Whenever you need to use the CAPS SHIFT in graphics mode the symbol ^ will be printed immediately before the number in question. For example [^ 1] indicates that you should, while in graphics mode, press the 1 key with the CAPS SHIFT held down. Each program is accompanied with its own 'typing tips' so you'll find this explanation repeated from time to time.

The only problem we've encountered using a different printer is that zeros appear without an oblique stroke through them. This is however unlikely to cause any confusion. If you ever need to type an upper-case letter 'O' or if a lower-case 'o' is used as a variable name you'll be alerted to this in the typing tips section. To avoid any confusion between lower case 'l' and the number 1, every 'l' used as a variable name is shown in capitals. However, you don't need to copy this convention when you type the programs in yourself!

Perfect programming

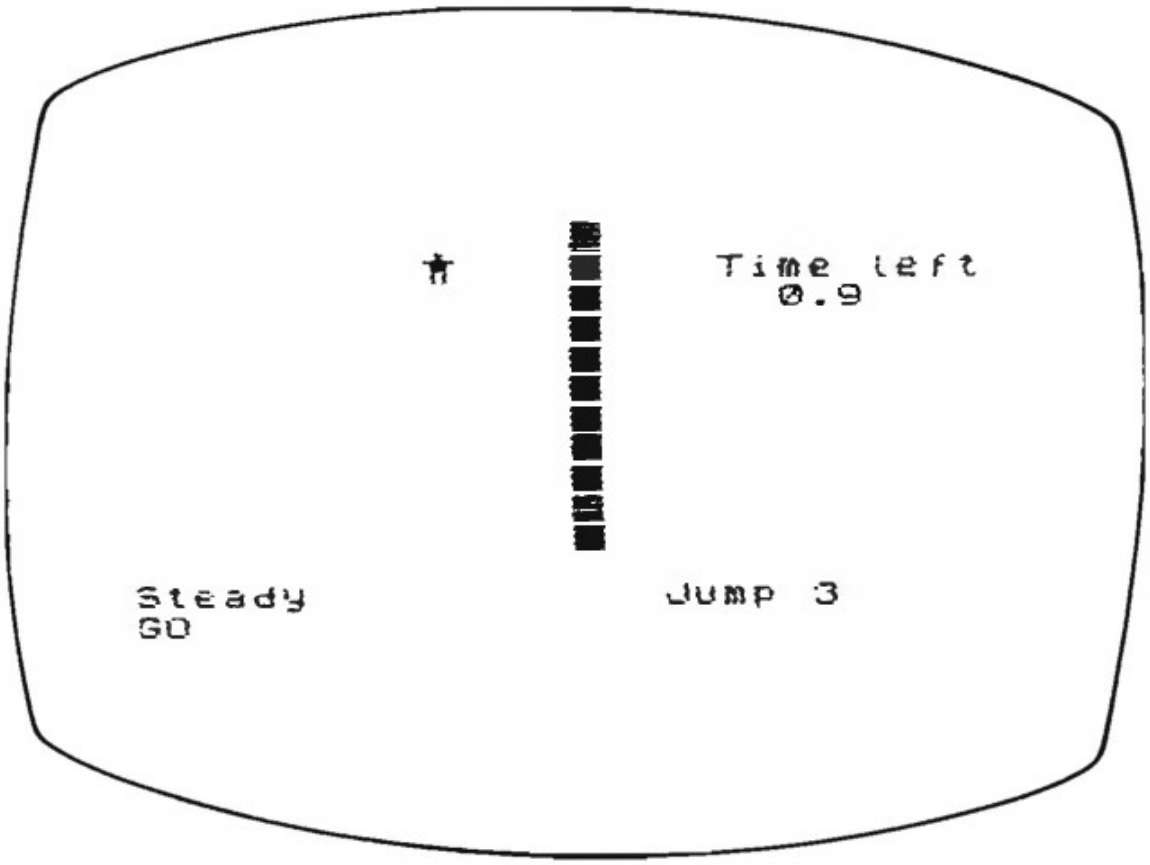
It's a well known fact that there is no such thing as the last bug in a program. No matter how careful we've been we'll never be able to guarantee twenty-one absolutely perfect programs. But it's a fact that bugs creep in whenever you enter a program – so if a program won't work when you've typed it 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. Particular points to look out for are null strings and rows of blank spaces – have you typed the correct number of them. To help you we've included REM statements that tell you how many spaces are needed when its difficult to see by eye. Another source of possible errors is the less than and greater than symbols – getting these round the wrong way will lead to chaotic results. Also, don't be tempted to change the line numbering when you type in the programs as there are far too many pitfalls involved!

Cassette tapes

Many of the programs in this book are fairly long. This is unavoidable as the games concerned include lots of features. Long programs can be a chore to type in and so to save you doing it all for

yourselves the programs, exactly as listed here, are also available on cassette tapes. 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
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 scramble up the remainder of the wall. Your 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 you will hear a bleeping sound – 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.

Typing tips

The letter 'l' is used as a variable name in this game. To avoid confusion with the number '1' it has been printed as 'L' in the program listing. You'll probably find it easier to type it in lower-case just like all the other variables. The 'a's and 'b's in square brackets denote the user-defined graphics characters and you should type them in graphics mode minus the brackets.

Subroutine structure

20	Play loop
1000	Countdown
1500	Cheat routine
2000	Prints wall
2060	Jump logic
2500	Fall down wall
2600	Prints man over wall
7000	Zeroes time
7100	Gets time
8000	Sets up game and defines man and brick characters
9000	Win/lose messages
9600	End of game

Programming details

This is a fairly straightforward application of low resolution dynamic graphics and its main programming interest is in the way

the Spectrum's internal clock is used as a reaction timer and a countdown device. The clock is zeroed by routine 7000 and read by routine 7100. You could use these routines in programs of your own. Another interesting point to note is that the repeat feature of the keyboard is disabled for this game by POKEing memory locations (in lines 8130–8140). The repeat key feature is restored by the end of game routine (lines 9650–9660).

Program

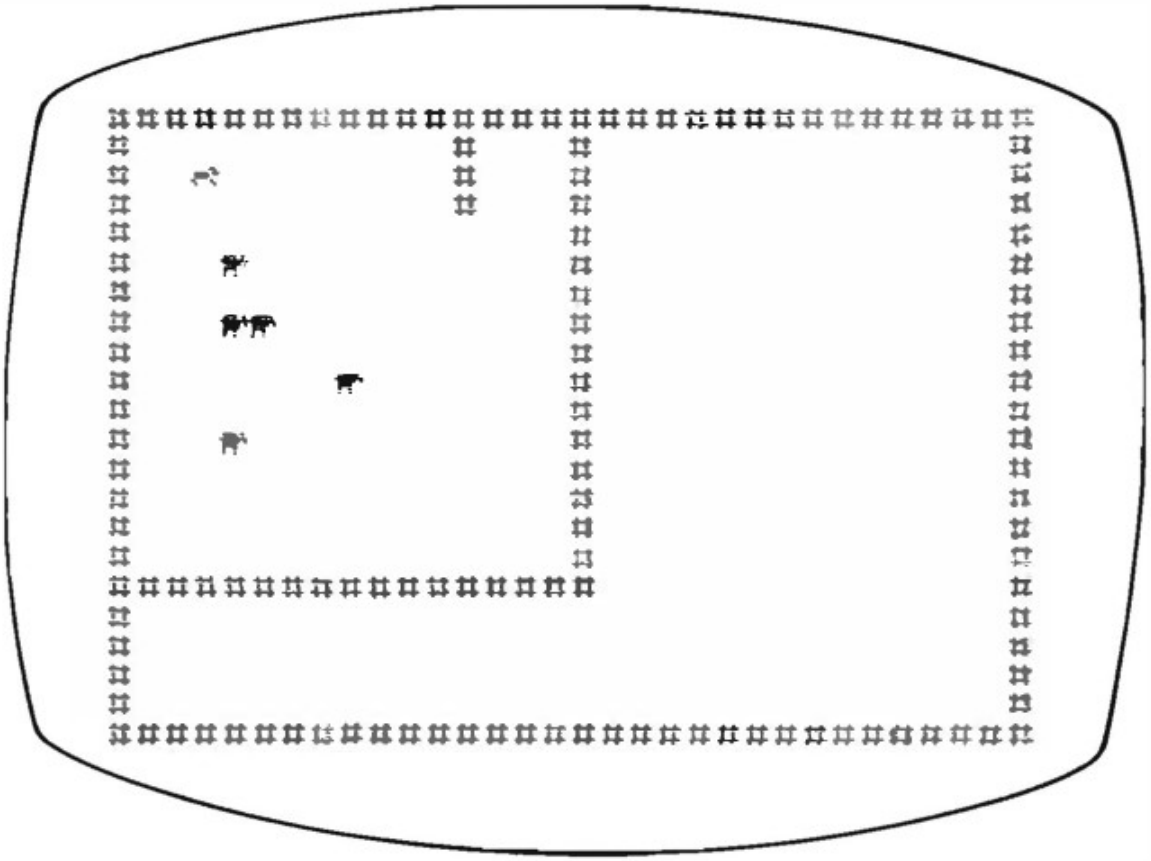
```
   10 REM Commando Jump
   20 GOSUB 8000
   30 GOSUB 2000
   50 GOTO 9000

 1000 PRINT AT 20,0; FLASH 1;"Ready";
      FLASH 0;"  ";AT 21,0;"  "
 1010 FOR i=1 TO RND*200+200
 1020 NEXT i
 1030 PRINT AT 20,0; FLASH 1;"Steady";
      FLASH 0;"  "
 1040 FOR i=1 TO RND*200+200
 1050 NEXT  i
 1060 IF INKEY$<>"" THEN GOTO 1500
 1070 GOSUB 7000
 1080 PRINT "GO"
 1090 IF INKEY$="" THEN GOTO 1090
 1100 GOSUB 7100
 1110 RETURN

 1500 PRINT AT 5,20 ; BRIGHT 1; FLASH 1;
      "Cheat"
 1510 BEEP 5,-8
 1520 PRINT AT 5,20;"      ";REM 5 spaces
 1530 LET t=5
 1540 RETURN
```

```
2000 PAPER 5: CLS : LET jump=1
2010 LET h=10+INT (RND*5)
2020 FOR i=18 TO 19-h STEP -1
2030 PRINT AT i,15; INK 2; PAPER 7;
     "[a]"
2040 NEXT i
2050 PRINT AT 18-h,15; INK 2; PAPER 7;
     "[^8]"

2060 PRINT AT 20,18;"Jump ";jump
2070 PRINT AT 18,18; INK 0;"[b]"
2100 GOSUB 1000
2110 FOR i=18 TO 18-h+INT (t*15) STEP -1
2120 PRINT AT i,18; INK 0;" "
2130 PRINT AT i-1,18; INK 0;"[b]"
2140 BEEP .01,20-i

2150 NEXT i
2160 LET j=i: LET L=INT i
2170 GOSUB 7100
2175 IF t>5 THEN GOTO 2500
2180 PRINT AT 9,20;"Time left";
     AT 10,22; INT ((5-t)*10)/10;"  "
2190 IF INKEY$="" THEN GOTO 2170
2200 PRINT AT INT L,18; INK 0;" "
2210 LET j=j-0.2
2215 LET L=INT j
2220 PRINT AT L,18; INK 0;"[b]"
2230 IF L<=17-h THEN PRINT AT L+1,18;" ":
     GOTO 2600
2340 IF INKEY$<>"" THEN BEEP .01,5:
     GOTO 2340
2350 GOTO 2170

2500 FOR i=L TO 18
2510 PRINT AT i-1,18; INK 0;" "
2520 PRINT AT i,18; INK 0;"[b]"
2530 BEEP .01,20-i
2540 NEXT i
2550 LET jump=jump+1
2560 PRINT AT 10,25;"    ": REM 4 spaces
2570 IF jump<=10 THEN GOTO 2060
2580 RETURN
```

```
2600 FOR i=18 TO 10 STEP -1
2610 PRINT AT L,i+1; INK 0;" "
2620 PRINT AT L,i; INK 0;"[b]"
2630 FOR k=1 TO 10: NEXT k
2640 NEXT i
2650 FOR i=L TO 18
2660 PRINT AT i-1,10; INK 0;" "
2670 PRINT AT i,10; INK 0;"[b]"
2675 BEEP .01,20-i
2680 FOR k=1 TO 10: NEXT k
2690 NEXT i
2700 RETURN

7000 POKE 23674,0
7010 POKE 23673,0
7020 POKE 23672,0
7030 RETURN

7100 LET t=(PEEK 23672+256*PEEK 23673)/50
7110 RETURN

8000 POKE USR "a",0
8010 FOR i=1 TO 7
8020 POKE USR "a"+i,255
8030 NEXT i
8040 POKE USR "b"+0,BIN 00011000
8050 POKE USR "b"+1,BIN 00011000
8060 POKE USR "b"+2,BIN 11111111
8070 POKE USR "b"+3,BIN 00111100
8080 POKE USR "b"+4,BIN 00111100
8090 POKE USR "b"+5,BIN 00100100
8100 POKE USR "b"+6,BIN 00100100
8110 POKE USR "b"+7,BIN 00100100
8120 LET t=0
8130 POKE 23561,255
8140 POKE 23562,255
8150 RETURN

9000 IF jump<=10 THEN GOTO 9500
9010 PRINT AT 0,10;"You "; FLASH 1;
BRIGHT 1;"FAILED!!!"
9020 GOTO 9600
9500 PRINT AT 0,10;"You took ";jump;
     " jumps to"
9510 PRINT AT 1,14;"clear the wall"
```

```
9600 INPUT "Another game ?";a$
9610 IF a$(1)="y" THEN RUN
9620 PAPER 7
9630 INK 0
9640 CLS
9650 POKE 23561,35
9660 POKE 23562,5
```

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 quite an 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 left hand side of their field in the minimum number of moves. To do

this you have to control the dog using the arrow keys (keys 5, 6, 7 and 8). If the dog approaches too close to the sheep they will scatter. (They 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

Wherever you see a character within square brackets remember that this indicates that you must use graphics mode. In this program you will find [s] in lines 1040 and 7350 and [d] in lines 1130 and 1660. The hash character used to print the fence in subroutine 800 is produced by typing the 3 key with SYMBOL SHIFT held down. The only other printing feature to look out for is the single space enclosed in double quotes in lines 1610 and 7300. You may be surprised by the inclusion of '+0' in lines 500 and 600. This is actually an unnecessary embellishment to the POKE USR statement and has only been included in order to make it easier for you to pick out the shapes of the user-defined graphics. For the same reason, you may find it easier to type these lines as they appear but if you prefer you can omit the '+0'.

Subroutine structure

500	Defines sheep graphics character
600	Defines dog graphics character
700	Initialises arrays
800	Prints fences
1000	Prints sheep
1100	Prints dog
1500	Moves dog
1700	Moves sheep and checks for end of game

7000 Move logic for sheep
7300 Prints sheep
8000 Scatters sheep
9000 Prints messages at end of game and offers another go

Programming details

Lines 7100 and 7150 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 8050 and 8070. Notice the use of ATTRibute statements in lines 1620 and 1650. They are used to make sure that the dog does not move into any of the picket fences. Similar statements are used in lines 7170 and 7175 where they ensure that the sheep do not move on to the dog or on to the picket fence.

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 7100 or you could add some obstacles such as a pond or a river that the sheep have to avoid or cross. Another suggestion is to modify the game to employ a time criterion, using the Spectrum's clock, instead of counting the number of moves needed.

Program

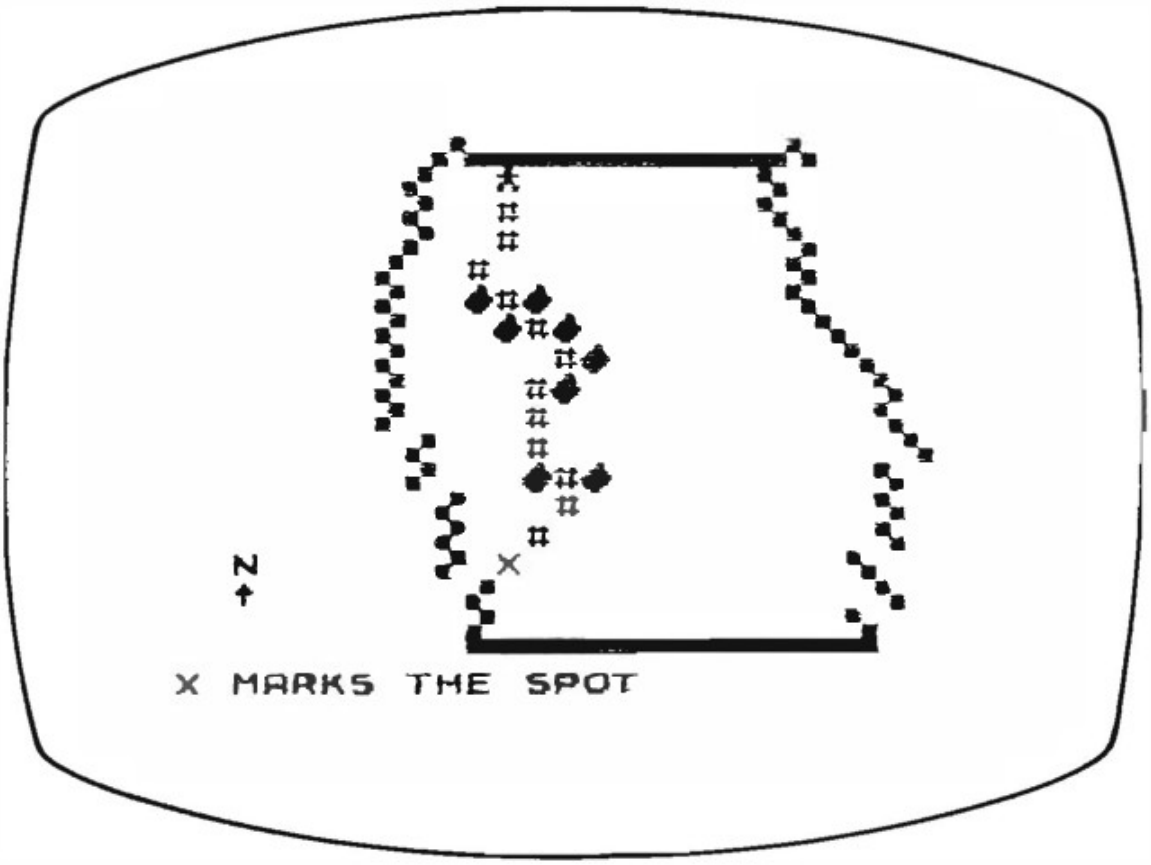
```
100 REM Sheepdog trial

500 POKE USR "s"+0,BIN 00000000
510 POKE USR "s"+1,BIN 00000000
520 POKE USR "s"+2,BIN 01111010
530 POKE USR "s"+3,BIN 11111111
540 POKE USR "s"+4,BIN 01111101
550 POKE USR "s"+5,BIN 01111000
560 POKE USR "s"+6,BIN 01001000
570 POKE USR "s"+7,BIN 01001000
```

```
600 POKE USR "d"+0,BIN 00000000
610 POKE USR "d"+1,BIN 00000000
620 POKE USR "d"+2,BIN 00000110
630 POKE USR "d"+3,BIN 01111011
640 POKE USR "d"+4,BIN 01111000
650 POKE USR "d"+5,BIN 10000100
660 POKE USR "d"+6,BIN 01000010
670 POKE USR "d"+7,BIN 00000000

700 DIM y (5)
710 DIM x (5)
720 LET m=0

800 PAPER 4:INK 4
810 CLS
820 FOR x=0 TO 15
830 PRINT AT 16,x; INK 0;"#"
840 NEXT x
850 FOR y=0 TO 16
860 PRINT AT y,16; INK 0;"#"
870 NEXT y
900 FOR y=0 TO 20
910 PRINT AT y,0; INK 0;"#"; AT y,31;"#"
920 NEXT y
930 FOR x=0 TO 31
940 PRINT AT 0,x; INK 0;"#"; AT 21,x;"#"
950 NEXT x
960 FOR y=1 TO 3
970 PRINT AT y,12; INK 0;"#"
980 NEXT y

1000 REM print sheep
1010 FOR s=1 TO 5
1020 LET y(s)=5+INT (RND*10)
1030 LET x(s)=4+INT (RND*6)
1040 PRINT AT y(s),x(s); INK 7;"[s]"
1050 NEXT s

1100 REM print dog
1110 LET yd=1+INT (RND*3)
1120 LET xd=1+INT (RND*3)
1130 PRINT AT yd,xd; INK 0;"[d]"
```

```
1500 REM move dog
1590 LET d$=INKEY$
1600 IF d$="" THEN GOTO 1590
1610 PRINT AT yd,xd;" "
1620 IF d$="5" AND ATTR (yd,xd-1)<>32
     THEN LET xd=xd-1
1630 IF d$="8" AND ATTR (yd,xd+1)<>32
     THEN LET xd=xd+1
1640 IF d$="6" AND ATTR (yd+1,xd)<>32
     THEN LET yd=yd+1
1650 IF d$="7" AND ATTR (yd-1,xd)<>32
     THEN LET yd=yd-1
1660 PRINT AT yd,xd; INK 0;"[d]"
1670 LET m=m+1
1680 PRINT INK 0; AT 10,20;"Move ";m

1700 GOSUB 7000
1820 IF f=0 THEN GOTO 9000
1850 GOTO 1500

7000 REM move sheep
7010 LET f=0
7050 FOR s=1 TO 5
7070 LET y=y(s):LET x=x(s)

7080 IF (ABS (x(s)-xd)<2 AND
     ABS (y(s)-yd)<2) OR (RND<.01) THEN
     GOSUB 8000
7100 IF ABS(x(s)-xd)>2+INT (RND*2)
     OR ABS(y(s)-yd)>2+INT (RND*2) THEN
     GOTO 7175
7150 LET x(s)=x(s)+SGN(x(s)-xd)
7160 LET y(s)=y(s)+SGN(y(s)-yd)
7170 IF ATTR (y(s),x(s))=39 THEN
     GOTO 7150
7175 IF ATTR (y(s),x(s))=32 THEN
     LET x(s)=x:LET y(s)=y(s)+1
7180 IF x(s)<1 THEN LET x(s)=1
7190 IF x(s)>14 THEN LET x(s)=14
7260 IF y(s)<1 THEN LET y(s)=1
7270 IF y(s)>14 THEN LET y(s)=14
```

```
7300 PRINT AT y,x; INK 4; " "
7350 PRINT AT y(s),x(s); INK 7; "[s]"
7360 IF x(s)>12 AND (y(s)=1 OR y(s)=2
     OR y(s)=3) THEN GOTO 7400
7370 LET f=1

7400 NEXT s
7500 RETURN

8000 REM scatter sheep
8050 LET x(s)=x(s)+((SGN (RND-.5))*
     (2+INT (RND*2)))
8070 LET y(s)=y(s)+((SGN (RND-.5))*
     (2+INT (RND*2)))
8100 RETURN

9000 PRINT AT 18,2;
9005 IF m<40 THEN PRINT INK 0;
     "Super Shepherd!!":GOTO 9080
9010 IF m<60 THEN PRINT INK 0;
     "Good Dog!!":GOTO 9080
9020 IF m<90 THEN PRINT INK 0;
     "Keep practising":GOTO 9080
9025 IF m<120 THEN PRINT INK 0;
"Better luck next time":GOTO 9080
9030 PRINT INK 0;"Hand in your crook !!"
9080 PRINT AT 20,2; INK 0;"You took ";m;
     " moves";
9090 INPUT "Another game (y/n) ";a$
9100 IF a$(1)="y" THEN RUN
9110 PAPER 7:INK 0
9120 STOP
```

3
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. 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 the path 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.

Typing tips

Following our usual conventions, characters enclosed in square brackets in this program are to be entered in graphics mode, and those in square brackets preceeded by an up-arrow, for example [↑3] in line 3140, are to be entered in graphics mode with the CAPS SHIFT held down. Notice the use of 'X' between double quotes in line 3410. Enter this as a straightforward capital 'X'.

Subroutine structure

100	Defines graphics character for parrot
200	Defines graphics character for natives
300	Defines graphics character for sail of ship
400	Defines graphics character for hull of ship
500	Defines graphics character for quicksand
600	Defines graphics character for man
810	Initialises variables and arrays
1050	Main play loop
1410	Logic for natives' attack
2000	Logic for parrot's help
3000	Prints map
3500	Moves man
3600	Prints quicksands on map
4000	Logic for quicksands
5000	Constructs island
5500	Constructs path

5650 Locates treasure
5700 Constructs quicksands
6000 Moves, prints pirate ship
7000 Prints island
8000 Help routine
9000 Treasure found routine
9990 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 5000. 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 alter the length of the PAUSE in line 8020. It is currently set at 50 fiftieths of a second plus a random factor. You can choose how many fiftieths of a second you would like as the fixed value of the PAUSE and substitute that value in place of the 50. You could also alter the amount that the pirate ship moves at each step by setting a different value for 'r' in line 6030. Currently it's set randomly to a value between 0 and 2.

Program

```
10 REM Treasure island
```

```
100 REM parrot
110 POKE USR "p"+0,BIN 00100000
120 POKE USR "p"+1,BIN 00110110
130 POKE USR "p"+2,BIN 00111110
140 POKE USR "p"+3,BIN 00011100
150 POKE USR "p"+4,BIN 00011110
160 POKE USR "p"+5,BIN 00100111
170 POKE USR "p"+6,BIN 01000000
180 POKE USR "p"+7,BIN 00000000

200 REM native
210 POKE USR "n"+0,BIN 00000001
220 POKE USR "n"+1,BIN 00011001
230 POKE USR "n"+2,BIN 11011001
240 POKE USR "n"+3,BIN 11111111
250 POKE USR "n"+4,BIN 11011001
260 POKE USR "n"+5,BIN 00011001
270 POKE USR "n"+6,BIN 00100101
280 POKE USR "n"+7,BIN 00100101

300 REM ship sail
310 POKE USR "j"+0,BIN 11111111
320 POKE USR "j"+1,BIN 01111110
330 POKE USR "j"+2,BIN 10100101
340 POKE USR "j"+3,BIN 11000011
350 POKE USR "j"+4,BIN 11000011
360 POKE USR "j"+5,BIN 10100101
370 POKE USR "j"+6,BIN 01111110
380 POKE USR "j"+7,BIN 11111111

400 REM ship
410 POKE USR "s"+0,BIN 00010000
420 POKE USR "s"+1,BIN 11111111
430 POKE USR "s"+2,BIN 11111111
440 POKE USR "s"+3,BIN 01111110
450 POKE USR "s"+4,BIN 01111110
460 POKE USR "s"+5,BIN 01111110
470 POKE USR "s"+6,BIN 00111100
480 POKE USR "s"+7,BIN 00111100
```

```
 500 REM quicksand
 510 POKE USR "q"+0,BIN 00001100
 520 POKE USR "q"+1,BIN 00011100
 530 POKE USR "q"+2,BIN 00111110
 540 POKE USR "q"+3,BIN 01111111
 550 POKE USR "q"+4,BIN 11111111
 560 POKE USR "q"+5,BIN 11111110
 570 POKE USR "q"+6,BIN 01111100
 580 POKE USR "q"+7,BIN 00111000

 600 REM man
 610 POKE USR "m"+0,BIN 00011000
 620 POKE USR "m"+1,BIN 00011000
 630 POKE USR "m"+2,BIN 01111110
 640 POKE USR "m"+3,BIN 00011000
 650 POKE USR "m"+4,BIN 00111100
 660 POKE USR "m"+5,BIN 01100110
 670 POKE USR "m"+6,BIN 01100110
 680 POKE USR "m"+7,BIN 00000000

 810 LET f=0
 820 LET xs=1: LET ys=1
 830 LET mes=0
 840 DIM v(10)
 850 DIM u(10)
 860 DIM L(20)
 870 DIM r(20)
 880 DIM x(20)

1050 GOSUB 5000
1060 LET xm=x(t+1)
1070 LET ym=t+1
1100 GOSUB 3000
1110 PAUSE 50+RND*20
1150 GOSUB 7000
1200 GOSUB 3570
1210 GOSUB 3500
1270 IF xm=xt AND ym=yt THEN GOTO 9000
1280 LET f=f+1
1290 IF INT (f/5)=f/5 THEN GOSUB 6000
1300 IF x(ym)=xm AND INT (f/5)=f/5 THEN
     GOTO 1150
1310 IF x(ym)=xm THEN GOTO 1210
```

```
1330 REM off path
1340 BEEP .3,-6: GOSUB 6000
1350 FOR q=1 TO 10
1360 REM test for quicksands
1370 IF v(q)=xm AND u(q)=ym
     THEN GOSUB 4000
1380 NEXT q
1400 IF RND<=.4 THEN GOTO 2000

1410 REM natives logic
1420 GOSUB 7000
1430 PRINT AT 19,1; PAPER 8; INK 0;
     "HOSTILE NATIVES AHEAD"
1440 FOR n=1 TO 3
1445 LET r=INT (RND*3)
1448 IF ym+r>=b THEN LET r=0
1450 PRINT AT ym+r,xm; PAPER 8;
     INK 0;"[n]"
1460 NEXT n
1470 LET ym=ym-3
1480 IF ym<=t+1 THEN LET ym=t+1
1490 LET xm=x(ym)
1500 PRINT AT ym,xm; PAPER 8;
     INK 0;"[m]"
1550 LET mes=1
1560 GOTO 1210

2000 REM parrot logic
2010 GOSUB 7000
2020 GOSUB 3570
2050 PRINT AT 19,1; PAPER 8; INK 0;
     "FOLLOW LONG JOHN SILVERS PARROT"
2100 LET yj=ym+1
2110 IF yj>p THEN LET yj=p
2120 LET xj=x(yj)
2130 PRINT AT yj,xj; PAPER 8; INK 2;"[p]"
2140 LET mes=1
2150 GOTO 1210
```

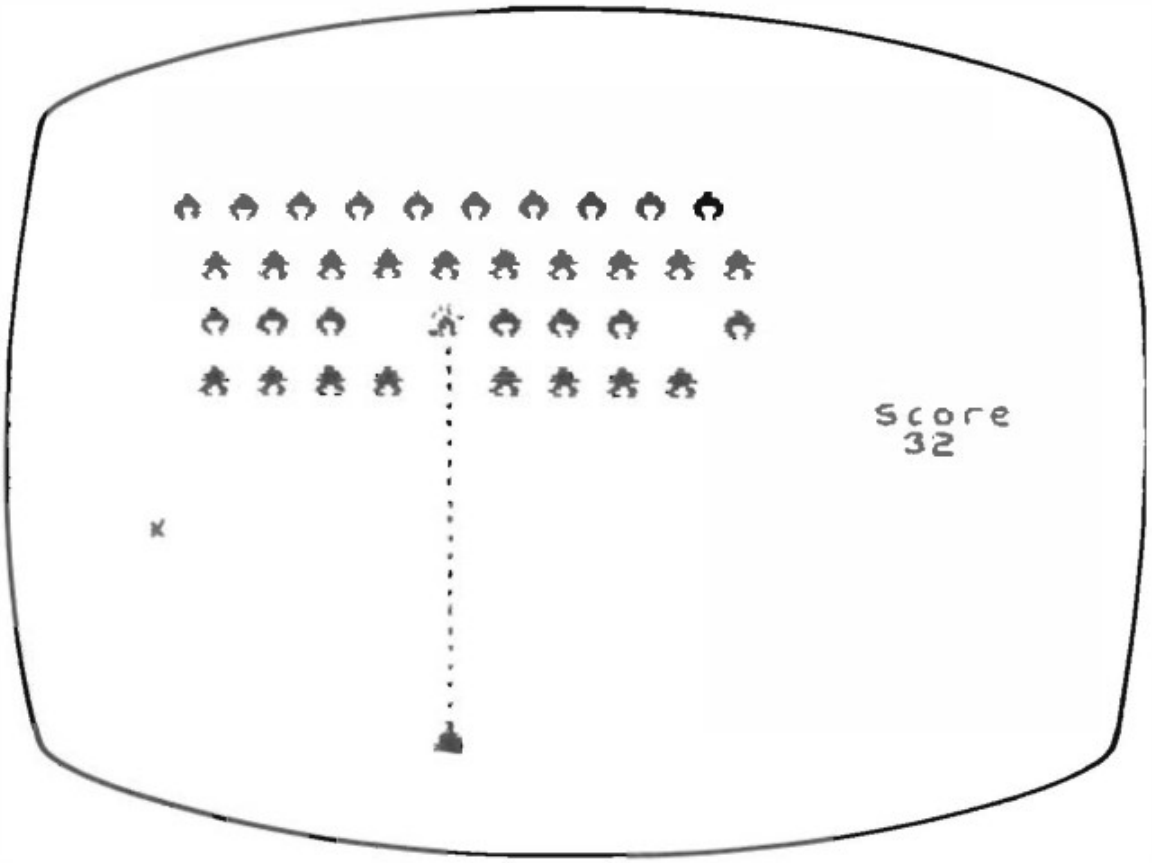
```
3000 REM print island
3110 PAPER 7: INK 0
3120 CLS
3130 FOR x=L(t) TO r(t)
3140 PRINT AT t,x;"[^3]"
3150 NEXT x
3160 FOR y=t TO b
3170 PRINT AT y,L(y);"[^6]";AT y,r(y);"[6]"
3180 NEXT y
3250 FOR x=L(y-1) TO r(y-1)
3260 PRINT AT y,x;"[3]"
3270 NEXT x

3300 REM print path
3350 FOR y=t+1 TO p
3360 PRINT AT y,x(y);"#"
3370 NEXT y
3400 PRINT AT 20,1;"X MARKS THE SPOT"
3410 PRINT AT p,x(p);"X"
3420 PRINT AT ym,xm;"[m]"
3440 RETURN
3460 GOTO 3510

3500 REM move man
3510 BEEP .1,6
3515 LET a$=INKEY$
3520 IF a$="" THEN GOTO 3515
3525 PRINT AT ym,xm; PAPER 8; INK 0;" "
3530 IF a$="h" THEN GOSUB 8000: RETURN
3535 IF a$="5" THEN LET xm=xm-1:
     GOTO 3560
3540 IF a$="8" THEN LET xm=xm+1:
     GOTO 3560
3550 IF a$<>"6" THEN GOTO 3510
3560 LET ym=ym+1
3570 PRINT AT ym,xm; PAPER 8; INK 0;"[m]"
3580 IF mes=0 THEN RETURN
3585 FOR i=0 TO 31
3590 PRINT AT 19,i; PAPER 8;" "
3595 NEXT i
3598 LET mes=0: RETURN
```

```
3600 REM print quicksands
3620 FOR q=1 TO 10
3650 PRINT AT u(q),v(q); INK 4; PAPER 8;
     "[q]"
3660 NEXT q
3700 PRINT AT 17,3; INK 0; PAPER 8;"[p]";
     AT 16,3;"[n]"
3800 LET xs=xs+1
3810 LET ys=ys+1
3900 RETURN

4000 REM quicksand
4010 GOSUB 7000
4020 PRINT AT ym,xm; INK 6; PAPER 8;"[q]"
4100 PRINT AT ym+1,xm+1; PAPER 8; INK 0;
     "AARGH"
4110 FOR i=5 TO -5 STEP -1
4120 BEEP .1,i
4150 NEXT i
4160 PRINT AT 19,1; INK 0; PAPER 8;
     "IN THE QUICKSAND"
4170 IF RND>.5 THEN PRINT INK 0: PAPER 8;
"A friendly native pulled you out":
     PAUSE 100: RETURN
4180 GOTO 9990

5000 REM calculate island
5120 LET L=INT (RND*3)+10
5130 LET t=INT (RND*3)+2
5140 LET w=10+INT (RND*3)
5150 LET b=INT (RND*2)+17
5230 FOR y=t TO b
5240 LET L(y)=L
5250 LET r(y)=L+w
5340 LET L=L-(SGN (10-y)*INT (RND*2))
5350 LET w=w+(SGN (10-y)*INT (RND*2))
5360 IF L(y)<1 THEN LET L(y)=1
5370 IF r(y)>30 THEN LET r(y)=30
5380 NEXT y
```

```
5500 REM path logic
5510 LET x(t+1)=L(t+1)+INT (RND*4)
5520 LET k=t+2
5550 FOR p=k TO b-1-INT (RND*3)
5610 LET x(p)=x(p-1)+INT (RND*3)-1
5620 IF x(p)>=r(p) THEN LET x(p)=r(p)-1
5630 IF x(p)<=L(p) THEN LET x(p)=L(p)+1
5640 NEXT p
5645 LET p=p-1

5650 REM treasure
5660 LET xt=x(p)
5670 LET yt=p

5700 REM quicksand logic
5710 FOR q=1 TO 10
5720 LET d=INT (RND*(p-t-2))+t+1
5750 LET u(q)=d
5760 LET v(q)=x(d)+SGN (RND-.5)
5770 IF v(q)<=L(d) THEN LET v(q)=v(q)+3
5780 IF v(q)>=r(d) THEN LET v(q)=v(q)-3
5790 NEXT q
5800 RETURN

6000 REM pirate ship
6010 INK 5: PAPER 5
6020 CLS
6030 LET r=INT (RND*3)
6100 LET xs=xs+r
6200 LET ys=ys+r
6210 PRINT AT 18,18; INK 4;"[q]"
6220 PRINT AT ys,xs; INK 0;"[j]"
6230 PRINT AT ys+1,xs; INK 0;"[s]"
6240 PAUSE 50
6250 IF ys<18 THEN RETURN
6280 PRINT AT 19,1; PAPER 8; INK 0;
     "THE PIRATES HAVE LANDED"
6290 PRINT PAPER 8; INK 0;"YOU ARE
     CAPTURED"
6300 GOTO 9990
```

```
7000 REM reprint island
7100 PAPER 5: INK 0
7110 CLS
7200 FOR x=L(t) TO r(t)
7210 PRINT AT t-1,x; PAPER 6;" "
7220 NEXT x
7230 FOR y=t TO b
7260 PRINT AT y,L(y)-1; PAPER 6;" "
7270 FOR x=L(y) TO r(y)
7280 PRINT AT y,x; PAPER 4;" "
7290 NEXT x
7330 PRINT AT y,x; PAPER 6;" "
7360 NEXT y
7400 FOR x=L(y-1) TO r(y-1)
7410 PRINT AT y,x; PAPER 6;" "
7420 NEXT x
7500 RETURN

8000 GOSUB 3000
8010 GOSUB 3600
8020 PAUSE 50+INT RND*100
8030 GOSUB 6000
8040 GOSUB 7000
8050 GOSUB 3570
8060 RETURN

9000 REM treasure found
9100 FOR c=0 TO 6
9150 PAPER c
9160 CLS
9170 BRIGHT 1
9200 PAUSE 5
9230 CLS
9240 BEEP .1,c*2
9260 PAUSE 5
9270 NEXT c
9300 PRINT AT 10,7;"YOU FOUND THE GOLD"
9310 BEEP 1,14
9350 BRIGHT 0
9360 PAUSE 20
9990 INPUT "Another game (y/n)?";a$
9991 IF a$(1)="y" THEN RUN
9992 PAPER 7
9993 INK 0
9994 CLS
```

4
Spectrum Invaders



Two types of alien ships are heading towards earth and you 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 passes this point the game is over – you will have lost. Your only hope is to wipe out 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 way that the game speeds up as the invaders get closer and the use of a two-tone throbbing sound. 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 5 and

8 keys – the ones with the appropriate arrows above them. Press the up arrow (key 7) to fire a missile. You score points for every alien you hit, the ones further away from you counting 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'.

Typing tips

The first sections of this program define four graphics characters. It is up to you whether you copy the +0 in lines 10, 100, 200 and 300 or leave it out. These two characters have no effect on the program. Instead they have a cosmetic value as they serve to produce a tidy listing that enables you to see the shape of the character you are trying to produce. Following our normal convention, when the graphics characters are used later in the program square brackets are used to indicate entering characters in graphics mode. One variable in this program has been labelled "xl". To avoid any confusion, it appears as "xL" in the program listing.

Notice the spacing of the graphics characters in lines 500 and 510. There is a space after the final character in line 500 and a space before the first character in line 510. The effect of this is to ensure that the rows of advancing ships are *staggered* with the ships in the odd rows appearing in the spaces between those in the even rows. This is very important for playing the game. The symbol between quotes in lines 1440, 1555, 1595 and 1635 is a full stop. Don't be worried by the way in which line 7020 ends with "TO)". This is perfectly correct. A similar use of the string slicer occurs in the last line of the program.

Subroutine structure

10	Define first alien ship graphics character
100	Define missile launcher graphics character
200	Define second alien ship graphics character
300	Define explosion graphics character
400	Initialise variable and set background paper and ink colours
500	Initialise strings
1000	Print aliens

1100 Print and move missile launcher
1400 Shoot missile
1500 Test whether alien hit and for end of game
7000 Remove aliens when hit
7510 Increment score when alien hit
7900 Print explosion, produce its sound and display new score
8000 Main play loop
8500 Losing message
9000 Winning message and offer of new game
9500 String function

Programming details

The alien ships are stored in strings and when they are hit a string slicing routine (line 7000) is used to replace the graphics characters by blanks.

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.

Program

```
 5 REM Spectrum invaders

10 POKE USR "i"+0,BIN 00011000
20 POKE USR "i"+1,BIN 00111100
30 POKE USR "i"+2,BIN 01111110
40 POKE USR "i"+3,BIN 11111111
50 POKE USR "i"+4,BIN 11000011
60 POKE USR "i"+5,BIN 11000011
70 POKE USR "i"+6,BIN 01100110
80 POKE USR "i"+7,BIN 00100100
```

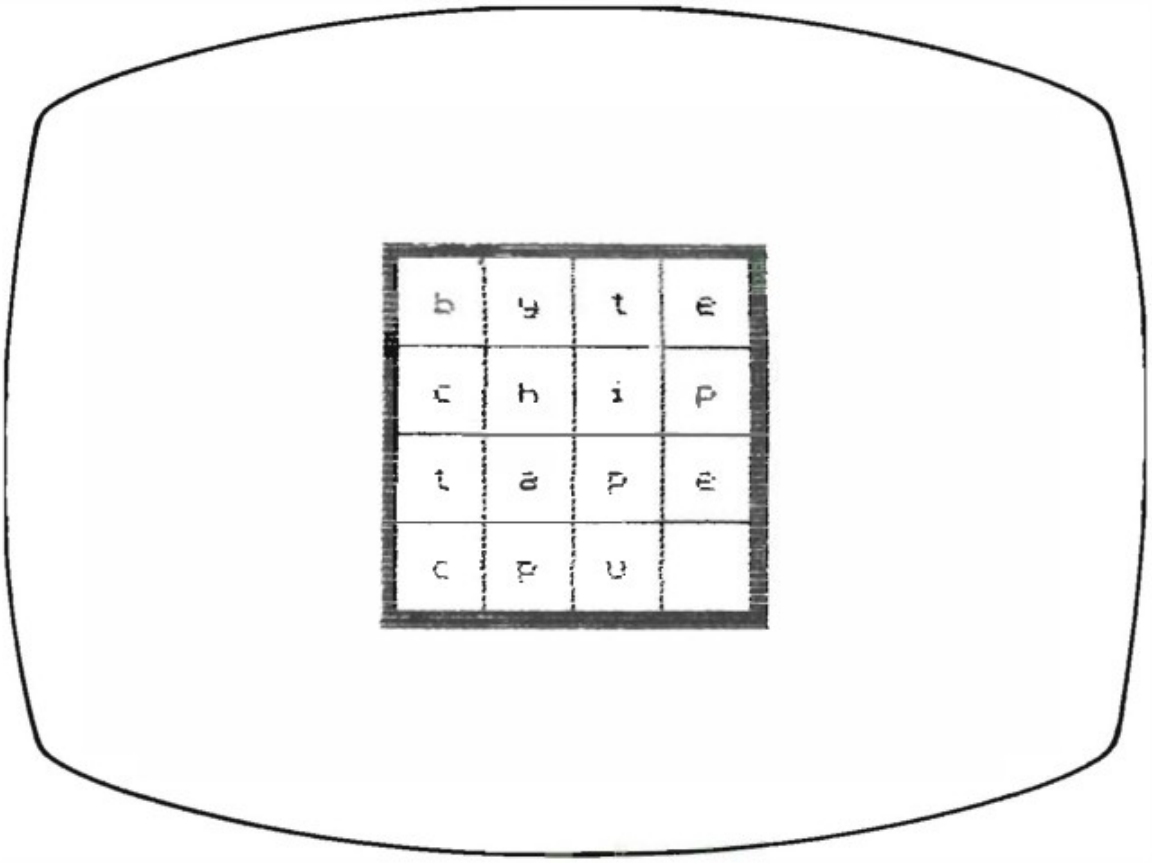
```
100 POKE USR "u"+0,BIN 00011 000
110 POKE USR "u"+1,BIN 00011000
120 POKE USR "u"+2,BIN 00011000
130 POKE USR "u"+3,BIN 00111100
140 POKE USR "u"+4,BIN 01111110
150 POKE USR "u"+5,BIN 01111110
160 POKE USR "u"+6,BIN 11111111
170 POKE USR "u"+7,BIN 11111111

200 POKE USR "j"+0,BIN 00011000
210 POKE USR "j"+1,BIN 00111100
220 POKE USR "j"+2,BIN 011 11110
230 POKE USR "j"+3,BIN 11111111
240 POKE USR "j"+4,BIN 00111100
250 POKE USR "j"+5,BIN 01100110
260 POKE USR "j"+6,BIN 11000011
270 POKE USR "j"+7,BIN 011 00110

300 POKE USR "k"+0,BIN 00101000
310 POKE USR "k"+1,BIN 10001000
320 POKE USR "k"+2,BIN 10010001
330 POKE USR "k"+3,BIN 00101000
340 POKE USR "k"+4,BIN 00011100
350 POKE USR "k"+5,BIN 00110100
360 POKE USR "k"+6,BIN 10100100
370 POKE USR "k"+7,BIN 10100100

400 LET y=1
420 LET xL=10
430 LET ym=0
440 LET t=0
450 LET s=0
460 PAPER 7: INK 0

500 LET a$="[i] [i] [i] [i] [i] [i] [i]
    [i] [i] [i] "
510 LET b$=" [j] [j] [j] [j] [j] [j] [j]
     [j] [j] [j]"
520 LET c$=a$
540 LET d$=b$
550 LET e$="                    ":
    REM 20 spaces
```

```
1000 REM print aliens
1010 CLS
1020 INK 6
1030 PRINT AT y,1;a$;AT y+1,1;c$
1040 INK 2
1050 PRINT AT y+2,1;b$;AT y+6,1;d$
1060 GOTO 8000

1100 REM print/move launcher
1110 INK 4
1120 LET t=t+1
1130 PRINT AT 21,xL;"[u]"
1150 LET L$=INKEY$
1160 IF L$="" THEN RETURN
1170 PRINT AT 21,xL;" "
1180 IF L$="5" AND xL>1 THEN LET xL=xL-1
1190 IF L$="8" AND xL<20 THEN LET xL=xL+1
1200 PRINT AT 21,xL;"[u]"

1400 REM shoot missile
1410 INK 6
1420 IF L$<>"7" THEN RETURN
1430 FOR m=19 TO y+6 STEP -1
1440 PRINT AT m,xL;".";AT m+1,xL;" "
1450 NEXT m
1460 PRINT AT m+1,xL;" "
```

```
1500 REM test if hit aliens
1510 LET f=0
1520 LET q$=d$ : LET r=6
1530 GOSUB 7000
1540 LET d$=q$
1550 IF f=1 THEN GOTO 1670
1555 PRINT AT y+5,xL;".";AT y+4,xL;".";
     AT y+5,xL;" ";AT y+4,xL;" "
1560 LET q$=c$ : LET r=4
1570 GOSUB 7000
1580 LET c$=q$
1590 IF f=1 THEN GOTO 1670
1595 PRINT AT y+3,xL;".";AT y+3,xL;".";
     AT y+2,xL;".";AT y+2,xL;" "
1600 LET q$=b$ : LET r=2
1610 GOSUB 7000
1620 LET b$=q$
1630 IF f=1 THEN GOTO 1670
1635 PRINT AT y+1,xL;".";AT y+1,xL;" ";
     AT y,xL;" "
1640 LET q$=a$ : LET r=0
1650 GOSUB 7000
1660 LET a$=q$
1670 IF a$=e$ AND b$=e$ AND c$=e$ AND
     d$=e$ THEN GOTO 9000
1680 REM no aliens left
1690 IF q$=e$ THEN LET y=y+2:
     PRINT AT y-2,1;e$
1700 GOTO 1100

7000 REM redefine strings
7010 IF q$(xL)=" " THEN RETURN
7020 LET q$=q$(1 TO xL-1)+" "+q$(xL+1 TO )
7030 LET f=1

7510 REM hit alien
7520 LET s=s+10-y
7530 LET q$=FN m$(q$)
7540 GOSUB 7900
7600 RETURN
```

```
7900 FOR m=1 TO 8
7910 PRINT INK 0; OVER 1; AT y+r,xL;"[k]"
7920 BEEP .01,-5
7930 NEXT m
7940 PRINT INK 0; AT 10,25;"Score"
7950 PRINT INK 0; AT 11,26;s;"      "
7960 LET t=t-1
7970 RETURN

8000 REM main play loop
8010 PRINT AT 14,0;"x"
8020 IF t>30+INT (RND*15) THEN  LET
     y=y+2: LET t=0 : PRINT AT y-2,1;e$
8030 LET a$=FN m$(a$)
8040 PRINT AT y,1; INK 6;a$
8045 BEEP .17-y/100,-10 : GOSUB 1100
8050 LET b$=FN m$(b$)
8060 PRINT AT y+2,1; INK 2;b$
8065 BEEP .17-y/100,-14 : GOSUB 1100
8070 LET c$=FN m$(c$)
8080 PRINT AT y+4,1; INK 6;c$
8085 BEEP .17-y/100,-10 : GOSUB 1100
8090 LET d$=FN m$(d$)
8100 PRINT AT y+6,1; INK 2;d$
8105 BEEP .17-y/100,-14 : GOSUB 1100
8110 IF y>8 AND d$<>e$ THEN GOTO 8500
8120 IF y>10 AND c$<>e$ THEN GOTO 8500
8130 IF y>12 AND b$<>e$ THEN GOTO 8500
8140 IF y>14 THEN GOTO 8500
8150 LET t=t+1
8160 GOTO 8000

8500 PRINT INK 0; AT 18,1;
     "they got you !!!! "
8520 GOTO 9020

9000 REM no aliens left
9010 PRINT INK 0; AT 20,1; "well done,
     you saved the world !!";
9020 INPUT "Another game (y/n)";z$
9030 IF z$(1)="y" THEN RUN
9040 INK 0 : PAPER 7
9050 STOP

9500 DEF FN m$(q$)=q$(2 TO )+q$(1)
```

5
Mirror Tile



Although your Spectrum 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 its 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 it is impossible to move on the diagonals.

These same rules apply to the Spectrum 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 16) of the square containing the piece you want to move. If you try to make a wrong move the Spectrum 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 RUN this program the first thing it asks you is whether you wish to input your own set of words. If you reply 'no' then the square will fill with the letters A to N. 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 TEN 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 Spectrum will count your moves and let you know how many you took at the end of the game.

Typing tips

You will have noticed that whereas we usually use only lower case letters for variable names, capital 'L' is used in this program. This is simply to avoid confusion between the lower-case 'l' and the number '1' which appear identical on our printer. So wherever you see a

capital 'L' you can type a lower-case one – remember the Spectrum treats them interchangeably so it recognises no difference between them.

Subroutine structure

10	Define arrays
40	Set up game
120	Main play loop
170	End of game
500	Check for end of game
1000	Set up default (alphabetic) board
2000	Print frame
2500	Shuffle routine
3000	Print number overlay
3500	Blank overlay and print error messages
4000	Move logic
5000	Locate empty space
5500	Print title and initial questions
6000	Do move
6500	Input selection of words
7000	Change attribute status
8000	Do move continued
9000	Define graphics characters used in frame

Programming details

This program is complicated both because of its length and also because it involves a lot of logic. However, its subroutine structure is very clear so if you follow it through section-by-section 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 actually scope for improvement. A routine that reminded the player of the target arrangement might be a very useful extra.

Program

```
  5 REM Mirror tile

 10 DIM b(4,4)
 20 DIM b$(16)
 30 DIM w$(16)

 40 GOSUB 5500
 50 GOSUB 1000
 60 IF wr=1 THEN GOSUB 6500
 70 LET move=0: LET error=0
 80 GOSUB 9000
 90 GOSUB 2000
100 GOSUB 5000
110 GOSUB 2500

120 GOSUB 4000
130 GOSUB 500
140 LET move=move+1
150 IF end<>0 THEN GOTO 120
160 PRINT AT 19,0; INK 0;
    "You did it in ";move;" moves"

170 INPUT "Another game ?";a$
180 IF a$(1)="y" THEN RUN
190 INK 0: PAPER 7
200 CLS
210 STOP

500 LET end=0: LET k=0
510 FOR i=1 TO 4
520 FOR j=1 TO 4
525 LET k=k+1
530 IF b$(b(i,j))<>w$(k) THEN LET end=1
540 NEXT j
550 NEXT i
560 RETURN
```

```
1000 LET k=0
1010 FOR i=1 TO 4
1020 FOR j=1 TO 4
1030 LET k=k+1
1040 LET b$(k)=CHR$(64+k)
1050 LET b(i,j)=k
1060 NEXT j
1070 NEXT i
1080 LET b$(16)=""
1090 LET w$=b$
1100 RETURN

2000 FOR i=0 TO 3
2010 FOR j=0 TO 3
2020 PRINT AT 4+i*3,10+j*3; INK 2;
     PAPER 6;"  [a]"
2030 NEXT j
2040 FOR j=0 TO 3
2050 PRINT AT 5+i*3,10+j*3; INK 0;
     PAPER 6;" "; INK 0;b$(b(i+1,j+1));
     INK 2;"[a]"
2060 NEXT j
2070 FOR j=0 TO 3
2080 PRINT AT 6+i*3,10+j*3; INK 2;
     PAPER 6;"[c][c][b]"
2090 NEXT j
2100 NEXT i
2110 FOR i=0 TO 11
2120 PRINT AT 3,10+i; INK 7; PAPER 2;"[3]"
2130 PRINT AT 16,10+i; INK 2; PAPER 7;"[3]"
2140 PRINT AT 4+i,9; INK 2; PAPER 7;"[5]"
2150 PRINT AT 4+i,22; INK 7; PAPER 2;"[5]"
2160 NEXT i
2170 PRINT AT 3,9; INK 2; PAPER 7;"[4]"
2180 PRINT AT 16,22; INK 7; PAPER 2;"[^2]"
2190 PRINT AT 3,22; INK 2; PAPER 7;"[^7]"
2200 PRINT AT 16,9; INK 2; PAPER 7;"[1]"
2210 RETURN

2500 LET is=4: LET js=4: LET io=0:
     LET jo=0
2510 FOR d=1 TO s
2520 LET i=is: LET j=js
```

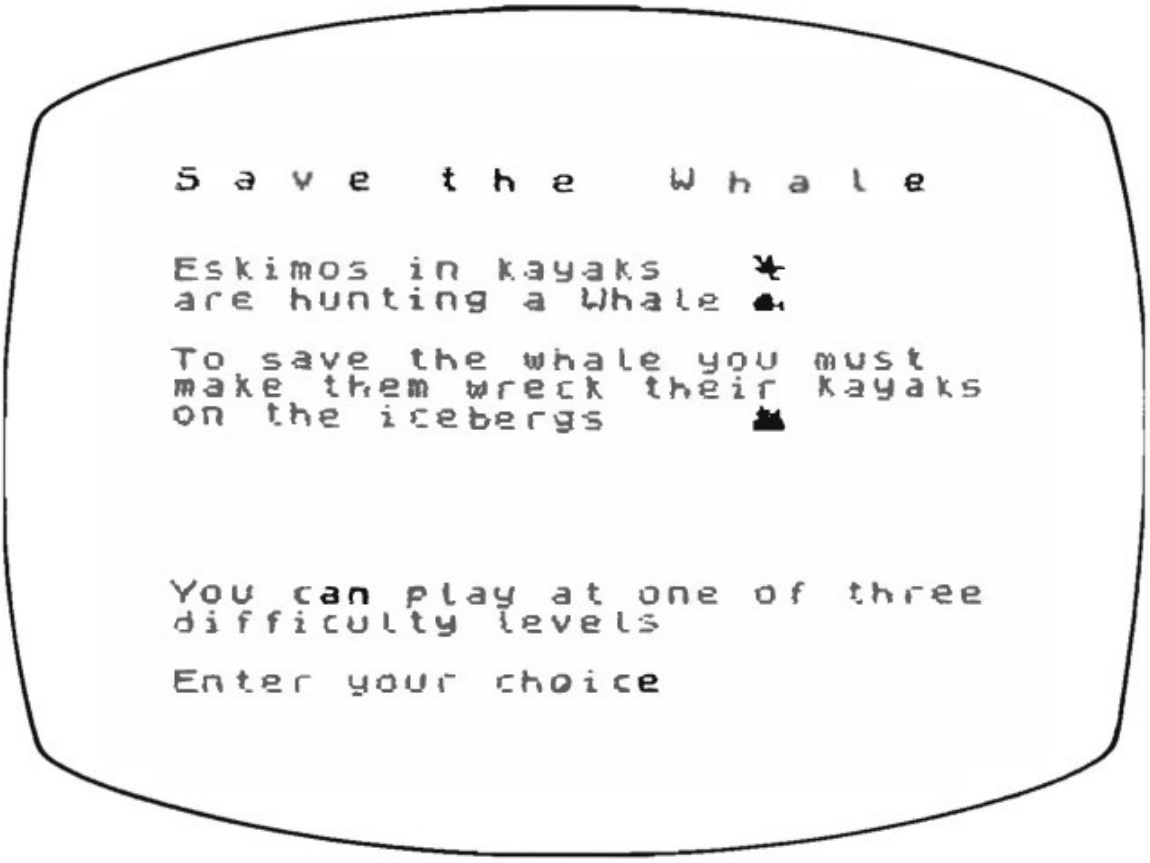
```
2530 IF RND>.5 THEN GOTO 2570
2540 LET i=is+INT (RND*2)*2-1
2550 IF i>4 OR i<1 THEN LET i=is:
     GOTO 2570
2560 GOTO 2590
2570 LET j=js+INT (RND*2)*2-1
2580 IF j>4 OR j<1 THEN LET j=js:
     GOTO 2520
2590 IF i=io AND j=jo THEN GOTO 2520
2600 LET io=is: LET jo=js
2610 GOSUB 6000
2620 NEXT d
2630 RETURN

3000 LET k=0
3010 FOR i=0 TO 3
3020 FOR j=0 TO 3
3030 LET k=k+1
3040 PRINT AT 4+i*3,10+j*3; INK 4;
     PAPER 6;k
3050 NEXT j
3060 NEXT i
3065 PAPER 4
3070 RETURN

3500 FOR L=0 TO 3
3510 FOR p=0 TO 3
3520 PRINT AT 4+L*3,10+p*3; PAPER 6;"   "
3530 NEXT p
3540 NEXT L
3550 INK 0: PAPER 6
3555 IF error=0 THEN RETURN
3560 PRINT AT 20,0;
3570 FOR L=1 TO 64
3580 PRINT PAPER 4;" ";
3590 NEXT L
3600 LET error=0
3610 RETURN
```

```
4000 PAPER 4
4004 INPUT "What is your move "; LINE m$
4005 IF m$="" THEN GOTO 4000
4006 IF LEN m$<2 THEN LET m$="0"+m$
4010 IF m$(1)<"0" OR m$(1)>"9"
     OR m$(2)<"0" OR m$(2)>"9" THEN
     GOSUB 3000: GOTO 4000
4020 LET m=VAL m$
4030 IF m>0 AND m<17 THEN GOTO 4070
4035 LET error=1
4040 PRINT AT 20,0; INK 0;"A move must
     be a number between "
4050 PRINT AT 21,0; INK 0;"1 and 16 as
     shown"
4060 GOSUB 3000
4065 GOTO 4000
4070 LET i=INT ((m-1)/4)
4080 LET j=m-i*4
4090 LET i=i+1
4100 IF ABS (i-is)+ABS (j-js)=1 THEN
     GOTO 6000
4110 BEEP .1,1 5
4115 LET error=1
4120 PRINT AT 20,0; INK 0;"You can only
     move a tile next to"
4130 PRINT AT 21,0; INK 0;"the space
           ": REM 9 spaces
4140 GOSUB 3000
4150 GOTO 4000

5000 LET k=0
5005 FOR L=1 TO 4
5010 FOR p=1 TO 4
5015 LET k=k+1
5020 IF b(p,L)=16 THEN LET is=L:
     LET js=p: LET ms=k
5030 NEXT p
5040 NEXT L
5050 RETURN
```

```
5500 INK 0: PAPER 7
5510 CLS
5520 PRINT TAB 5;"M i r r o r   T i l e"
5530 PRINT AT 10,0;"Do you want to
     input your"
5540 PRINT "own set of words ?";
5550 INPUT a$
5555 IF a$="" THEN GOTO 5550
5560 IF a$(1)="y" THEN LET wr=1: GOTO 5590
5570 IF a$(1)="n" THEN LET wr=0: GOTO 5590
5580 GOTO 5550
5590 PRINT a$
5600 PRINT AT 15,0;"How many shuffles ?";
5610 INPUT s
5620 IF s<1 THEN GOTO 5600
5630 PRINT s
5640 RETURN

6000 GOSUB 3500
6002 INK 0
6005 LET f1=(js-1)*3+(is-1)*3*32+5*32+
     22528+11
6020 LET f=f1: LET c=128
6030 GOSUB 7000
6040 LET f2=(j-1)*3+(i-1)*3*32+5*32+
     22528+11
6050 LET f=f2
6060 GOSUB 7000
6070 GOSUB 8000
6080 PRINT AT 5+(i-1)*3,11+(j-1)*3;
     INK 0; PAPER 6; FLASH 1;b$(b(i,j))
6090 PRINT AT 5+(is-1)*3,11+(js-1)*3;
     INK 0; PAPER 6; FLASH 1;b$(b(is,js))
6100 LET f=f1: LET c=-128
6110 GOSUB 7000
6120 LET f=f2
6130 GOSUB 7000
6140 RETURN
```

```
6500 CLS
6510 PRINT AT 5,0;"Choose 3 four-letter
     words"
6520 PRINT "and 1 three-letter word."
6530 INPUT "Type the first four-letter
     word ";a$
6540 IF LEN a$<>4 THEN GOTO 6530
6550 LET w$(1 TO 4)=a$
6560 PRINT "First word= ";a$
6570 INPUT "Type the second four-letter
     word ";a$
6580 IF LEN a$<>4 THEN GOTO 6570
6590 LET w$(5 TO 8)=a$
6600 PRINT "Second word= ";a$
6610 INPUT "Type the third four-letter
     word ";a$
6620 IF LEN a$<>4 THEN GOTO 6610
6630 LET w$(9 TO 12)=a$
6640 PRINT "Third word= ";a$
6650 INPUT "Type the three-letter
     word ";a$
6660 IF LEN a$<>3 THEN GOTO 6650
6670 LET w$(13 TO 15)=a$
6680 PRINT "Fourth word= ";a$
6690 PAUSE 20
6700 LET w$(16)=" "
6710 LET b$=w$
6720 RETURN

7000 POKE f,PEEK f+c
7050 RETURN

8000 LET b(is,js)=b(i,j)
8030 LET b(i,j)=16
8040 LET t=is: LET is=i: LET i=t
8050 LET t=j: LET j=js: LET js=t
8060 RETURN

9000 For i=0 TO 7
9010 POKE USR "a"+i,BIN 00000001
9020 POKE USR "b"+i,BIN 00000001
9030 POKE USR "c"+i,0
9040 NEXT i
9050 POKE USR "b"+7,BIN 11111111
9060 POKE USR "c"+7,BIN 11111111
9070 PAPER 4
9080 CLS
9090 RETURN
```

6
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. Your selection governs the starting positions of 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 it is 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.

Typing tips

There are three user-defined graphics characters in this game. The whale graphic uses the letter 'u' key, the icebergs use the letter 'i' key and the eskimos use the letter 'e' key. You will find these letters enclosed in square brackets throughout the program, for example [e] occurs in line 1030. Remember, do not type the square brackets. They are there simply to indicate that you must enter graphics mode and then type the letter in question. As usual, when defining graphics characters a redundant '+0' has been included in the first 'POKE USR' line. This is simply to produce a neat listing which allows you to see the shape of the graphics character. If you like you can leave out the +0 but leaving it in will not harm the program.

Subroutine structure

10	Defines arrays
500	Define whale graphics character
600	Define iceberg graphics character
700	Define eskimo in kayak graphics character
1000	Print title frame and set difficulty level
2000	Print eskimos in initial positions
2100	Print icebergs in initial positions
2200	Print whale in initial position
2300	Main play loop
2400	Check for game over
3000	Move whale routine
4000	Move eskimos routine
7000	End game – prints final message and offers new game

Programming details

The initial positions of the kayaks are set at random within a band at the edges of the screen (lines 2020 and 2030). The initial positions of the icebergs are set in a similar fashion (lines 2120 and 2130) but

account is also taken of the difficulty factor input at 1110. ATTRibute statements are used in lines 4090 and 4110. They detect whether a kayak has landed on an iceberg – in which case that is the end of the kayak – or landed on the whale – in which case that is the end of the game.

Scope for improvement

An imaginative user could have great fun with the graphics used in this game. How about adding a water-spout to the whale that was printed every other move?!!

Program

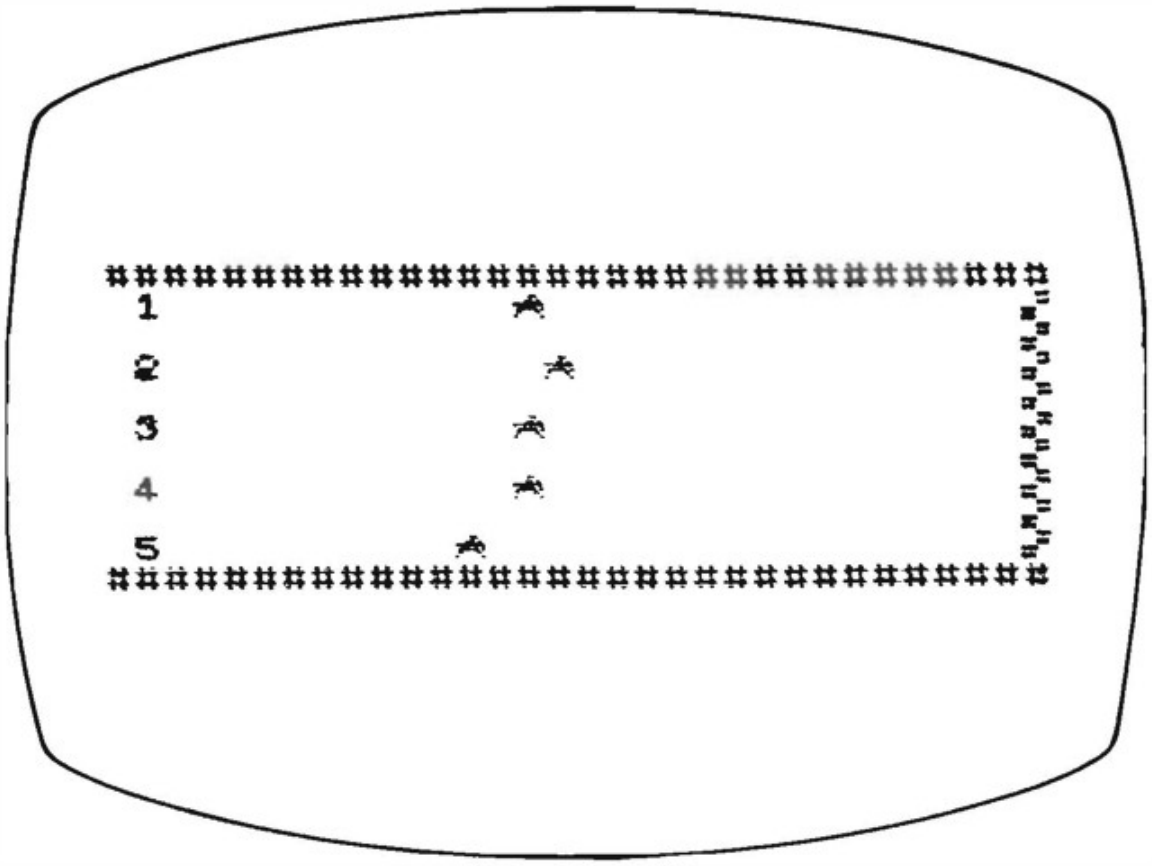
```
  5 REM Save the whale

 10 DIM x(20)
 20 DIM y(20)
500 POKE USR "u"+0,BIN 00000000
510 POKE USR "u"+1,BIN 00000000
520 POKE USR "u"+2,BIN 00110000
530 POKE USR "u"+3,BIN 01111000
540 POKE USR "u"+4,BIN 11111001
550 POKE USR "u"+5,BIN 11111111
560 POKE USR "u"+6,BIN 11111001
570 POKE USR "u"+7,BIN 00000000

600 POKE USR "i"+0,BIN 00000000
610 POKE USR "i"+1,BIN 00100000
620 POKE USR "i"+2,BIN 01110010
630 POKE USR "i"+3,BIN 01110110
640 POKE USR "i"+4,BIN 01111110
650 POKE USR "i"+5,BIN 11111111
660 POKE USR "i"+6,BIN 11111111
670 POKE USR "i"+7,BIN 11111111

700 POKE USR "e"+0,BIN 00000000
710 POKE USR "e"+1,BIN 11001000
720 POKE USR "e"+2,BIN 01011000
730 POKE USR "e"+3,BIN 00111000
740 POKE USR "e"+4,BIN 11111111
750 POKE USR "e"+5,BIN 00111100
760 POKE USR "e"+6,BIN 00001000
770 POKE USR "e"+7,BIN 00000110
```

```
1000 INK 0: PAPER 5
1010 CLS
1020 PRINT AT 2,2;"S a v e   t h e
     W h a l e"
1030 PRINT AT 5,2;"Eskimos in kayaks ";
     INK 2;"[e]"
1040 PRINT "  are hunting a Whale [u]"
1050 PRINT '"  To save the whale you must"
1060 PRINT "  make them wreck their kayaks"
1070 PRINT "  on the icebergs  ";
     INK 7;"[i]"
1080 PRINT AT 16,2;"You can play at one
     of three"
1090 PRINT  "  difficulty levels"
1100 PRINT '"  Enter your choice"
1110 INPUT "  1=difficult,2=medium,
     3=easy ?";d
1120 IF d<1 OR d>3 THEN GOTO 1110

1500 PAPER 5: INK 5
1550 CLS

2000 REM print kayaks
2010 FOR c=1 TO 20
2020 LET x=((SGN (RND-.5))*((RND*4)+11))+15
2030 LET y=((SGN (RND-.5))*((RND*4)+8))+9
2040 INK 2
2050 PRINT AT y,x;"[e]"
2055 LET x(c)=x: LET y(c)=y
2060 NEXT c

2100 REM print icebergs
2110 FOR c=1 TO 20
2120 LET x=((SGN (RND-.5))*((RND*4)+5-d))+15
2130 LET y=((SGN (RND-.5))*((RND*4)+4-d))+9
2140 INK 7
2150 PRINT AT y,x;"[i]"
2170 NEXT c

2200 REM print whale
2210 INK 0
2220 LET x=INT ((RND*2)+10)
2230 LET y=INT ((RND*2)+10)
2240 PRINT AT y,x;"[u]"
```

```
2300 GOSUB 3000
2305 LET f=0
2310 FOR c=1 TO 20
2320 IF x(c)=0 THEN GOTO 2340
2325 LET f=1
2330 GOSUB 4000
2340 NEXT c

2400 IF f=0 THEN GOTO 8000
2410 GOTO 2300

3000 REM move whale
3010 BEEP .1,10
3020 LET z=x: LET v=y
3040 LET a$=INKEY$
3050 IF a$="" THEN GOTO 3040
3060 IF a$="5" THEN LET x=x-1
3070 IF a$="8" THEN LET x=x+1
3080 IF a$="7" THEN LET y=y-1
3090 IF a$="6" THEN LET y=y+1
3105 PRINT AT v,z; INK 5;" "
3110 PRINT AT y,x; INK 0;"[u]"
3120 RETURN

4000 REM move kayaks
4010 PRINT AT y(c),x(c); INK 5;" "
4030 LET e=0: LET d=0
4040 LET d=SGN (x(c)-x)
4050 LET x(c)=INT (x(c)-d)
4060 LET e=SGN (y(c)-y)
4070 LET y(c)=INT (y(c)-e)
4090 IF ATTR (y(c),x(c))=47 THEN
     LET x(c)=0: GOTO 4500
4110 IF ATTR (y(c),x(c))=40 THEN GOTO 7000
4120 INK 2
4130 PRINT AT y(c),x(c); "[e]"
4500 RETURN

7000 PRINT AT 20,1; PAPER 7; "the whale
     is dead";
7005 PRINT AT y(c),x(c); " "
7010 GOTO 9000
```

```
8000 PRINT AT 20,1; PAPER 7; "you escaped
     this time ";
9000 INPUT "another game (y/n)";q$
9010 IF q$(1)="y" THEN RUN
9020 STOP
```

7
Spectrum Ledger



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' but 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 Spectrum 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 Spectrum keeps a tally of your winnings and losses and will tell you if you go broke. During the race, your horse is the white one but as the race is run automatically and at random there's nothing you can do to make your horse win.

Typing tips

Pay very careful attention to the stops and commas in the DATA statements in subroutine 6000. If you make mistakes in typing in this subroutine, the program may still run but the tune may be unrecognisable or sound discordant! The character between quotes in line 1220 is the hash symbol (SYMBOL SHIFT and 3). As usual square brackets denote graphics mode and the ^ symbol in line 1310 indicates that, once in graphics mode you press the CAPS SHIFT as well as the 6 key.

Subroutine structure

500	Defines graphics character for horse
1000	Prints race course
1500	Prints horses
1560	Runs race
2000	Title frame
3000	Betting routine
4000	Moves horses
5000	Plays whole tune
6000	Plays one note of tune
8000	Win/lose routine
9000	Another race or end of game

Programming details

This is the only program in this collection that plays a tune so it is worth drawing your attention to the details of lines 6000–6080. 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 1 or .5 or .25 – representing a minim, a crochet or a semiquaver. The number –99 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 6080 signals the end of the tune. After detecting 99,99 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 5000 which plays all the notes one after the other. However, because the Spectrum cannot do

two things at once – it can't play a note and move the horses – the notes of the tune are produced by a call directly to subroutine 6000 which plays one note in the tune and then moves the pointer to the next note 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 6000 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.

Scope for improvement

If you get tired of 'Camptown Races' as the background music you can substitute 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

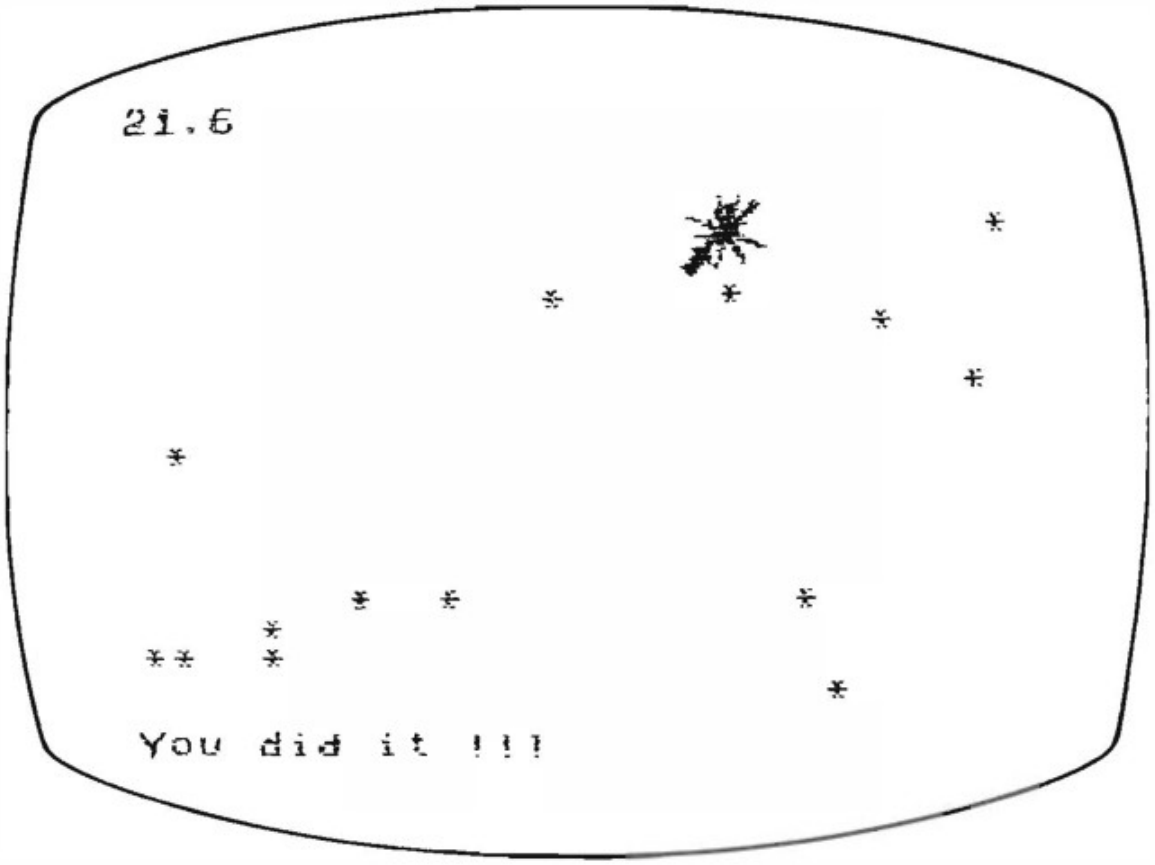
```
 10 REM Spectrum ledger

500 POKE USR "h"+0,BIN 00000000
510 POKE USR "h"+1,BIN 00001100
520 POKE USR "h"+2,BIN 00011010
530 POKE USR "h"+3,BIN 11111111
540 POKE USR "h"+4,BIN 01111101
550 POKE USR "h"+5,BIN 01000010
560 POKE USR "h"+6,BIN 10000001
570 POKE USR "h"+7,BIN 00000000
600 GOSUB 5000
610 LET total=100
```

```
1000 PAPER 4: INK 0
1100 CLS
1130 DIM x(5)
1140 DIM y(5)
1200 FOR x=0 TO 31
1220 PRINT AT 1,x; INK 0; "#";
     AT 11,x;"#"
1230 NEXT x
1300 FOR y=2 TO 10
1310 PRINT AT y,31; INK 0;"[^6]"
1320 NEXT y

1500 FOR y=1 TO 5
1520 LET x(y)=2
1530 LET y(y)=y*2
1540 PRINT AT y(y),x(y)-1; INK 7;y;
     INK 0; "[h]"
1550 NEXT y

1560 GOSUB 3000
1570 LET tempo=8
1710 GOSUB 4000
1720 GOTO 1710

2000 INK 0
2010 PAPER 7
2020 CLS
2040 BORDER 1
2050 PRINT AT 10,1;"S p e c t r u m
     L e d g e r"
2060 RETURN

3000 INPUT "Which horse do you want to"'
     "bet on ?";b
3010 IF b<1 OR b>5 THEN PRINT AT 18,0;
     "no such horse": GOTO 3000
3020 INPUT "you have ";(total)
     '"How much do you want to bet ?";m
3030 IF total-m<0 THEN PRINT AT 18,0;
     "you don't have enough money!":
     BEEP .1,10: GOTO 3020
3040 LET total=total-m
3050 PRINT AT 18,0;"
                    ": REM 30 spaces
```

```
4000 REM move horses
4010 FOR z=1 TO 5
4050 PRINT AT y(z),x(z); INK 4;" "
4110 LET x(z)=X(z)+(1+RND*.6)
4120 PRINT AT y(z),x(z); INK (z=b)*7;"[h]"
4130 IF x(z)>30 THEN GOTO 8000
4135 GOSUB 6000
4138 IF t=99 THEN RESTORE
4140 NEXT z
4150 RETURN

5000 GOSUB 2000
5005 LET tempo=4
5010 LET i=1
5015 GOSUB 6000
5020 IF t=99 THEN PRINT AT 15,i+1;" ":
     RESTORE: RETURN
5030 PRINT AT 15,i;" h"
5035 PAUSE 2
5040 LET i=i+.4
5050 GOTO 5015

6000 DATA 9,.5,9,.5,9,.5,6,.5,9,.5,11,.5,
     9,.5,6,.5,-99,.5,6,.5,4,1.5,6,.5,4,1
6010 DATA 9,.5,9,.5,6,.5,9,.5,11,.5,9,.5,
     6,.5
6020 DATA -99,.5,6,.25,4,.25,2,.25,4,.25,
     6,.5,4,.5,2,1.5
6030 DATA -99,1,2,.75,2,.25,6,.5,9,.5,
     14,1.5
6040 DATA -99,.5,11,.75,11,.25,14,.5,
     11,.5,9,1.5
6050 DATA 6,.25,7,.25,9,.5,9,.5,6,.25,
     6,.25
6060 DATA 9,.25,9,.25,11,.5,9,.5,6,1
6070 DATA 4,.5,6,.5,7,.25,6,.5,4,.25,
     4,.25,2,1.5
6080 DATA 99,99
6090 READ p,t
6100 IF t=99 THEN RETURN
6110 LET t=t/tempo
6120 IF p=-99 THEN PAUSE INT (t*50):
     RETURN
6130 BEEP t,p
6140 RETURN
```

```
8000 IF z=b THEN PRINT AT 18,0;"
     you win ";: LET total =total+INT
     (5*m): PRINT INT (5*m)
8010 IF z<>b THEN PRINT AT 18,0;
     "you lose ";m
8020 IF total<=0 THEN PRINT AT 18,0;
     FLASH 1;"you're broke    ":PAUSE 500:
     GOTO 9020

9000 INPUT "you have ";(total)
     '"Another race";a$
9005 RESTORE
9010 IF a$(1)="y" THEN GOTO 1000
9020 INK 0
9030 PAPER 7
9040 BORDER 7
9050 CLS
```

8
Laser Attack



This is a very exciting game that uses 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 ot 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. It

is shaped like an arrow-head 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

Look out as usual for graphics characters enclosed in square brackets. Remember to type the letters so indicated in graphics mode and without the brackets. Also, as elsewhere in this book, where 'l' is used as a variable name, e.g. in line 2000, it has been printed out as a capital just to avoid confusion with the number '1'.

Subroutine structure

20	Main play loop
1000	Fires or rotates direction of arrow
2000	Laser zap and detect hit logic
3000	Moves target
4000	Moves arrow
7000	Title frame
7500	Gets and prints time
7900	Prints stars
8000	Inititialises variables and defines arrow graphics
8600	Defines cartwheel graphics
8810	Zeroes time
9000	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. Also notice that eight versions of the

arrow graphic are used in order to allow it to be moved in any of eight different directions – also an interesting technique. The way the direction of movement and velocity is set 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 GOSUB 7000
  30 GOSUB 8000
  40 GOSUB 7500
  50 IF INT t>1000 THEN GOTO 9000
  60 GOSUB 1000
  70 GOSUB 3000
  80 IF h=1 THEN GOTO 9000
  90 GOSUB 4000
 100 GOTO 40

1000 LET a$=INKEY$
1010 IF a$="" THEN RETURN
1020 IF a$="7" THEN GOTO 2000
1030 LET k=k+1
1040 IF k>8 THEN LET k=1
1050 RETURN
```

```
2000 LET xL=x*8+4
2010 LET yL=175-y*8-4
2015 GOSUB 2500
2050 PLOT xL,yL
2060 LET dx=0
2070 IF v(k)=1 THEN LET dx=255-xL
2080 IF v(k)=-1 THEN LET dx=-xL
2090 LET dy=0
2100 IF w(k)=1 THEN LET dy=-yL
2120 IF w(k)=-1 THEN LET dy=175-yL
2130 IF v(k)*w(k)=0 THEN GOTO 2200
2140 IF ABS dx<ABS dy THEN
     LET dy=ABS dx*SGN dy: GOTO 2200
2150 LET dx=ABS dy*SGN dx
2200 DRAW dx,dy
2210 PLOT xL,yL
2215 BEEP .01,10
2220 DRAW OVER 1;dx,dy

2230 IF h=0 THEN RETURN
2235 LET mx=b*8+4: LET my=175-a*8-4
2240 FOR i=1 TO RND*5+20
2250 PLOT mx,my
2260 LET dx=10-RND*20
2270 IF mx+dx>255 OR mx+dx<0 THEN
     GOTO 2330
2280 LET dy=10-RND*20
2290 IF my+dy>175 OR my+dy<0 THEN
     GOTO 2330
2300 PLOT mx,my
2310 DRAW dx,dy
2320 BEEP .01,-10
2330 NEXT i
2340 RETURN
2500 LET h=0
2510 LET dy=a-y
2520 LET dx=b-x
2530 IF w(k)*dx<>v(k)*dy THEN RETURN
2540 IF ABS v(k)*SGN dx<>v(k)
     OR ABS w(k)*SGN dy<>w(k) THEN RETURN
2550 LET h=1
2560 RETURN
```

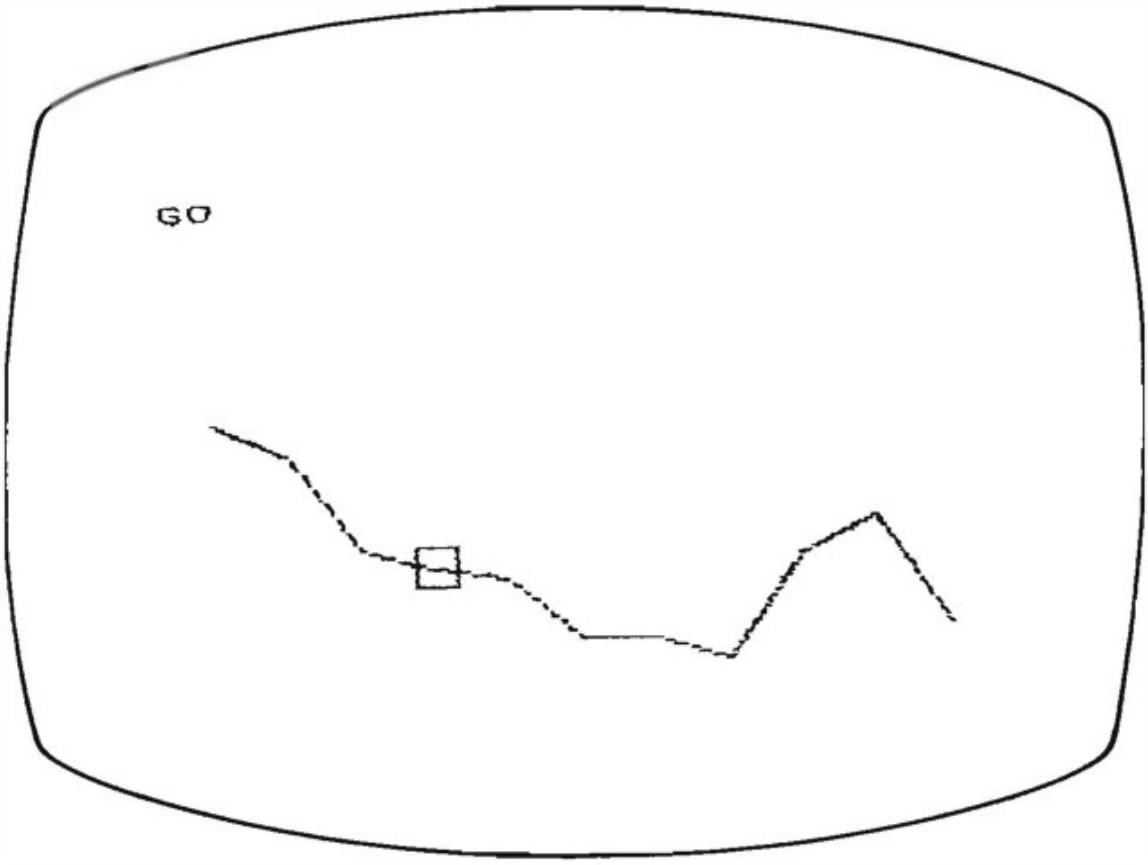
```
3000 IF nb1=0 THEN PRINT AT y,x;" "
3010 LET x=x+v(k)
3020 LET y=y+w(k)
3030 IF x<0 THEN LET x=31
3040 IF x>31 THEN LET x=0
3050 IF y<0 THEN LET y=21
3060 IF y>21 THEN LET y=0
3065 IF ATTR (y,x)<>15 THEN BEEP .1,-10:
     LET nb1=1: GOTO 1030
3070 PRINT AT y,x;m$(k)
3080 LET nb1=0
3090 RETURN

4000 IF RND>1.05-df/20 THEN LET z=z+1
4010 IF z>8 THEN LET z=1
4020 IF nb2=0 THEN PRINT AT a,b;" "
4030 LET a=a+v(z)
4040 LET b=b+w(z)
4050 IF b<0 THEN LET b=31

4060 IF b>31 THEN LET b=0
4070 IF a<0 THEN LET a=21
4080 IF a>21 THEN LET a=0
4085 IF ATTR (a,b)<>15 THEN BEEP .1,-10:
     LET nb2=1: LET z=z+1: GOTO 4010
4090 PRINT AT a,b;w$(r+1)
4100 LET r=NOT r
4110 LET nb2=0
4120 RETURN
```

```
7000 PAPER 1
7010 BORDER 5
7020 INK 7
7030 CLS
7040 PRINT AT 2,5;"L a s e r   A t t a c k"
7050 PRINT AT 5,0;"You are in control
     of an"
7060 PRINT "advanced laser attack ship in"
7070 PRINT "pursuit of an enemy craft"
7080 PRINT ''"Shoot it down before your
     time"
7090 PRINT TAB 5;"is up !!!!"
7100 INPUT "Select the difficulty level
     - 1 (easy) to 10 (difficult) ?";df
7110 IF df<1 OR df>10 THEN GOTO 7100
7120 CLS
7130 RETURN

7500 LET t=(PEEK 23672+256*PEEK 23673+
     65536*PEEK 23674)/5
7510 PRINT AT 0,0; INT t/10;"  "
7520 RETURN

7900 IF RND>.1+df/50 THEN RETURN
7910 PRINT AT RND*21,RND*31; INK 6;"*"
7920 RETURN

8000 DIM w(8): DIM v(8)
8010 DIM s$(8,8): DIM r$(8,8)
8020 DATA 0,1,1,1,1,0,1,-1,0,-1,-1,-1,
     -1,0,-1,1
8030 FOR i=1 TO 8
8040 READ w(i),v(i)
8050 NEXT i
```

```
8060 LET k=1
8070 LET x=20
8080 LET y=10
8090 LET v=v(k)
8100 LET w=w(k)
8110 LET s$(1)="00011000"
8120 LET s$(2)="00111100"
8130 LET s$(3)="01111110"
8140 LET s$(4)="11111111"
8150 LET s$(5)="00111100"
8160 LET s$(6)="00111100"
8170 LET s$(7)="00111100"
8180 LET s$(8)="00111100"
8190 LET r$(1)="11111000"
8200 LET r$(2)="11110000"
8210 LET r$(3)="11111000"
8220 LET r$(4)="11111100"
8225 LET r$(5)="10111110"
8230 LET r$(6)="00011111"
8240 LET r$(7)="00001110"
8250 LET r$(8)="00000100"
8260 FOR i=1 TO 8
8270 LET a=0: LET b=0: LET c=0
8275 LET d=0: LET e=0: LET f=0
8280 FOR j=1 TO 8
8290 LET a=a*2+VAL (s$(i,j))
8300 LET b=b*2+VAL (s$(9-j,i))
8310 LET c=c*2+VAL (s$(j,i))
8320 LET d=d*2+VAL (r$(i,j))
8330 LET e=e*2+VAL (r$(9-j,i))
8340 LET f=f*2+VAL (r$(9-i,9-j))
8345 GOSUB 7900
8380 NEXT j
8390 POKE USR "a"+i-1,a
8400 POKE USR "b"+8-i,a
8410 POKE USR "c"+i-1,b
8420 POKE USR "d"+i-1,c
8430 POKE USR "e"+i-1,d
8440 POKE USR "f"+8-i,d
8450 POKE USR "g"+i-1,e
8460 POKE USR "h"+i-1,f
8490 NEXT i
```

```
8500 LET m$="[c][h][b][f][d][e][a][g]"
8510 LET a=10
8520 LET b=10
8530 LET z=INT (RND*8)+1

8600 POKE USR "i"+0,BIN 00100000
8610 POKE USR "i"+1,BIN 01000010
8620 POKE USR "i"+2,BIN 00100101
8630 POKE USR "i"+3,BIN 00011000
8640 POKE USR "i"+4,BIN 00011000
8650 POKE USR "i"+5,BIN 10100100
8660 POKE USR "i"+6,BIN 01000010
8670 POKE USR "i"+7,BIN 00000100
8680 POKE USR "j"+0,BIN 00001110
8690 POKE USR "j"+1,BIN 10001000
8700 POKE USR "j"+2,BIN 10001000
8710 POKE USR "j"+3,BIN 11111000
8720 POKE USR "j"+4,BIN 00011111
8730 POKE USR "j"+5,BIN 00001001
8740 POKE USR "j"+6,BIN 00001001
8750 POKE USR "j"+7,BIN 00111000
8760 LET w$="[i][j]"
8770 LET r=0
8780 INK 7
8790 LET nb1=0
8800 LET nb2=0

8810 POKE 23674,0
8820 POKE 23673,0
8830 POKE 23672,0
8840 LET h=0
8900 RETURN

9000 IF h<>1 THEN GOTO 9500
9010 PRINT AT 21,0;"You did it !!!"
9020 GOTO 9600
9500 PRINT AT 21,0;"Your time is up"
9600 INPUT "Another game y/n ?";a$
9610 IF a$(1)="y" THEN RUN
9620 PAPER 7
9630 INK 0
9640 BORDER 7
9650 CLS
```

9
Spectrum Guideline



This is a game of skill in which you have to guide a square along an irregular wavy wire. Whereas if 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 Spectrum 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 of 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 there is a short delay and then it automatically starts to move right. Press the up and down arrow keys to change direction. Keeping your finger down on an arrow key will keep the square moving in the same direction.

Typing tips

The symbol <> is one that crops up very often in the programs in this book. Remember that it is entered as a single key press ('W' with the SYMBOL SHIFT held down.

Subroutine structure

20	Main play loop
105	End of game
1000	Initialises variables and prints title frame
2000	Plots wire
5000	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 2000 is responsible for plotting the line for the wire and the square is constructed by a collection of DRAW statements at the beginning of subroutine 5000. The POINT function is used in this subroutine to test whether the square ring is actually on target around the wire.

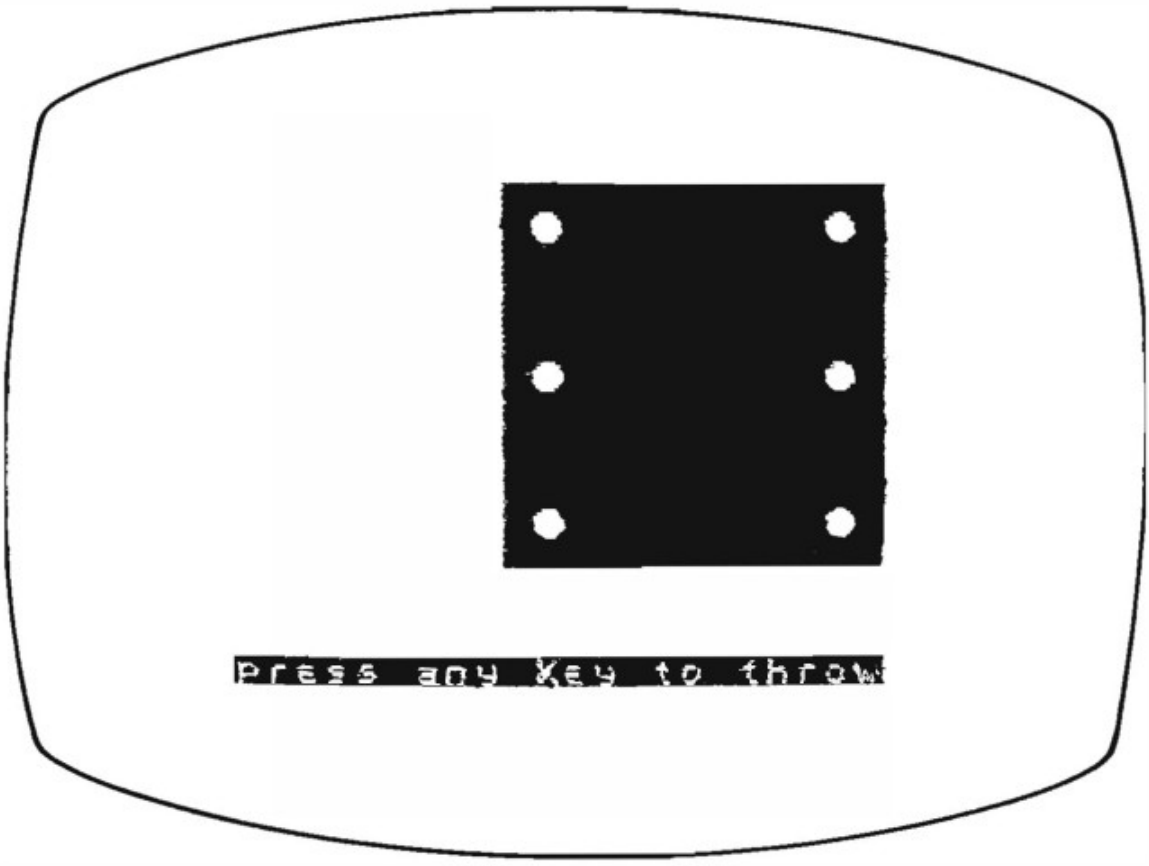
Program

```
 10 REM Spectrum guideline

 20 GOSUB 1000
 30 GOSUB 2000
 40 PAUSE 100
 50 PRINT AT 1,0; BRIGHT 1; "GO"
 60 BEEP .1,5
 70 LET y=100: LET x=10
 80 LET b=y: LET r=12-df
 90 LET r2=r/2: LET v=2
100 GOSUB 5000
```

```
 105 OVER 0
 110 PRINT AT 1,0;"You were on target ";
     AT 2,10;INT(hit/(hit+miss)*10000)/100;
     "% of the time"
 120 INPUT "Another game ";a$
 130 IF a$="y" THEN RUN
 140 IF a$<>"n" THEN GO TO 120
 150 BORDER 7
 160 CLS
 170 STOP

1000 LET x=10
1010 LET y=100
1020 LET d=30
1030 LET p=0
1040 INK 0
1050 PAPER 7
1060 BORDER 4
1070 CLS
1080 PRINT AT 2,7;"G U I D E L I N E"
1090 PRINT AT 5,0;"You must guide a ring
     along the"
1100 PRINT "wavy 'wire' using the up and"
1110 PRINT "down arrow keys"
1120 PRINT
1130 PRINT "You will be marked on how"
1140 PRINT "accurate you are"
1150 INPUT "Select the difficulty level"'
     " - 1 (easy) to 5 (difficult) ?";df
1160 IF df<1 OR df>5 THEN GO TO 1150
1170 CLS
1180 OVER 1
1190 RETURN

2000 PLOT x,y
2010 FOR i=1 TO 10
2020 LET r=d-INT (RND*2*d)
2030 LET y=y+r
2040 IF y>160 THEN LET y=y-r: LET r=-r
2050 IF y<10 THEN LET y=y+r: LET r=-r
2060 DRAW 20,r
2070 NEXT i
2080 LET hit=0
2090 LET miss=0
2100 RETURN
```

```
5000 PLOT x-r2,y-r2
5010 DRAW 0,r
5020 DRAW r,0
5030 DRAW 0,-r
5040 DRAW -r,0
5100 LET a$=INKEY$
5110 IF a$="6" AND y-r2>2*v
     THEN LET b=b-2*v
5120 IF a$="7" AND y+r<175-2*v
     THEN LET b=b+2*v
5510 PLOT x-r2,y-r2
5520 DRAW 0,r
5530 DRAW r,0
5540 DRAW 0,-r
5550 DRAW -r,0
5560 LET x=x+v
5565 IF x>210 THEN RETURN
5570 LET y=b
5580 FOR i=-r+1 TO r-1
5590 IF POINT (x,y+i) THEN LET hit=hit+1:
     PAUSE 2: GOTO 5000
5600 NEXT i
5610 LET miss=miss+1
5700 BEEP .01,3
5710 GOTO 5000
```

10
Spectrum 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 could not 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 Spectrum to absorb you, but think again. They are actually very enjoyable games for lots of players especially if you don't have to spend so 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 Spectrum join in the game and take over from the dice.

Of course, your Spectrum 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 – and you'll be impressed by the way it even sounds like a dice rolling over a wooden surface and actually slows down before it comes to a final halt.

How to use the program

Using this program is simplicity itself. Type RUN and, when your Spectrum 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. When you've finished with the program press the BREAK key to stop it running.

Typing tips

In the subroutine that starts at line 1000, you'll notice that "[a]" appears a number of times. The square brackets around the letter 'a' indicate that it is a graphics character. Do not type the brackets, instead get into graphics mode, so that you see the G cursor, (you do this by pressing the 9 key with the CAPS SHIFT held down) and then type an 'a'. Remember to type the double quotation marks around it.

In line 2010, notice that you have to type 13 spaces between the quotation marks. Count these carefully as your dice display will not work with the wrong number.

Subroutine structure

10	Main play loop
1000	Prints and unprints dots
2000	Draws yellow square for dice
5000	Defines dot character [a] and sets colours
6000	Emits click 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 110, 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 1000). The program goes to one of the six line numbers 1100, 1200, 1300, 1400, 1500, 1600, according to the value of 'r'. At 1100 one dot is printed, at 1200 two dots are printed and so on.

The other use of the random number generator is to give the Spectrum Dice realistic suspense. When a human throws a dice it will turn just a few times or quite a number of times and before it actually stops it will slow down. Your Spectrum Dice copies both the features by incorporating lines 60 to 80. Another random number, 't', with a value between 6 and 16 is selected. This governs the number of turns the dice makes and each time it rolls over the pause before the dots reappear lengthens.

This program makes use of both colour and sound. Colour is largely looked after in lines 5080 to 5120. However, the colour that the dots will be printed in, red, is set in line 90. The dots are actually made to disappear by overprinting them in the dice's background colour (yellow). This colour is determined in line 2010. You may have to listen fairly carefully to detect the clicking sound the dice makes as it rolls over. This is because the noise comes from the speaker on your Spectrum which is hidden on its underside. Lines 6000 to 6020 are responsible for this very dice-like sound.

Scope for improvement

This program could be incorporated into many self-contained games. You could devise a gambling game – see for example Spectrum Ledger – where bets were placed on which face of the dice would show.

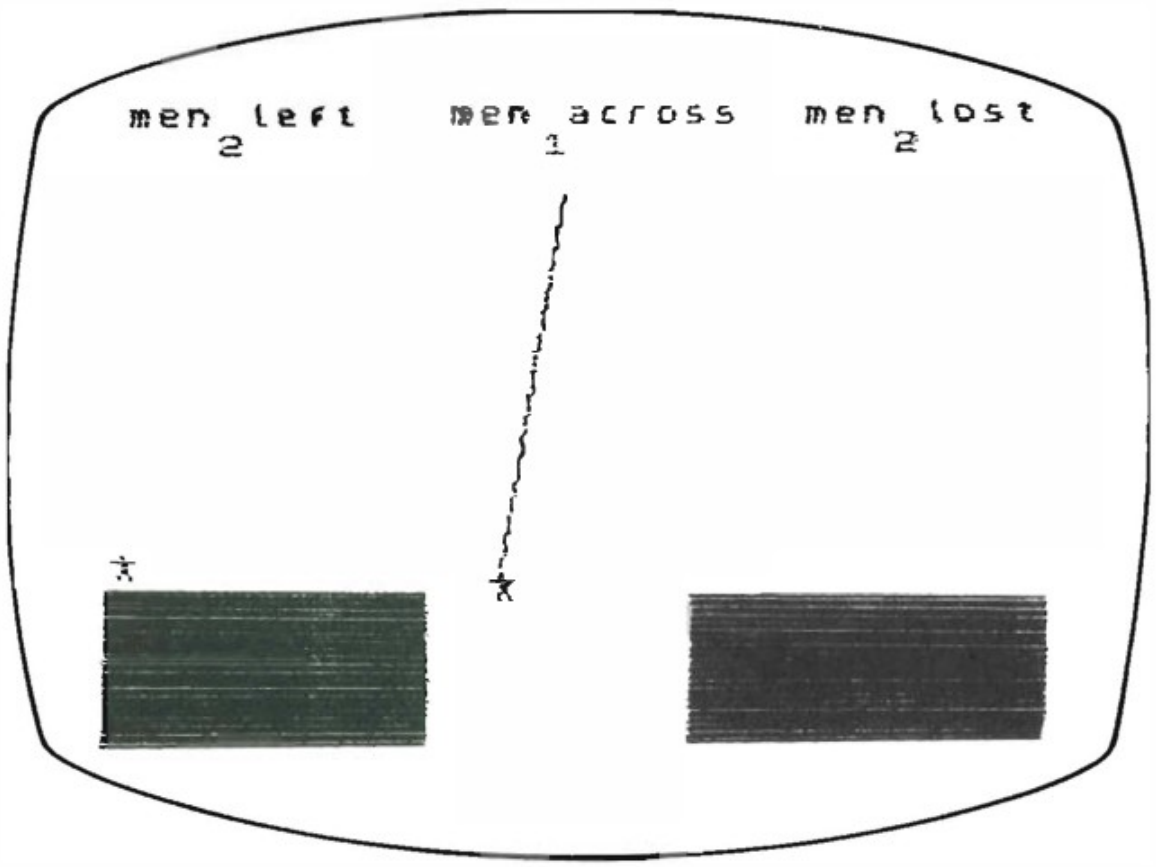
Program

```
5 REM Spectrum dice
```

```
  10 GOSUB 5000
  20 GOSUB 2000
  30 INPUT " "
  40 PRINT AT 20,0; PAPER 0; INK 6;
     "press any key to throw"
  50 PAUSE 0
  60 LET t=RND*10+5
  70 FOR i=1 TO t
  80 PAUSE 1+i
  90 INK 6
 100 GOSUB 1000
 110 LET r=INT (RND*6)+1
 120 INK 2
 130 GOSUB 1000
 140 NEXT i
 150 GOTO 50

1000 GOSUB 6000
1010 GOTO 1000+r*100
1100 PRINT AT 10,15;"[a]"
1110 RETURN
1200 PRINT AT 5,10;"[a]"
1210 PRINT AT 15,20;"[a]"
1220 RETURN
1300 GOSUB 1100
1310 GOTO 1200
1400 PRINT AT 5,20;"[a]"
1410 PRINT AT 15,10;"[a]"
1420 GOTO 1200
1500 GOSUB 1400
1510 GOTO 1100
1600 PRINT AT 10,10;"[a]"
1610 PRINT AT 10,20;"[a]"
1620 GOTO 1400

2000 FOR i=1 TO 13
2010 PRINT AT 3+i,9; PAPER 6;"
        "; REM 13 spaces
2020 NEXT i
2030 LET r=1
2040 GOTO 1000
```

```
5000 POKE USR "a"+0,BIN 00111100
5010 POKE USR "a"+1,BIN 01111110
5020 POKE USR "a"+2,BIN 11111111
5030 POKE USR "a"+3,BIN 11111111
5040 POKE USR "a"+4,BIN 11111111
5050 POKE USR "a"+5,BIN 11111111
5060 POKE USR "a"+6,BIN 01111110
5070 POKE USR "a"+7,BIN 00111100
5080 INK 2
5090 PAPER 0
5100 CLS
5110 PAPER 6
5120 BORDER 0
5130 RETURN

6000 OUT 254,16
6010 OUT 254,0
6020 RETURN
```

11
Across the Ravine



This is a colour graphics game with a difference – not least because it features a combination of low resolution and high resolution graphics. 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.

Typing tips

As this program uses high resolution graphics you'll find it uses some commands – PLOT, DRAW and OVER – and some functions – SIN, COS and PI – you may be unfamiliar with. Remember that all these are entered by a single key press.

Subroutine structure

20	Initialises arrays
40	Main play loop
1500	Swings rope
2000	Calculates positions of rope
3000	Print ravine
4000	Plots man on end of rope
5000	Man jump routine
6000	Get next man ready
7000	Man fall in water routine
7500	Set up game
9000	End of game

Programming details

There are several interesting features of this game. The first point to note is that the co-ordinates of the positions of the swinging rope are first calculated by subroutine 2000 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. The second interesting technique is in subroutine 4000 which, instead of using a low resolution user-defined graphic, actually plots the man figure using high resolution graphics. This means that he can appear anywhere on the screen and therefore move smoothly rather than just be printed at set positions which would result in jerky movement.

Program

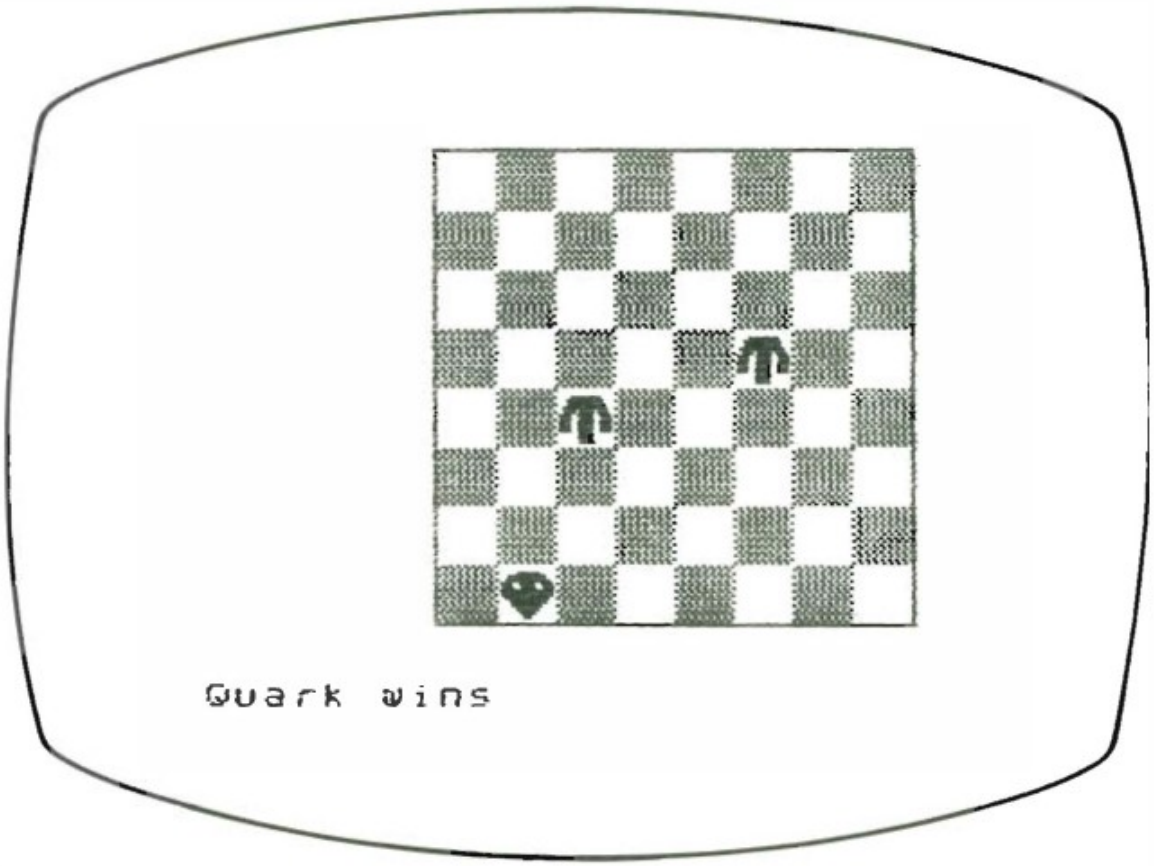
```
  10 REM Across the Ravine
  20 DIM x(16)
  30 DIM y(16)

  40 GOSUB 3000
  50 GOSUB 2000
  60 GOSUB 7500
  70 GOSUB 6000
  80 GOSUB 1500
  90 IF men=0 THEN GOTO 9000
 100 GOTO 80

1500 FOR t=1+r TO n-r
1510 PLOT 127,150
1520 DRAW x(t),y(t)
1525 LET s=1: GOSUB 4000
1530 PLOT 127,150
1540 DRAW OVER 1;x(t),y(t)
1550 IF j=1 THEN LET s=2: GOSUB 4000
1560 NEXT t
1570 IF c=1 THEN GOTO 1680
1600 FOR t=n-r TO 1+r STEP -1
1610 PLOT 127,150
1620 DRAW x(t),y(t)
1625 LET s=3: GOSUB 4000
1630 PLOT 127,150
1640 DRAW OVER 1;x(t),y(t)
1650 IF j=1 THEN LET s=4: GOSUB 4000
1660 NEXT t
1670 RETURN
1680 LET across=across+1
1690 LET dx=(across-1)*10+5
1700 LET dy=50
1710 GOSUB 4020
1720 LET c=0
1730 LET men=men-1
1735 IF men=0 THEN GOTO 6050
1740 GOSUB 6000
1750 RETURN
```

```
2000 LET n=0
2010 FOR t=-PI/6 TO PI/6 STEP.1
2020 LET n=n+1
2030 LET x(n)=-105*SIN t
2040 LET y(n)=-105*COS t
2050 NEXT t
2060 RETURN

3000 PAPER 7
3010 INK 0
3020 CLS
3040 FOR i=17 TO 21
3050 FOR j=0 TO 31
3060 IF j>10 AND j<20 THEN GOTO 3090
3070 PRINT AT i,j; INK 4; "[^8]"
3080 GOTO 3110
3090 IF i<19 THEN GOTO 3110
3100 PRINT AT i,j; PAPER 1;" "
3110 NEXT j
3120 NEXT i
3130 LET c=0
3140 LET j=0
3150 RETURN

4000 IF c=0 THEN GOTO 5000
4010 LET dx=127+x(t): LET dy=150+y(t)-2
4020 PLOT OVER 1;dx,dy
4030 PLOT OVER 1;dx,dy-1
4040 PLOT OVER 1;dx-3,dy-2
4050 DRAW OVER 1;+6,0
4060 PLOT OVER 1;dx,dy-3
4070 DRAW OVER 1;+2,-4
4080 PLOT OVER 1;dx+1,dy-3
4090 DRAW OVER 1;-2,-4
4100 RETURN

5000 IF INKEY$="" AND j=0 THEN PAUSE 15:
      RETURN
5010 LET j=1
5020 LET dy=my
5030 LET dx=mx
5040 GOSUB 4020
```

```
5050 LET mx=mx-10
5060 LET dy=my
5070 LET dx=mx
5080 GOSUB 4020
5090 IF ABS (mx-127-x(t))<10 AND s=2 THEN
     LET c=1: GOTO 4020
5200 IF mx<160 THEN GOTO 7000
5210 RETURN

6000 LET my=50
6010 LET mx=240
6020 LET dy=my
6030 LET dx=mx
6040 LET j=0
6050 GOSUB 4020
6060 PRINT AT 0,1;
     "men left   men across  men lost"
6070 PRINT AT 1,4;men;TAB 15;across;
     TAB 27;lost
6080 RETURN

7000 GOSUB 4020
7010 LET my=my-2
7020 LET dy=my
7030 GOSUB 4020
7040 BEEP .01,my/2
7050 GOSUB 4020
7060 IF my>30 THEN GOTO 7010
7070 BEEP 1,-10
7080 LET lost=lost+1
7090 LET c=0
7100 LET j=0
7110 LET men=men-1
7120 IF men=0 THEN GOTO 6060
7130 GOSUB 6000
7140 RETURN
```

```
7500 LET men=5
7510 LET lost=0
7520 LET across=0
7530 LET j=0
7540 LET c=0
7550 LET r=INT (RND*4)
7560 PRINT AT 15,0;"                ";
     REM 10 spaces
7570 PRINT AT 16,0;"                ";
     REM 10 spaces
7580 RETURN

9000 INPUT "You lost ";(lost)
     '"Another game ?";a$
9010 IF a$="y" THEN GOTO 60
9020 INK 0
9030 PAPER 7
9040 CLS
```

12
Capture the Quark



What on earth is a quark? Well you may 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 pieces (determined at random) 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 pieces (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 pieces is flashing. 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 it's your turn again. The Spectrum 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 "yes" or "no" to the question "resign?"

Typing tips

The "IF" statement in line 1010 looks as though it's wrong as there are no relational signs. It is however correct – the values stored in the array are either '1' or '0' and the Spectrum treats these as equivalent to 'true' or 'false'. Notice that there should be two spaces between each of the double quotes in line 2000 and those in line 5500. In lines 7020 and 7060 there should also be two spaces – in the first case before two graphics 'a's and in the second after two graphics 'a's. Notice the use of 'O' in subroutines 1000 and 7000.

Subroutine structure

20	Initialises variables
50	Main play loop
1000	Quark move logic
5000	Moves hounds and validates their moves
6000	Selects which hound to move
6500	Starts position flashing
6600	Stops position flashing
7000	Draws board
7500	Draws initial positions of hounds and quark and initialises board
8000	Defines graphics characters
9000	End of game

Programming details

Although this program runs in black and white the board is drawn up in white and hatched shading which gives an extra tone. This is defined at the beginning of subroutine 8000. Notice the way that subroutines 6500 and 6600 cause characters already on the screen to start and stop flashing. This is done by manipulating the attributes area of memory.

Program

```
 10 REM Capture the quark

 20 DIM d(10,10)
 30 DIM x(4): DIM y(4)
 40 DIM a(4): DIM b(4)

 50 GOSUB 8000
 60 GOSUB 7000
 70 LET h=INT (RND*3)+2: GOSUB 7500
 80 GOSUB 6000
 90 GOSUB 1000
100 GOTO 80
```

```
1000 PRINT AT 21,0; "quarks move"
1010 IF d(qi+2,qj+2) AND d(qi+2,qj)
     AND d(qi,qj+2) AND d(qi,qj)
     THEN GOTO 9000
1020 LET m=1
1030 GOSUB 3000
1035 IF qi+n<1 OR qi+n>8 THEN GOTO 1100
1040 IF d(qi+n+1,qj+m+1) THEN GOSUB 2500
1050 IF d(qi+n+2,qj+m+2)=1
     AND d(qi+n,qj+m+2)=1
     AND qj<7 AND qj>1 THEN LET m=-m
1100 IF d(qi+n+1,qj+m+1) THEN GOSUB 2500
1110 IF oi=qi AND oj=qj THEN LET m=-m:
     LET n=SGN (RND-.5): GOTO 1035
1120 LET oi=qi: LET oj=qj
2000 PRINT AT qy,qx;"  ";AT qy+1,qx;"  "
2010 LET qx=qx+n*2
2020 LET qy=qy+m*2
2030 PRINT AT qy,qx;"[b][c]";AT qy+1,qx;
     "[d][f]"
2040 LET d(qi+n+1,qj+m+1)=2
2050 LET d(qi+1,qj+1)=0
2060 LET qi=qi+n
2070 LET qj=qj+m
2080 IF qj=8 THEN GOTO 9500
2090 RETURN
2500 LET n=-n
2510 IF d(qi+n+1,qj+m+1)<>1 THEN RETURN
2520 LET m=-m
2525 IF qj<4 THEN LET n=1
2530 IF d(qi+n+1,qj+m+1)<>1 THEN RETURN
2540 LET n=-n
2550 RETURN
3000 LET n=(qi<5)-(qi>=5)
3005 LET r=RND
3010 IF qj>6 THEN RETURN
3020 IF r>.5 AND qi>3 THEN GOTO 3050
3025 IF qi>7 THEN GOTO 3035
3030 IF (d(qi+3,qj+3)=0 OR d(qi+2,qj+3)=0)
     AND d(qi+2,qj+2)=0 THEN LET n=-1: RETURN
3035 IF qi<4 OR r>.5 THEN RETURN
3050 IF (d(qi-3,qj+3)=0 OR d(qi-2,qj+3)=0)
     AND d(qi-2,qj+2)=0 THEN LET n=1: RETURN
3060 IF r>.5 THEN GOTO 3025
3070 RETURN
```

```
5000 LET a(hm)=a(hm)+m
5010 LET b(hm)=b(hm)-1
5020 IF d(a(hm)+1,b(hm)+1)<>0 THEN GOTO 5100
5050 LET d(a(hm)-m+1,b(hm)+2)=0
5060 LET d(a(hm)+1,b(hm)+1)=1
5070 GOTO 5500
5100 LET a(hm)=a(hm)-m
5110 LET b(hm)=b(hm)+1
5120 BEEP 1,-10
5130 GOTO 6000
5500 PRINT AT y(hm),x(hm); FLASH 0;"  ";
     AT y(hm)+1,x(hm);"  "
5510 LET y(hm)=y(hm)-2
5520 LET x(hm)=x(hm)+m*2
5530 PRINT AT y(hm),x(hm); FLASH 1;
     "[g][i]"; AT y(hm)+1,x(hm);"[h][j]"
5540 RETURN

6000 PRINT AT 21,0;"your move":
     REM 3 spaces
6005 BEEP .1,6
6010 LET m$=INKEY$
6020 IF m$="" THEN GOTO 6010
6025 IF m$="r" THEN GOTO 6100
6030 IF m$="5" THEN LET m=-1: GOTO 5000
6040 IF m$="8" THEN LET m=+1: GOTO 5000
6050 GOSUB 6200
6060 GOTO 6000
6100 INPUT "resign y/n";a$
6110 IF a$="y" THEN GOTO 9500
6120 GOTO 6000
6200 GOSUB 6600
6210 LET hm=hm+1
6220 IF hm>h THEN LET hm=1
6230 GOSUB 6500
6240 RETURN

6500 LET f=22528+x(hm)+y(hm)*32
6510 POKE f,PEEK f+128
6520 POKE f+1,PEEK (f+1)+128
6530 POKE f+32,PEEK (f+32)+128
6540 POKE f+33,PEEK (f+33)+128
6550 RETURN
```

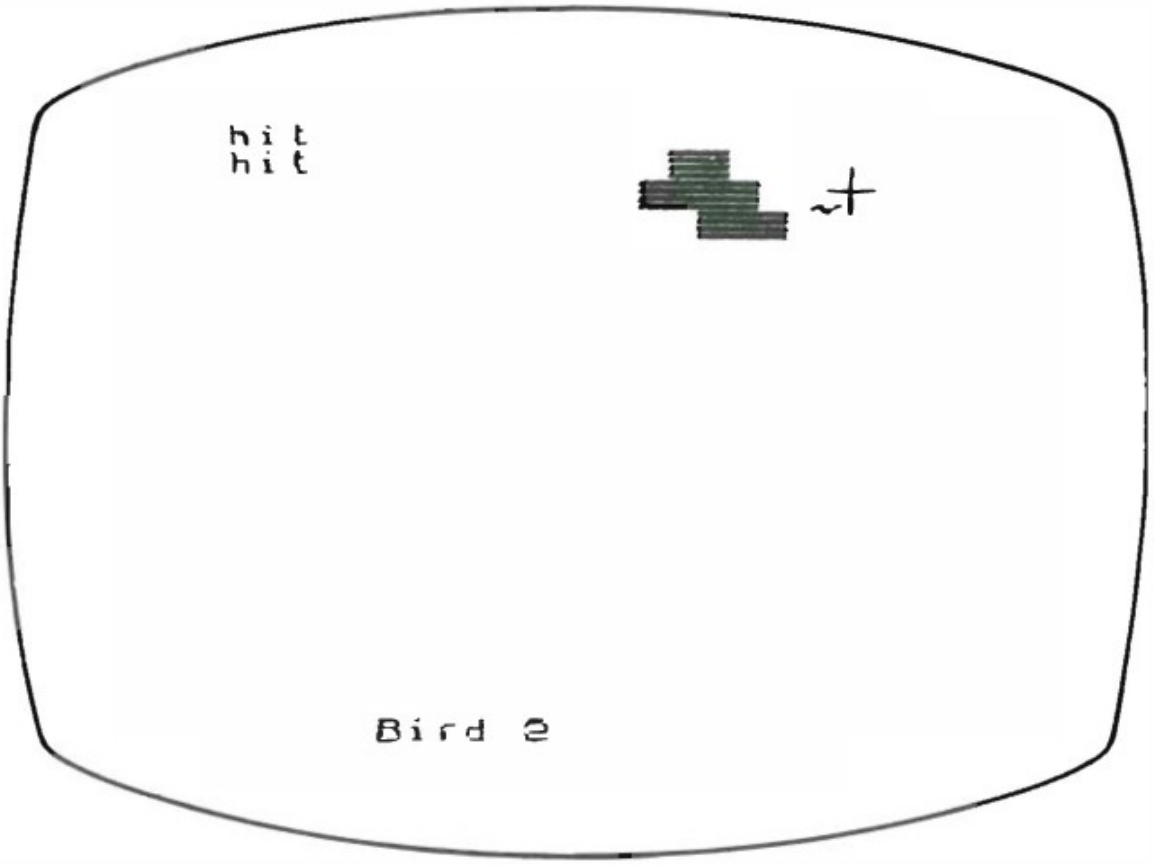
```
6600 LET f=22528+x(hm)+y(hm)*32
6610 POKE f,PEEK f-128
6620 POKE f+1,PEEK (f+1)-128
6630 POKE f+32,PEEK (f+32)-128
6640 POKE f+33,PEEK (f+33)-128
6650 RETURN

7000 PRINT AT 3,8;
7005 FOR i=1 TO 4
7010 FOR j=1 TO 8
7020 PRINT "  [a][a]";
7030 IF j/4=INT (j/4) THEN PRINT:
     PRINT TAB 8;
7040 NEXT j
7050 FOR j=1 TO 8
7060 PRINT "[a][a]  ";
7070 IF j/4=INT (j/4) THEN PRINT:
     PRINT TAB 8;
7080 NEXT j
7090 NEXT i
7100 PLOT 63,23
7110 DRAW 129,0
7120 DRAW 0,129
7130 DRAW-129,0
7140 DRAW 0,-129
7150 RETURN
```

```
7500 FOR i=1 TO h
7510 LET x=i*4+6
7520 PRINT AT 17,x;"[g][i]";
     AT 16,x; "[h][j]"
7530 LET d(i*2+1,9)=1
7540 LET x(i)=x
7550 LET y(i)=17
7560 LET hm=1
7570 LET a(i)=i*2
7580 LET b(i)=8
7590 NEXT i
7600 GOSUB 6500
7610 LET qi=5
7620 LET qj=1
7630 LET qx=qi*2+6
7640 LET qy=3
7650 PRINT AT qy,qx;"[b][c]";
     AT qy+1,qx;"[d][f]"
7660 FOR i=1 TO 10
7670 LET d(i,1)=1
7680 LET d(1,i)=1
7690 LET d(10,i)=1
7700 LET d(i,10)=1
7710 NEXT i
7720 LET d(qi+1,qj+1)=2
7730 LET oi=0
7740 LET oj=0
7750 RETURN
```

```
8000 FOR i=0 TO 7 STEP 2
8010 POKE USR "a"+i,BIN 01010101
8020 POKE USR "a"+i+1,BIN 10101010
8030 NEXT i
8050 POKE USR "b"+0,BIN 00000000
8060 POKE USR "b"+1,BIN 00000000
8070 POKE USR "b"+2,BIN 00000111
8080 POKE USR "b"+3,BIN 00011111
8090 POKE USR "b"+4,BIN 00111111
8100 POKE USR "b"+5,BIN 01110011
8110 POKE USR "b"+6,BIN 01110011
8120 POKE USR "b"+7,BIN 01111111
8200 POKE USR "c"+0,BIN 00000000
8210 POKE USR "c"+1,BIN 00000000
8220 POKE USR "c"+2,BIN 11100000
8230 POKE USR "c"+3,BIN 11111000
8240 POKE USR "c"+4,BIN 11111100
8250 POKE USR "c"+5,BIN 11001110
8260 POKE USR "c"+6,BIN 11001110
8270 POKE USR "c"+7,BIN 11111110
8300 POKE USR "d"+0,BIN 01111111
8310 POKE USR "d"+1,BIN 00111111
8320 POKE USR "d"+2,BIN 00011111
8330 POKE USR "d"+3,BIN 00001111
8340 POKE USR "d"+4,BIN 00000111
8350 POKE USR "d"+5,BIN 00000011
8360 POKE USR "d"+6,BIN 00000001
8370 POKE USR "d"+7,BIN 00000000
8400 POKE USR "f"+0,BIN 11111110
8410 POKE USR "f"+1,BIN 11111100
8420 POKE USR "f"+2,BIN 11111000
8430 POKE USR "f"+3,BIN 11110000
8440 POKE USR "f"+4,BIN 11100000
8450 POKE USR "f"+5,BIN 11000000
8460 POKE USR "f"+6,BIN 10000000
8470 POKE USR "f"+7,BIN 00000000
8500 POKE USR "g"+0,BIN 00000000
8510 POKE USR "g"+1,BIN 00000000
8520 POKE USR "g"+2,BIN 00001111
8530 POKE USR "g"+3,BIN 00011111
8540 POKE USR "g"+4,BIN 00011111
8550 POKE USR "g"+5,BIN 00111111
8560 POKE USR "g"+6,BIN 00111011
8570 POKE USR "g"+7,BIN 01110011
```

```
8600 POKE USR "h"+0,BIN 01110011
8610 POKE USR "h"+1,BIN 01110011
8620 POKE USR "h"+2,BIN 01110011
8630 POKE USR "h"+3,BIN 01110011
8640 POKE USR "h"+4,BIN 00000011
8650 POKE USR "h"+5,BIN 00000011
8660 POKE USR "h"+6,BIN 00000011
8670 POKE USR "h"+7,BIN 00000000
8700 POKE USR "i"+0,BIN 00000000
8710 POKE USR "i"+1,BIN 00000000
8720 POKE USR "i"+2,BIN 11110000
8730 POKE USR "i"+3,BIN 11111000
8740 POKE USR "i"+4,BIN 11111000
8750 POKE USR "i"+5,BIN 11111100
8760 POKE USR "i"+6,BIN 11011100
8780 POKE USR "i"+7,BIN 11001110
8800 POKE USR "j"+0,BIN 11001110
8810 POKE USR "j"+1,BIN 11001110
8820 POKE USR "j"+2,BIN 11001110
8830 POKE USR "j"+3,BIN 11001110
8840 POKE USR "j"+4,BIN 11000000
8850 POKE USR "j"+5,BIN 11000000
8860 POKE USR "j"+6,BIN 11000000
8870 POKE USR "j"+7,BIN 00000000
8900 INK 0
8910 PAPER 7
8920 CLS
8930 BORDER 7
8940 RETURN
9000 PRINT AT 21,0;"you win    ":
      REM 4 spaces
9010 GOTO 9900
9500 PRINT AT 21,0;"Quark wins "
9900 INPUT "Another game";a$
9910 IF a$="y" THEN RUN
9920 CLS
9930 PRINT "Bye"
```

13
Spectrum Shoot



All the target practice you could ever want and the magic bird lives to fly on – however many times you score a direct hit. 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 it's not that simple. For one thing, every time you fire your rifle 'kicks' to one side or the other so you have to realign your sight for another shot. Another problem is that the bird (and your sight too) disappear behind the cloud when they reach it. You can continue to fire, but you won't be able to see what is happening. 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 distinctive sound every time you fire your rifle and another sound when you hit the bird. The word 'hit' also appears to confirm a good shot.

How to play

To hit the target you have to line up the cross point of your sight 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.

Typing tips

Line 45 should end AT 0,5 with nothing to follow. The effect of this is to move the position at which the next character will be printed to the top of the screen. There are two spaces within the double quotes marks in line 3040, four in line 3050 and three in 3060.

Subroutine structure

20	Main play loop
1000	Plots cross
2000	Moves sight
3000	Plots cloud and set up game
4000	Plots bird
5000	Shoots, tests for hit and recoils sight
8000	Calculates path of bird
9000	End of game

Programming details

This is a high-resolution dynamic graphics game. The rifle sight is plotted as a high-resolution cross, using PLOT and DRAW, in subroutine 1000. The bird is also plotted as a collection of high resolution points, in subroutine 4000. The use of high-resolution graphics 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 subroutine 5000 to discover if the target has been hit. Notice the extensive use of OVER 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' 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.

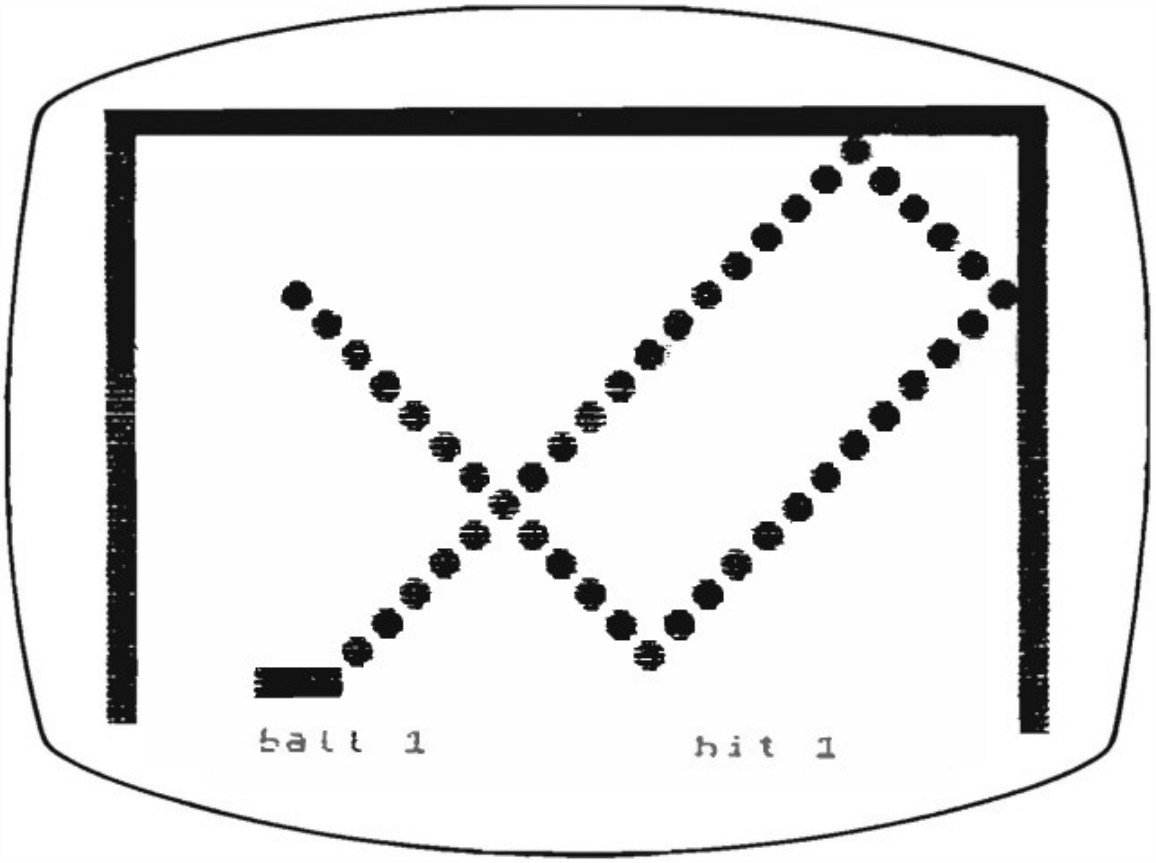
Program

```
   10 REM Spectrum shoot

   20 GOSUB 8000
   30 FOR g=1 TO 5
   40 GOSUB 3000
   45 PRINT AT 21,5; "Bird ";g;AT 0,5
   50 GOSUB 2000
   60 NEXT g
   70 GOTO 9000
   90 STOP

 1000 PLOT x-6,y
 1010 DRAW 12,0
 1020 PLOT x,y-6
 1030 DRAW 0,12
 1060 RETURN

 2000 LET a$=INKEY$
 2005 GOSUB 1000
 2006 LET j=b(i): GOSUB 4000
 2010 IF a$="5" AND x>10 THEN  LET x=x-2
 2020 IF a$="6" AND y>10 THEN  LET y=y-2
 2030 IF a$="7" AND y<160 THEN  LET y=y+2
 2040 IF a$="8" AND x<240 THEN LET x=x+2
 2050 LET i=i+1: LET j=b(i): GOSUB 4000
 2060 IF a$="f" THEN GOSUB 5000
 2070 GOSUB 1000
 2080 IF end=1 THEN LET end=0: RETURN
 2100 GOTO 2000
```

```
3000 CLS
3020 LET j=INT (RND*3)+1
3030 LET r=INT (RND*10)
3040 PRINT AT j,r+9; INK 7;
     PAPER 7;"  ": REM 2 spaces
3050 PRINT AT j+1,r+8; INK 7;
     PAPER 7;"    ": REM 4 spaces
3060 PRINT AT j+2,r+10; INK 7;
     PAPER 7;"   ": REM 3 spaces
3080 LET end=0
3090 LET x=INT (RND*60)+40
3100 LET y=75
3110 PAPER 5
3120 INK 0
3140 LET i=3
3150 GOSUB 1000
3160 LET j=b(i): GOSUB 4000
3170 RETURN

4000 INK 8: PAPER 8
4005 PLOT i-3,j+2
4010 PLOT i-2,j+2
4020 PLOT i-1,j+1
4030 PLOT i,j
4040 PLOT i+1,j+1
4050 PLOT i+2,j+2
4060 PLOT i+3,j+2
4070 PLOT i-4,j+1
4080 IF i>250 THEN LET end=1
4090 RETURN

5000 BEEP .1,-5
5010 IF POINT (x,y)<>1 THEN
     LET x=x+10-INT (RND*20): RETURN
5020 LET x=x+10-INT (RND*20)
5030 BEEP .2,-5
5040 PRINT "hit"
5050 LET h(g)=h(g)+1
5060 RETURN
```

```
8000 LET hit=0
8010 CLS
8020 PRINT AT 10,2;
     "S P E C T R U M   S H O O T"
8060 DIM b(254)
8070 FOR i=1 TO 254
8080 LET b(i)=150-(125-i)*(125-i)/200
8100 NEXT i
8120 OVER 1
8130 DIM h(5)
8140 RETURN

9000 CLS
9010 LET t=0
9020 FOR i=1 TO 5
9030 PRINT AT i+5,10;"Bird ";i;" hit ";
     h(i)
9040 LET t=t+h(i)
9050 NEXT i
9060 PRINT AT 12,10;"Total hits= ";t
9070 INPUT "Another game y/n ?";a$
9075 IF a$="y" THEN RUN
9080 PAPER 7
9090 INK 0
9100 OVER 0
```

14
Rainbow Squash



This is a colourful version of the popular computer squash game – on a colour TV you'll find yourself playing on a white, red and yellow court within a blue border. It 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 Spectrum 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 (5 and 8). 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 bat 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.

Typing tips

The graphics characters used for the bat are solid blocks, produced by pressing CAPS SHIFT and the 8 key while in graphics mode. Notice that in line 110 you type double quotes, followed by one space, followed by three of these solid blocks, followed by another space and finally double quotes. In lines 170 and 470 five blank spaces, between double quotes, are required to blank out the bat.

Subroutine structure

5	Sets colours and initialise variables
100	Main loop – prints bat and score line
200	Gets movements of bat
300	Moves balls and controls bouncing off walls
400	Controls bounce against bat and movement of bat up screen
500	Prints walls of court
800	Sets colours of court
930	Defines graphics character for ball
2000	End of game – prints final score, offers another game and re-runs or restores screen display

Programming details

There are some interesting uses of colour commands in this program. In lines 110 and 360 you'll find "PAPER 8". Colour 8 is *transparent* in the sense that it allows whatever colour is already present on the screen to show through. So by using "PAPER 8" in both the bat and ball printing routines, the background colour is

unaltered. The ball graphic defined at subroutine 930 is one that you may find useful in your own programs.

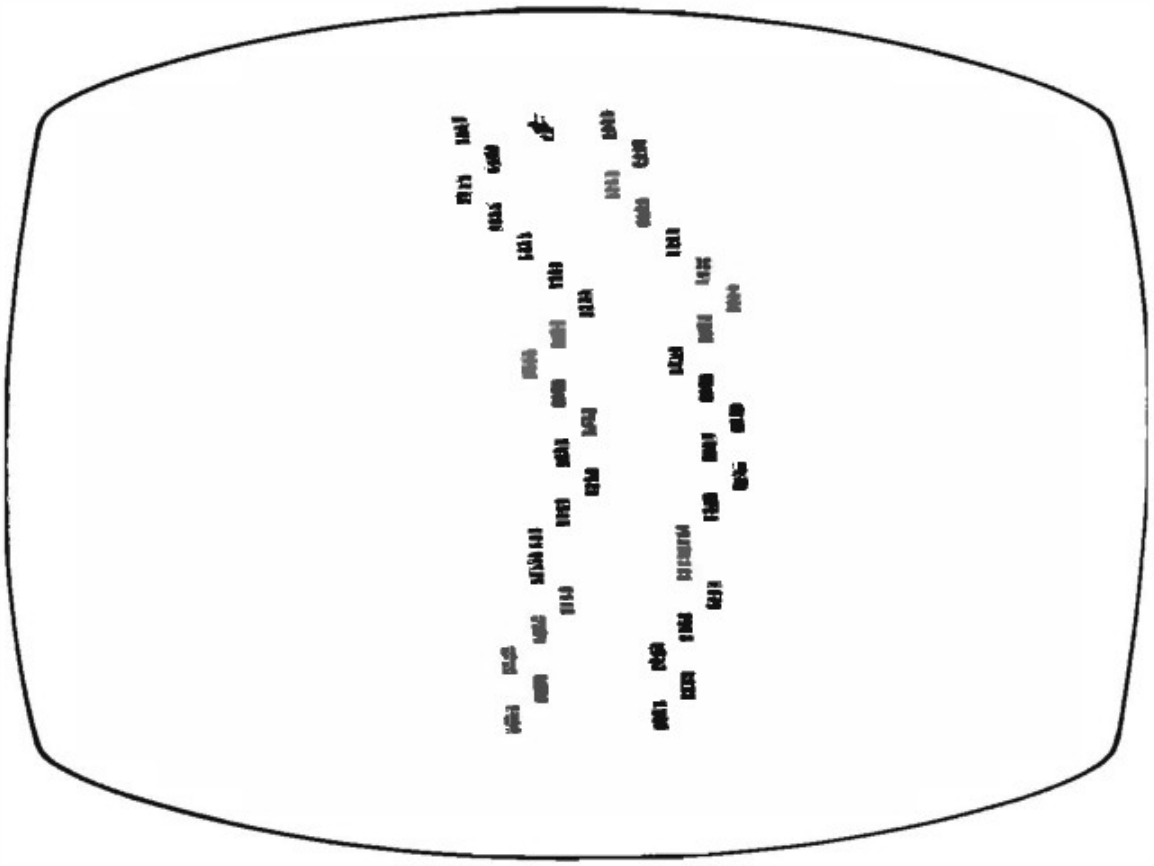
Program

```
  1 REM Rainbow squash

  5 BORDER 1: PAPER 7: INK 0
 10 LET h=0: LET ht=0: LET d=19:
    LET ball=0
 15 GOSUB 800: GOSUB 500
 20 LET ball=ball+1
 30 IF ball>10 THEN GOTO 2000
 40 LET a=5
 50 LET b=5
 60 LET v=1
 70 LET w=1
 80 LET x=10: LET y=d
 90 INPUT " "

100 PRINT AT 21,5; "ball ";ball;
110 PRINT AT y,x; PAPER 8; INK 0;
    "  [^8][^8][^8] "
120 PRINT AT 21,20; "hit ";ht
130 GOSUB 300
140 GOSUB 200
150 IF b+w<>y THEN LET y=d: GOTO 110
160 BEEP 1,0
170 PRINT AT y,x; PAPER 8; "     ":
    REM 5 spaces
180 PRINT AT b,a; PAPER 8; " "
190 IF d<19 THEN LET d=d+1
194 LET h=0
195 GOTO 20

200 LET a$=INKEY$
210 IF a$="5" AND x>0 THEN LET x=x-1
220 IF a$="8" AND x<27 THEN LET x=x+1
230 RETURN
```

```
300 PRINT AT b,a; PAPER 8; INK 8; " "
310 LET a=a+v
320 LET b=b+w
330 IF a=30 OR a=1 THEN LET v=-v:
    BEEP .1,15
340 IF b=1 THEN LET w=-w: BEEP .1,15
350 IF b+w=y THEN GOTO 400
360 PRINT AT b,a; PAPER 8; "[a]"
370 RETURN

400 LET r=a-x
410 IF r<1 OR r>3 THEN GOTO 360
420 LET w=-w
430 BEEP .1,15
440 LET h=h+1: LET ht=ht+1
450 IF h<>2 THEN GOTO 360
460 LET h=0: LET d=d-1
470 PRINT AT y,x; PAPER 8; "     ":
    REM 5 spaces
490 GOTO 300

500 FOR i=0 TO 31
510 PRINT AT 0,i; PAPER 0; " "
520 NEXT i
530 FOR i=0 TO 20
540 PRINT AT i,0; PAPER 0; " ";
    AT i,31; " "
550 NEXT i
560 RETURN
```

```
 800 PRINT AT 7,0;
 810 LET c=2: GOSUB 850
 820 LET c=6: GOSUB 850
 830 GOSUB 930
 840 RETURN
 850 FOR i=1 TO 7*32
 860 PRINT PAPER c;" ";
 870 NEXT i
 880 RETURN
 930 POKE USR "a"+0,BIN 00111100
 940 POKE USR "a"+1,BIN 01111110
 950 POKE USR "a"+2,BIN 11111111
 960 POKE USR "a"+3,BIN 11111111
 970 POKE USR "a"+4,BIN 11111111
 980 POKE USR "a"+5,BIN 11111111
 990 POKE USR "a"+6,BIN 01111110
1000 POKE USR "a"+7,BIN 00111100
1010 PAPER 7
1020 RETURN

2000 INK 0
2010 PAPER 7
2030 CLS
2040 PRINT AT 10,10; "You scored"
2050 PRINT AT 12,14;ht
2060 INPUT "Another game y/n ? ";a$
2070 IF a$(1)="y" THEN RUN
2080 CLS: BORDER 7
2090 PRINT "Bye"
```

15
Bobsleigh



In this game you can choose to steer your red bobsleigh down a random 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 course.

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.

Typing tips

This game involves a user-defined graphics character and a supplied graphics character. Both are listed within square brackets. Do not type in the brackets. Instead type in the appropriate letter, 'b', or number '5' in graphics mode. Notice that you have to type two apostrophes at the ends of lines 1320 and 3100.

Subroutine structure

500	Define bobsleigh graphics character
1000	Print track and bobsleigh
2400	Scroll screen
5000	Move bobsleigh
7000	Title frame
8000	Win/lose messages
9000	End of game

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 5320 ATTRribute 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.

Scope for improvement

If you find this game too fast-moving you can slow it down by introducing a PAUSE into subroutine 2400.

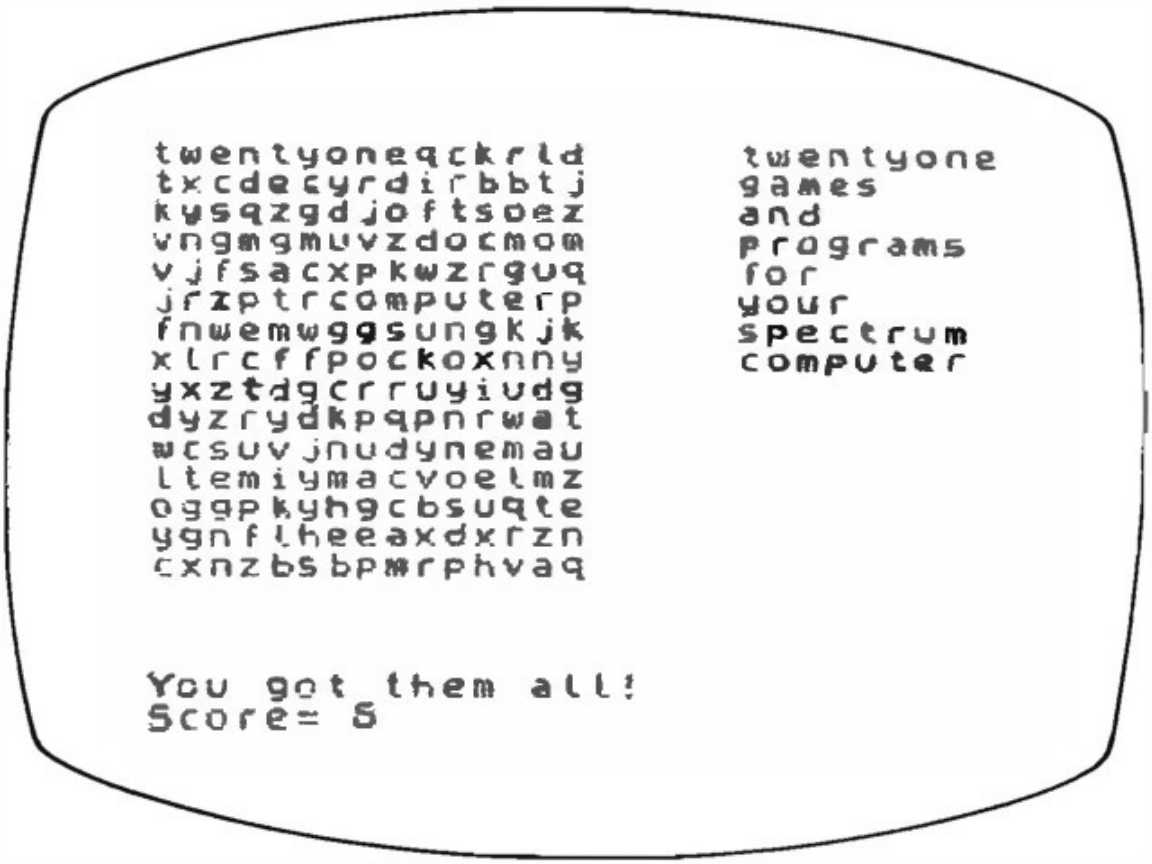
Program

```
5 REM Bobsleigh
```

```
  20 GOSUB 7000
 500 POKE USR "b"+0,BIN 00100000
 510 POKE USR "b"+1,BIN 00101000
 520 POKE USR "b"+2,BIN 11101000
 530 POKE USR "b"+3,BIN 11111100
 540 POKE USR "b"+4,BIN 01111110
 550 POKE USR "b"+5,BIN 00111110
 560 POKE USR "b"+6,BIN 00011110
 570 POKE USR "b"+7,BIN 00001110
 800 LET yb=1

1000 REM print track
1010 PAPER 7: INK 7
1020 CLS
1050 LET x=(RND*10)+5
1060 LET xb=x+2
1100 FOR y=1 TO 20
1140 PRINT AT y,x; INK 0; "[5]";
     AT y,x+d; "[5]"
1150 LET x=x+(SGN (RND-.5))
1160 IF x>31 THEN LET x=31
1170 IF x<0 THEN LET x=0
1200 NEXT y
1220 PRINT AT yb,xb; INK 2; "[b]"
1230 PAUSE 50
1250 FOR z=1 TO 20
1260 LET x=x+(SGN (RND-.5))
1270 IF x>31 THEN LET x=31
1280 IF x<0 THEN LET x=0
1310 POKE 23692,41
1320 PRINT AT 21,x; INK 0; "[5]";
     AT 21,x+d; "[5]";''
2000 GOSUB 5000

2400 NEXT z
3000 FOR z=1 TO 20
3100 PRINT AT 21,1''
3150 GOSUB 5000
3200 NEXT z
3500 GOSUB 8000
```

```
5000 REM move bobsleigh
5100 LET a$=INKEY$
5150 PRINT AT yb-1,xb;" "
5200 IF a$="5" THEN LET xb=xb-1
5210 IF a$="8" THEN LET xb=xb+1
5320 IF ATTR (yb,xb)=56 THEN GOTO 8500
5330 PRINT AT yb,xb; INK 2; "[b]"
5500 RETURN

7000 INK 0
7010 PAPER 7
7020 PRINT AT 1,5;"B o b s l e i g h"
7030 PRINT AT 5,0;"You must steer your
     bobsleigh"
7040 PRINT "down a dangerous course"
7050 INPUT "Select the difficulty level -"
     '"from 1 (easy) to 5 (difficult) ?";d
7060 IF d<1 OR d>5 THEN GOTO 7050
7070 LET d=9-d
7080 RETURN

8000 PRINT AT 20,1; INK 0;
     "Congratulations, you made it"
8100 GOTO 9000
8500 PRINT AT 20,1; INK 0; "You crashed"
8510 BEEP 1,-10

9000 INPUT "another game (y/n)";b$
9020 IF b$(1)="y" THEN RUN
9500 INK 0
9510 PAPER 7
9520 CLS
```

16
Spectrum 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 Spectrum 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, using a

flashing cursor as a pointer to identify the *first* letter of each word you find. If you're correct you score a point, otherwise you hear a dismal tone.

How to play

The Spectrum 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 it a little time and 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 "yes" or "no" to the next question which gives the option of having the list of hidden words displayed on the screen beside the complete grid. When the grid appears, you'll see that the top left hand position is bright. 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 Spectrum 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

Line 7820 appears to have the majority of the typing traps in this program. There are ten blanks between the double quotes and despite its odd appearance you are meant to type

```
(TO 10-LEN w$)
```

in that line – there should be nothing before the TO. In this program you'll find the letter 'l' used as the name of an array. It has been printed as a capital in the program listing to avoid any confusion. Also notice that 'list' is used as a variable name so make sure that you enter it a letter at a time in line 5050. There is a single space between

double quotes in lines 3180, 5020, 5030, and 6050 so don't type in a null string instead.

Subroutine structure

20	Main play loop
1000	Asks for words to be input
2000	Finds longest word left in list
3000	Constructs puzzle
5000	Prints puzzle
7000	Controls cursor
7500	Asks for guess of word
7600	Word correct routine
7800	Checks for word in word list
7900	Checks for word in word square
8000	Defines dot used in grid and sets score to zero
9000	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 3120–3130) and also by choosing directions at random (lines 3140–3150). Each word is then tested against the square position-by-position 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.

There is a clever use of the SCREEN$ function at lines 7020 and 7100. The SCREEN$ function is used to find out what letter is actually on the screen at x,y and then its PRINTed back on the screen in the same position with either BRIGHT 1 or BRIGHT 0. This is how the cursor is produced.

Line 1070 appears to be a little strange as it refers to itself. It is however, perfectly correct. The program will not move on unless you input a word of ten letters maximum.

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 Spectrum scramble

   20 GOSUB 1000
   30 GOSUB 3000
   40 GOSUB 8000
   50 GOSUB 6000
   60 GOSUB 7000

 1000 DIM L$(10,10)
 1005 DIM c(10)
 1006 LET list=0
 1010 INPUT "how many words ?";w
 1020 IF w<2 OR w>10 THEN BEEP .1,10:
       GOTO 1010
 1040 FOR i=1 TO w
 1050 PRINT AT 5+i,5;"Word number ";i;"=";
 1060 INPUT w$
 1070 IF LEN w$>10 THEN INPUT "maximum of
      ten letters!";w$: GOTO 1070
 1080 IF LEN w$<1 THEN GOTO 1060
 1090 PRINT w$
 1100 LET L$(i)=w$
 1110 LET c(i)=LEN w$
 1120 NEXT i
 1130 RETURN

 2000 LET m=0: LET j=0
 2010 FOR z=1 TO w
 2020 IF m<c(z) THEN LET m=c(z): LET j=z
 2030 NEXT z
 2040 RETURN
```

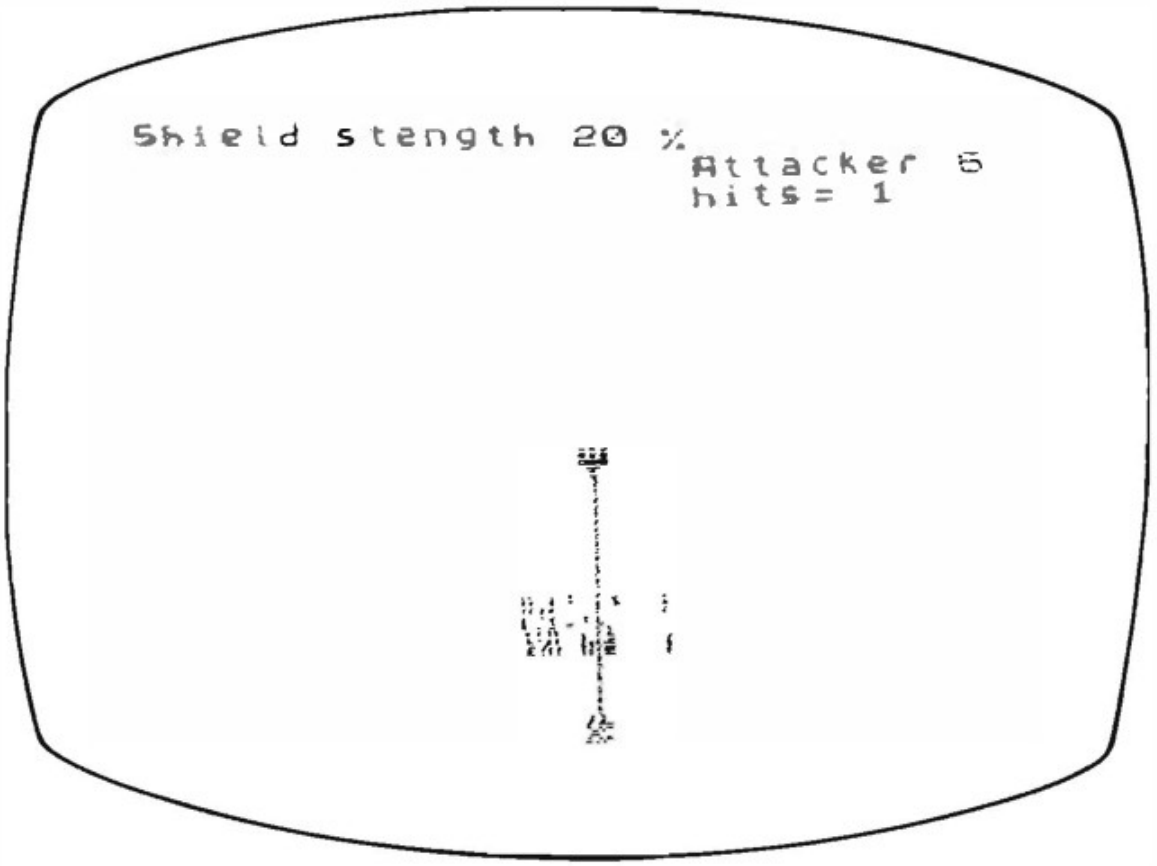
```
3000 DIM d$(15,15)
3010 CLS
3080 FOR i=1 TO w
3090 GOSUB 2000
3100 LET L=c(j)
3110 LET c(j)=0
3120 LET x=INT (RND*(15-L))+1
3130 LET y=INT (RND*(15-L))+1
3140 LET v=INT (RND*3)-1
3150 LET u=INT (RND*3)-1
3155 IF u=0 AND v=0 THEN GOTO 3140
3160 LET a=x: LET b=y
3165 IF v<0 THEN LET a=a+L
3166 IF u<0 THEN LET b=b+L
3170 FOR k=1 TO L
3180 IF d$(a,b)<>" " AND d$(a,b)<>L$(j,k)
     THEN GOTO 3120
3190 LET a=a+v
3200 LET b=b+u
3210 NEXT k
3300 PRINT "Working"
3310 LET a=x: LET b=y
3315 IF v<0 THEN LET a=a+L
3316 IF u<0 THEN LET b=b+L
3320 FOR k=1 TO L
3330 LET d$(a,b)=L$(j,k)
3340 LET a=a+v
3350 LET b=b+u
3360 NEXT k
3370 NEXT i
3380 RETURN

5000 CLS
5005 FOR m=1 TO 15
5010 FOR n=1 TO 15
5020 IF d$(m,n)=" " THEN PRINT "a";
5030 IF d$(m,n)<>" " THEN PRINT d$(m,n);
5040 NEXT n
5050 IF list=0 OR m>10 THEN PRINT
5055 IF list=1 AND m<=10 THEN PRINT
     TAB 20;L$(m)
5060 NEXT m
5070 RETURN
```

```
6000 INPUT "Do you want to preview"'
     " the puzzle ?";a$
6010 IF LEN a$=0 THEN GOTO 6000
6020 IF a$(1)="y" THEN GOSUB 5000
6030 FOR i=1 TO 15
6040 FOR j=1 TO 15
6050 IF d$(i,j)=" " THEN
     LET d$(i,j)=CHR$ (INT (RND*26)+97)
6060 NEXT j
6070 NEXT i
6080 INPUT "Do you want to display"'
     " the words beside the puzzle ?";a$
6090 IF LEN a$=0 THEN GOTO 6080
6100 IF a$(1)="y" THEN LET list=1
6110 GOSUB 5000
6120 RETURN

7000 LET x=0
7010 LET y=0
7020 PRINT AT y,x; BRIGHT 1;SCREEN$ (y,x);
7040 LET a$=INKEY$
7045 IF INKEY$="" THEN GOTO 7040
7050 IF a$="w" THEN GOTO 7500
7055 PRINT AT y,x; BRIGHT 0;SCREEN$ (y,x);
7060 IF a$="5" AND x>0 THEN LET x=x-1
7070 IF a$="8" AND x<14 THEN LET x=x+1
7080 IF a$="6" AND y<14 THEN LET y=y+1
7090 IF a$="7" AND y>0 THEN LET y=y-1
7095 IF a$="r" THEN GOSUB 9000
7100 PRINT AT y,x; BRIGHT 1;SCREEN$ (y,x);
7110 GOTO 7040
```

```
7500 INPUT "What is the word ?";w$
7510 IF LEN w$=0 OR LEN w$>15 THEN
     BEEP .5,20: GOTO 7500
7515 GOSUB 7800
7516 IF match=0 THEN BEEP .5,10:
     GOTO 7040
7520 FOR u=-1 TO 1
7530 FOR v=-1 TO 1
7540 IF u=0 AND v=0 THEN GOTO 7570
7550 GOSUB 7900
7560 IF match=1 THEN GOTO 7600
7570 NEXT v
7580 NEXT u
7590 BEEP .5,-20
7595 GOTO 7040

7600 LET score=score+1
7605 PRINT AT 19,0;"Score= ";score;" "
7610 BEEP.2,20
7620 LET L$(word)=" "
7630 IF score=w THEN GOTO 9500
7640 GOTO 7040

7800 LET match=0
7810 FOR i=1 TO w
7820 IF w$+"              "( TO 10-LEN w$)=
     L$(i) THEN LET match=1: LET word=i:
     RETURN
7830 NEXT i
7840 RETURN

7900 LET match=0
7910 LET a=x+1
7920 LET b=y+1
7930 FOR i=1 TO LEN w$
7940 IF w$(i)<>d$(b,a) THEN RETURN
7950 LET a=a+u
7960 LET b=b+v
7970 IF a<1 OR a>15 THEN RETURN
7980 IF b<1 OR b>15 THEN RETURN
7990 NEXT i
7995 LET match=1
7996 RETURN
```

```
8000 POKE USR "a"+0,BIN 00000000
8010 POKE USR "a"+1,BIN 00000000
8020 POKE USR "a"+2,BIN 00000000
8030 POKE USR "a"+3,BIN 00011000
8040 POKE USR "a"+4,BIN 00011000
8050 POKE USR "a"+5,BIN 00000000
8060 POKE USR "a"+6,BIN 00000000
8070 POKE USR "a"+7,BIN 00000000
8080 LET score=0
8090 RETURN

9000 INPUT "Are you sure that you"'
     " can find no more words ?";a$
9010 IF a$(1)<>"y" THEN RETURN

9020 CLS
9030 PRINT AT 8,10;"Final Score= ";score
9040 INPUT "Another game y/n ";a$
9050 IF a$="y" THEN RUN
9060 IF a$<>"n" THEN GOTO 9040
9070 STOP
9500 PRINT AT 18,0;"You got them all!"
9510 GOTO 9040
```

17
Mighty Missile



Your weapon can destroy anything – that is anything 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 percent of it left you will no longer have any protection from the enemies' lasers. This exciting 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 percent you will be vulnerable to attack. Once the shield is so eroded the enemy lasers can 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 3060, notice the two spaces after the percentage sign and before the double quotes. These serve the important function of blanking out previous figures as the number displayed gets smaller and so occupies fewer positions on the screen. The shield is composed of two rows of five solid blocks but you only need type the characters in once – in line 6010. Remember that [^8] means that you get into graphics mode and then press the 8 key with the CAPS SHIFT down.

Subroutine structure

20	Set up
46	Main play loop
175	End of game
2000	Print attacker and fire laser
2500	Check to see if player has activated missile
3000	Calculate shield strength
3500	Move and fire missile and test for hit or miss
4000	Explosion graphics and sound
5000	Set up attack orbit
6000	Print shield
8000	User-defined graphics

Programming details

The first interesting point to note about this program is the way in which the path of the attacking ship is calculated by subroutine 5000 and stored in a pair of arrays (x) and (y) to be used repeatedly for the various orbits used during the game. The 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 3000, 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 white.

Program

```
10 REM Mighty missile

20 GOSUB 8000
30 GOSUB 5000
40 GOSUB 6000
45 GOSUB 3000
```

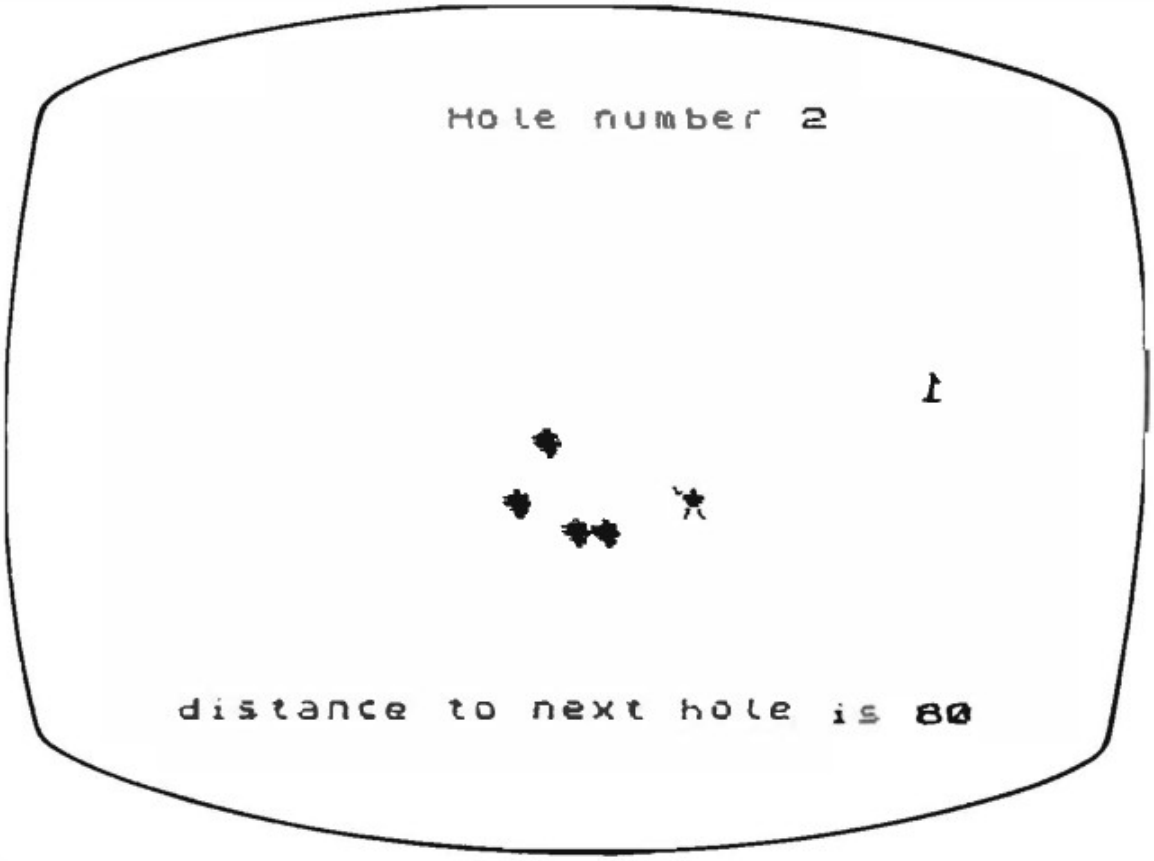
```
 46 FOR a=1 TO 10
 47 LET dir=SGN (RND-.5)
 48 PRINT AT 1,19;"Attacker ";a
 49 PRINT AT 2,19;"hits=";hit
 50 LET r=7-INT (RND*14)
 55 IF r<0 THEN LET s=3-r: LET e=29
 60 IF r>=0 THEN LET s=3: LET e=29-r
 65 IF dir=-1 THEN LET t=s: LET s=e:
    LET e=t
 70 FOR i=s TO e STEP dir
 80 GOSUB 2000
 85 GOSUB 2500
 86 IF f=1 THEN GOSUB 3500
 90 NEXT i
100 PRINT AT y(i+r),x(i);" "
105 IF f=1 THEN LET end=1
110 GOSUB 3000
115 IF end=2 THEN LET a=
120 IF f=1 THEN LET f=0: PRINT AT my,mx;
    FLASH 1;"[c]": BEEP 1,-10
125 PRINT AT my,mx;" "
130 LET mx=15: LET my=20
140 PRINT AT my,mx;"[b]"
150 IF end=0 THEN GOTO 70
160 LET end=0
170 NEXT a

175 IF end=2 THEN PRINT AT 10,10;
    "They got you"

180 INPUT "You hit ";(hit)''"Another
    game?";a$;
190 IF a$="y" THEN RUN
200 STOP
```

```
2000 PRINT AT y(i+r),x(i);" "
2010 PRINT AT y(i+r+dir),x(i+dir);"[a]"
2020 IF RND<.5 THEN RETURN
2025 IF c<=20 AND mx=x(i+dir) THEN
     LET end=2: GOTO 4000
2030 PLOT x(i+dir)*8+4,170-y(i+r+dir)*8
2035 LET d=10-INT (RND*20)
2040 DRAW d,-50
2045 BEEP .01,20
2050 PLOT INVERSE 1;x(i+dir)*8+4,
     170-y(i+r+dir)*8
2060 DRAW INVERSE 1;d,-50
2070 RETURN

2500 IF f=1 THEN RETURN
2505 LET a$=INKEY$
2510 IF a$="" THEN RETURN
2515 PRINT AT my,mx;" "
2520 IF a$="5" THEN LET mx=mx-8: GOTO 2540
2530 IF a$="8" THEN LET mx=mx+8: GOTO 2540
2535 RETURN
2540 PRINT AT my,mx;"[b]"
2550 LET f=1
2560 RETURN

3000 LET c=0
3010 LET j=175-16*8-1
3020 FOR i=13*8 To 17*8+8
3030 LET c=c+POINT (i,j)
3040 NEXT i
3050 LET c=c/40*100
3060 PRINT AT 0,0;"Shield strength ";c;
    " %  "
3070 RETURN

3500 PRINT AT my,mx;" "
3510 LET my=my-1
3515 IF my<2 THEN LET f=0: LET end=1:
     GOTO 4020
```

```
3520 PRINT AT my,mx;"[b]"
3530 IF mx<>x(i+dir) THEN RETURN
3540 IF my-y(i+r+dir)>2 OR my-y(i+r+dir)<0
     THEN RETURN
3550 LET end=1
3555 PRINT AT my,mx;" "
3560 LET mx=x(i+dir)
3570 LET my=y(i+r+dir)
3580 LET hit=hit+1
3590 LET f=0

4000 PLOT x(i+dir)*8+4,170-y(i+r+dir)*8
4010 DRAW 0,y(i+r+dir)*8-my*8
4020 PRINT FLASH 1;AT my,mx;"[c]"
4030 BEEP 1,-10
4050 PRINT AT y(i+r+dir),x(i+dir);" "
4060 LET i=e+dir
4070 RETURN

5000 DIM x(50): DIM y(50)
5010 LET x=0: LET y=0
5020 LET n=31
5030 FOR i=1 TO n
5040 LET x=x+1
5050 LET y=11-INT (((16-x)*(16-x))/20)
5060 LET x(i)=x
5070 LET y(i)=y
5080 NEXT i
5090 LET i=1
5100 LET hit=0
5110 RETURN

6000 FOR i=0 TO 1
6010 PRINT AT 17-i,13;
     "[^8][^8][^8][^8][^8]"
6020 NEXT i
6040 LET mx=15
6050 LET my=20
6060 PRINT AT my,mx;"[b]"
6070 LET f=0
6080 LET end=0
6090 RETURN
```

```
8000 POKE USR "a"+0,BIN 11011011
8010 POKE USR "a"+1,BIN 11011011
8020 POKE USR "a"+2,BIN 11011011
8030 POKE USR "a"+3,BIN 11111111
8040 POKE USR "a"+4,BIN 11111111
8050 POKE USR "a"+5,BIN 00111100
8060 POKE USR "a"+6,BIN 00011000
8070 POKE USR "a"+7,BIN 00011000
8100 POKE USR "b"+0,BIN 00011000
8110 POKE USR "b"+1,BIN 00111100
8120 POKE USR "b"+2,BIN 01111110
8130 POKE USR "b"+3,BIN 00011000
8140 POKE USR "b"+4,BIN 00011000
8150 POKE USR "b"+5,BIN 00111100
8160 POKE USR "b"+6,BIN 01111110
8170 POKE USR "b"+7,BIN 11100111
8200 POKE USR "c"+0,BIN 00100100
8210 POKE USR "c"+1,BIN 01100100
8220 POKE USR "c"+2,BIN 01001111
8230 POKE USR "c"+3,BIN 01001011
8240 POKE USR "c"+4,BIN 00110100
8250 POKE USR "c"+5,BIN 01110000
8260 POKE USR "c"+6,BIN 01001011
8270 POKE USR "c"+7,BIN 11010100
8300 BORDER 2
8310 RETURN
```

18
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 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. 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 three user-defined graphics characters, you will also find a capital 'O' used within double quotes in this program – in line 8100. Remember to enter the letters you find enclosed in square brackets in graphics mode but just enter the 'O' as a straightforward capital. You'll find the three functions SIN, COS and PI used in this program. All these are entered by a single key press while in extended mode. Look for them on the Q, W and M keys respectively.

Subroutine structure

500	Defines graphics character for flag
600	Defines graphics character for golfer
700	Defines graphics character for bunker
790	Initialises variables
2000	Sets up and plays each hole
4900	Reports score for each hole
4930	End of game
5000	Effectiveness of hit routine
5500	Plots balls' flight
6500	Detects overshoot
7000	Displays swinging club
8000	Putting routine
8500	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 subroutine 7000 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 5000) 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

500 REM flag
510 POKE USR "f"+0,BIN 00001000
520 POKE USR "f"+1,BIN 00001100
530 POKE USR "f"+2,BIN 00001110
540 POKE USR "f"+3,BIN 00001000
550 POKE USR "f"+4,BIN 00001000
560 POKE USR "f"+5,BIN 00001000
570 POKE USR "f"+6,BIN 00001000
580 POKE USR "f"+7,BIN 00111110

600 REM golfer
610 POKE USR "g"+0,BIN 00011000
620 POKE USR "g"+1,BIN 00111100
630 POKE USR "g"+2,BIN 01011010
640 POKE USR "g"+3,BIN 00111100
650 POKE USR "g"+4,BIN 00011000
660 POKE USR "g"+5,BIN 00100100
670 POKE USR "g"+6,BIN 00100100
680 POKE USR "g"+7,BIN 01000010
```

```
 700 REM bunker
 710 POKE USR "b"+0,BIN 00011000
 720 POKE USR "b"+1,BIN 00111110
 730 POKE USR "b"+2,BIN 11111110
 740 POKE USR "b"+3,BIN 11111111
 750 POKE USR "b"+4,BIN 01111111
 760 POKE USR "b"+5,BIN 00111110
 770 POKE USR "b"+6,BIN 00001110
 780 POKE USR "b"+7,BIN 00001100

 790 DIM t(9)
 800 LET b=1
 810 LET xh=0: LET xc=0
 820 LET yh=0: LET ht=0: LET yc=0
1000 INK 0
1010 PAPER 4
```

```
2000 FOR h=1 TO 9
2010 CLS
2040 PRINT AT 0,10;"Hole number ";h
2050 FOR z=1 TO 5
2060 LET xb=RND*5+10
2070 LET yb=RND*5+10
2080 PRINT AT yb,xb; PAPER 8; INK 6;"[b]"
2090 NEXT z
2100 REM drive section
2120 LET x=INT (RND*5)+1
2130 LET y=INT (RND*5+12)
2150 LET xt=INT (RND*15+15)
2170 LET yt=INT (RND*10)+1
2180 LET d=SGN (xt-x)*SQR ((xt-x)*
     (xt-x)+(yt-y)*(yt-y))
2190 PRINT AT yt,xt; PAPER 8; INK 8;
     "[f]"
2200 PRINT AT y,x; PAPER 8; INK 0; "[g]"
2300 PRINT AT 20,1; INK 0; "distance to
     next hole is ";INT d*10;"   ";
     REM 3 spaces
2310 INPUT INK 0;"which club (1 to 8)";c
2315 IF c<1 OR c>8 THEN GOTO 2310
2320 LET c=INT ((9-c)/b)+1
2325 GOSUB 7000
2330 GOSUB 5000
2335 LET b=1
2340 LET d=SGN (xt-xht)*SQR ((xt-xht)*
     (xt-xht)+(yt-yht)*(yt-yht))
2360 IF d<-3 THEN PRINT AT 20,1;INK 0;
     "you overshot-try another hole":
     PAUSE 100: GOTO 2010
2370 IF d<3 THEN PRINT AT 20,1;INK 0;
     "on the green": GOTO 8000
2380 PRINT AT y,x;" "
2390 LET x=xht
2400 LET y=yht
2500 GOTO 2190

4900 PRINT AT 2,10;"You took ";t(h);
     " strokes"
4910 PAUSE 100
4920 NEXT h
```

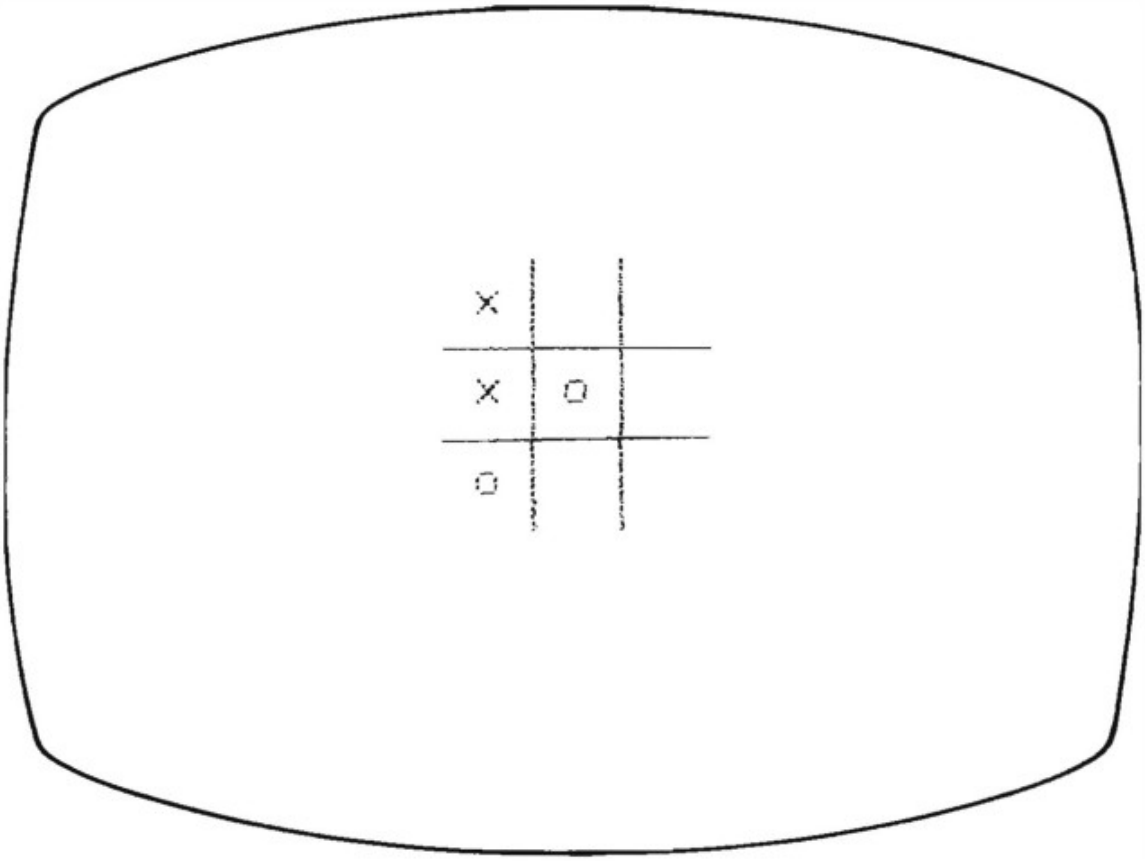
```
4930 CLS
4940 PRINT AT 2,10;"This round"
4945 PRINT
4950 FOR i=1 TO 9
4960 PRINT TAB 8;"hole";i;
4970 IF t(i)=-1 THEN PRINT " lost ball":
      GOTO 4990
4980 PRINT " ";t(i);" strokes"
4990 NEXT i
4995 INPUT "Another round ?";a$
4996 IF a$(1)="y" THEN RUN
4999 PAPER 7: INK 0: CLS: STOP

5000 REM hit routine
5010 PLOT INK 8; PAPER 8; INVERSE 1;
     xh+xc,yh+ht+yc
5100 LET vt=(c*(2+RND*.3))
5160 LET ht=0
5170 LET xh=0
```

```
5500 REM plot ball
5530 LET q=(y-yt)/(xt-x)
5600 LET vv=vt*(SIN (45*PI/180))
5610 LET xc=(x+1)*8
5620 LET yc=21-y: LET yc=yc*8
5630 LET vh=vt*(COS (45*PI/180))
5650 LET ht=ht+vv
5660 LET yh=q*xh
5670 LET vv=vv-2.5
5800 LET xh=xh+vh
5810 LET yh=q*xh
5820 IF xh+xc>255 THEN LET yh=0:
     LET yt=0: LET yc=0: LET xh=0:
     LET xc=0: GOTO 6500
5830 IF yh+ht+yc>175 THEN LET yh=0:
     LET yt=0: LET yc=0: LET xh=0:
     LET xc=0: GOTO 6500
5850 IF ht<=0 THEN GOTO 5900
5860 PLOT PAPER 8; OVER 1; INK 8;
     xh+xc,yh+ht+yc
5870 PAUSE 2
5880 PLOT PAPER 8; OVER 1; INK 8;
     xh+xc,yh+ht+yc
5890 GOTO 5650
5900 LET xht=INT ((xh+xc)/8)
5910 LET yht=INT ((175-yh-ht-yc)/8)
5999 IF POINT (xh+xc,yh+ht+yc)=1 THEN
     GOTO 8500
6000 PLOT INK 8; PAPER 8;xh+xc,yh+ht+yc
6010 RETURN

6500 PRINT AT 1,6;"You've lost your
     ball!!!"
6510 BEEP 2,-10
6520 LET t(h)=-1
6530 PAUSE 75
6540 GOTO 4920
```

```
7000 REM swing
7010 LET t(h)=t(h)+1
7100 LET xs=x*8+4
7110 LET ys=21-y: LET ys=ys*8+4
7200 FOR s=-5 TO -40 STEP -5
7220 LET a=s/30*PI
7230 LET sx=7*SIN a: LET sy=7*COS a
7240 PLOT INK 8; PAPER 8;xs,ys:
     DRAW INK 8; PAPER 8;
     OVER 1;sx,sy
7245 IF s<>-30 THEN PAUSE 5
7250 IF s=-30 THEN BEEP .1,-2
7270 PLOT INK 8; PAPER 8;xs,ys:
     DRAW INK 8; PAPER 8;
     OVER 1;sx,sy
7280 NEXT s
7500 RETURN
```

```
8000 REM putting
8010 PAPER 4
8020 CLS
8030 LET xg=INT (RND*5)+1
8040 LET yg=15
8050 LET xh=INT (RND*15)+10
8060 LET yh=15
8070 LET d=xh-xg
8080 IF d<0 THEN LET d=ABS (d)
8100 PRINT AT yh,xh; INK 0; "O"
8110 PRINT AT yg,xg; INK 0; "[g]"
8120 PRINT AT 18,1; "DISTANCE TO HOLE
     IS ";d;"    "
8130 INPUT "WHICH CLUB (1 TO 8)";c
8135 IF c<1 OR c>8 THEN GOTO 8130
8140 LET t(h)=t(h)+1
8145 LET h1=9-c+INT (RND*2)
8160 LET d=d-h1
8170 FOR z=xg+1 TO xg+h1
8180 PRINT AT yh,z; INK 0; "."
8190 PAUSE 10
8200 PRINT AT yh,z;" "
8210 NEXT z
8220 PRINT AT yg,xg;" "
8230 LET xg=xg+h1
8240 IF xg=xh THEN GOTO 8300
8250 IF d<0 THEN CLS: LET xg=xh+d:
     GOTO 8070
8260 GOTO 8100

8300 REM in the hole
8310 GOTO 4900

8500 REM in the bunker
8600 PRINT AT 20,1; INK 0;"in the
     bunker              "; REM 12 spaces
8610 PAUSE 100
8620 LET b=2
8630 RETURN
```

19
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 Spectrum an opponent who can be beaten. The Spectrum will make sensible moves but is not infallible so it is worth playing on until you beat the computer. It's actually a very good way of learning about game-playing strategy.

How to play

The 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 and the column number but do not leave a space between them. 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 4,1, the Spectrum 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 Spectrum will display "I win" if it has been successful, "You win" if you've been successful and "Drawn" if it's stalemate. In the case of a draw you have to fill the board for the game to be over.

Typing tips

There are no graphics characters in this game. High resolution graphics are used for the grid and capital Xs for the player's cross and capital Os for the Spectrum's nought.

Subroutine structure

20	Main play loop
4000	Evaluates computer's move
5000	Tries each move
6000	Gets player's move
7000	Prints board
8000	Plots frame
9000	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 when deciding its move – in other words it does not try to take account of the next move you will make – which is why it slips up sometimes and allows you to win!

Scope for improvement

If you want to brighten up the display of this program you could add a colourful frame around the grid (if you look at Mirror Tile you'll find a method of doing this.)

Program

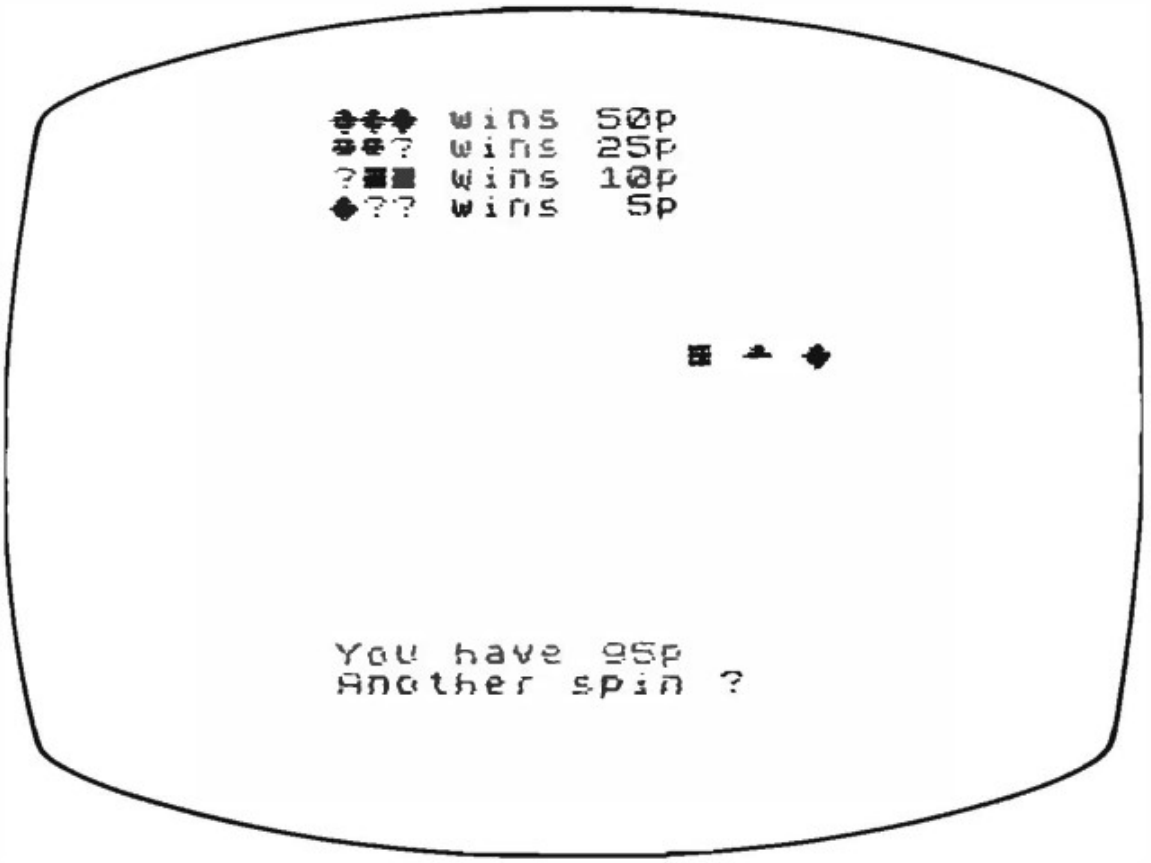
```
   10 REM  Noughts and crosses

   20 GOSUB 8000
   30 GOSUB 6000
   40 GOSUB 7000
   50 GOSUB 5000
   60 IF end=1 THEN GOTO 9500
   70 IF end=2 THEN GOSUB 7000:GOTO 9000
   75 IF dr=1 THEN GOTO 9100
   80 GOSUB 7000
   90 GOTO 30
   99 STOP

 4000 DIM x(4):DIM y(4)
 4020 FOR L=1 TO 3
 4030 LET s=0
 4040 LET t=0
 4050 FOR k=1 TO 3
 4060 IF a(L,k)=1 THEN LET s=s+1
 4070 IF b(L,k)=1 THEN LET t=t+1
 4080 NEXT k
 4090 IF s=0 THEN LET y(t+1)=y(t+1)+1
 4100 IF t=0 THEN LET x(s+1)=x(s+1)+1
 4105 NEXT L
 4110 FOR L=1 TO 3
 4120 LET t=0
 4125 LET s=0
 4130 FOR k=1 TO 3
 4140 IF a(k,L)=1 THEN LET s=s+1
 4150 IF b(k,L)=1 THEN LET t=t+1
 4160 NEXT k
 4170 IF s=0 THEN LET y(t+1)=y(t+1)+1,
 4180 IF t=0 THEN LET x(s+1)=x(s+1)+1
 4185 NEXT L
 4190 GOSUB 4300
 4200 GOSUB 4400
 4210 IF x(4)=1 THEN LET end=1: RETURN
```

```
4215 IF y(4)=1 THEN LET end=2
4220 LET e=128*y(4)-63*x(3)+31*y(3)-15
     *x(2)+7*y(2)
4230 RETURN
4300 LET t=0
4310 LET s=0
4320 FOR k=1 TO 3
4330 LET t=t+a(k,k)
4340 LET s=s+b(k,k)
4350 NEXT k
4360 IF s=0 THEN LET x(t+1)=x(t+1)+1
4370 IF t=0 THEN LET y(s+1)=y(s+1)+1
4380 RETURN
4400 LET t=0
4410 LET s=0
4420 FOR k=1 TO 3
4430 LET t=t+a(4-k,k)
4440 LET s=s+b(4-k,k)
4450 NEXT k
4460 IF s=0 THEN LET x(t+1)=x(t+1)+1
4470 IF t=0 THEN LET y(s+1)=y(s+1)+1
4480 RETURN

5000 LET m=-256: LET dr=1
5005 FOR j=1 TO 3
5010 FOR i=1 TO 3
5015 IF a(i,j)=1 OR b(i,j)=1 THEN
     GOTO 5040
5016 LET dr=0: LET b(i,j)=1
5020 GOSUB 4000
5025 IF end=1 THEN RETURN
5030 IF e>m THEN LET m=e: LET a=i:
     LET b=j
5035 LET b(i,j)=0
5040 NEXT i
5050 NEXT j
5060 LET b(a,b)=1
5070 RETURN

6000 INPUT "Your move (row col) ?";a$
6005 IF LEN a$<>2 THEN BEEP .5,-5:
     GOTO 6000
6010 LET j=VAL a$(1): LET i=VAL a$(2)
```

```
6020 IF i<1 OR i>3 THEN BEEP .5,5:
     GOTO 6000
6030 IF j<1 OR j>3 THEN BEEP .5,5:
     GOTO 6000
6040 IF a(i,j)=1 THEN GOTO 6100
6050 IF b(i,j)=1 THEN GOTO 6100
6060 LET a(i,j)=1
6070 PRINT AT 21,0;"
                   ";REM 25 spaces
6080 RETURN
6100 PRINT AT 21,0;"Position already
     occupied";
6110 BEEP .5,10
6120 GOTO 6000

7000 FOR j=1 TO 3
7010 FOR i=1 TO 3
7020 IF a(i,j)=1 THEN PRINT AT j*3+3,i*3+8
     ;"X";
7030 IF b(i,j)=1 THEN PRINT AT j*3+3,i*3+8
     ;"O";
7040 IF a(i,j)+b(i,j)=0 THEN PRINT AT
     j*3+3,i*3+8;" ";
7050 NEXT i
7060 PRINT
7070 NEXT j
7080 RETURN

8000 DIM a(3,3)
8005 DIM b(3,3)
8010 GOSUB 7000
8040 PLOT 104,135
8050 DRAW 0,-72
8060 PLOT 128,135
8070 DRAW 0,-72
8080 PLOT 80,111
8090 DRAW 72,0
8100 PLOT 80,87
8110 DRAW 72,0
8120 LET end=0
8130 LET dr=0
8200 RETURN
```

```
9000 PRINT "I WIN"
9010 GOTO 9600
9100 PRINT "DRAW"
9110 GOTO 9600
9500 PRINT "YOU WIN"
9600 INPUT "Another game y/n?"'a$
9610 IF a$="y" THEN RUN
9620 CLS
```

20
Fruit Machine



Here's a way of playing the fruit machines without losing a penny – the Spectrum gives you 100 pence to start with, takes 10 pence for every go, and awards you a sum between 5 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, this program includes some really clever graphics techniques so that you see the drum of the fruit machine rotate smoothly. 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 – diamond, circle, square and triangle. The combination for the jackpot is three diamonds. Two circles and any other symbol wins 25 pence; two squares and any

other symbol wins 10 pence and one diamond and any other symbol wins 5 pence. These combinations are winners wherever they occur on the line and not just as in the pattern suggested by the screen display. 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 and once you run out of money the game is over.

It is important not to break into this program while it is running because of the alterations it makes to the user-defined graphics table. If by accident you do interrupt its running then you will need to switch your Spectrum on and off again to restore normality.

Typing tips

Notice that there are two spaces after the 'p' in line 70. It is important that you copy this when you type the program in or you could be misled about the value of your winnings. The two spaces are needed to blank out any previous values that were larger than the current value. The only other point to remember is to enter graphics mode whenever you see a letter enclosed in square brackets – and not to type the brackets. Leave graphics mode before typing the '?'s in the PRINT statements.

Subroutine structure

15	Main play loop
70	Offers another spin and end of game
1000	Rotates drum
2000	Sets starting points of drum
3000	Decides winning combinations
4000	Tests for win
5000	Signals jackpot
7000	Sets up initial conditions
8000	Defines graphics characters
8500	Replicates graphics
9000	Signals when broke

Programming details

The program uses some very tricky programming techniques –

which is why it achieves its effect in so short a length of BASIC. The rotating drum is produced by moving the starting point of the area of memory used to define the shapes of the user-defined graphics characters. The effect of this is that the graphics characters can be displayed starting part way through their definitions. This produces the visual illusion of a smoothly rotating drum.

Scope for improvement

If you like adding graphics to programs there is a lot of scope in this game – not just for the symbols used but also for a surround that looks like a one-armed-bandit.

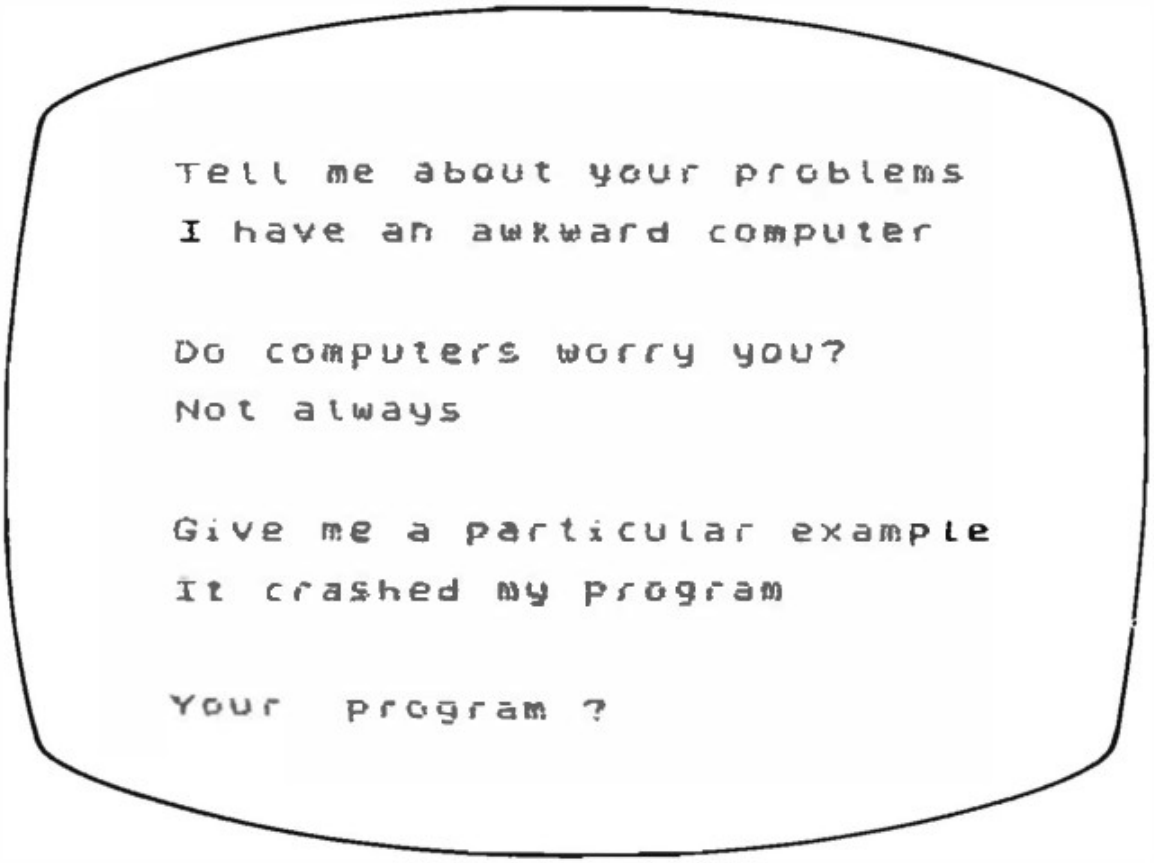
Program

```
  10 REM Fruit machine

  15 GOSUB 8000
  20 GOSUB 7000
  35 GOSUB 3000
  40 GOSUB 2000
  45 LET m=m-10
  50 GOSUB 1000
  60 GOSUB 4000
  65 IF m<=0 THEN GOTO 9000

  70 PRINT AT 20,0;"You have ";m;"p    "
  80 PRINT "Another spin ?"
  90 LET a$=INKEY$
 100 IF a$="" THEN GOTO 90
 110 IF a$="y" THEN GOTO 40
 120 PRINT AT 21,0;"You take home ";m;"p"
 130 STOP

1000 FOR i=d TO d+33
1010 PRINT AT 10,12;CHR$ (p);" ";
     CHR$ (q);" ";CHR$ (r)
1020 POKE udg,i
1030 NEXT i
1040 POKE udg,d
1050 RETURN
```

```
2000 LET p=144+INT (RND*4)
2010 LET q=144+INT (RND*4)
2020 LET r=144+INT (RND*4)
2030 RETURN

3000 PRINT AT 2,0;"[a][a][a] wins 50p"
3010 PRINT "[c][c]? wins 25p"
3020 PRINT "?[d][d] wins 10p"
3030 PRINT "[a]?? wins  5p"
3040 RETURN

4000 IF p=144 AND q=144 AND r=144 THEN
     LET m=m+50: GOTO 5000
4010 IF (p=146)+(q=146)+(r=146)=2 THEN
     LET m=m+25: BEEP .1,20

4020 IF (p=147)+(q=147)+(r=147)=2 THEN
     LET m=m+10: BEEP .1,20: RETURN
4030 IF (p=144)+(q=144)+(r=144)=2
     THEN LET m=m+5: BEEP .1,20: RETURN
4040 RETURN

5000 FOR i=1 TO 20
5010 BEEP .01,20
5020 NEXT i
5030 RETURN

7000 LET udg=23675
7010 LET d=PEEK udg
7020 LET m=100
7030 RETURN
```

```
8000 POKE USR "a"+0,BIN 00000000
8010 POKE USR "a"+1,BIN 00011000
8020 POKE USR "a"+2,BIN 00111100
8030 POKE USR "a"+3,BIN 01111110
8040 POKE USR "a"+4,BIN 11111111
8050 POKE USR "a"+5,BIN 01111110
8060 POKE USR "a"+6,BIN 00111100
8070 POKE USR "a"+7,BIN 00011000
8100 POKE USR "b"+0,BIN 00000000
8110 POKE USR "b"+1,BIN 00011000
8120 POKE USR "b"+2,BIN 00111100
8130 POKE USR "b"+3,BIN 01111110
8140 POKE USR "b"+4,BIN 11111111
8150 POKE USR "b"+5,BIN 00000000
8160 POKE USR "b"+6,BIN 00000000
8170 POKE USR "b"+7,BIN 00000000
8200 POKE USR "c"+0,BIN 00000000
8210 POKE USR "c"+1,BIN 00111100
8220 POKE USR "c"+2,BIN 01111110
8230 POKE USR "c"+3,BIN 01111110
8240 POKE USR "c"+4,BIN 01111110
8250 POKE USR "c"+5,BIN 00111100
8260 POKE USR "c"+6,BIN 00000000
8270 POKE USR "c"+7,BIN 00000000
8300 POKE USR "d"+0,BIN 00000000
8310 POKE USR "d"+1,BIN 01111110
8320 POKE USR "d"+2,BIN 01111110

8330 POKE USR "d"+3,BIN 01111110
8340 POKE USR "d"+4,BIN 01111110
8350 POKE USR "d"+5,BIN 01111110
8360 POKE USR "d"+6,BIN 01111110
8370 POKE USR "d"+7,BIN 00000000

8500 FOR i=0 TO 7
8510 POKE USR "e"+i,PEEK (USR "a"+i)
8520 POKE USR "f"+i,PEEK (USR "b"+i)
8530 POKE USR "g"+i,PEEK (USR "c"+i)
8540 POKE USR "h"+i,PEEK (USR "d"+i)
8550 NEXT i
8990 RETURN

9000 CLS
9020 PRINT AT 10,5; FLASH 1;
     "You are broke"
```

21
Spectrum Smalltalk



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

Coping with the syntax of the English language is a very complicated problem which this program has to contend with. 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 Spectrum.

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 Spectrum 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

Both when typing this program in and when using it, do be careful about your spelling. If you type in either the initial program or subsequent responses with misspellings 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 Spectrum and, while typing in the program, notice their use not only within words but also to throw lines of space on the screen – as in lines 2202, 2220 and 9830. Another point to notice in the listing is the occurrence of '(TO' and 'TO)' for string slicing.

Subroutine structure

20	Main program loop
1000	Initial message and set up
2000	Input human's sentence
3000	Divides sentences into words
3600	Changes tense/pronouns
3800	Tense/pronouns data
5000	Finds keywords in sentence
6000	Keywords data
7000	Keyword responses
9800	Prints computer's response
9900	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 6000, the keywords are paired with the line number of the subroutine that responds to them. Also it is 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 Spectrum smalltalk

20 GOSUB 1000
30 GOSUB 2000
40 GOSUB 3000
50 GOSUB 5000
60 IF num<>0 THEN GOSUB num
70 GOSUB 9800
80 GOTO 30
```

```
1000 CLS
1010 PRINT AT 1,2;"Hi! My name is
     ZX Spectrum"
1030 PRINT
1040 PRINT "I would like you to talk to me"
1050 PRINT "but I don't have ears so will"
1060 PRINT "you type sentences on my"
1070 PRINT "keyboard"
1074 PRINT
1075 PRINT "Don't use any punctuation apart"
1076 PRINT "from apostrophies which are"
1078 PRINT "important"
1080 PRINT
1110 PRINT "When you have finished typing"
1115 PRINT "press ENTER"
1120 PRINT AT 18,0;BRIGHT 1;
     "Tell me about your problems"
1130 LET r$=""
1140 LET m$=""
1150 LET d$=""
1160 DIM n$(3,32)
1170 LET n$(1)="Please go on"
1180 LET n$(2)="I'm not sure I
     understand you"
1190 LET n$(3)="Tell me more"
1200 DIM i$(3,40)
1210 LET i$(1)="Let's talk some more
     about your"
1220 LET i$(2)="Earlier you spoke of your "
1230 LET i$(3)="Does that have anything
     to do with your "
1235 DIM j$(2,32)
1240 LET j$(1)="Are you just being negative"
1250 LET j$(2)="I see"
1990 RETURN
```

```
2000 LET a$=""
2010 LET b$=INKEY$
2015 POKE 23692,255
2020 IF b$="" THEN GOTO 2010
2025 IF INKEY$<>"" THEN GOTO 2025
2030 IF CODE(b$)=13 THEN GOTO 2200
2040 IF CODE (b$)=12 AND a$<>"" THEN LET
     a$=a$( TO LEN a$-1): GOTO 2090
2050 IF CODE b$<32 OR CODE b$>126 THEN
     GOTO 2000
2060 LET a$=a$+b$
2090 PRINT AT 20,0;a$;" "
2100 GOTO 2010
2200 IF a$(LEN a$)=" " THEN
     LET a$=a$( TO LEN a$-1): GOTO 2200
2202 IF a$=r$ THEN PRINT AT 21,0; "You're
      repeating  yourself!"'''':GOTO 2000
2203 LET r$=a$
2204 IF a$="" THEN GOTO 2000
2205 LET a$=" "+a$
2215 IF CODE (a$(2))<97 THEN
     LET a$(2)=CHR$ (CODE (a$(2))+32)
2220 PRINT AT 21,0;'''
2230 RETURN

3000 DIM w(20,2)
3005 LET n=1
3010 LET b=0
3020 FOR i=1 TO LEN a$
3040 IF (a$(i)=" " OR a$(i)=",") AND b=0
     THEN LET b=1
3050 IF (a$(i)<>" " OR a$(i)<>",") AND b<=1
     THEN LET w(n,1)=i: LET b=2
3060 IF (a$(i)=" " OR a$(i)=",") AND b=2
     THEN LET w(n,2)=i-1: LET n=n+1: LET b=0
3070 NEXT i
3080 LET w(n,2)=LEN a$
```

```
3600 FOR i=1 TO n
3605 RESTORE 3800
3610 READ b$
3620 IF b$="s" THEN GOTO 3690
3630 IF b$<>a$(w(i,1) TO w(i,2)) THEN
     GOTO 3680
3640 READ c$
3650 LET a$=a$( TO w(i,1)-1)+c$+
     a$(w(i,2)+1 TO )
3654 LET w(i,2)=w(i,2)+LEN c$-LEN b$
3655 FOR j=i+1 TO n
3660 LET w(j,2)=w(j,2)+LEN c$-LEN b$
3664 LET w(j,1)=w(j,1)+LEN c$-LEN b$
3665 NEXT j
3670 GOTO 3690
3680 READ b$
3685 GOTO 3630
3690 NEXT i
3700 RETURN

3800 DATA "my","your*","I","you*","i","you*"
3810 DATA "mum","Mother","dad","Father"
3820 DATA "dreams","dream","you","I*",
     "me","you*"
3830 DATA "your","my*","myself","yourself*",
     "I'm","you're"
3840 DATA "yourself","myself*",
     "I'm","you're*"
3850 DATA "you're","I'm*","am","are*"
3870 DATA "i'm","you're*"
3880 DATA "were",was"
3885 DATA "are","am"
3890 DATA "s","s"
```

```
5000 RESTORE 6000
5010 READ b$,num
5020 IF b$="s" THEN GOTO 5710
5025 FOR i=1 TO n
5030 IF a$(w(i,1) TO w(i,2))<>b$ THEN
     GOTO 5700
5040 LET t$=a$(w(i,2)+1 TO )
5050 RETURN
5700 NEXT i
5705 GOTO 5010
5710 LET num=0
5720 IF m$<>"" THEN GOTO 5800
5730 LET p$=n$(INT (RND*3)+1)
5740 RETURN
5800 LET p$=i$(INT (RND*3)+1)+m$
5900 RETURN

6000 DATA "computer",7000,"machine",7000
6010 DATA "like",7100,"same",7100,
     "alike",7100
6020 DATA "if",7200,"everbody",7300
6024 DATA "can",8200,"certainly",8250
6025 DATA "how",8100,"because",8150
6026 DATA "always",7800
6030 DATA "everyone",7300,"nobody",7300
6034 DATA "was",7500
6035 DATA "I*",8800
6040 DATA "no",7400
6060 DATA "your*",7600
6070 DATA "you're*",8500,"you*",8650
6110 DATA "hello",8300,"maybe",8350
6120 DATA "my*",8370,"no",8420
6130 DATA "yes",8250,"why",8450
6140 DATA "perhaps",8350,"sorry",8400
6160 DATA "what",8450
6900 DATA "s",0
```

```
7000 LET p$="Do computers worry you?"
7010 RETURN
7100 LET p$="In what way ?"
7110 RETURN
7200 LET p$="Why talk of possibilities"
7210 RETURN
7300 LET p$="Really "+b$+" ?"
7310 RETURN
7400 IF i=n THEN GOTO 7450
7410 LET i=i+1
7420 IF a$(w(i,1) TO w(i,2))="one" THEN
     LET b$=b$+" one":GOTO 7300
7450 LET p$=j$(INT (RND*2)+1)
7460 RETURN
7500 IF i=n THEN GOTO 9900
7510 LET i=i+1
7515 IF i>n THEN GOTO 5720
7520 IF a$(w(i,1) TO w(i,2))<>"you*" THEN GOTO 7550
7530 LET p$="What if you were "+
     a$(w(i,2)+1 TO )+" ?"
7540 RETURN
7550 IF a$(w(i,1) TO w(i,2))<>"I*" THEN
     GOTO 5720
7560 LET p$="Would you like to believe I
     was "+a$(w(i,2)+1 TO )
7570 RETURN
```

```
7600 LET i=i+1
7605 IF i>n THEN GOTO 7450
7610 IF a$(w(i,1) TO w(i ,2))="Mother" THEN
     GOTO 7700
7620 IF a$(w(i,1) TO w(i,2))="Father" THEN
     GOTO 7700
7630 IF a$(w(i,1) TO w(i,2))="sister" THEN
     GOTO 7700
7640 IF a$(w(i,1) TO w(i,2))="brother" THEN
     GOTO 7700
7650 IF a$(w(i,1) TO w(i,2))="wife" THEN
     GOTO 7700
7660 IF a$(w(i,1) TO w(i,2))="husband" THEN
     GOTO 7700
7670 IF a$(w(i,1) TO w(i,2))="children" THEN
     GOTO 7700
7680 IF LEN t$>10 THEN LET m$=t$
7690 LET p$="your "+t$+" ?"
7695 RETURN
7700 LET p$="Tell me more about your family"
7710 RETURN
7800 LET p$="Give me a particular example"
7810 RETURN
7900 LET i=i+1
7905 IF i>n THEN LET p$="Am I what ?":RETURN
7920 LET p$="Why are you interested in
     whether I am "+a$(w(i,1) TO )+" or not?"
7930 RETURN
8000 LET p$="Do you think you are "+
     a$(w(i,2) TO )
8030 RETURN
8100 LET p$="Why do you ask ?"
8110 RETURN
8150 LET p$="Tell me about any other reasons"
8160 RETURN
8200 LET i=i+1
8205 IF i>n THEN LET p$="What ?": RETURN
8210 IF a$(w(i,1) TO w(i,2))="I*" THEN
     LET p$="Do you believe I can"+
     a$(w(i,2)+1 TO )+" ?": RETURN
8220 IF a$(w(i,1) TO w(i,2))="you*" THEN
     LET p$="Do you believe you can "+
     a$(w(i,2)+1 TO )+" ?":RETURN
8230 GOTO 5720
8250 LET p$="You seem very positive"
8260 RETURN
```

```
8300 LET p$="Pleased to meet you - let's
     talk about your problems"
8310 RETURN
8350 LET p$="Could you try to be more
     positive"
8360 RETURN
8370 LET p$="Why are you concerned about
     my "+t$
8380 RETURN
8400 LET p$="You don't have to apologise
     to me"
8410 RETURN
8450 LET p$="Some questions are difficult
       to answer..."
8460 RETURN
8500 LET i=i+1
8505 IF i>n THEN GOTO 5720
8510 LET p$="I am sorry to hear that you are"
     +a$(w(i,1) TO w(i,2))
8520 IF a$(w(i,1) TO w(i,2))="sad" THEN
     RETURN
8530 IF a$(w(i,1) TO w(i,2))="unhappy" THEN
     RETURN
8540 IF a$(w(i,1) TO w(i,2))="depressed" THEN
     RETURN
8550 IF a$(w(i,1) TO w(i,2))="sick" THEN
     RETURN
8560 LET p$="How have I helped you to be "+
     a$(w(i,1) TO w(i,2))
8570 IF a$(w(i,1) TO w(i,2))="happy" THEN
     RETURN
8580 IF a$(w(i,1) TO w(i,2))="elated" THEN
     RETURN
8590 IF a$(w(i,1) TO w(i,2))="glad" THEN
     RETURN
8600 IF a$(w(i,1) TO w(i,2))="better" THEN
     RETURN
8610 LET p$="Is it because you are "+
     a$(w(i,1) TO  )+" you would like
     to talk to me ?"
8620 RETURN
```

```
8650 IF i=1 THEN GOTO 8660
8654 IF a$(w(i-1,1) TO w(i-1,2))="are*" THEN
     GOTO 8000
8655 LET i=i+1
8656 IF i>n THEN GOTO 9900
8660 IF a$(w(i,1) TO w(i,2))="are*" THEN
     GOTO 8500
8670 IF a$(w(i,1) TO w(i,2))="want" OR
     a$(w(i,1) TO w(i,2))="need" THEN
     LET p$="what would it mean if you got "
     +a$(w(i,2)+1 TO ): RETURN
8675 IF a$(w(i,1) TO w(i,2))="think" THEN
     LET p$="do you really think so": RETURN
8680 IF a$(w(i,1) TO w(i,2))="can't" OR
     a$(w(i,1) TO w(i,2))="cannot" THEN
     LET p$="How do you know you can't "
     +a$(w(i,2)+1 TO ): RETURN
8690 IF a$(w(i,1) TO w(i,2))="feel" THEN LET
     p$="tell me more about hou you feel":
     RETURN
8700 GOTO 5720
8800 IF i-1<1 THEN GOTO 8805
8804 IF a$(w(i,1) TO w(i,2))="am" THEN
     GOTO 7900
8805 LET i=i+1
8810 IF i>=n THEN LET p$="What am I ?":
     RETURN
8820 IF a$(w(i,1) TO w(i,2))="am" THEN
     LET p$="Why do you think so ?": RETURN
8830 LET p$="Is that what you think of me !"
8840  RETURN

9800 FOR j=1 TO LEN p$
9810 IF p$(j)<>"*" THEN PRINT BRIGHT 1;p$(j);
9820 NEXT j
9830 PRINT AT 21,0''''
9840 RETURN

9900 LET p$="Please talk sensibly !"
9910 RETURN
```

Other books of interest from Granada:

THE ZX SPECTRUM And How To Get The Most From It
Ian Sinclair
0 246 12018 5

THE SPECTRUM PROGRAMMER
S. M. Gee
0 246 12025 8

THE SPECTRUM BOOK OF GAMES
M. James, S. M. Gee and K. Ewbank
0 246 12047 9

THE BBC MICRO— AN EXPERT GUIDE
Mike James
0 246 12014 2

THE COMPLETE PROGRAMMER
Mike James
0 246 12015 0

PROGRAMMING WITH GRAPHICS
Garry Marshall
0 246 12021 5

SIMPLE INTERFACING PROJECTS
Owen Bishop
0 246 12026 6

COMPUTING FOR THE HOBBYIST AND SMALL BUSINESS
A. P. Stephenson
0 246 12023 1

COMPUTER LANGUAGES AND THEIR USES
Garry Marshall
0 246 12022 3

INTRODUCING SPECTRUM MACHINE CODE
Ian Sinclair
0 246 12082 7

THE DRAGON 32 And How To Make The Most Of It
Ian Sinclair
0 246 12114 9

THE DRAGON 32 BOOK OF GAMES
M. James, S. M. Gee and K Ewbank
0 246 12102 5

21 GAMES FOR THE BBC MICRO
M. James, S. M. Gee and K Ewbank
0 246 12103 3

COMMODORE 64 COMPUTING
Ian Sinclair
0 246 12030 4

Z-80 MACHINE CODE FOR HUMANS
Alan Tootill and David Barrow
0 246 12031 2

DATABASES FOR FUN AND PROFIT
Nigel Freestone
0 246 12032 0

CHOOSING A MICROCOMPUTER
F. X. Samish
0 246 12029 0

APPLE II PROGRAMMERS HANDBOOK
R. C. Vile
0 246 12027 4

THE ORIC - 1 And How To Get The Most From It
Ian Sinclair
0 246 12130 0

THE DRAGON PROGRAMMER
S. M. Gee
0 246 12133 5

LYNX COMPUTING
Ian Sinclair
0 246 12131 9

21 QUALITY GAMES FOR YOUR ZX SPECTRUM

Here is a selection of twenty-one exciting, high quality games written specially for the ZX Spectrum – both 16K and 48K models. These games make full use of the Spectrum's facilities, and are fully tested and crash-proofed.

Among those included are variants of popular arcade games such as Spectrum Invaders, Rainbow Squash and Mighty Missile; board games such as Capture the Quark; a compelling adventure game Treasure Island; a conversational game in which the Spectrum answers you back; and a commando game for you to test your skill. Each program is presented to appeal to all Spectrum owners, no matter how old or young, whether you are completely new to computing or fairly experienced.

Each program is accompanied by an explanation of how to play the game and how the program works, including tips on how to modify or 'personalise' it for your own special use.

Normally games of this quality are only available individually on cassette. This book therefore gives you remarkable value.

The Authors

Mike James is the author of several very successful books on programming, and is a regular contributor to *Computing Today* and *Electronics and Computing Monthly*.

S. M. Gee has written two other books and is a regular contributor to *Computing Today*.

Kay Ewbank is an experienced programmer who has been involved in many joint projects with the other two authors.