



GAMES TO PLAY ON YOUR ZX SPECTRUM

Martin Wren-Hilton

SHIVA'S
micro puzzle books

GAMES TO PLAY ON YOUR ZX SPECTRUM

GAMES TO PLAY ON YOUR ZX SPECTRUM

Martin Wren-Hilton



Shiva Publishing Limited

SHIVA PUBLISHING LIMITED

4 Church Lane, Nantwich, Cheshire CW5 5RQ, England

© Martin Wren-Hilton, 1982

Reprinted 1983 (twice)

ISBN 0 906812 28 3

All rights reserved. No part of this publication may be reproduced, stored in a retrieval system, or transmitted in any form or by any means, electronic, mechanical, photocopying, recording and/or otherwise, without the prior written permission of the Publishers.

This book is sold subject to the Standard Conditions of Sale of Net Books and may not be resold in the UK below the net price given by the Publishers in their current price list.

Typeset and printed by Devon Print Group, Exeter

CONTENTS

	Introduction	8
1	Breakout	9
2	String Art	12
3	Helicopter	14
4	Worm Race	18
5	Flower	21
6	Mastermind	22
7	Monitor	25
8	Bomber	27
9	Kaleidoscope	31
10	Customer	33
11	Spiral	39
12	Stunt Bike	40
13	Draughts	44

INTRODUCTION

The ZX Spectrum, whether it be 16K or 48K, is a very impressive machine. By that, I mean that it has great potential – potential for doing more than you may think. In writing this book, I have brought together a number of games and utilities which should amuse or amaze you, from the brilliant BREAKOUT program to the mind-boggling MASTERMIND game. As well as being fun to run, these programs will also teach you some interesting programming techniques as well as a couple of commands which were left out of the *Manual* altogether!

In learning how to use the machine, through the machine, I believe that it is up to you to modify any of the programs listed by adding extra lines where you want to. That is not to say that these programs have errors which need correcting, but a quick alteration here or there may help to "personalise" the program.

It should be noted that *all* of these programs will run on either model of the Spectrum. If you do not feel up to entering all of them, you may like to know that some of the games are available on cassette, for further details contact Shiva Publishing Limited.

Finally, I hope that you enjoy programming your Spectrum with these listings, and that you gain many hours of pleasure playing the games.

Martin Wren-Hilton
Blackpool, August 1982

Note the following conventions used to represent graphics in the program listings:

Small g means "graphics character": it is followed by the letter or number of the right key, so "g6" is . ■

Characters accessed by CAPS SHIFT get a "c" added on the end, so "g6c" is ■ .

A box □ is used to represent a space when one is not otherwise obviously needed.

Boxes around characters denote inverse video.

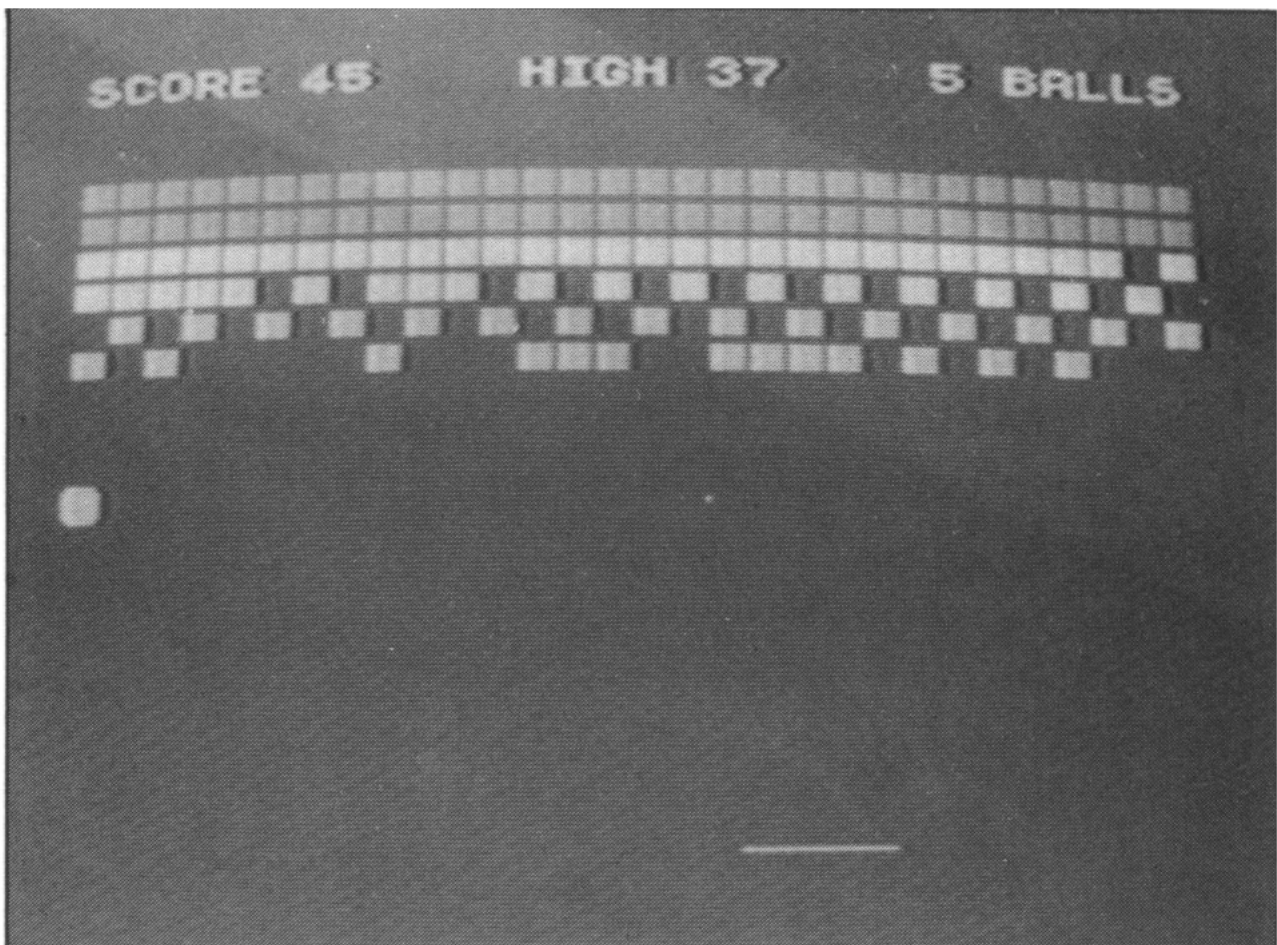
*A game designed to drive
you up the wall...*

1 BREAKOUT

This game, based upon the pub game of the same name, seems to be extremely popular, judging by the reaction of my friends when they see it. For those of you unfortunates that have never played BREAKOUT before, the game involves knocking bricks out of a wall with a bat (which you control). It may seem easy in principle, but in practice, it is much more of a challenge.

The program should be entered as it is listed, but it is worth noting that the lines in 100 and 115 are actually underline characters – four of them in each program line.

The 5 key moves your bat left and the 8 moves it right. Can you beat my high score of 243?




```

10  LET HS = 0: OVER 0

20  LET SC = 0: RESTORE: BORDER 0: PAPER 0: INK 1: BRIGHT 1: CLS:
    LET S = INT (RND * 5 + 16): LET X = 10: LET Y = 5: LET A1 =
    -1: LET B1 = 1: LET N = 9

30  FOR B = USR "A" TO USR "A" + 15: READ C: POKE B, C: NEXT B

50  PRINT INK 6; "□ SCORE □"; SC; TAB 13; "HIGH □"; HS; TAB 24;
    "9 BALLS" ' ' '

60  FOR A = 1 TO 6: READ B: INK B

70  PRINT "□ gB [30 times]"

80  NEXT A

85  INK 1

90  LET Z = ATTR (X, Y) < > 65 AND ATTR (X, Y) < > 67

95  PRINT AT X, Y; INK 5; "gA"

100 LET S = S + (INKEY$ = "8") - (INKEY$ = "5"): LET S = S + (S =
    -1) - (S = 26): PRINT AT 21, S; INK 3; "□ _ _ _ _ □"

110 LET A1 = A1 - 2 * A1 * Z - 2 * (X = 21) + 2 * (X = 2): LET B1
    = B1 + 2 * (Y = 1) - 2 * (Y = 30): IF Z THEN LET SC = SC + 1:
    BEEP .01, 45 - 2 * x: PRINT AT 0, 7; INK 6; SC

115 LET S = S + (INKEY$ = "8") - (INKEY$ = "5"): LET S = S + (S =
    -1) - (S = 26): PRINT AT 21, S; INK 3; "□ _ _ _ _ □"

120 PRINT AT X, Y; "□": LET X = X + A1: LET Y = Y + B1: IF (Y < S
    + 1 OR Y > S + 4) AND X = 21 THEN GO TO 140

125 IF SC = 180 THEN PRINT AT 3, 0; : RESTORE 1020: GO TO 60

130 GO TO 90

140 PRINT AT 10, 11; INK 2; PAPER 7; "□ MISSED! □"; AT X, Y; INK
    2; PAPER 0; "gA": BEEP 1, -10

150 LET N = N - 1

160 IF N = 0 THEN GO TO 200

170 PRINT AT 0, 24; INK 6; N

180 FOR Q = 1 TO 250: NEXT Q: PRINT AT 10, 11; "[10 spaces]"; AT
    X, Y; "□"

185 LET X = 10

```

```

190  GO TO 90

200  PRINT AT 10, 7; INK 6; PAPER 1; "YOUR SCORE IS □"; SC

205  IF SC > HS THEN LET HS = SC

210  PRINT AT 0, 18; INK 6; HS; AT 12, 3; INK 3; PAPER 7; FLASH 1;
      "HIT ANY KEY TO PLAY AGAIN"

215  IF INKEY$ < > " " THEN GO TO 215

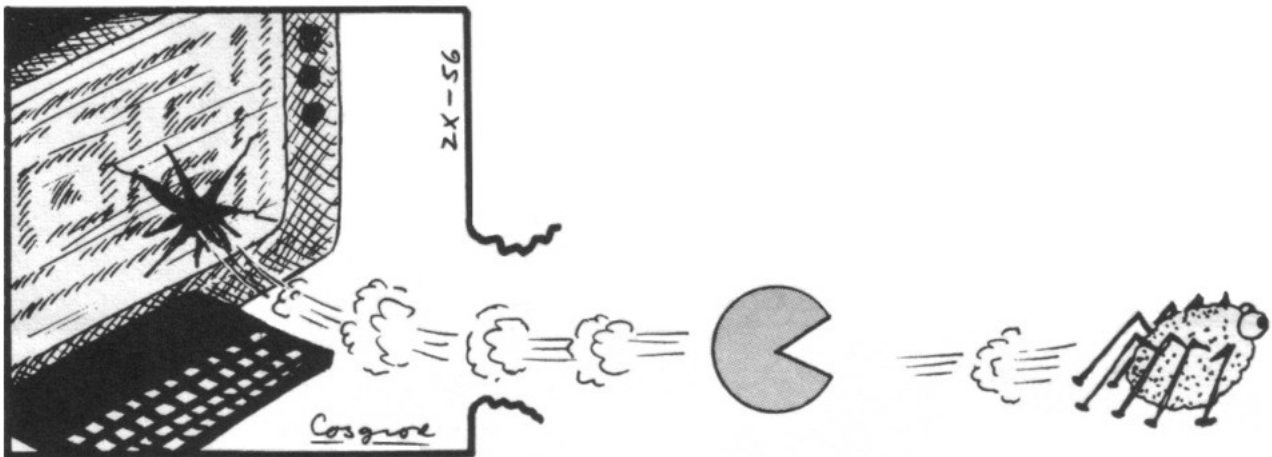
220  IF INKEY$ = " " THEN GO TO 220

230  GO TO 20

1010  DATA 126, 255, 255, 255, 255, 255, 255, 126, 0, 0, 127, 127,
      127, 127, 127, 127

1020  DATA 2, 2, 6, 6, 4, 4

```

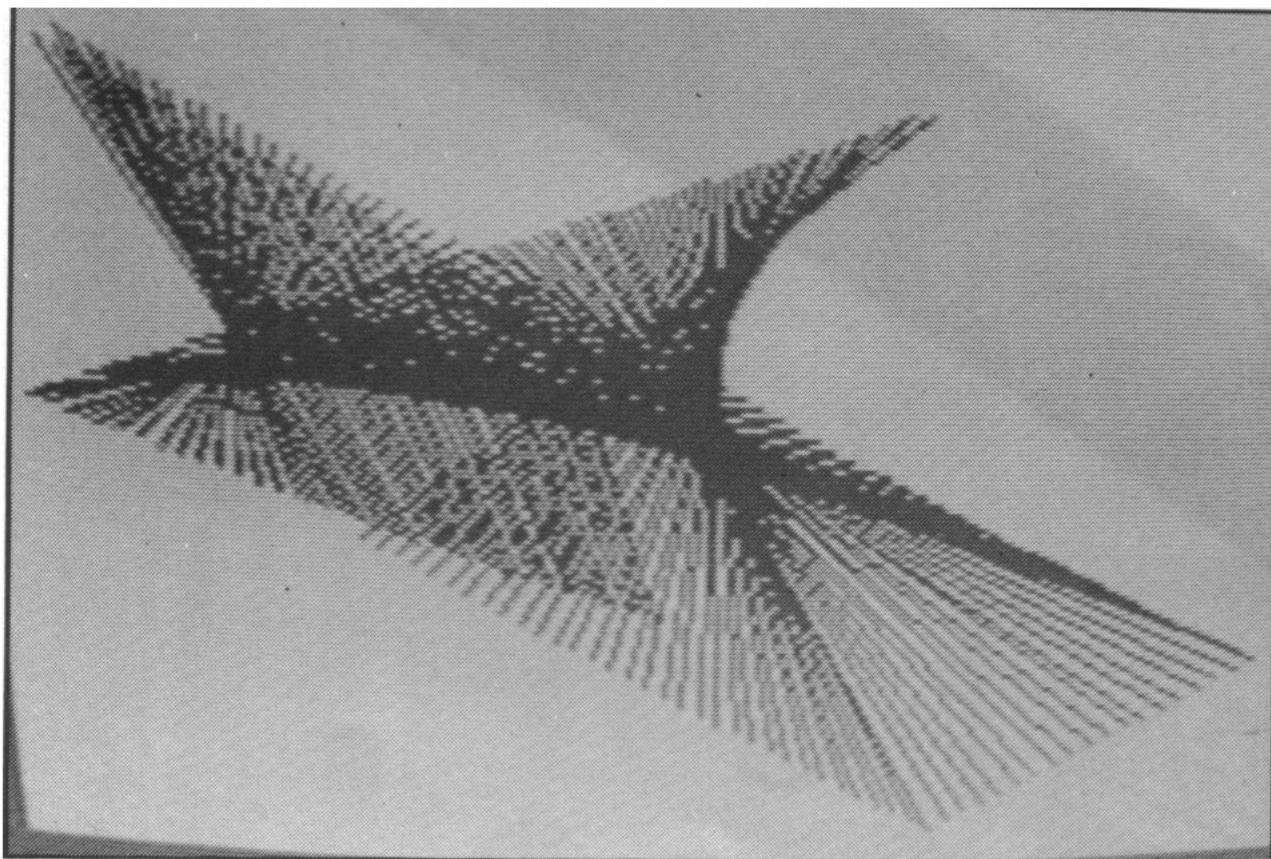


*You've read about Picasso's
Blue Period...*

2 STRING ART

This short program is a most entertaining graphical demonstration of the Spectrum's ability to generate "string art". For the uninitiated, string art involves hammering nails into a cloth-covered board, in a series of lines, and subsequently joining the nails with a coloured thread. This simple technique produces quite attractive results, considering the materials used. In this simulation, the computer draws a series of lines which gradually grow, shrink, change directions and bounce off the sides of the screen. For those who have played the arcade game *Qix*, this sort of movement of lines will be familiar.

There are no fundamental peculiarities in this program although a line which randomly changed the colour of the "thread" might improve the overall effect.



```
10  BORDER 1: PAPER 7: INK 0: CLS
20  LET X = RND * 255: LET Y = RND * 175
30  LET L = RND * 255: LET M = RND * 175
40  LET U = 4 - RND * 8
50  LET V = 4 - RND * 8
60  LET P = 4 - RND * 8
70  LET Q = 4 - RND * 8
120 FOR K = 1 TO 150
130  BEEP .02, L/5: PLOT X, Y: DRAW L - X, M - Y
140  IF L + P > 250 OR L + P < 5 THEN LET P = -P
150  IF M + Q > 170 OR M + Q < 5 THEN LET Q = -Q
160  IF X + U > 250 OR X + U < 5 THEN LET U = -U
170  IF Y + V > 170 OR Y + V < 5 THEN LET V = -V
180  LET X = X + U
190  LET Y = Y + V
200  LET L = L + P
210  LET M = M + Q
230  NEXT K
240  IF INKEY$ = " " THEN GO TO 240
250  RUN
```

Up a bit, left a bit...

3 HELICOPTER

Ever piloted a helicopter? No? Well neither have I, but here's your chance. The game's aim is to land your helicopter on the landing pad of a ship, without running out of fuel. You start with 500 gallons and this is represented by a green band. As the amount in your tanks falls, the band becomes yellow, at 200 gallons, and red at 100. If you leave the controls alone, your helicopter loses ground and its height above sea-level decreases. Try the game for yourself, using the keys 1 and 0. But don't forget that the computer cannot sense if you are pressing two keys simultaneously, so remember, one at a time! At first, you will probably sink a number of times, but as you pick up the knack, you should aim to use as little fuel as possible. Definitely, a challenging game!

Apart from the actual messages printed, all the other letters in PRINT statements are graphic symbols. If you enter lines 1000 to 1060 first, then 10 to 30 and press RUN, the graphic symbols should make more sense.

```
10 LET S = 500: RESTORE: PAPER 5: BORDER 3: CLS

15 LET F$ = "g3c [28 times]"

20 FOR A = 1 TO 7: READ A$

30 FOR B = 0 TO 7: READ C: POKE USR A$ + B, C: NEXT B

40 NEXT A

50 INK 0

60 PRINT AT 0, 0; INK 2; PAPER 7; "FUEL" ' ' ' ' ' ' ' '

70 PRINT "□ □ □ □ □ □ g4c g3 g7 [28 spaces] g2c g8c g8c g8c g1c
[24 spaces] g5 □ □ g8c gD gD gD g8c □ □ □ g7 g3 g4c [18
spaces] gB g8c HMS g8c SPECTRUM g8c [18spaces] gB g8c gD g8c
gD g8c gD g8c gD g8c gD g8c gD gA [19 spaces] gB g8c [10
times] gA"

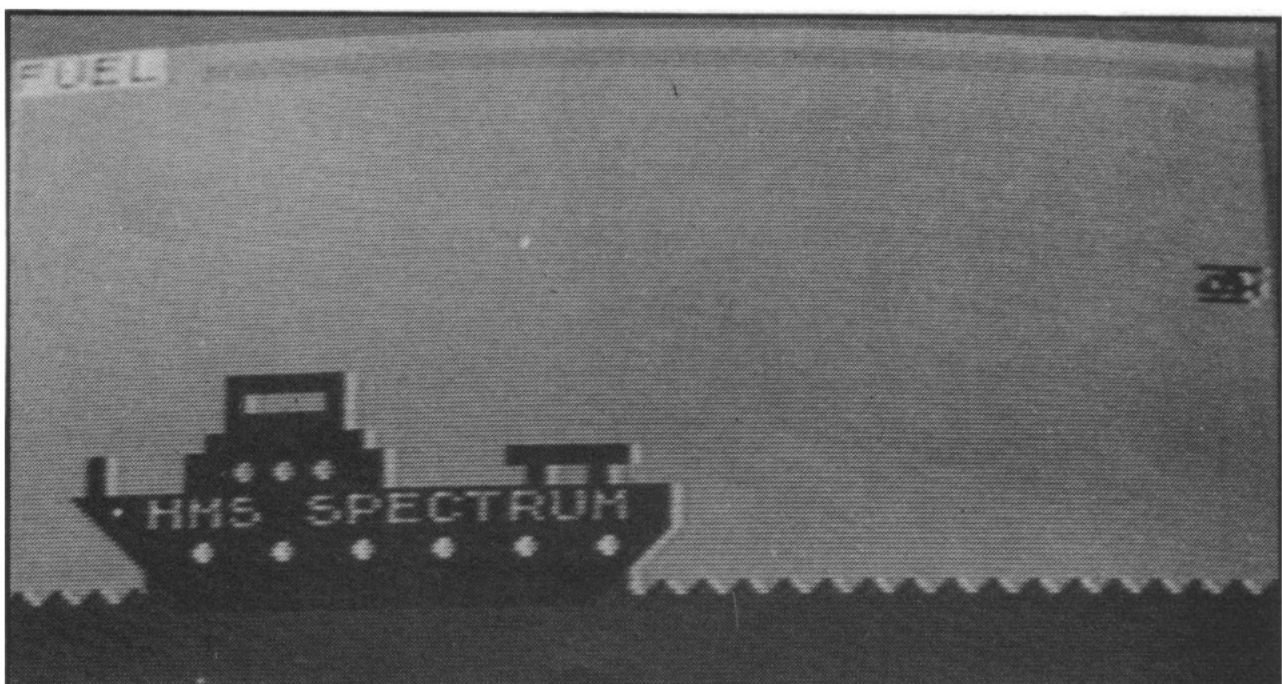
80 PRINT AT 14, 0; INK 1; "gC gC gC gC"; AT 14, 16; "gC [16
times]"; PAPER 1; INK 0; AT 14, 4; "gB"; AT 14, 15; "gA"
```



```

90  FOR A = 1 TO 7
100 PRINT PAPER 1; "[32 spaces]"
110 NEXT A
120 PRINT AT 13, 5; PAPER 6; "gD g8c gD g8c gD g8c gD g8c gD g8c
    gD"
130 LET X = 5: LET Y = 30
135 IF SCREEN$ (X, Y) < > "□" THEN GO TO 400
140 PRINT AT X, Y; "gE gF"
145 IF INT (X + .5) = 10 AND (INT Y = 14 OR INT Y = 13 OR INT Y =
    12) THEN GO TO 500
150 PAUSE 10
155 PRINT AT 14, 0; INK 1; "gG gG gG gG"; AT 14, 16; "gG [16
    times]"
160 PRINT AT 0, 5; INK 4; (CHR$ 16 AND S < 200); (CHR$ 6 AND (S <
    200 AND S > = 100)); (CHR$ 2 AND S < 100); F$ (TO S/18.6);
    "[29 spaces]": IF S = 0 THEN GO TO 600
165 LET S = S - 4
170 PRINT AT X, Y; "□ □"
175 LET X = X - (INKEY$ = "1") + (INT X = 2): LET Y = Y - (INKEY$
    = "0") 180 LET Y = Y + .25 - (INT Y = 30): LET X = X + .5

```



```

180 LET Y = Y + .25 - (INT Y = 30): LET X = X + .5
190 PRINT AT 14, 0; INK 1; "gC gC gC gC"; AT 14, 16; "gC [16
    times]"
200 GO TO 135
300 PRINT AT X, Y; INK 1; "gC gC"
310 FOR A = 1 TO 7
320 PRINT AT 14 + A, Y; PAPER 1; INK 7; "gE gF"
325 IF A > 1 THEN PRINT AT 13 + A, Y; PAPER 1; "□ □"
330 BEEP .25, 1 - A: PAUSE 10
340 NEXT A
350 PRINT AT 2, 0; "Oh dear, you've just sunk to the bottom of
    the ocean."
360 INPUT "HIT ENTER TO TRY AGAIN □"; Q$
370 RUN
400 IF INT X = 13 THEN GO TO 300
410 FOR A = 1 TO 10
420 PAUSE 5: PRINT AT X, Y; INK 2; "gE gF"
430 PAUSE 5: PRINT AT X, Y; "□ □"
440 NEXT A
450 PRINT AT X, Y - 1; INK 2; PAPER 7; BRIGHT 1; "BOOM!"
460 PRINT AT 2, 0; "That's no way to treat a heli- □ □ copter!"
470 GO TO 360
500 PRINT AT 2, 0; "Well done! You have successfully landed your
    helicopter, with □ □ □ □"; S; "□ gallons of fuel left"
510 FOR A = 1 TO 3
520 BEEP .2, 6: PAUSE 10
530 NEXT A
540 GO TO 360
600 PRINT AT 2, 0; "Sorry, you're out of fuel."

```

```
610  GO TO 360

1000  DATA "A", 255, 254, 252, 248, 240, 224, 192, 128

1010  DATA "B", 127, 63, 31, 15, 7, 3, 1, 0

1020  DATA "C", 0, 0, 0, 96, 240, 249, 255, 255

1030  DATA "D", 255, 255, 231, 195, 195, 231, 255, 255

1040  DATA "E", 127, 1, 31, 39, 100, 127, 8, 127

1050  DATA "F", 252, 128, 192, 200, 248, 200, 128, 224

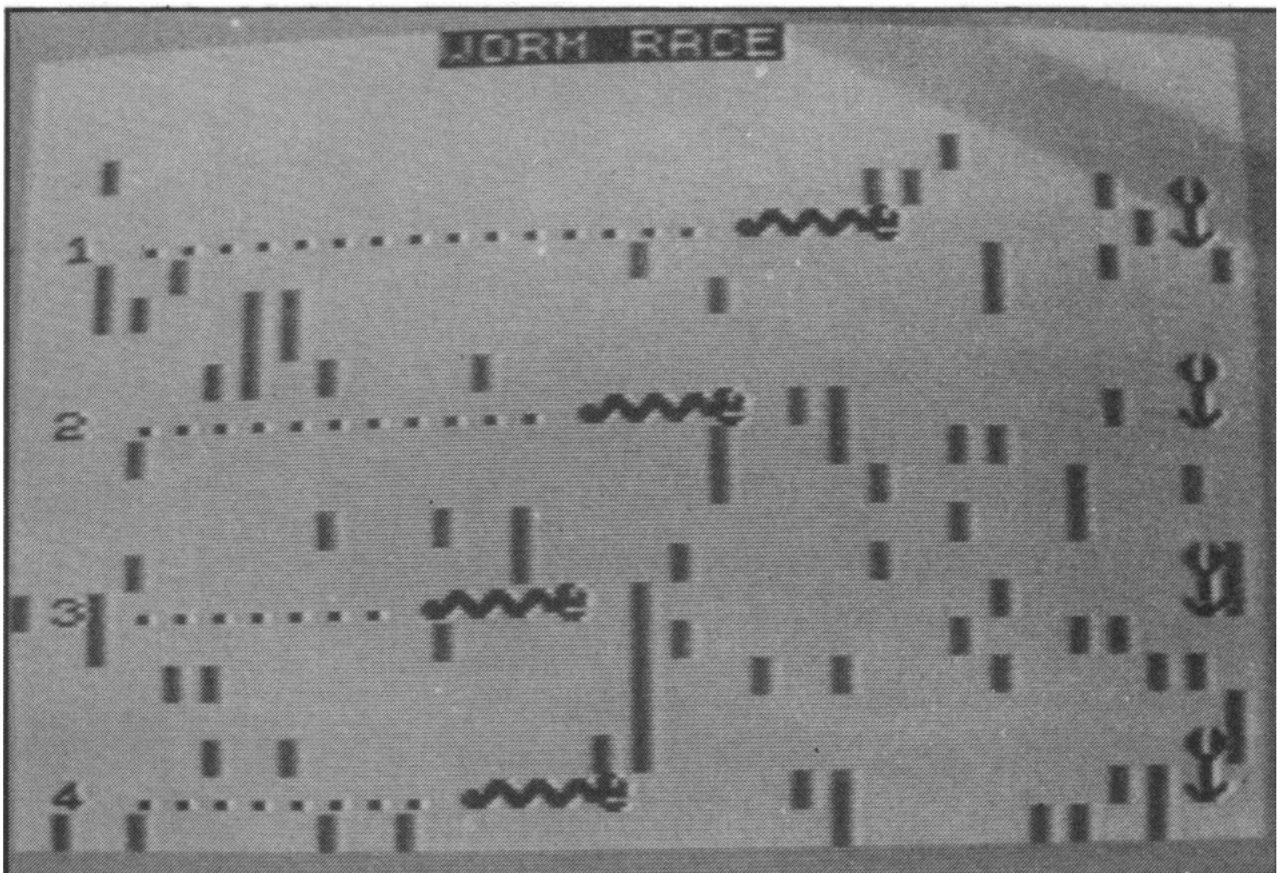
1060  DATA "G", 0, 0, 0, 6, 15, 159, 255, 255
```

You've heard of flatworms – but what about worms on the flat?

4 WORM RACE

This was the first program that I ever wrote on a computer (a ZX81 actually). It appeals to me because it is just so daft! The aim – well there isn't one really – is to back one of four worms, and watch it race across a lawn, towards its own tulip. Try giving them names of relatives/friends/teachers/bosses etc... and watch them wind their way up the lawn. The sound effects are particularly good, although simple, and give the game a whole new meaning. A night at the Worm Races is highly recommended.

See if you can work out how the sounds are produced.



```

10  DIM L (4): DIM N$ (4, 7): RESTORE: INK 1: PAPER 7: BORDER 6:
    CLS

20  PRINT AT 0, 11; INK 1; FLASH 1; "WORM RACE" ' ' '

30  FOR A = 1 TO 5

40  READ A$

50  FOR N = 0 TO 7: READ B: POKE USR A$ + N, B: NEXT N

60  NEXT A

70  FOR A = 1 TO 19

80  PRINT "[33 spaces]"

90  NEXT A

100 FOR A = 1 TO 100

110 PRINT AT 3 + INT (RND * 19), INT (RND * 32); INK 4; "g5"

120 NEXT A

130 FOR A = 1 TO 4

140 PRINT AT A * 5, 3; INK 0; "gB gA gA gA gC"; AT A * 5, 1; A;
    AT A * 5, 30; "gD"; INK 2; AT A * 5 - 1, 30; "gE"

150 INPUT "Enter name (7 letters)"; N$ (A)

160 LET L (A) = 2

170 NEXT A

180 LET M = 1 + INT (RND * 4)

190 LET L (M) = L (M) + 1

200 IF L (M) = 25 THEN GO TO 240

210 PRINT AT M * 5, L (M); INK 0; PAPER 7; ". gB gA gA gA gC"

220 BEEP .03, L (M)

230 GO TO 180

240 PRINT AT M * 5, 3; PAPER 7; "WINNER - □"; N$ (M)

250 FOR A = 1 TO 4

260 PRINT AT M * 5 - 1, 30; INK 5; "gE"

```



```
270  BEEP .5, 10
280  PRINT AT M * 5 - 1, 30; INK 2; "gE"
290  PAUSE 25
300  NEXT A
310  INPUT "HIT ENTER TO RACE AGAIN!"; Q$
320  RUN
1000 DATA "A", 0, 14, 31, 191, 251, 241, 224, 0
1010 DATA "B", 0, 0, 0, 1, 3, 3, 1, 0
1020 DATA "C", 56, 116, 118, 254, 254, 224, 62, 28
1030 DATA "D", 24, 24, 24, 24, 153, 219, 126, 60
1040 DATA "E", 36, 102, 231, 231, 231, 102, 60, 24
```

*If you don't have green fingers
then read on...*

5 FLOWER

The ability of the Spectrum to DRAW arcs quickly is rarely used in programs. FLOWER draws a flower on the screen, using this facility. By changing lines in the program, you can vary the number, size and shape of the leaves. I won't tell you which variables to change, as you can get some very interesting results by accident!

```
5   BORDER 0: PAPER 0: INK 1 + RND * 7: CLS: FOR t = 124 TO 136
    STEP 4

10  FOR a = 0 TO 2 * PI STEP 2 * PI / 12

15  LET x = t + SIN a * 80: LET y = 85 + COS a * 80

20  PLOT x, y

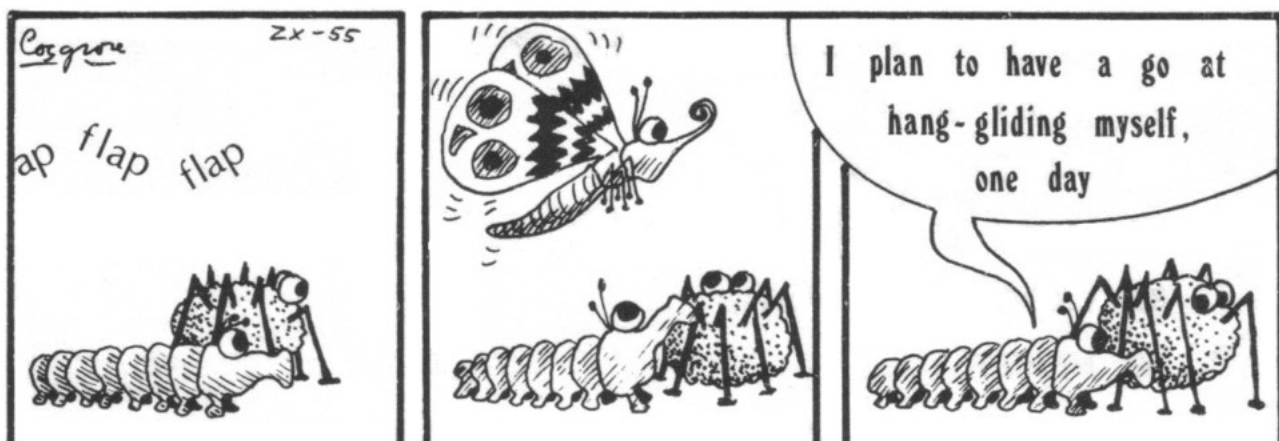
30  LET a1 = a - 2 * PI / 3

35  LET x1 = t + SIN a1 * 80: LET y1 = 85 + COS a1 * 80

40  BEEP .03, y1 / 8: DRAW x1 - x, y1 - y, - 2 * PI / 3

50  NEXT a

55  NEXT t
```



Inspired guesswork is required here...

6 MASTERMIND

This one is particularly suited for use on colour televisions, as the ability to tell the difference between a number of hues is desirable (well, necessary really). If you have never played *Mastermind* before, then here is your opportunity. Your opponent – the Spectrum – guesses a series of four colours out of the first six and by guessing (and hopefully using some powers of deduction!), you must match the computer's code in eight goes or less. For every 'peg' (pegs are used in the original game of the same name), you have in the right column, and (of course) the right colour, you get a black mark and every right colour but in the wrong place you get a white mark. The computer does not repeat any colours in its code, nor does it leave any holes blank. Confused? If so, you should find that the game unravels as you play it. If you are still having difficulty, it might be worth asking around if any friends have played the game.

There is only the graphic A used in this program, to represent a peg. This graphic (as you might expect), appears in the PRINT statements, including the inverse graphics in line 70, which are used to represent the holes that the pegs go in. To help you sort the program out, the lines 235-245 set up the four random colours and lines 310-350 work out your score.

```
10  DIM G (4): BORDER 4: INK 0: PAPER 7: CLS: RESTORE: LET X = 8

20  FOR A = 0 TO 7: READ B: POKE USR "A" + A, B: NEXT A

30  PRINT "g8C [15 times]"

40  PRINT "g8c g8c MASTERMIND! g8c g8c"

50  PRINT "g8c [15 times]"

60  FOR F = 1 TO 8

70  PRINT "g8c gA g8c gA g8c gA g8c gA g8c g8c □ □ □ □ g8c"

80  PRINT "g8c [15 times]"

90  NEXT F

220 PRINT AT 1, 16; INK 6; PAPER 1; "□"; X; "□ GOES LEFT □"
```

```

230 PRINT AT 3, 16; INK 7; PAPER 2; FLASH 1; "□ PRESS A COLOUR □"

235 DIM A (4): FOR N = 1 TO 4

240 LET A (N) = INT (RND * 6) + 1: FOR M = 1 TO 4: IF A (M) = A
    (N) AND N < > M THEN GO TO 240

245 NEXT M: NEXT N

250 FOR N = 1 TO 4

260 LET I$ = INKEY$: IF I$ = " " THEN GO TO 260

270 IF I$ < "1" OR I$ > "6" THEN GO TO 260

280 PRINT AT X * 2 + 1, N * 2 - 1; PAPER VAL I$; "gA": LET G (N)
    = VAL I$

290 IF INKEY$ < > " " THEN GO TO 290

300 NEXT N

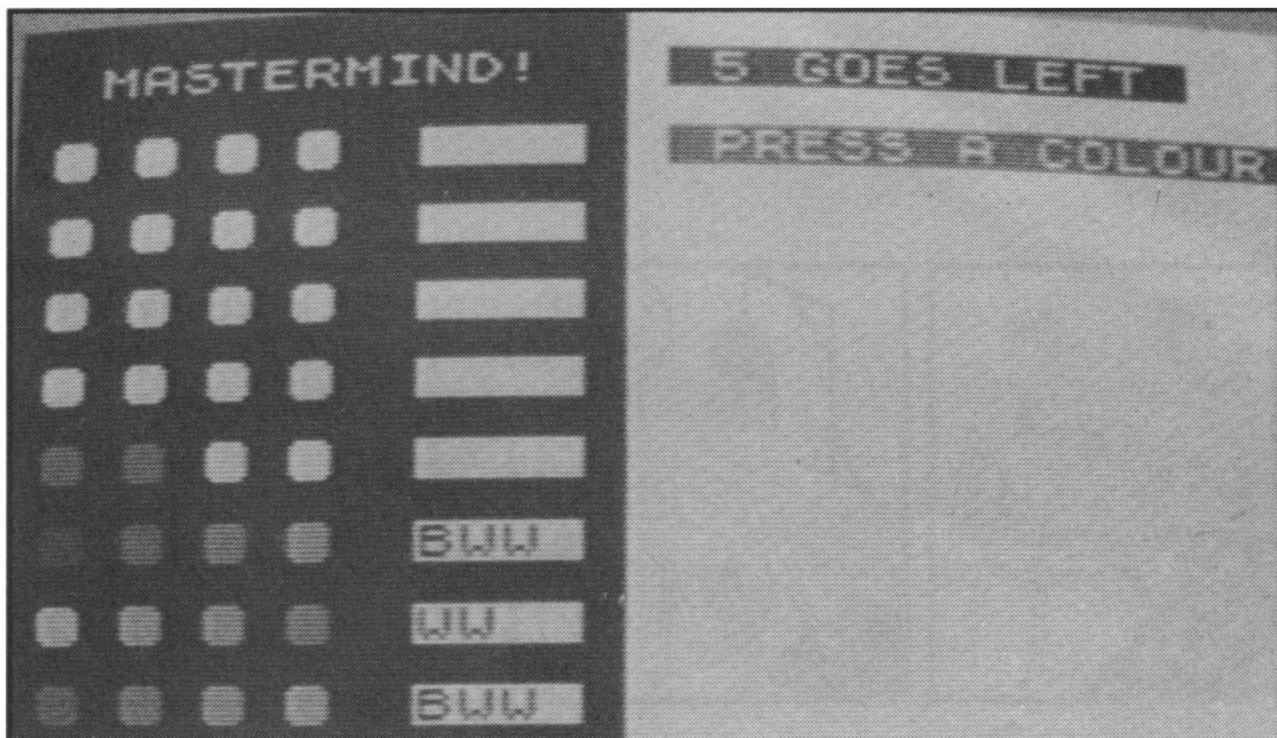
310 LET B = 0: LET W = 0

320 FOR N = 1 TO 4: IF G (N) = A (N) THEN LET B = B + 1: GO TO
    350

330 FOR M = 1 TO 4: IF N < > M AND G (N) = A (M) THEN LET W = W +
    1: GO TO 350

340 NEXT M

```



```

350  NEXT N

360  PRINT AT X * 2 + 1, 10;

370  FOR A = 1 TO B: PRINT "B"; : NEXT A

380  FOR A = 1 TO W: PRINT "W"; : NEXT A

390  IF B = 4 THEN GO TO 420

400  LET X = X - 1

405  PRINT AT 1, 17; INK 6; PAPER 1; X: IF X = 0 THEN GO TO 500

410  GO TO 250

420  PRINT AT 3, 16; INK 3; PAPER 7; "You've got it! □ □"

430  INPUT "HIT ENTER TO PLAY AGAIN □"; Q$

440  RUN

500  PRINT AT 20, 0; "THE ANSWER WAS □";

510  FOR A = 1 TO 4

520  PRINT INK A (A); "gA □";

530  NEXT A

540  GO TO 430

1000 DATA 126, 255, 255, 255, 255, 255, 255, 126

```



*A program to help you write
your own games...*

7 MONITOR

Are you a Machine Code buff or HEX freak? If so, you will find this short monitor routine useful. It displays an address in HEX, the contents of that address in HEX and allows you to alter the contents by simply entering the new code. Or, you can skim through memory, quickly, looking for a particular instruction(s). If you are not too interested in Machine Code, then this program isn't for you. If you would like to learn about Machine Code, then I suggest you pick up a copy of *"Machine Code and better Basic"* by Ian Stewart and Robin Jones (a Shiva title).

Some interesting program lines here – 90 forces a scroll and 160 and 170 perform HEX to decimal/decimal to HEX conversions. No more PEEKing, no more POKEing and no more converting laboriously to and from hexadecimal.

```
10  REM RUN WITH CAPS LOCK ON
20  BORDER 1: PAPER 1: INK 7: CLS
30  LET H$ = " ": LET A = 0
40  PRINT AT 21, A; FLASH 1; "?"
50  PAUSE 0: LET I$ = INKEY$
60  PRINT AT 21, A; I$: LET H$ = H$ + I$
70  IF A < > 3 THEN LET A = A + 1: GO TO 40
80  GO SUB 160: LET D = X
90  POKE 23692, 255: PRINT AT 21, 31; "□"
100 LET V = D: GO SUB 170
110 PRINT A$; "□ □"; : LET V = PEEK D
120 GO SUB 170
125 PRINT A$ (3 TO); "□ □";
130 PAUSE 0: LET I$ = INKEY$: IF I$ = " " THEN GO TO 130
```

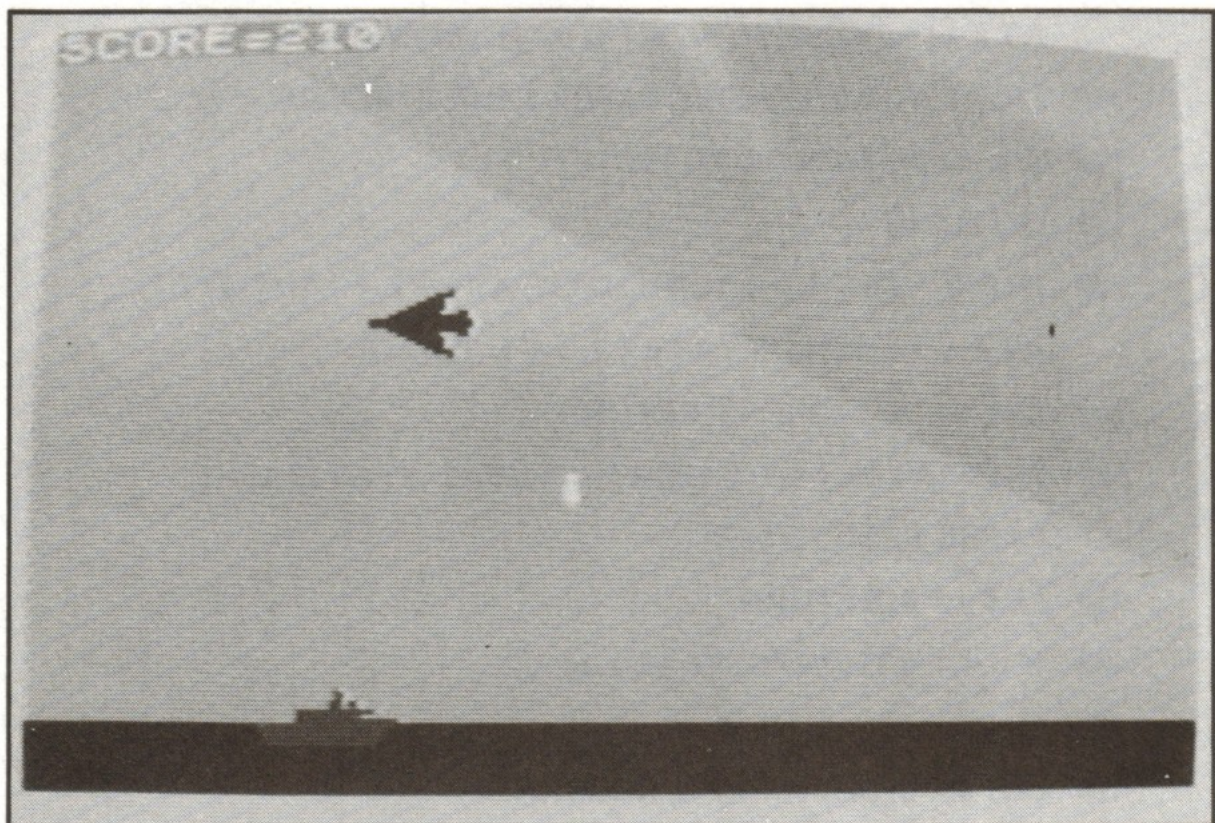

A dawn sky, the deep throb of engines...

8 BOMBER

This game is almost the opposite of HELICOPTER. Here, your aim is to bomb as many ships as you can, in the time available. One added bonus - the higher up you are when you drop a bomb, the greater the score when you hit. Using the keys 6 for down, 7 for up and 0 for fire, you have full control over the aeroplane's movements and actions.

Due to the large number of graphics characters used in BOMBER it is advisable to load the graphics by entering lines 10 to 80 and 905 to 985 and RUNning those parts. Note the use of ATTR in this program to detect whether the missile has hit anything or not. If you wish to change the sound generated when the ship blows up, then change the data in line 905.

If you are interested in writing your own "action games", it is worthwhile remembering that ATTR (a, b) can be used to detect the colour attributes of the square at (a, b), hence telling what is actually in that square. Note that line 30 uses "truth values" to move the 'plane up and down.



```

10  BORDER 6: INK 0: PAPER 5

20  CLS

30  RESTORE 905

40  FOR a = 0 TO 15: READ a$

50  FOR b = 0 TO 7: READ c

60  POKE b + USR a$, c

70  NEXT b

80  NEXT a

90  FOR a = 20 TO 21: PRINT AT a, 0; PAPER 1; "[32 spaces]": NEXT
    a

100  LET p = 0: LET x = 19: LET y = 0: LET u = 2: LET v = 29: LET
    a = 0: LET b = 0: LET s = 0

115  LET p = p + 1

120  PRINT AT x, y; INK 5; "□"; INK 2; "□ gB gC gD"; AT x + 1, y;
    PAPER 1; INK 2; "□ gE gF gF gH"

130  LET u = u + (INKEY$ = "6" AND u < 10) - (INKEY$ = "7" AND u >
    2): LET v = v - 1

140  IF v = 0 THEN GO SUB 200

150  LET y = y + 1: IF y = 27 THEN GO SUB 300

155  IF INKEY$ = "0" THEN GO SUB 400

157  IF b < > 0 THEN GO SUB 500

160  PRINT AT u - 1, v; "□ □ □ □"; AT u + 2, v; "□ □ □ □"; AT u,
    v; "gL gJ gK □"; AT u + 1, v; "gL gM gN □"

165  PRINT AT 0, 0; FLASH 1; PAPER 4; INK 7; "SCORE ="; s

170  IF p < 600 THEN GO TO 115

180  GO TO 800

200  PRINT AT u - 1, v + 1; "□ □ □ □"; AT u + 2, v + 1; "□ □ □ □";
    AT u, v + 1; "□ □ □"; AT u + 1, v + 1; "□ □ □"

210  LET v = 29: LET u = 2: RETURN

```

```

300  PRINT AT 19, y - 1; PAPER 5; "□ □ □ □ □"; AT 20, y - 1; PAPER
    1; "□ □ □ □ □"

310  LET y = 0: RETURN

400  IF b < > 0 THEN PRINT AT a, b; "□"

405  LET d = u

410  LET a = u: LET b = v + 1: RETURN

500  LET a = a + 1: IF a = 21 THEN LET b = 0: RETURN

503  PRINT AT a - 1, b; "□": IF a = 20 AND ATTR (a, b) < > 15 THEN
    RETURN

505  IF ATTR (a, b) = 42 THEN GO TO 530

510  PRINT AT a, b; INK 7; "gP"

515  BEEP .05, - a / 10

520  RETURN

530  PRINT AT a, b; ".": FOR z = 0 TO 20: NEXT z: PRINT AT a, b;
    "gR": FOR z = 0 TO 20: NEXT z: PRINT AT a, b; "gS": FOR z = 0
    TO 20: NEXT z

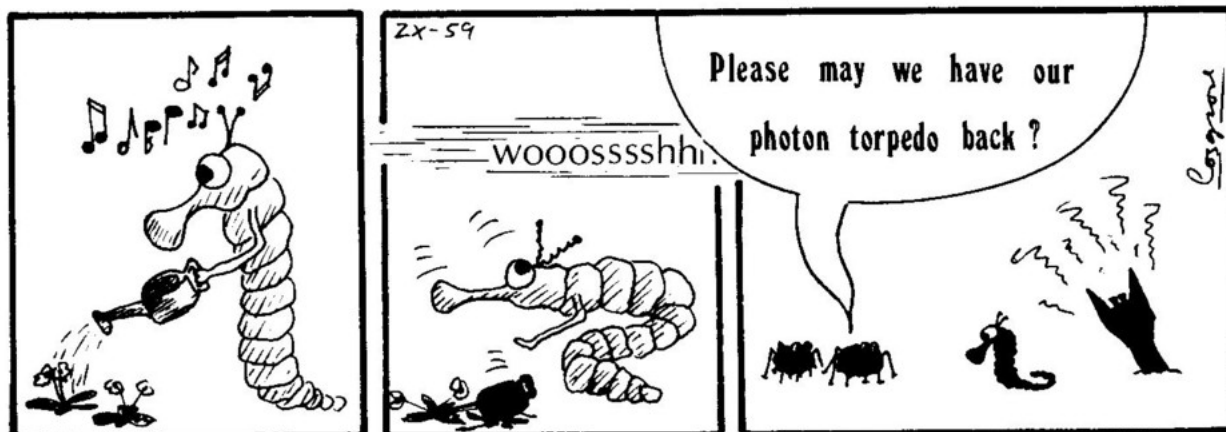
535  RESTORE 900: FOR z = 0 TO 5: READ c: BEEP .5, c: NEXT z

537  PRINT AT x, y; PAPER 5; "□ □ □ □ □"; AT x + 1, y; PAPER 1; "□
    □ □ □ □": LET y = 0

539  RESTORE

540  LET b = 0: LET s = s + (10 * (11 - d)): RETURN

```



```
800  FOR a = 0 TO 2: PAUSE 25: BEEP .5, -5: NEXT a: INPUT "HIT
    ENTER TO TRY AGAIN! □"; a$: CLS: GO TO 90

900  DATA 35, 20, 35, 20, 35, 20

905  DATA "b", 0, 0, 0, 0, 0, 63, 63, 63

915  DATA "c", 32, 48, 48, 118, 118, 255, 255, 255

920  DATA "d", 0, 0, 0, 0, 0, 224, 0, 255

925  DATA "e", 63, 63, 31, 31, 15, 0, 0, 0

930  DATA "f", 255, 255, 255, 255, 255, 0, 0, 0

935  DATA "h", 254, 252, 248, 240, 224, 0, 0, 0

940  DATA "i", 0, 0, 0, 0, 0, 3, 15, 255

945  DATA "j", 0, 3, 15, 63, 255, 254, 255, 255

950  DATA "k", 64, 192, 0, 0, 0, 24, 248, 252

955  DATA "l", 255, 15, 3, 0, 0, 0, 0, 0

960  DATA "m", 255, 255, 254, 255, 63, 15, 3, 0

965  DATA "n", 252, 248, 24, 0, 0, 0, 192, 64

970  DATA "p", 0, 56, 16, 56, 56, 56, 56, 16

975  DATA "q", 0, 0, 0, 24, 24, 0, 0, 0

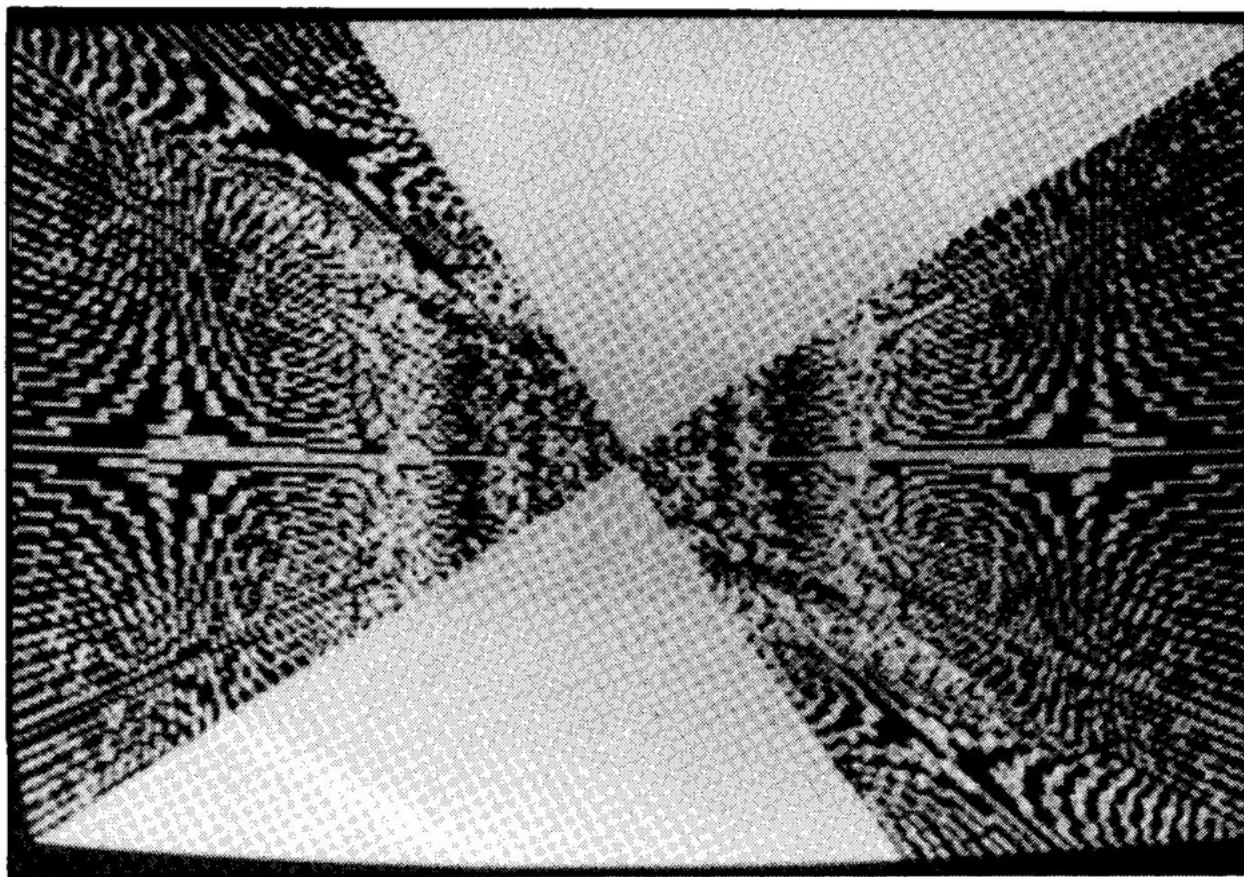
980  DATA "r", 0, 0, 16, 56, 28, 8, 0, 0

985  DATA "s", 0, 144, 86, 60, 60, 88, 16, 0
```

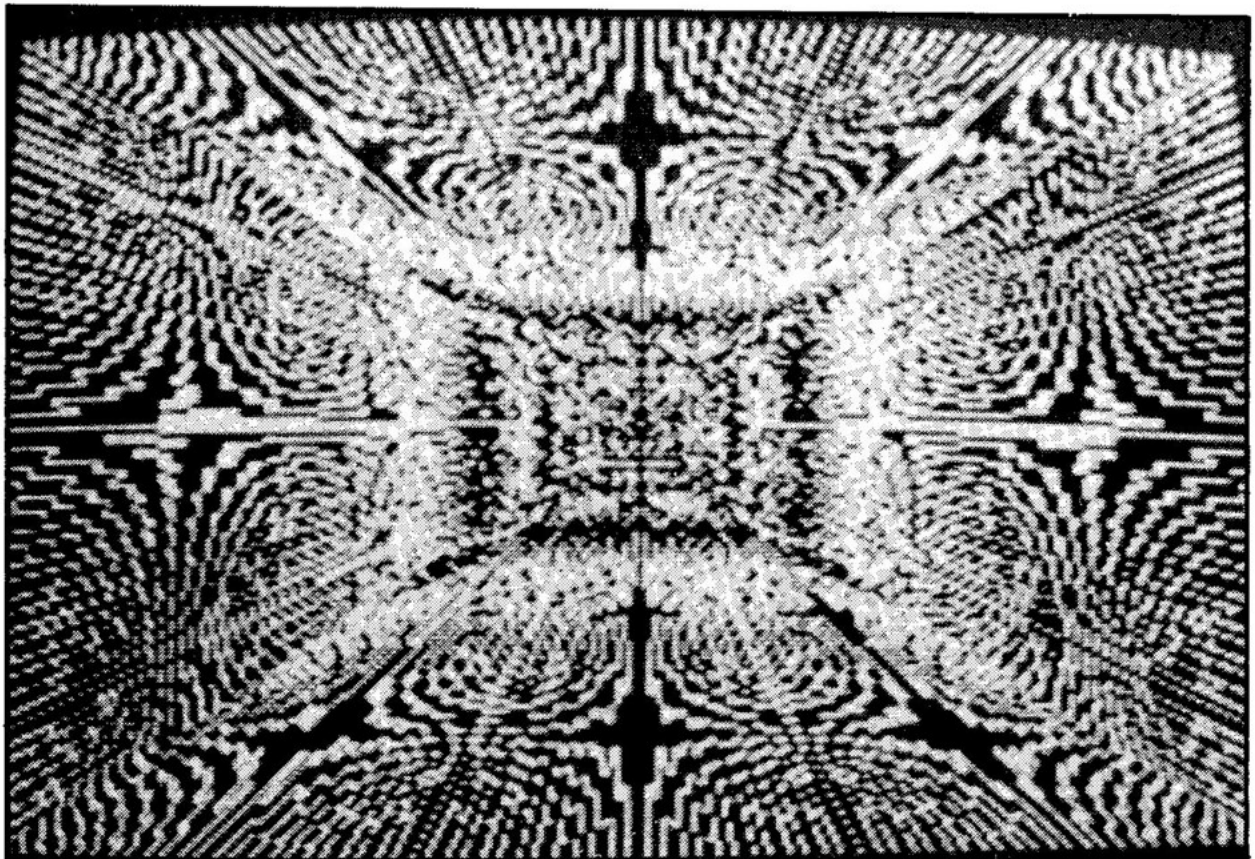

*Hmmm...Spectrum...Spectrum...there must
be a better name.*

9 KALEIDOSCOPE

This program is a fabulous demonstration of the graphics capabilities of the Spectrum. The patterns generated by KALEIDOSCOPE rely upon the fact that the Spectrum *cannot* draw straight lines! The OVER 1 command is used, so that when any dots are drawn over each other, they disappear. These two features enable you to DRAW OVER other lines that the Spectrum has already drawn, producing some amazing effects. I was going to call this program OVERDRAWN, but it might have reminded me about my bank account!



```
10  INK 1 + RND * 7: PAPER 0: BORDER 0: OVER 1: CLS
15  LET R = 1 + .5 * RND: FOR A = 1 TO 22: PRINT "g8c [32
    times]": NEXT A
20  FOR A = -87 TO 87 STEP R
30  BEEP .02, A/3: PLOT 128, 88: DRAW 127, -A: PLOT 128, 88: DRAW
    -128, A
40  NEXT A
50  FOR A = 127 TO -127 STEP -R
60  BEEP .02, A/3: PLOT 128, 88: DRAW A, -88: PLOT 128, 88: DRAW
    -A, 87
70  NEXT A
```



For future reference...

10 CUSTOMER

Strictly speaking, this is not a game, although it provides an interesting demonstration of how a computer can be used to handle "files". What, I can hear you say, is a file? A file as you or I know it is a box in which information is held. In a similar way, the Spectrum can store information on cassette, using data files. In this example (which can be easily modified for virtually any purpose), a 128 byte file is stored on cassette, with the following information: Name, address, telephone number, order number, notes, and date of order. Although designed for a small business, you may want to store friends and relatives 'phone numbers and addresses.

Note: F\$ (128) is split up within the program to store 8 "fields". Each field is assigned to a particular part of the file, so, for example, field 2 is the first line of the address. Entry of the program is straightforward.

When using this program, the date is automatically SAVED with the file, and when asked for "NEXT CUSTOMER NO.:", if you simply press "ENTER", you will get the number following the previous number.

```
10 LET D = 0: INK 7: PAPER 1: BORDER 1

15 CLS: DIM F$ (128)

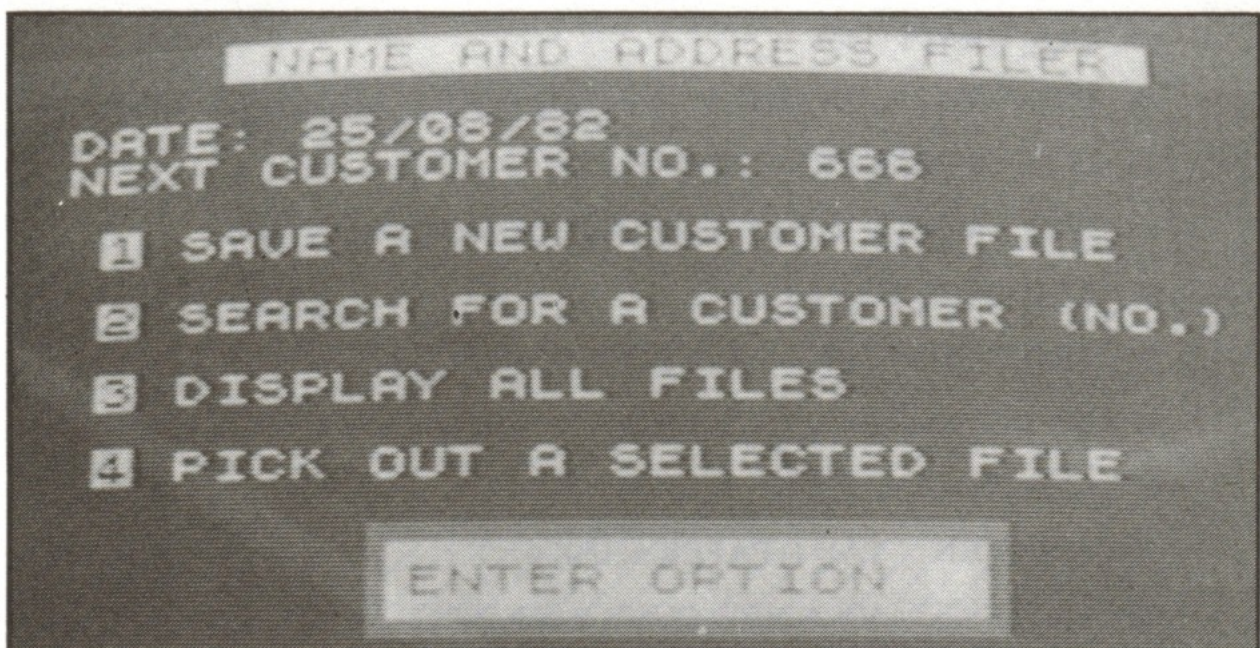
20 PRINT TAB 4; BRIGHT 1; "g8c NAME AND ADDRESS FILER g8c"

30 IF D = 0 THEN PRINT "DATE:  SAVE A NEW CUSTOMER FILE"
```

```

90  PRINT ' TAB 1; PAPER 6; INK 0; "2";
100 PRINT "□ SEARCH FOR A CUSTOMER (NO.)"
110 PRINT ' TAB 1; PAPER 6; INK 0; "3";
120 PRINT "□ DISPLAY ALL FILES"
130 PRINT ' TAB 1; PAPER 6; INK 0; "4";
140 PRINT "□ PICK OUT A SELECTED FILE"
150 PRINT INK 2; PAPER 6; AT 13, 8; "g4c g3 [14 times] g7"; AT
    14, 8; "g5c ENTER OPTION □ □ g5"; AT 15, 8; "g1c g3c [14
    times] g2c"
160 PRINT AT 14, 22; INK 4; PAPER 7; "?"
165 PAUSE 15
170 IF INKEY$ < > " " THEN GO TO 210
180 PRINT AT 14, 22; INK 4; PAPER 7; "□"
185 PAUSE 15
190 IF INKEY$ < > " " THEN GO TO 210
200 GO TO 160
210 LET I$ = INKEY$
220 IF I$ < "1" OR I$ > "4" THEN GO TO 160

```



```

225  LET D = 1

230  CLS

240  GO TO 1000 * VAL I$

1000  RESTORE 9000: PRINT "CREATE A NEW FILE" ' "g3 [17 times]"

1005  LET A$ = "No □" + STR$ CN

1010  LET A = USR "A"

1020  FOR B = 0 TO 7

1030  READ C: POKE A + B, C

1040  NEXT B

1050  LET A = USR "B"

1060  FOR B = 0 TO 7

1070  READ C: POKE A + B, C

1080  NEXT B
1085  PRINT "CUSTOMER NO.: □"; CN

1090  PRINT ' ' "NAME:" ' TAB 9; "gA [20 times] gB" ' "ADDRESS:" '
      TAB 9; "gA [20 times] gB" ' ' TAB 9; "gA [20 times] gB" ' '
      TAB 9; "gA [20 times] gB"

1100  PRINT "PHONE:" ' TAB 9; "gA [15 times] gB"

1110  PRINT "ORDER:" ' TAB 9; "gA gA gA gB" ' "NOTES:" ' TAB 9; "gA
      [22 times] gB"

1120  RESTORE 9020

1125  LET T = 0: LET L = 8

1130  FOR A = 0 TO 6

1135  LET T = T + L - 8

1140  READ L

1150  LET R = 5 + 2 * A: LET C = 9

1160  GO SUB 9500

1170  IF CODE I$ = 12 THEN PRINT AT R, C; "□": LET C = C - 1: GO TO
      1160

```

```

1180  IF CODE I$ = 13 THEN GO TO 1230

1190  LET F$ (T + C - 8) = I$

1200  IF C = L THEN GO TO 1230

1210  LET C = C + 1

1220  GO TO 1160

1230  NEXT A

1240  SAVE A$ DATA F$ ( )

1250  LET CN = CN + 1

1260  GO TO 15

2000  INPUT "ENTER CUSTOMER NUMBER: "; N

2010  PRINT ' "SEARCHING FOR CUSTOMER "; FLASH 1; N

2020  LOAD "No " + STR$ N DATA F$ ( )

2040  GO SUB 9100

2050  IF INKEY$ = " " THEN GO TO 2050

2060  GO TO 15

3000  PRINT ' "SEARCHING"

3005  INK 1: PAPER 7

3010  LOAD " " DATA F$ ( )

3020  CLS: PRINT ' "NAME: "; F$ (TO 20) ' ' "ADDRESS: "; F$ (21
    TO 40) ' TAB 9; F$ (41 TO 60) ' TAB 9; F$ (61 TO 80) ' '
    "PHONE: "; F$ (81 TO 95) ' ' "ORDER: "; F$ (96 TO 98) ' '
    "NOTES: "; F$ (99 TO 120) ' ' "DATE OF ORDER: "; F$ (121 TO
    128)

3030  GO TO 3010

4000  PRINT ' "WHICH FIELD? (1 TO 8)"

4005  INPUT "FIELD: "; F

4010  RESTORE 9020

4015  LET T = 0

4020  FOR A = 1 TO F: READ L: LET T = T + L - 8: NEXT A

```

```

4030 DIM T$ (L - 8)

4040 PRINT ' "□ ENTER FIELD □"; F; "□ TEXT"

4050 INPUT T$ (TO L - 8)

4060 PRINT ' "□ SEARCHING FOR FIELD □"; FLASH 1; F; FLASH 0; ' '
      TAB 1; FLASH 1; T$

4070 LOAD " " DATA F$ ( )

4080 IF F$ (T - L + 9 TO T) < > T$ THEN GO TO 4090

4085 CLS: PRINT ' "NAME: □"; F$ (TO 20) ' ' "ADDRESS: □"; F$ (21
      TO 40) ' TAB 9; F$ (41 TO 60) ' TAB 9; F$ (61 TO 80) ' '
      "PHONE: □"; F$ (81 TO 95) ' ' "ORDER: □"; F$ (96 TO 98) ' '
      "NOTES: □"; F$ (99 TO 120) ' ' "DATE OF ORDER: □"; F$ (121 TO
      128)

4090 GO TO 4070

8999 STOP

9000 DATA 128, 255, 0, 0, 0, 0, 0, 0

9010 DATA 128, 128, 0, 0, 0, 0, 0, 0

9020 DATA 28, 28, 28, 28, 23, 11, 35

9100 CLS: PRINT "CUSTOMER NUMBER □"; N ' ' "NAME: □"; F$ (TO 20) '
      ' "ADDRESS: □"; F$ (21 TO 40) ' TAB 9; F$ (41 TO 60) ' TAB 9;
      F$ (61 TO 80) ' ' "PHONE: □"; F$ (81 TO 95) ' ' "ORDER:
      □"; F$ (96 TO 98) ' ' "NOTES: □"; F$ (99 TO 120) ' ' "DATE OF
      ORDER: □"; F$ (121 TO 128)

9110 RETURN

9500 PRINT AT R, C; INK 6; "*"

9510 PAUSE 15

9520 LET I$ = INKEY$: IF I$ < > " " THEN GO TO 9560

9530 PRINT AT R, C; "□"

9540 PAUSE 15

9550 LET I$ = INKEY$: IF I$ < > " " THEN GO TO 9560

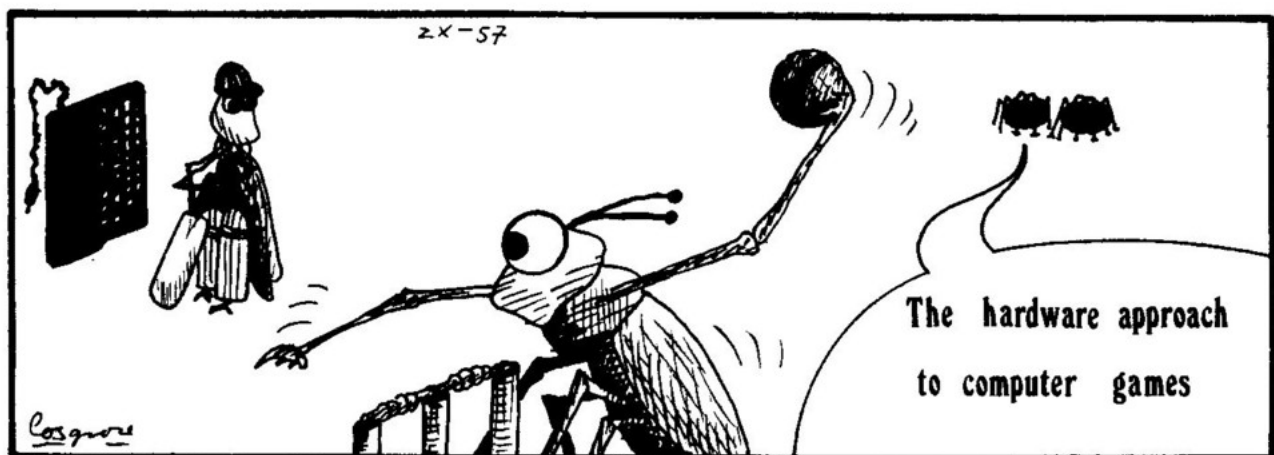
9555 GO TO 9500

9560 IF CODE I$ = 13 THEN PRINT AT R, C; "□"

```

9565 PRINT AT R, C; I\$

9570 RETURN



*Ever felt that you're going
round in circles?*

11 SPIRAL

This most interesting program draws all sorts of magnificent designs. Don't underestimate this program by its length. Enter it and RUN it. Enter your own values; try 3, 4, 6, 8, to see how it works – press v to clear the screen and start again or any other key to start without clearing the screen. Now, try 3.5, 6.66666666, 2.9, 4.95 and any value.

Q: What is the difference between 4.95 and 4.05?
See if you can work out why this happens.

```
10  BORDER 0: PAPER 0: INK 1 + RND * 7: CLS
15  LET r = 80
20  INPUT n
25  LET x = 128: LET y = r + 85
30  PLOT x, y
40  FOR b = 0 TO 1000
50  LET a = PI * 2 / n * b
55  LET x1 = 128 + SIN a * r: LET y1 = 85 + COS a * r
60  DRAW x1 - x, y1 - y: BEEP .05, (80 - R) / 3
65  LET r = r - 1: IF r < 5 THEN GO TO 80
70  LET x = x1: LET y = y1
75  NEXT b
80  LET a$ = INKEY$: IF a$ = " " THEN GO TO 80
95  IF a$ = "v" THEN RUN
100 GO TO 15
```

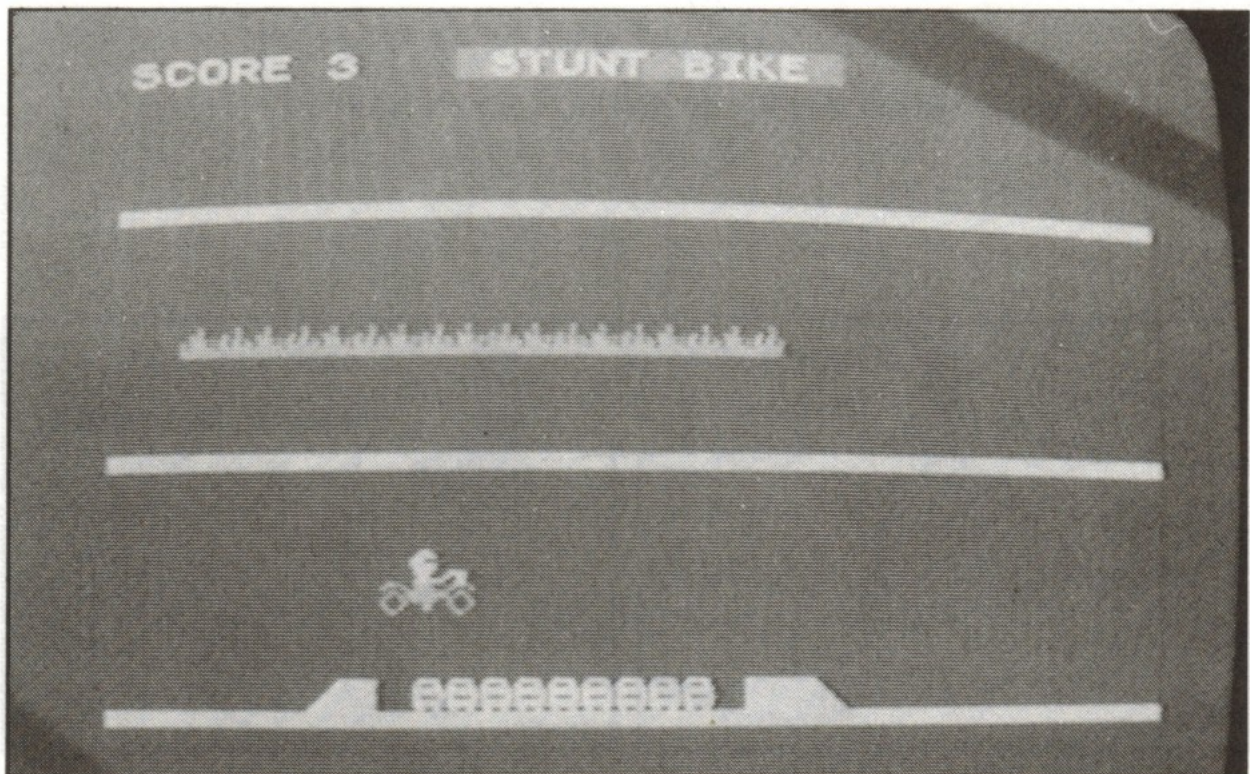

Can you out-jump Evel Knievel?

12 STUNT BIKE

STUNT BIKE is a game of skill, where you race through fire-tunnels, jump buses and pull "wheelies" whilst accelerating hard. Use the keys 1 to brake and 0 to accelerate. There are two specific hazards – do not try to pull wheelies in the fire-tunnel, or you will crash, and you must take your hand off the throttle, well before hitting the take-off ramp or you will lose height!

As with the other games, get the graphics loaded first and enter graphics characters in lines 105, 110, 130, 150, 155, 160 and 240.

A number of techniques are used in this program that you may not have seen before. The GO TO in line 155 jumps to 170 if you have not hit the roof of the fire-tunnel and 300 if you have. The LET command in 170 uses logical values of 0 and 1 to alter the bike's acceleration according to which keys you press. Finally, the most unusual lines are 272 and 300. The PRINT#1 prints on top of the bottom lines, as does PRINT#0, PRINT#2 prints normally and PRINT#3 sends output to the ZX Printer. This is the ZX Spectrum's stream mechanism, and the PRINT# command is really intended for use with the RS 232/net interface board.



If you wish to change the function of any of these commands, you can, using OPEN#n, a\$, where n is a number 0-15 and a\$ is K, S or P according to whether the peripheral attached to stream n is a keyboard, screen or printer. If you are not quite sure how the stream facility works, I suggest you experiment for yourself.

```
10  FOR A = 0 TO 95

20  READ B: POKE USR "A" + A, B

30  NEXT A

40  FOR B = 1 TO 9

50  LET V = 0: LET R = 0: LET X = 4: LET Y = 1

60  BORDER 0: PAPER 0: INK 7: BRIGHT 1

70  PRINT AT 0, 0; "SCORE □"; B - 1; TAB 10; PAPER 1; INK 6; "□
    STUNT BIKE □"

80  FOR A = 0 TO 31

90  PRINT AT 5, A; INK 4; "g3"; AT 13, A; "g3"; AT 21, A; "g3"

100 NEXT A

105 FOR A = 0 TO B + 4: PRINT AT 9, A * 2 + 2; INK 2; "gK gL":
    NEXT A

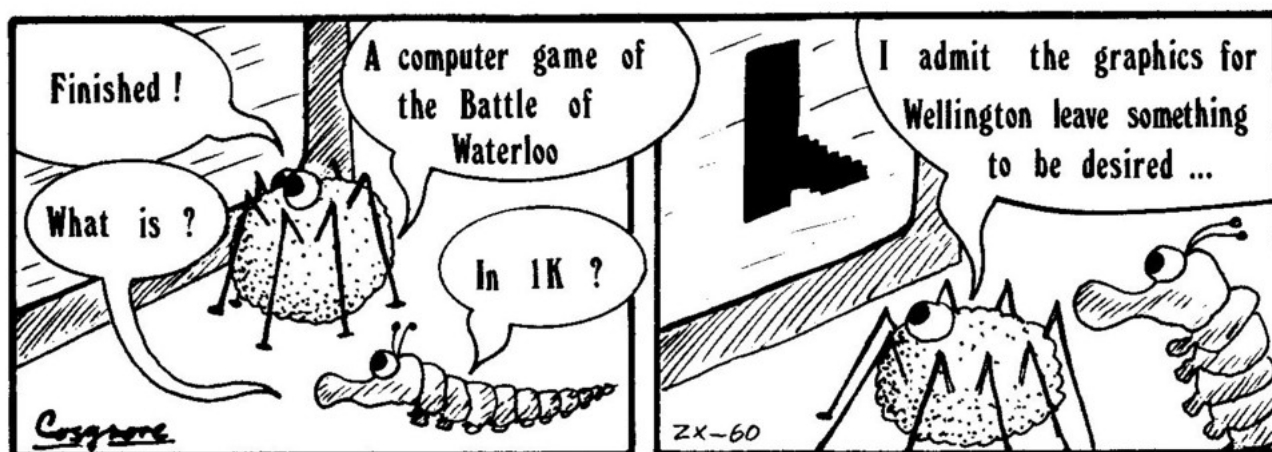
110 PRINT AT 20, 6; INK 4; "gG g8c □";

120 FOR A = 1 TO B + 5

130 PRINT INK 5; "gF";

140 NEXT A

150 PRINT INK 4; "□ g8c g8c gH"
```



```

155 IF R > .7 THEN PRINT AT X - 2, Y; INK 6; "□ gC gE"; AT X - 1,
    Y - 1; "□ gJ gI gA"; AT X, Y - 1; "□ gB □ □": GO TO 170 + 130
    * (X = 12 AND Y < B * 2 + 10.5)

160 PRINT AT X - 1, Y - 1; INK 6; "□ □ gC gE"; AT X, Y - 1; "□ gB
    gD gA"; AT X - 2, Y; "□ □ □"

170 LET R = R + .2 * (INKEY$ = "0") - .1 * (INKEY$ = "1" AND V >
    0) - .12 * (V > 0): LET V = V + R: IF V > 10 THEN LET V = 10

175 IF V < 0 THEN LET V = 0

180 LET Y = Y + V / 10: IF INT Y = 28 THEN PRINT AT X - 2, Y; "□
    □ □"; AT X - 1, Y - 1; "□ □ □ □"; AT X, Y - 1; "□ □ □ □":
    LET X = X + 8: LET Y = 1

190 BEEP .01, V * 3 - 20: IF INT (Y + .5) = 4 AND X = 20 THEN GO
    TO 220

200 IF X < 27 THEN GO TO 155

210 NEXT B: CLS: PRINT "CONGRATULATIONS! YOU HAVE JUST □ □
    CLEARED ALL 15 BUSES ON YOUR □ □ □ □ STUNT BIKE, AND
    QUALIFIED FOR A PLACE IN THE GUINNESS BOOK OF □ □ □
    RECORDS!": STOP

220 PRINT AT 18, 3; "□ □ □" ' "□ □ □ □ □ □ □" ' "□ □ □ □ □ □":
    IF R < .2 THEN LET R = .2

225 LET X = X - V / 7: IF X < 16 THEN LET X = 16

230 LET V = V - R / 2 - .42: LET Y = Y + 1: LET X = X + .3

235 IF X > = 20 THEN GO TO 270

240 PRINT AT X - 1, Y - 1; INK 6; "□ □ gC gE"; AT X, Y - 1; "□ gB
    gD gA"; AT X - 2, Y; "□ □ □"

245 BEEP .03, 10 - 3 * (X - 15)

250 PRINT AT X - 1, Y - 1; "□ □ □ □"; AT X, Y - 1; "□ □ □ □"; AT
    X - 2, Y; "□ □ □"

260 GO TO 225

270 IF Y > B + 17 THEN GO TO 210

272 PRINT#1; PAPER 2; FLASH 1; "You have crashed into the Buses
    Hit ENTER to play again □ □ □ □ □ □ □ □"

280 IF INKEY$ = CHR$ 13 THEN RUN

```

```
290  GO TO 305

300  PRINT#1; PAPER 2; FLASH 1; "You have crashed in the Fire □ □
    □ □ Tunnel. □ □ Hit ENTER to play again"

305  IF INKEY$ = CHR$ 13 THEN RUN

310  GO TO 305

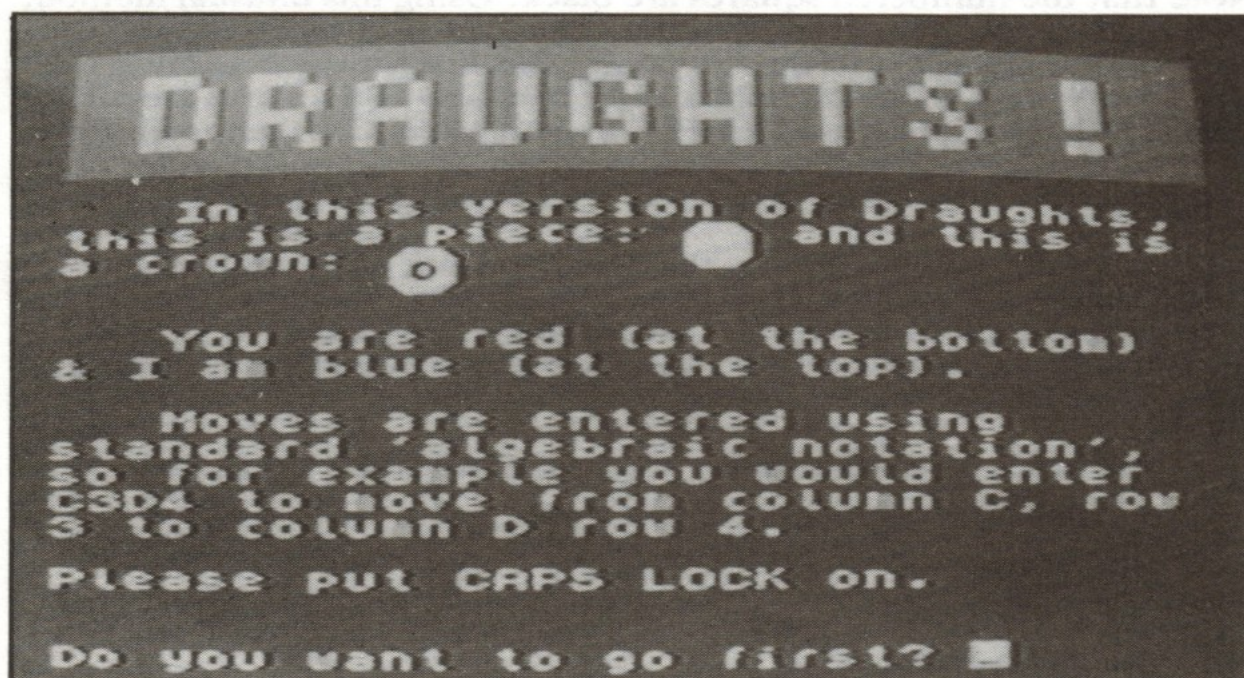
1000 DATA 60, 66, 152, 164, 66, 66, 36, 24
1010 DATA 60, 66, 25, 37, 66, 66, 36, 24
1020 DATA 56, 124, 224, 252, 124, 120, 48, 127
1030 DATA 120, 124, 255, 255, 252, 24, 24, 0
1040 DATA 0, 0, 0, 0, 0, 120, 152, 24
1050 DATA 62, 66, 127, 127, 65, 127, 127, 34
1060 DATA 1, 3, 7, 15, 31, 63, 127, 255
1070 DATA 128, 192, 224, 240, 248, 252, 254, 255
1080 DATA 63, 127, 254, 252, 248, 252, 236, 192
1090 DATA 0, 0, 0, 1, 3, 7, 15, 31
1100 DATA 8, 8, 40, 61, 189, 191, 255, 255
1110 DATA 4, 4, 36, 109, 109, 127, 255, 255
```

*As the weather draws in,
sit round the fire and play a game of...*

13 DRAUGHTS

Draughts, or checkers as it is called outside the UK, is a board game for two players. It was played in Southern Europe in medieval times and appears to have been derived from much older games played in the Middle East. The objective of the game is to take all of the Spectrum's pieces, or to get the Spectrum to concede by trapping its pieces. The rest of the instructions are in the game.

This program takes up nearly all the program space in a 16K machine, and as a result, has no "look ahead" or real "move evaluation" that an intelligent game like Draughts deserves. As a result, the standard of play of the Spectrum is abysmally low — although it makes an entertaining opponent. If you own a 48K Spectrum, try including some real move logic. As the author of ZX SPECTRUM CHESS, Published by Prism Computing, I would recommend you start by incorporating a routine to evaluate the present board situation, i.e. how strong your position is now, and how would each possible move put you in danger. In order to do this, you would need a separate piece-move routine and an evaluation routine.



Back to the game – entry is straightforward, with graphics used to represent pieces. Note that a crown, or king, is generated automatically when you reach the opposite side. Although the rules of the game say that you must capture if you can, the Spectrum does not check for possible capture of its pieces, so it does not force you to move, but it always captures if it can.

The program is subdivided into a number of blocks and each one has a title REM telling you what it does. The only other thing you really need to know is how the board is numbered. It looks like this:

	72		71		70		69
66		65		64		63	
	59		58		57		56
53		52		51		50	
	46		45		44		43
40		39		38		37	
	33		32		31		30
27		26		25		24	

Note that the numbered squares are black. Using this unusual method, diagonally opposite pieces have the value of 6 or 7 more or less, than each other, according to the direction. This is what the X array is used for. The A array is used to hold a 10 by 10 board – 36 squares are off the edges. Other variables used are H = human piece, C = computer piece, W = human crown and K = computer crown, B = off board. Also, LS = lose flag, CS = computer score, HS = human score and MO is a move flag.

It should be fairly easy from the above to work out exactly what each part of the program does. Q: Where is the multiple jump routine and the routine that guesses a move if it cannot find a good one?

If you add extra lines to make the game a bit of a challenge, I would be *most* interested to hear from you!

```

10  BORDER 0: POKE 23624, 70: OVER 0: BRIGHT 1: INVERSE 0: PAPER
    0: INK 6: CLS

20  LET BD = 0

30  GO SUB 6000

40  GO SUB 4000

50  GO SUB 9000

60  LET MO = 0

70  GO SUB 5000

1000 REM ACCEPT PLAYERS MOVE

1010 LET MO = 0

1020 INPUT "ENTER MOVE: "; LINE B$

1030 PRINT AT 21, 0; "Your move -"; B$ (TO 2); " to "; B$ (3
    TO); " "

1040 LET Z (1) = CODE B$ (1) * 10 + CODE B$ (2)

1050 LET Z (2) = CODE B$ (3) * 10 + CODE B$ (4)

1060 FOR F = 1 TO 2

1070 RESTORE 9160: BEEP .02, 35

1080 READ X, Y

1090 IF X = 0 THEN PRINT AT 21, 0; INK 2; PAPER 7; "ILLEGAL
    MOVE!"; PAPER 0; "[15 spaces]": BEEP .5, -10: GO TO 1020

1100 IF Z (F) = X THEN LET B (F) = Y: GO TO 1120

1110 GO TO 1080

1120 NEXT F

1160 LET MO = 0

1170 LET A (B (2)) = A (B (1))

1180 LET A (B (1)) = B

1190 IF ABS (B (1) - B (2)) > 7 THEN LET MO = 1: LET A ((B (1) + B
    (2)) / 2) = B

1200 GO SUB 5000

```

```

2000  REM THE 'INTELLIGENT' BIT

2020  FOR F = 24 TO 72: BEEP .015, F/2: BEEP .008, F/2 + 1

2050  IF A (F) < > C AND A (F) < > K THEN GO TO 2100

2060  FOR D = 1 TO 4

2070  IF A (F) < > K AND D > 2 THEN GO TO 2100

2080  IF (A (F + X (D)) = H OR A (F + X (D)) = W) AND A (F + 2 * X
    (D)) = B THEN LET Q = X (D): GO TO 2120

2090  NEXT D

2100  NEXT F

2110  GO TO 2240

2120  LET A (F + 2 * Q) = A (F)

2130  LET CS = CS + 1

2140  LET A (F + Q) = B

2150  LET P = F + 2 * Q: LET A (F) = B

2160  LET M = 0: FOR D = 1 TO 4

2170  IF (A (P + X (D)) = H OR A (P + X (D)) = W) AND A (P + 2 * X
    (D)) = B THEN LET M = X (D)

2180  IF A (P) < > K AND D > 2 THEN GO TO 2200

2190  IF M = 0 THEN NEXT D

2200  IF M = 0 THEN GO TO 60

2210  GO SUB 5000: LET A (P + M) = B: LET A (P + 2 * M) = A (P):
    LET A (P) = B: LET S = S + 1

2230  GO TO 60

2240  LET Y = 0

2250  BEEP .07, -2: LET F = 24 + INT (RND * 49): LET Y = Y + 1: IF
    Y > 400 THEN LET LS = 1: GO TO 5000

2260  IF A (F) < > C AND A (F) < > K THEN GO TO 2250

2270  FOR D = 1 TO 4

2280  IF A (F) = C AND D > 2 THEN GO TO 2250

```



```

2290 IF A (F + X (D)) = B THEN LET A (F + X (D)) = A (F): LET A
      (F) = B: GO TO 60

2300 NEXT D

2310 GO TO 2250

3000 REM CONVERT TO CROWNS

3010 FOR F = 24 TO 72

3020 IF A (F) = H AND F > 68 THEN LET A (F) = W

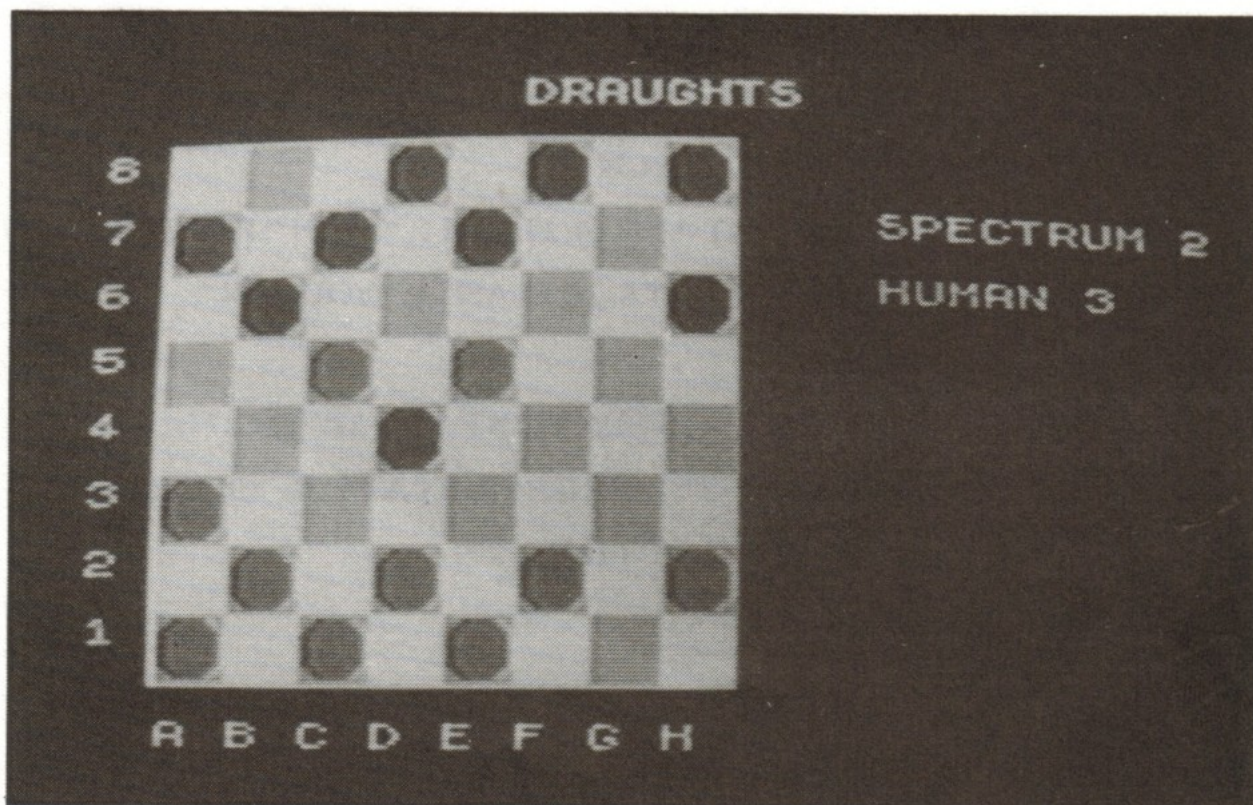
3030 IF A (F) = C AND F < 28 THEN LET A (F) = K

3040 NEXT F

3050 RETURN

4010 PRINT PAPER 1; INK 4; "[34 spaces] g3c g7c □ g3c g7c □ g4 g7c
      □ g7c g4 □ g4 g7c □ g7c g4 □ g3c g3c g7c □ g3c □ □ g3c □ □
      □ □ g5c g5 □ g5c g5 □ g5c g5 □ g5c g5 □ g5c g1 □ g5c g5 □
      □ g5c □ g1 g7c g2 □ □ g8c □ □ □ □ g5c g5 □ g4c g5c □ g4c
      g7 □ g5c g5 □ g5c g7 □ g4c g7 □ □ g5c □ g4 g1 g7c □ □ g3 □ □
      □ □ g3 g2 □ g2 g1 □ g2 g1 □ g1 g2 □ g1 g2 □ g2 g1 □ □ g2 □
      □ g3 □ □ g3 [36spaces]"

```




```

4020 PRINT ' "□ □ □ In this version of Draughts, this is a piece:
gA gB and this is a crown: gE gF □ □ □ □ □ □ gC gD [22
spaces] gG gH □" ' ' "□ □ □ You are red (at the bottom) □ □ &
I am blue (at the top)." ' ' "□ □ □ Moves are entered using □
□ □ □ □ standard 'algebraic notation', □ □ so for example
you would enter □ □ C3D4 to move from column C, row 3 to
column D row 4."

4030 PRINT ' "Please put CAPS LOCK on."

4040 RETURN

5000 REM DISPLAY BOARD

5010 GO SUB 3000: BEEP .09, 20: BEEP .05, 15: IF MO = 1 THEN LET
HS = HS + 1

5020 PRINT AT 0, 12; INK 6; PAPER 1; "DRAUGHTS"; PAPER 0; INK 6;
AT 4, 22; "SPECTRUM □"; INK 7; CS; AT 6, 22; INK 6; "HUMAN
□"; INK 7; HS: IF BD = 1 THEN GO TO 5080

5030 FOR F = 3 TO 15 STEP 4

5040 PRINT INK 4; PAPER 7; AT F - 1, 2; "□ □ g8c g8c □ □ g8c g8c □
□ g8c g8c □ □ g8c g8c"; AT F, 2; "□ □ g8c g8c □ □ g8c g8c □
□ g8c g8c □ □ g8c g8c"; AT F + 1, 2; "g8c g8c □ □ g8c g8c □ □
g8c g8c □ □ g8c g8c □ □"; AT F + 2, 2; "g8c g8c □ □ g8c g8c □
□ g8c g8c □ □ g8c g8c □ □"

5050 PRINT AT F - 1, 0; CHR$ (57 - F/2); AT F + 1, 0; CHR$ (56 -
F/2)

5060 NEXT F: LET BD = 1

5070 PRINT AT 19, 2; "A □ B □ C □ D □ E □ F □ G □ H": INK 6: PLOT
15, 160: DRAW 129, 0: DRAW 0, -129: DRAW -129, 0: DRAW 0,
129: INK 6

5080 LET F = 0: FOR E = 0 TO 7: LET S = (E/2 = INT (E/2)): LET F =
F + 7 * S + 6 * NOT S: LET D = 2 * S

5090 FOR J = 0 TO 3: LET V = A (79 - F - J)

5100 PRINT INK 1 + (V < C); PAPER 4; AT E * 2 + 2, 4 * J + 2 + D;
("□ □" AND V = B); ("gA gB" AND (H = V OR C = V)); ("gE gF"
AND (K = V OR W = V)); AT E * 2 + 3, 4 * J + 2 + D; ("□ □"
AND B = V); ("gC gD" AND (H = V OR C = V)); ("gG gH" AND (K =
V OR W = V))

5150 NEXT J: NEXT E

5160 PRINT AT 21, 0; "[32 spaces]"

```

```

5170 GO SUB 7000: RETURN

6000 REM g8c INITIALISE U.D.G.s g8c

6010 RESTORE 9190

6020 FOR F = 1 TO 8: READ A$

6030 FOR G = 0 TO 7

6040 READ H: POKE USR A$ + G, H

6050 NEXT G

6060 NEXT F

6070 RETURN

7000 REM g8c VARIOUS CHECKS g8c

7010 IF LS = 1 THEN PRINT AT 8, 22; "I give up!": INPUT "Hit ENTER
to play again ☐"; LINE A$: RUN

7020 IF CS = 12 THEN PRINT AT 6, 24; FLASH 1; " I WIN": PAUSE 0:
RUN

7030 IF HS = 12 THEN PRINT AT 6, 24; FLASH 1; "YOU WIN": PAUSE 0:
RUN

7040 LET Q$ = "☐": IF MO = 1 THEN PRINT AT 6, 22; "again?": INPUT
"Can you jump? ☐"; LINE Q$: PRINT AT 8, 22; "☐ ☐ ☐ ☐ ☐ ☐":
IF Q$ (1) = "Y" THEN GO TO 1000

7050 LET MO = 0

7060 RETURN

9000 REM

9010 RESTORE 9150: DIM A (100): LET Y = 0

9020 DIM X (5)

9030 FOR F = 1 TO 5: READ G: LET X (F) = G: NEXT F

9040 LET H = 0: LET C = 128: LET W = 1: LET K = 129: LET B = 255:
LET Q = 0

9050 FOR F = 1 TO 100

9060 LET A (F) = 9: IF F < 73 AND F > 55 AND F < > 68 AND F < > 67
AND F < > 62 AND F < > 61 AND F < > 60 THEN LET A (F) = C: GO
TO 9090

```

```

9070 IF F < 54 AND F > 42 AND F < > 49 AND F < > 48 AND F < > 47
    THEN LET A (F) = B: GO TO 9090

9080 IF F < 41 AND F > 23 AND F < > 36 AND F < > 35 AND F < > 34
    AND F < > 29 AND F < > 28 THEN LET A (F) = H

9090 NEXT F

9100 LET LS = 0: LET CS = 0: LET HS = 0

9110 DIM B$ (4): DIM B (2): DIM Z (2)

9120 LET MO = 0: LET B$ (3) = "K"

9130 INPUT "Do you want to go first? ☐"; LINE A$: CLS: IF A$ (1) =
    "Y" THEN RETURN

9140 GO SUB 5000: LET Z = 57 + INT (RND * 3): LET Q = INT (RND *
    2) - 7: LET A (Z + Q) = C: LET A (Z) = B: RETURN

9150 DATA -6, -7, 6, 7, 9

9160 DATA 699, 27, 701, 40, 703, 53, 705, 66, 710, 33, 712, 46,
    714, 59, 716, 72, 719, 26, 721, 39, 723, 52, 725, 65

9170 DATA 730, 32, 732, 45, 734, 58, 736, 71, 739, 25, 741, 38,
    743, 51, 745, 64, 750, 31, 752, 44, 754, 57, 756, 70

9180 DATA 759, 24, 761, 37, 763, 50, 765, 63, 770, 30, 772, 43,
    774, 56, 776, 69, 0, 0

9190 DATA "A", 0, 15, 31, 63, 127, 127, 127, 127

9200 DATA "B", 0, 240, 248, 252, 254, 254, 254, 254

9210 DATA "C", 127, 127, 127, 127, 63, 31, 15, 0

9220 DATA "D", 254, 254, 254, 254, 252, 248, 240, 0

9230 DATA "E", 0, 15, 31, 63, 127, 124, 120, 121

9240 DATA "F", 0, 240, 248, 252, 254, 62, 30, 158

9250 DATA "G", 121, 120, 124, 127, 63, 31, 15, 0

9260 DATA "H", 158, 30, 62, 254, 252, 248, 240, 0

```

Other titles of interest

Computer Puzzles: For Spectrum and ZX81

Ian Stewart & Robin Jones

Brainteasers for BASIC Computers

Gordon Lee

Easy Programming for the ZX Spectrum

Ian Stewart & Robin Jones

'... will take you a long way into the mysteries of the Spectrum; is written with a consistent and humorous hand; and shares the affection the authors feel for the computer' – *ZX Computing*

Further Programming for the ZX Spectrum

Ian Stewart & Robin Jones

Spectrum Machine Code

Ian Stewart & Robin Jones

Spectrum in Education

Eric Deeson

PEEK, POKE, BYTE & RAM! Basic Programming for the ZX81

Ian Stewart & Robin Jones

'Far and away the best book for ZX81 users new to computing'
– *Popular Computing Weekly*

'... the best introduction to using this trail-blazing micro'
– *Computers in Schools*

'One of fifty books already published on the Sinclair micros, it is the best introduction accessible to all computing novices'
– *Laboratory Equipment Digest*

The ZX81 Add-On Book

Martin Wren-Hilton

Shiva Software

Spectrum Special 1

Ian Stewart & Robin Jones

A selection of 10 educational games and puzzles.

Spectrum Specials 2 & 3

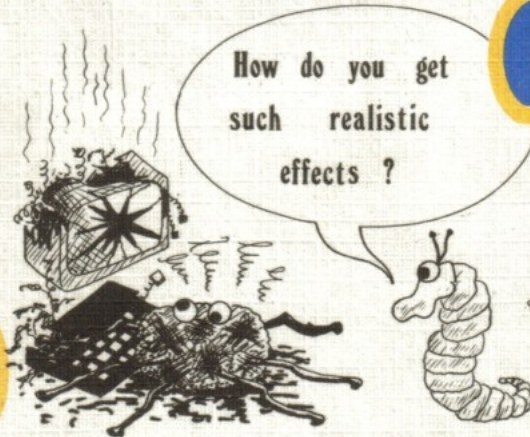
Ian Stewart & Robin Jones

Can you:

- crack the secret code
- break out of your cell
- leap over nine buses on your stunt cycle
- and blow up the enemy ship?

Or maybe gardening and worm racing are more your style. These games are designed to test your faculties and the Spectrum's facilities to the full. Also included are a couple of programs to help you write your own games.

All programs will RUN on the standard 16K Spectrum.



Shiva Publishing Limited

ISBN 0 906812 28 3

UK price £1.95 net