THE COMMODORE 64 GAMES BOOK

21 Sensational Games

Other books of interest from Granada:

The Apple II

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

ATARI

GET MORE FROM THE ATARI
Ian Sinclair
0 246 12149 1

THE ATARI BOOK OF GAMES
M. James, S. M. Gee and K. Ewbank
0 246 12277 3

The BBC Micro

INTRODUCING THE BBC MICRO
Ian Sinclair
0 246 12146 7

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

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

BBC MICRO GRAPHICS AND SOUND
Steve Money
0 246 12156 4

DISCOVERING BBC MICRO MACHINE CODE
A. P. Stephenson
0 246 12160 2

THE BBC BASIC PROGRAMMER
S. M. Gee and M. James
0 246 12158 0

The Colour Genie

MASTERING THE COLOUR GENIE
Ian Sinclair
0 246 12190 4

The Commodore 64

COMMODORE 64 COMPUTING
Ian Sinclair
0 246 12030 4

THE COMMODORE 64 GAMES BOOK
Owen Bishop
0 246 12258 7

SOFTWARE 64
Practical Programs for the Commodore 64
Owen Bishop
0 246 12266 8

The Dragon 32

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

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

DRAGON GRAPHICS AND SOUND
Steve Money
0 246 12147 5

THE DRAGON 32
How to Use and Program
Ian Sinclair
0 586 06103 7

The IBM Personal Computer

THE IBM PERSONAL COMPUTER
James Aitken
0 246 12151 3

The Jupiter Ace

THE JUPITER ACE
Owen Bishop
0 246 12197 1

The Lynx

LYNX COMPUTING
Ian Sinclair
0 246 12131 9

The NewBrain

THE NEWBRAIN
And How To Make The Most Of It
Francis Samish
0 246 12232 3

The ORIC-1

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

THE ORIC-1 BOOK OF GAMES
M. James, S. M. Gee and K. Ewbank
0 246 12155 6

THE ORIC-1 PROGRAMMER
M. James and S. M. Gee
0 246 12157 2

ORIC MACHINE CODE HANDBOOK
Paul Kaufman
0 246 12150 5

Continued on inside back cover

The Commodore 64 Games Book

The Commodore 64 Games Book

Owen Bishop

in collaboration with

Audrey Bishop

GRANADA
London Toronto Sydney New York

Granada Technical Books
Granada Publishing
8 Grafton Street, London

British Library Cataloguing in Publication Data
Bishop, Owen
The Commodore games book
1. Electronic games
2. Commodore 64 (Computers)—Programming
I. Title
794 GV1469.2

ISBN 0-246-12253-7

First published in Great Britain 1983 by Granada Publishing

Typeset by V & M Graphics Ltd, Aylesbury, Bucks
Printed in Great Britain

Contents

	Key Points		1
	1	Black Hole	5
	2	Tight-rope	12
	3	Magic Jigsaw	18
	4	Cops and Robbers	24
	5	Mind Over Electrons	30
	6	Nimble Thimble	34
	7	Sand Castle	39
	8	Bombing Run	46
	9	Breakaway	52
	10	Snipers	59
	11	Nine Men Morris	65
	12	Ball Maze	71
	13	Singalong	77
	14	Pintable	82
	15	Rail Runner	88
	16	Computer Clues	96
	17	Spellbound	102
	18	Minefield	107
	19	Vibrations	113
	20	Poker-Face	118
	21	Snorkel	124

Also by Owen Bishop

SOFTWARE 64
Practical programs for the Commodore 64

Get your Commodore 64 working for you! The powerful Commodore 64, with its large memory and versatile graphics, is ideal for both home and office. This book provides thirteen practical programs to help you organise your life better, run a small business, and perhaps even help you win a fortune.

0 246 12266 8

Key Points

The listings with each game have all been made directly from working programs, using a Commodore VIC-1515 printer attached to the Commodore 64. This has been done to avoid the errors which always seem to occur when listings are independently typeset. All the programs should work perfectly, provided that they are correctly keyed in. The description of each game has a section dealing with keying it in. Below are given some general points that apply to all programs.

A mistake which is easy to make (it happened quite often during the writing of this book!) is to type 'o' (letter 'oh') instead of '0' (figure 'zero'). The letter 'oh' is used BASIC words such as 'FOR' and 'POKE', but never in this book as a variable. In the listings, the figure has a stroke across it to distinguish it from the letter. Another confusion is between 'I' (letter 'eye') and '1' (figure 'one'). The letter is not used alone as a variable in any of the listings, but it has not been possible to avoid using it in the two time variables, TI and TI$. To save confusion, none of the programs use a variable called T1 ('tee one').

The semicolon ';' is an essential item in many PRINT statements. Keep a sharp eye open for this, especially at the ends of lines. It may be even more difficult to spot in a multistatement line such as:

```
FOR J=1 TO 10:PRINTM$(J);:NEXT
```

If you leave out the semi-colon, the display on the screen will be spoiled.

The Commodore 64 requires frequent use of the POKE statement for controlling its visual and sound effects, especially in games programs. POKEing the wrong number to the right address may have dire consequences, and POKEing the right number to the wrong address can have even worse! If the computer 'hangs up' when you RUN the program, it is probable that you have made a mistake with one of the POKE statements. This can never damage the computer, of

course, but it is tedious to have to switch off the machine and type in the program again.

To save space, we frequently use a variable such as S (for sound, or SID) and V (for vision, or VIC II chip) in a POKE statement. At the beginning of the program, for example, we have a statement such as 'V=53248' and then use statements such as 'POKE V + 17' throughout the program. If you have forgotten to type in the statement which defines V, any statement such as 'POKE V+7' later in the program will have strange results.

Several of the programs take advantage of the 64's facilities for specially defined graphics. In order to do this the program alters the way in which the computer uses its memory. The result is that, should you stop the program when it is running or try to use the computer when the program is finished, some or all of the ordinary keyboard characters may appear as featureless smudges on the screen. Do not worry, the computer has not been harmed. Just press 'RUN/STOP' with 'RESTORE', and they will appear normally.

Programs which rearrange memory always reduce the amount available for the BASIC program. This action is *not* cancelled by pressing 'RUN/STOP' with 'RESTORE'. The effect persists even when you have 'NEWED' the program and are trying to load the next one. If this is a very lengthy one you could get an 'OUT OF MEMORY' error at this stage. What has happened is that the new program does not fit into the restricted amount of memory allocated for the previous program. The easiest thing to do is to switch off the computer, then switch on and LOAD the new program.

Games programs need attractive displays and some require machine code routines. These are typed in as DATA statements. Many of these are long and you need to take special care to get every figure right. It is well worth while to SAVE the program on to tape or disk before RUNning it. There might be a small error in your typing (for example, a POKE to the wrong address) which causes the computer to stall when the program is RUN. The only remedy may be to switch off the computer and start again. With the program already on tape, it takes only a few moments to re-load it and look for the error.

One of the features of Commodore computers is their use of control characters. These are characters, such as the ' ▭ ' reversed heart which appear in PRINT statements and tell the computer to do something. In the example above, the character has the same effect as pressing 'CLR/HOME' with 'SHIFT'. Such characters affect the computer, and most of them appear in listings, but they never actually

appear on the screen when the program is running. The VIC 1515 printer reproduces most of these characters but with a low degree of resolution, which often makes them difficult to recognise. For this reason, the 'Keying in' section of each game lists all the control characters used in that program and the lines on which they occur. If a character occurs more than once in a line, the number in brackets tells you how many times they occur on that line. Below is a table to show you how to obtain each of these characters, when typing a PRINT (or INPUT) statement.

Character	*Its effect*	*Key presses*
	Clear screen	CLR/HOME
	Clear screen, and send the cursor home	CLR/HOME and SHIFT
	Cursor down	up-down CRSR
	Cursor right	left-right CRSR
	Cursor up	up-down CRSR and SHIFT
	Cursor left	left-right CRSR and SHIFT
	Black print	CRTL and 1
	White print	CTRL and 2
	Red print	CTRL and 3
	Cyan print	CTRL and 4
	Purple print	CTRL and 5
	Green print	CTRL and 6
	Blue print	CTRL and 7
	Yellow print	CTRL and 8
	Reverse on	CTRL and 9
	Reverse off	CTROL and 0

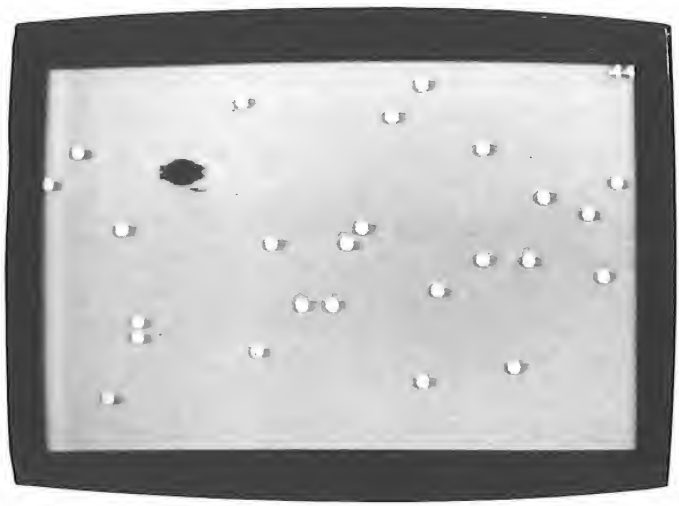
Note that all control characters appear as 'reverse' characters. If you get a normal character, when typing the above, you may have forgotten to type the " at the beginning of the PRINT statement.

You will see that the list above does *not* include the control characters for the colours obtained by using the 'Commodore' key. This is the key marked with the Commodore Logo, situated at the extreme left of the front row of keys. We will refer to it as the 'C=' key, since the symbol on it has a similar appearance. The VIC-1515 ignores these symbols when it lists a program, so they do not appear in the listings given in this book. Where such characters have been used (which is not often) they are noted under the heading 'unlisted' in the

'Keying in' section. The table below shows how to obtain these characters, and what they do.

Character	*Its effect*	*Key presses*
	Orange print	C = and 1
	Brown print	C = and 2
	Pink print	C = and 3
	Grey 1 print (dark grey)	C = and 4
	Grey 2 print (medium grey)	C = and 5
	Light green print	C = and 6
	Light blue print	C = and 7
	Grey 3 print (Light grey)	C = and 8

1
Black Hole



You are in command of a spacecraft exploring a distant part of the Galaxy. This region is of special attraction to explorers because it teems with nuggets of pure gold! Most of them are as big as a house, or even larger! Space explorers such as yourself, find a rich harvest there, just for the taking.

It is thought that these nuggets were produced by nuclear fusion during the last stages of collapse of a supergiant star which once existed in this region. Now all that remains are a few traces of gas, the nuggets – and the Black Hole.

Unfortunately, the last stages of collapse also produced a Black Hole, into which the greater part of the mass of the star disappeared. Black holes need not be large – this one is too small to be seen until you are within a kilometre, or so, of it. Yet the strong pull of the condensed matter of the former star makes itself felt at distances of thousands of kilometres. Any object within this range, including your spacecraft, is drawn forcibly toward the Black Hole at ever increasing velocity, never to be seen again.

Gold hunting in space is a hazardous business but, in readiness for this, your craft is equipped with powerful booster rockets which can overcome the enormous gravitational attraction of the Black Hole –

provided that you act in time. Since you cannot see the Hole, you can only guess where it is by noting the extent to which your craft is being pulled out of its intended course. Act quickly, for a few moments delay will almost certainly lead to disaster.

How to play

When the program is run you are asked to choose at which level of difficulty you wish to play. Level 1 is the easiest; level 10 is the hardest. The higher the level the faster the game, and the stronger the effect of the Black Hole. Type in your chosen level and press 'RETURN'.

The screen becomes blue, with a black border. Scattered at random over the screen you see 20 gold nuggets. After a few seconds, your spacecraft appears, flashing its green and red lights and bleeping at the same time. Both the position of your craft and that of the Black Hole are randomly selected, so no two games are ever exactly alike.

The top right-hand corner of the screen displays the elapsed time, in seconds. You have exactly 60 seconds in which to collect as many nuggets as possible, preferably without being drawn into the Black Hole. The craft is moved around the screen by pressing one of these keys:

Z to move left
X to move right
; to move up
/ to move down

Each time you press one of these keys, the craft moves one step (that is, to the next row or column on the screen) in the required direction. You cannot move it completely off the screen. You collect a nugget by moving the craft so that the middle of its upper half is over the nugget. The program keeps score of how many nuggets you have collected.

Sooner or later, you will find that, as well as moving as you intended, the craft moves unexpectedly and sometimes very quickly. Your spacecraft is being attracted by the Black Hole. When you press the control keys now, their effect is automatically boosted, and you move 3 steps at a time. This is enough to overcome any effect that the Black Hole might have, *provided* you press the right key and press it in time. If you manage to place your craft beyond the range of the Black Hole, it then moves one step for each key-press, as

before. While you are in the vicinity of the hole your movements will be erratic, like a frenzied tug-of-war between you and the force of the Hole. But it is still possible to collect nuggets, either by accident or design and, as a bonus, these nuggets score three times as much as those out of range of the Hole. It is worth risking danger to improve your score, but do not get sucked into the Black Hole if you can help it, for this carries a penalty.

One point to remember is that, whereas *your* commands affect the craft only when you press a key, the gravitational force of the Hole acts all the time. If the craft is within range of a Hole and you do nothing to save it, it will move steadily toward the Hole of its own accord and soon be lost.

Your departure into the Black Hole is accompanied by suitable visual and sound effects, after which you are told your score, the best score so far, and the length of time for which the game lasted. If you manage to survive for the full 60 seconds, there is a more colourful finale.

When you are ready to play again, press the 'Y' key. You will then be taken back to the beginning and given another opportunity to select the level of play. The Hole will probably be in a different position next time. The 'Best Score' value is retained when you play again, though it is lost if you re-run the program.

Winning tactics

Nuggets score 1 point if they are beyond the range of the Hole (more than about 10 rows or 20 columns away), depending on the level, or 3 points if they are within range. The final score is calculated on a 'points per minute' basis. It is therefore best to collect nuggets as quickly as you can, just in case you soon find yourself in the Hole. If you fall into the Hole, you *lose* 10 points, so you could end up with a negative score. If time is nearly running out and you have collected most of the nuggets, it might be better not to try to get those which are near the Hole for, although they are worth 3 points each, there is a strong risk of losing 10 points if you fail. Safer to let the 'clock' bring the game to an end. However, you may need that extra point or, two, to better your previous score or that of someone else, in which case the risk may be worth while. Or perhaps you just like running risks!

If the Hole is near the top or bottom, or to one side, it is fairly easy to pick up nuggets from the distant areas. If the Hole is near the

centre it is a tricky business to circumnavigate it to get the nuggets from the opposite side. Try making a dash for it! Incidentally, the command keys are read by the GET statement, which has an input buffer. Commands sent in quick succession are stored and executed later. If you become frenzied and press keys more times than you intended, the craft continues to obey these extra commands after you have ceased to press the keys. This may well waste valuable seconds of flight time.

If you think you know where the Hole is, it is worth venturing closer to it to collect the valuable nuggets in that region. With practise, it is often possible to position yourself with a nugget between you and the Hole, and then let the Hole draw you on to the nugget, so capturing it. Then a quick burst of boosted power takes you safely out of range again. Happy hunting!

Keying in

The symbol PRINTed on line 190 to represent the nuggets, is the disk ('SHIFT' with 'Q').

Control characters used are:

CLEAR: lines 20, 100, 520, 670, 690
CTRL/2: line 100
CTRL/8: line 160, 760
CRSR DOWN: lines 610(3), 620, 630, 640(5)
CRSR RIGHT: lines 610(3), 620(3), 630(3), 640(3)

Program design

The main sections of the program are:

20–170	Initialising, including asking the user to key in the playing level.
180–200	Displaying nuggets randomly scattered on the screen.
210–240	Selecting position of the Hole (row R, column C) and the initial position of the spacecraft (row RC, column CC) at random. Line 240 rejects any spacecraft initial position which is too close to the Hole.
250	Display craft at initial position (omitting first stage of subroutine 1000).
260	Initialise the clock.

270	The re-entry point for each move.
270–290	Update time display and check for 'time up'.
300–350	Bleep and flash spacecraft lights (alternating red or green).
360–410	Accept command from keyboard, calculate position of craft and move it (subroutine 1000).
420–510	Calculate effect of Hole (if any) on position of craft, and move it (subroutine 1000).
520–680	Final routine following capture by Hole, begins with sound effects, then displays score. Invites repeated play.
690–790	Final routine when time is up. Jumps to other final routine for the display of score.
1000–1110	Moving the spacecraft. Checks if on edge of screen and rejects moves completely beyond edge. Increments score if it replaces a nugget (line 1080).
2000	DATA line for spacecraft sprite.

Points of interest

The spacecraft is a multicolour sprite. The body of the craft is black (border colour). It has two sets of 'lights', one red, one green. These are alternately coloured or black. Thus the lights appear to flash on and off in time with the bleeping.

Random numbers are used in setting up the game. The expression in line 140 uses the time variable TI to seed the random number generator so that it produces a different sequence of random numbers each time the game is played. Line 150 defines a random number function. Given the parameter X, it produces a random number between zero and X. This function saves program space, for it is used five times (lines 190, 210 and 220) with differing parameters.

The program

```
10 REM ** BLACK HOLE **
20 PRINT"□":POKE 53280,6:POKE 53281,0:G=
1024:H=55296:V=53248:S=54272
30 POKE 52,48:POKE 56,48
40 FOR J=0 TO 38:READ X:POKE 12288+J,X:N
EXT
50 FOR J=0 TO 23:POKE 12338+J,0:NEXT
```

```
60 POKE 2040,192:POKE V+28,1:POKE V+37,1
0:POKE V+0,13:POKE V+39,0
70 GOSUB 1500
80 POKE S+3,8:POKE S+14,164:POKE S+18,32
:POKE S+24,143
90 FR=10000:POKE S+5,133:POKE S+6,240
100 PRINT"■"
110 INPUT"LEVEL (1-10)";L$
120 L= VAL(L$)
130 IF L<1 OR L>10 THEN 100
140 X=RND(-TI):N=1:L=(10-L)/2
150 DEF FNR(X) = INT(RND(1)*(X+1))
160 PRINT"■■":POKE H+38,7:POKE H+39,7
170 POKE 53280,0:POKE 53281,6
180 FOR J=1 TO 25
190 PRINTSPC(FNR(60));"●";
200 NEXT
210 R=FNR(24):C=FNR(39)
220 RC=FNR(24):CC=FNR(39)
230 IF ABS(R-RC)<13 OR ABS(C-CC)<21 THEN
 220
240 POKE V+21,1
250 GOSUB 1050
260 TI$="000000"
270 POKE G+38,ASC(MID$(TI$,5,1))
280 POKE G+39,ASC(RIGHT$(TI$,1))
290 IF VAL(RIGHT$(TI$,3))=100 THEN 690
300 GOSUB 900
310 POKE V+37,0:POKE V+38,13
320 FOR K=0 TO 5*L:NEXT
330 GOSUB 900
340 POKE V+37,10:POKE V+38,0
350 FOR K=0 TO 150:NEXT
360 GET A$
370 IF A$=";" THEN RC=RC-N
380 IF A$="/" THEN RC=RC+N
390 IF A$="Z" THEN CC=CC-N
400 IF A$="X" THEN CC=CC+N
410 GOSUB 1000
420 RA=ABS(R-RC):CA=ABS(C-CC)
430 IF RA<2 AND CA<2 THEN T$=RIGHT$(TI$,
2):GOTO 520
440 IF RA>12 OR CA>20+L THEN N=1:GOTO 27
0
450 N=3
460 IF RA<6 THEN RC=RC+SGN(R-RC)
470 IF RA<13 THEN RC=RC+SGN(R-RC)
480 IF CA<11 THEN CC=CC+SGN(C-CC)
490 IF CA<21+L THEN CC=CC+SGN(C-CC)
500 GOSUB 1000
510 GOTO 270
520 PRINT"■":POKE 53281,0:GOSUB 1500
530 POKE S+24,15:POKE S+6,248:POKE S+4,1
29
540 FOR FQ=10 TO 20000 STEP 100
550 HF=INT(FQ/256):LF=FQ-HF*256
560 POKE S,LF:POKE S+1,HF
570 NEXT
580 POKE V+21,0:GOSUB 1500
590 SC=INT(SC*60/VAL(T$))-10
600 IF SC>BS THEN BS=SC
610 PRINT"■■■■■■■YOUR SCORE =";SC
620 PRINT"■■■■BEST SCORE =";BS
630 PRINT"■■■■TIME = ";T$;" SECONDS"
640 PRINT"■■■■■■■■PRESS SPACE BAR TO CON
TINUE"
650 SC=0
```

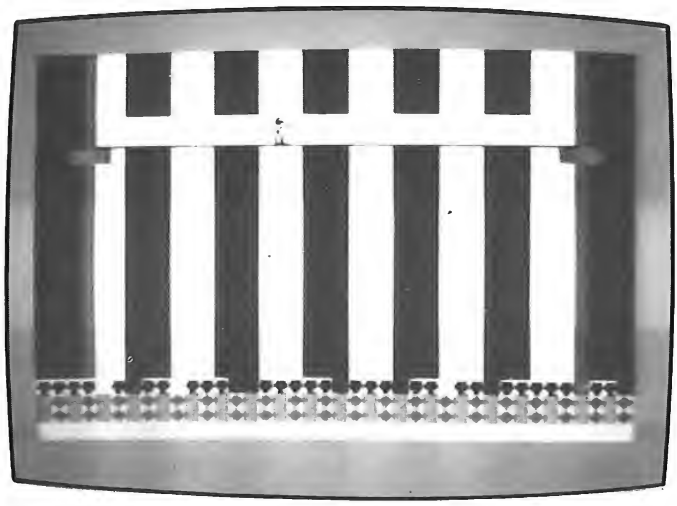
```
660 GET A$:IF A$<>" " THEN 660
670 PRINT"□":POKE 53281,6
680 POKE V+21,0:GOTO 60
690 PRINT"□":POKE 53280,7:POKE 53281,6
700 POKE V+21,1:POKE V,174:POKE V+1,200:
POKE V+38,13
710 POKE S+6,252
720 GOSUB 900
730 FOR K=1 TO 1000:NEXT
740 GOSUB 1500
750 T$="60"
760 PRINT"■":GOTO 600
900 FQ=FR+PEEK(S+27)*4
910 HF=INT(FQ/256):LF=FQ-HF*256
920 POKE S+4,33
930 POKE S,LF:POKE S+1,HF
940 POKE S+4,32
950 RETURN
1000 IF RC<0 THEN RC=0
1010 IF RC>24 THEN RC=24
1020 IF CC<0 THEN CC=0
1030 IF CC>39 THEN CC=39
1040 X=CC*8+16:Y=RC*8+50
1050 POKE V+16,INT(X/255)
1060 POKE V,X-INT(X/255)*255
1070 POKE V+1,Y
1080 Z=CC+40*RC:IF PEEK(G+Z)=81 THEN SC=
SC+N:POKE H+Z,6
1090 RETURN
1500 FOR J=0 TO 24:POKE S+J,0:NEXT:RETUR
N
2000 DATA 0,168,0,2,170,0,10,186,128
2010 DATA 170,170,168,153,185,152,170,17
0,168
2020 DATA 34,186,32,170,170,168,153,185,
152
2030 DATA 170,170,168,10,186,128,2,170,0
,0,168,0
```

Variations

If you are keen on defining your own graphics, the nuggets could be given a different shape. To add interest, these could be defined in two or three different shapes or colours, with different scores for each.

There is plenty of scope for varying the scoring. As an alternative approach, you could dispense with the clock and go for high scores. When a given number (say 15) of the nuggets have been collected, the supply could be replaced by returning to lines 180 to 200. A skilful player could amass a huge score before being finally sucked into the Black Hole.

2
Tight-rope



The scene is the Big Top, with a tight-rope stretched between the two tent-poles. The crowd waits expectantly below for you to begin your walk. Swaying, you begin to cross. It takes 31 steps to get from the platform on one pole to that on the other. Precise timing is essential. If you take a step when you are swaying too far in one direction, you could lose your balance and fall.

How to play

You are first asked at which level you wish to play. Level 1 is the easiest. With increasing level, a breeze begins to blow against you, making it more difficult to keep balanced. More important, the angle at which you finally overbalance is reduced with increasing playing level, so that you are more likely to fall.

When you have typed in the playing level and pressed 'RETURN', the screen displays the interior of the Big Top, with the heads of the crowd just visible above the Ring. The tight-rope walker stands on the left-hand platform.

You have three keys to control the wire walker:

S makes you take one step forward (no turning back!).
Z adjusts your balance to the left. This effect is cumulative; the more times you press Z, the further you tilt to the left.
X adjusts balance to the right.

Choose the moment with care, press S, and you are off on your perilious journey.

Winning tactics

If you are the kind of person who takes risks, select a higher playing level than you think you can manage. See how many steps you can take before you fall off. When you fall, the screen displays the number of steps you have taken. It also displays the greatest number of steps taken by any walker during the current run of the game.

Alternatively, try to cross in the shortest possible time. Select a level which is not too difficult (or too easy!). When you reach the other side, the screen displays your crossing time, and also the shortest time in the current session.

The regular swaying, which you can gauge before you begin to walk, is not enough to throw you off the wire, even at level 10. If other effects are added to this swaying at the wrong moment, you may readily overbalance. The walker always begins by stepping off with the left foot. This causes an additional tilt toward the left. So beware of taking the first step when the pointer is moving to the left. Stepping causes a second oscillation to be added to the constant swaying, but this gradually dies away, and it is then safe to take the next step. If you are trying for the shortest possible time it is a matter of judging the minimum safe interval between taking successive steps.

Adjusting your balance is useful for reducing the amplitude of the regular swinging when you are about to take a step. Timing is important. Adjusting balance at the wrong moment can lead to disaster. There is an initial strong effect, to counter a large swing immediately. The effect is then gently reduced but a small permanent bias remains. This may be countered by pressing the other balance key.

The breeze alters your balance in a random way and changes frequently. Its effect is not strong, but it adds an element of chance to the game.

Keying in

Lines 170, 210 and 250 look peculiar. These are the PRINT statements for the main display. It is essential to type these in exactly as listed, otherwise a very strange display may result. Do not worry about the 'R's and 'Z's in these lines. They do not appear on the screen as letters but as white and red blocks respectively. This is because the program re-defines key 'R' as a space and key 'Z' as a solid block of colour (reversed space).

Control characters used are:

CLEAR: lines 40, 80, 680, 740
CTRL/7: lines 80, 170(2), 210(2), 250
CRSR DOWN: lines 80, 700(3), 720(2), 760(3), 770(2), 780(5)
CRSR RIGHT: lines 80(2), 700(2) 720(2), 760(2), 770(2), 780(5)
CTRL/3: lines 170(3), 210(2), 250(6), 760
CTRL/1: line 250(7)
CTRL/5: line 270
CTRL/6: lines 270, 780

Program design

20–110	Initialising.
120–160	Put the values required for generating motion into three arrays, A() for regular sway, S() for the effects of taking a step, and B() for the temporary effects of correcting balance.
170–280	Set up the main feature of the display.
290–320	POKE a row of 'invisible' (white on white) up-arrows across the bottom of the display, ready for the tilt display. Make the central one red.
330–360	Display the walker, arms raised, on the left-hand platform; display the tight-rope.
370	Set the block to zero.
380–670	Generate the motion, in four phases.
690–730	The final display routine following a successful crossing. This continues at line 780.
740–770	The final display routine following a fall.
780–800	Inviting the player to try again.
1000–1020	Subroutine to check for a fall.
2000–2060	Subroutine to accept commands from keyboard and put them into effect.

3000–3050	Performs the changes in the display which animate the figure walking across the tight-rope.
4000–4070	DATA for the arrays which control motion.
5000–5040	DATA for the special graphics characters.

The Program

```
10 REM ** TIGHT-ROPE **
20 G=0:H=0:D=0:PD=0:S=0:B=0:BP=0
30 W=0:NS=0:FF=0:POKE 649,1
40 PRINT"■":POKE 53280,7:POKE 53281,1
50 POKE 52,48:POKE 56,48:CLR:POKE 56334,
PEEK(56334)AND254:POKE 1,PEEK(1)AND251
60 FOR J=0 TO 111:READ X:POKE 12416+J,X:
NEXT
70 POKE 1,PEEK(1)OR4:POKE 56334,PEEK(563
34)OR1
80 INPUT"■■■■LEVEL (1-10)";L$
90 L=VAL(L$):IF L<1 OR L>10 THEN 80
100 PRINT"■";:POKE 53272,(PEEK(53272)AND
240)+12
110 DIM A(24),S(24),B(24):FT=1E6:G=1024:
H=55296:LR=-1
120 FOR J=0 TO 24
130 READ A(J)
140 READ S(J)
150 READ B(J)
160 NEXT
170 D$="■ZZZ■ZRR■ZZZRRRZZZRRRZZZRRRZZZRR
RZZZRRR■Z■ZZ"
180 FOR J=0 TO 5
190 PRINT D$;
200 NEXT
210 PRINT "ZZ■ZZZR■ZZZRRRZZZRRRZZZRRRZZZ
RRRZZZRR■ZZZ■ZZ";
220 FOR J=0 TO 13
230 PRINT D$;
240 NEXT
250 PRINT "■[[[[R[■Z■[[R[[[■ZZ■[[[[[■Z■[
[[■Z■[[R[[■Z■[[[[[■Z■[[■E";
260 FOR J=0 TO 39
270 PRINT "■Z■E";
280 NEXT
290 FOR J=0 TO 22
300 POKE G+968+J,29
310 NEXT
320 POKE H+979,2
330 POKE H+164,9:POKE H+204,9:POKE G+164
,16:POKE G+204,17
340 FOR J=165 TO 195
350 POKE H+J,1:POKE G+J,18:POKE H+J+40,9
:POKE G+J+40,19
360 NEXT
370 TI$="000000"
380 J=0
390 D=A(J)+S*S(J)+B*B(J)+BP+W
400 GOSUB 1000
410 IF FF=1 THEN 740
420 J=J+1:IF J<25 THEN 390
430 GOSUB 2000
440 IF NS=31 THEN 680
```

```
450 J=0
460 D=A(24-J)+S*S(J)+B*B(J)+BP+W
470 GOSUB 1000
480 IF FF=1 THEN 740
490 J=J+1:IF J<25 THEN 460
500 GOSUB 2000
510 IF NS=31 THEN 680
520 J=0
530 D=-A(J)+S*S(J)+B*B(J)+BP+W
540 GOSUB 1000
550 IF FF=1 THEN 740
560 J=J+1:IF J<25 THEN 530
570 GOSUB 2000
580 IF NS=31 THEN 680
590 J=0
600 D=-A(24-J)+S*S(J)+B*B(J)+BP+W
610 GOSUB 1000
620 IF FF=1 THEN 740
630 J=J+1:IF J<25 THEN 600
640 GOSUB 2000
650 IF NS=31 THEN 680
660 W=(L-1)*(INT(RND(1)*7)-3)
670 GOTO 380
680 PRINT"■":POKE 53280,6:POKE 53281,9:P
OKE 53272,21
690 T=INT(TI/60)
700 PRINT"■■■■■■■SUCCESSFUL CROSSING IN "
;T;" SECONDS"
710 IF T<FT THEN FT=T
720 PRINT"■■■■FASTEST TIME IS ";FT;" SEC
ONDS"
730 GOTO 780
740 PRINT"■":POKE 53280,4:POKE 53281,13:
POKE 53272,21
750 IF MS<NS THEN MS=NS
760 PRINT"■■■■■■■YOU FELL, AFTER ";NS;"ST
EPS"
770 PRINT"■■■■LONGEST WALK IS ";MS;" STE
PS"
780 PRINT"■■■■■■■■■■■■PRESS SPACE BAR TO
CONTINUE"
790 GET A$:IF A$<>" " THEN 790
800 RESTORE:CLR:GOTO 40
1000 IF ABS(D)>200-7*L THEN FF=1
1010 POKE H+979+PD/20,1:POKE H+979+D/20,
2:PD=D
1020 RETURN
2000 S=S*-.5
2010 IF B<>0 THEN BP=BP+10*SGN(B):B=0
2020 GET A$
2030 IF A$="S" THEN S=LR:LR=-SGN(LR):NS=
NS+1:GOSUB 3000
2040 IF A$="X" THEN B=1
2050 IF A$="Z" THEN B=-1
2060 RETURN
3000 POKE G+163+NS,22:POKE G+164+NS,24:P
OKE G+203+NS,23:POKE G+204+NS,25
3010 POKE H+164+NS,9:POKE H+204+NS,9
3020 FOR K=1 TO 400:NEXT
3030 POKE G+163+NS,18:POKE G+164+NS,20:P
OKE G+203+NS,19:POKE G+204+NS,21
3040 IF NS=31 THEN POKE G+195,16:POKE G+
235,17:FOR K=1 TO 5000:NEXT
3050 RETURN
4000 DATA 129,153,189,153,90,60,24,24,24
,24,36,36,36,36,102,255
```

```
4010 DATA 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
255
4020 DATA 0,48,96,120,112,34,25,160,160,
32,80,72,72,72,108,255
4030 DATA 3,6,7,39,18,15,2,2,2,5,4,4,4,4
,2,255
4040 DATA 0,0,128,0,0,0,128,64,32,128,64
,128,128,64,0,255
4050 DATA 255,255,255,255,255,255,255,25
5
4060 DATA 0,60,60,126,126,60,24,24,24,60
,126,255,255,126,60,24
4070 DATA 24,126,219,153,24,24,24,24
5000 DATA 3,2,25,9,12,46,16,21,50,22,30,
49,28,38,47
5010 DATA 34,46,44,40,55,41,45,60,37,51,
65,33,56,69,28
5020 DATA 61,72,25,66,74,23,71,75,21,75,
74,18,79,72,17
5030 DATA 83,69,15,86,65,13,89,60,12,92,
55,17,94,46,11
5040 DATA 96,38,10,98,30,10,99,21,10,100
,12,10,100,2,10
```

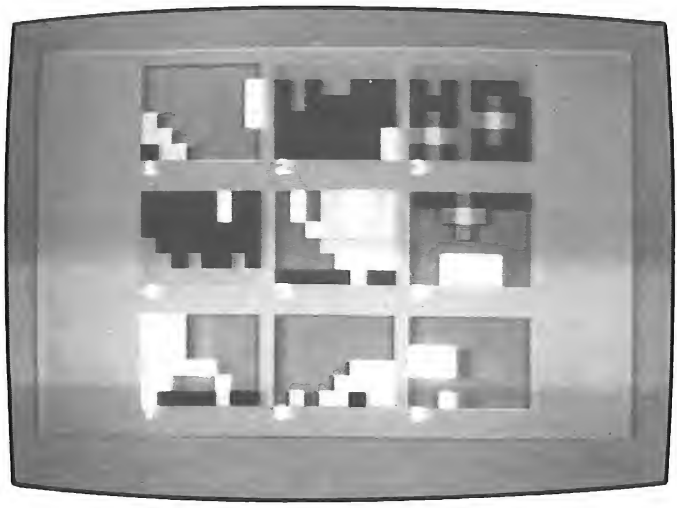
Variations

The values in line 1000 can be altered to vary the range of playing levels. If LR were to be altered to −1.2 or perhaps less (line 110), this would make S take the values −1.2 and +1.2 at alternate steps, so increasing the disturbance of balance due to taking a step.

To increase the element of chance, alter line 660 to increase the breeze factor, W. As listed, the expression 'INT(RND(1)*7)−3' gives random numbers between −3 and +3. To get numbers between −X and +X, alter the '7' to 'twice X plus 1', and alter the '3' to 'X'.

There is scope for making the most of the COMMODORE 64's sound chip. The roll of drums before the walk begins, and at each daring step; the triumphant chord at the end of the walk and the applause of the crowd would all add to the realism of the game.

3
Magic Jigsaw



Have you heard the story about poor old Humpty Dumpty? This is what happened ...

Humpty Dumpty sat on a wall.
Humpty Dumpty had a big fall.
All the King's horses and all the King's men,
Couldn't put Humpty Dumpty together again.

He was very sad to be broken into so many pieces. The King's horses and the King's men tried so hard to put him together, but they could not.

Could you put him together again?
You only have to try!

How to play

The jigsaw has 9 pieces, which appear on the screen when the program is run. They are not in their right places. Some of them are upside down, too.

When all the pieces are showing on the screen, you hear two

musical tones – the first two notes of the Humpty Dumpty tune. This tells you that the computer is ready for you to begin.

You can choose which of two things you wish to do. One is to turn a piece the other way up. The other thing is to move a piece to another place on the screen.

Find the number of the piece you want to turn or move. The number is there on the screen just below the piece. Find a key (in the top row of keys) which has the same number. Press the key once.

Now you will hear four notes – the first 4 notes of Humpty Dumpty's tune. This tells you that the computer is waiting to know what you want it to do. If you want to move the piece, decide which place you want it to go to. Then press the key which has the same number as this place. If you just want to turn the piece round, press any key which has a letter on it.

Before you begin to 'put Humpty together again', try moving and turning some pieces a few times. Do not worry, it is not possible to lose any of the pieces of this kind of jigsaw.

Now try to work out where each piece should go. One at a time, move each piece to its proper place. Make sure that they are all the right way up. When all the pieces are in their right places and are all the right way up, you will see all the pieces of Humpty Dumpty joining together again. Then Humpty will play you the whole of his tune.

If you want to hear the tune again, press any key. You can hear the tune as many times as you like.

Winning tactics

The sky is blue, so look for pieces that might have sky on them and move these to the top places of the screen. Turn them so that the sky is at the top of each piece.

Grass is green, so this is likely to be at the bottom of the picture.

Keying in

Some kind parent or elder brother or sister will have to key this in for the youngsters. They will also have to read the introduction to them and explain to them how to play.

Control characters used are:

CLEAR: line 20

Program design

30–210	Initialising, displaying jigsaw pieces.
220–250	Detecting if all pieces are in their correct positions and the right way up. If so, go to the final routine at line 460.
260–330	Requesting input.
350–390	Swapping two pieces.
400–450	Turning a piece the other way up.
460–500	Displaying the pieces put together to make the final picture.
510–540	Playing the whole tune.
900–910	Subroutine to calculate, the RAM address of the top left corner of each piece.
1000–1050	Subroutine to display a piece the right way up.
2000–2060	Subroutine to display a piece upside down.
3000–3040	Subroutine to play the first two notes of the tune (plays the next two notes if entered at line 3010 without a previous RESTORE).
3500–3560	Subroutine used to find the DATA for a single piece and then call the subroutines needed to display it.
4000–4060	Subroutine to control the sound generator.
5000–5170	DATA for the pieces.
6000–6040	DATA for the tune.

The program

```
10 REM ** MAGIC JIGSAW **
20 PRINT"□":G=1024:H=55296:S=54272:POKE
53280,6:POKE 53281,10
30 FOR J=0 TO 25:POKE S+J,0:NEXT:POKE S+
24,15:POKE S+6,240:POKE 649,1
40 FOR P=1 TO 9
50 N=INT(RND(1)*9)+1:F=0
60 FOR J=1 TO 9
70 IF F(J,0)=N THEN F=1
80 NEXT
90 IF F=1 THEN 50
100 F(P,0)=N
110 GOSUB 900
120 Q=INT(RND(1)*2)+1:F(P,1)=Q
130 IF N=1 THEN 150
140 FOR J=1 TO 48*(N-1):READ X:NEXT
150 ON Q GOSUB 1000,2000
160 RESTORE
170 NEXT
180 FOR J=0 TO 2
190 FOR K=0 TO 2
200 POKE G+287+9*K+320*J,49+K+3*J:POKE H
+287+9*K+320*J,1
210 NEXT:NEXT
```

```
220 FF=1:FOR N=1 TO 9
230 IF F(N,0)<>N OR F(N,1)<>1 THEN FF=0
240 NEXT
250 IF FF=1 THEN 460
260 RESTORE:GOSUB 3000
270 GET A$:IF A$="" THEN 270
280 P=VAL(A$):IF P<1 OR P>9 THEN 220
290 RESTORE:GOSUB 3000
300 GOSUB 3010
310 RESTORE
320 GET A$:IF A$="" THEN 320
330 R=VAL(A$):RESTORE
340 IF R=0 THEN 400
350 PT=F(P,0):QT=F(P,1)
360 F(P,0)=F(R,0):F(P,1)=F(R,1)
370 F(R,0)=PT:F(R,1)=QT
380 GOSUB 3500
390 P=R:GOSUB 3500:GOTO 220
400 IF F(P,0)=1 THEN 420
410 FOR K=1 TO (F(P,0)-1)*48:READ X:NEXT
420 IF F(P,1)=1 THEN F(P,1)=2:GOTO 440
430 F(P,1)=1:GOSUB 900
440 ON F(P,1) GOSUB 1000,2000
450 GOTO 220
460 PRINT"□":RESTORE
470 FOR N=1 TO 9
480 D=128+240*(INT((N+2)/3-1))+8*(N-INT(
(N-1)/3)*3-1)
490 GOSUB 1000
500 NEXT
510 FOR J= 1 TO 38:GOSUB 4000:NEXT
520 GET A$:IF A$="" THEN 520
530 RESTORE:FOR J=1 TO 432:READ X:NEXT
540 GOTO 510
900 D=47+320*(INT((P+2)/3-1))+9*(P-INT((
P-1)/3)*3-1)
910 RETURN
1000 FOR L=D TO 200+D STEP 40
1010 FOR M=0 TO 7
1020 READ X
1030 POKE G+L+M,160
1040 POKE H+L+M,X
1050 NEXT:NEXT
1060 RETURN
2000 FOR L=200+D TO D STEP -40
2010 FOR M=7 TO 0 STEP -1
2020 READ X
2030 POKE G+L+M,160
2040 POKE H+L+M,X
2050 NEXT:NEXT
2060 RETURN
3000 FOR J=1 TO 432:READ X:NEXT
3010 FOR J=1 TO 2
3020 GOSUB 4000
3030 NEXT
3040 RETURN
3500 IF F(P,0)=1 THEN 3540
3510 FOR J=1 TO (F(P,0)-1)*48
3520 READ X
3530 NEXT
3540 GOSUB 900
3550 ON F(P,1) GOSUB 1000,2000
3560 RESTORE
3570 RETURN
4000 READ LF,HF
4010 POKE S,LF:POKE S+1,HF
4020 READ X
```

```
4030 POKE S+4,17
4040 FOR K=1 TO 100*X:NEXT
4050 POKE S+4,16
4060 RETURN
5000 DATA 6,6,6,6,6,6,6,6,6,6,6,6,6,6,6,
6,6,6,6,6,6,6,6,6
5010 DATA 6,6,6,6,6,6,7,7,7,6,6,6,7,7,7,
7,6,7,6,7,7,0,7,7
5020 DATA 6,6,6,6,6,6,6,6,6,6,6,6,6,6,6,
1,6,6,6,6,6,6,6,1
5030 DATA 7,6,6,6,6,6,6,1,7,7,6,6,6,6,6,
6,0,7,7,6,6,6,6,6
5040 DATA 6,6,6,6,6,6,6,6,6,6,6,6,6,6,6,
6,1,1,1,6,6,6,6,6
5050 DATA 1,1,1,6,6,6,6,6,6,6,6,6,6,6,6,
6,6,6,1,6,6,6,6,6
5060 DATA 6,3,6,7,7,1,7,7,6,3,6,7,7,7,7,
7,6,6,3,7,7,7,7,7
5070 DATA 6,6,6,4,4,4,4,4,6,6,6,6,3,3,3,
3,2,2,2,2,2,3,2,2
5080 DATA 1,7,7,6,6,6,6,6,7,7,7,6,6,6,6,
6,7,7,7,6,6,6,6,6
5090 DATA 4,4,4,3,3,6,6,6,3,3,6,6,6,3,6,
6,3,2,2,2,2,7,2,2
5100 DATA 6,6,1,1,1,1,6,6,6,6,1,1,1,1,6,
6,6,6,6,6,6,6,6,6
5110 DATA 6,6,6,6,8,6,6,6,6,6,6,8,7,8,6,
6,2,2,2,2,8,2,2,2
5120 DATA 2,2,2,2,2,3,2,2,2,2,2,2,2,3,2,
2,2,2,2,2,9,9,2,2
5130 DATA 5,2,2,2,2,2,2,2,5,5,2,5,2,2,5,
2,5,5,5,5,5,5,5,5
5140 DATA 3,2,2,2,2,2,2,2,3,2,2,2,2,2,2,
2,9,9,2,2,2,2,2,2
5150 DATA 2,2,2,2,2,2,5,2,2,2,2,5,5,2,5,
2,5,5,5,5,5,5,5,5
5160 DATA 2,2,2,2,5,2,8,2,2,2,8,2,5,8,7,
8,2,8,7,8,5,2,8,2
5170 DATA 2,2,8,2,5,2,2,2,5,2,2,2,5,2,5,
2,5,5,5,5,5,5,5,5
6000 DATA 195,16,4,31,21,2,209,18,4,96,2
2,2,31,21,2,30,25,2,165,31,2,135,33,6
6010 DATA 31,21,4,30,25,2,96,22,4,49,28,
2,30,25,2,31,21,2,195,16,2,209,18,6
6020 DATA 31,21,2,96,22,2,30,25,2,96,22,
2,30,25,2,49,28,2,30,25,2,49,28,2
6030 DATA 165,31,2,135,33,4,135,33,1,162
,37,1,62,42,2,135,33,2,135,33,2
6040 DATA 193,44,2,193,44,2,62,42,2,162,
37,2,135,33,2,165,31,2,135,33,6
```

Variations

It is easy to replace the picture and tune with entirely new ones. To change the picture, follow these steps:

(1) Draw a grid of squares, 24 across and 18 down.

(2) Subdivide the grid into 9 large squares, each consisting of a grid of 8 × 6 smaller squares.

(3) Lightly sketch your design on this grid.

(4) Colour in the squares, using any or all of the sixteen colours:

(0) black	(8) orange
(1) white	(9) brown
(2) red	(10) pink
(3) cyan (blue–green)	(11) dark grey
(4) purple	(12) medium grey
(5) green	(13) light green
(6) blue	(14) light blue
(7) yellow	(15) light grey

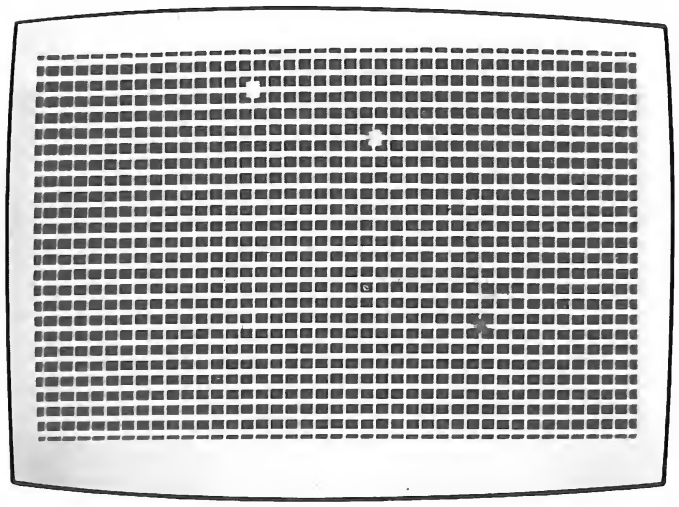
(5) when you have completed your design, take each of the 9 main squares in turn and work out a pair of DATA lines for each. The pair of lines will contain 48 numbers altogether, one for each of the smaller squares. Run along the top rows of the large squares from left to right, writing down the number of the colour each small square contains. Continue in this way to the second row, and so on down to the bottom row. Repeat this for each of the 9 large squares. In the end, you will have 18 DATA lines to replace lines 5000 to 5170 of this program.

The tune can be changed by altering the values in lines 6000 to 6040. The values are in threes

Low Frequency
High Frequency } as in Appendix M of the 64 User Manual.
Duration

You can adjust the tempo by altering the '100' in line 4040 to some other value. You also need to alter the maximum value for J in line 510 to the number of notes in your tune.

4
Cops and Robbers



This is a game for two players. One player takes the part of the cops, while the other takes the part of the robbers. The aim is for the cops to chase the robbers through a city and to try to catch them before they escape to their hideaway.

The streets of the city run north to south and east to west. The cops have the advantage of a faster car, but they must obey the traffic laws. At intersections, they may only go ahead or turn left. No right turns or U-turns are allowed. The robbers ignore all traffic laws so, although their car is slower than the cops' car, they are better able to manoeuvre.

You can choose the speed of the game. At its fastest, it is a hair-raising chase, through the streets. Played more slowly, it becomes an interesting game of strategy.

How to play

The player who is taking the part of the robbers sits on the left, the other player on the right. When asked to select the speed of the game, key in any number between '1' (the slowest) and '10' (the fastest).

Then press 'RETURN'. The screen displays the street grid. The intersection which is displayed in purple is the location of the robber's hideaway. This is randomly placed in the central region of the top half of the screen. The cops' car is blue, and appears at the bottom right corner. The robbers' car is red and is at the bottom left corner. Both cars are pointing north (to top of screen) to begin with.

When you are ready, press key 'G' and the game starts. Moves are paced by the computer. The pacing is indicated by alternate tones, like a police siren. It is the robbers' turn to move when the higher note is sounding. The robbers have first move. A turn lasts for as long as the note is sounding. If you do not make any move during that turn, you car is automatically moved one square forward. It is part of the game to remember the direction in which your car is facing.

Moves other than one square forward are made by pressing a key:

Robbers	A	turn left
	S	turn right
	Z	U-turn
'space bar'		stop
Cops	F3	two squares forward
	F5	turn left
	F7	stop

The effect of these keys lasts only for one turn. You can see that the cops have the option of moving one square forward (no key-press) or two squares forward (press function key F3).

Now the chase begins. The robbers must get to their hideaway before they are caught by the cops. To catch the robbers, the cops must come up from behind them and then move on to the same square. Note that both cars must be travelling in the same direction. If the cars are travelling in different directions, there is a head-on crash and neither player wins. A skilful cop may be able to avoid a crash by moving two squares forward, safely by-passing (jumping over) the robbers's car.

The game ends with escape (robbers win), capture (cops win), or collision (a draw). The final display tells you the number of moves for which the game lasted, excluding all moves when the robbers were in the country. The display also shows the greatest and least numbers of moves in the current series of games. You can score by seeing who is the robber able to evade capture for the greatest numbers of moves, or who is the cop who makes a capture in the fewest moves. Players take alternate turns at being cops or robbers.

Winning tactics

The balance between the speed advantage of the cops and the freedom of movement of the robbers makes this an interesting game. The cops should avoid having to change direction to the right when they are near the edge of the city. Turning right involves repeated left turns and this may take the car beyond the city limits, where is it all too easy to get lost.

Moving off the screen gives the robbers a chance to escape from view and to make their way stealthily to a point just outside the edge of town which is close to their hideout. Then a quick dash takes them safely home. The danger of this strategy is that a false move may send the car off in the wrong direction. The robbers gain no advantage by deliberately getting themselves lost, for moves are not counted while they are off the board. The cops must avoid becoming lost in the country at all costs, for this leaves the robbers free to make their way to their hideout without opposition.

Keying in

The value of LM in line 20 is '1E6', or 1 million! This is the initial setting of the 'least number of moves' variable.

Control characters used are:

CLEAR: lines 50, 660, 770, 800
CTRL/2: line 50
CRSR DOWN: lines 50(2), 660(2), 700(2), 710(2), 720(2), 730(3), 770(2), 800(2)
CRSR RIGHT: lines 50(2), 660(2), 700(2), 710(2), 720(2), 730(4), 770(2), 800(2)

Unlisted control characters are:

C=/1: lines 770, 800

Program design

20–30	Setting variables, etc.
40–60	Inputting speed.
70–200	Displaying street, hideout and cars.
210–220	Waiting to start.

230 Checking if robbers have been off screen for too long.
240 Re-entry point for each move.
260–280 Timed wait for robbers' move.
290 Clear robbers car from screen.
300–330 Analysing robbers' input.
340–430 Making robbers' move and displaying it.
460–480 Timed wait for cops' move.
490 Clear cops' car from screen.
500–530 Analysing cops' input.
540–630 Making cops' move and displaying it.
650–660 "Robbers escape" message.
670–720 Calculating and displaying moves.
730–750 Inviting a repeat game.
760–780 "Robbers captured" message.
790–810 "Head-on crash" message.
1000–1020 Subroutine for preventing RD (robbers' direction) from exceeding 3.
1100–1110 Subroutine for preventing CD (cops' direction) from exceeding 3.

The Program

```
10 REM ** COPS AND ROBBERS **
20 G=1024:H=55296:C=160:B=219:S=54272:LM
=1E6
30 FOR J=0 TO 24:POKE S+J,0:NEXT
40 POKE S+24,15:POKE S+5,18:POKE S+6,242
:POKE 53280,5:POKE 53281,8:M=0:MC=0:POKE
 S+3,1
50 INPUT"■■■■■■■■SPEED (1 TO 10)";A$
60 L=11-VAL(A$):IF L<1 OR L>10 THEN 50
70 PRINT"■":POKE 53281,7
80 FOR J=0 TO 999
90 POKE J+G,B
100 POKE J+H,0
110 NEXT
120 HX=INT(RND(1)*21)+10
130 HY=INT(RND(1)*12)
140 POKE H+HX+HY*40,4
150 RM=0:RX=0:RY=24:RD=0
160 CM=0:CX=39:CY=24:CD=0
170 RP=960:POKE G+RP,C
180 POKE H+RP,2
190 CP=999:POKE G+CP,C
200 POKE H+CP,6
210 GET A$:IF A$="" THEN 210
220 IF A$<>"G" THEN 210
230 IF MC=31 THEN MC=0:POKE S+4,64:POKE
G+CP,B:POKE H+CP,0:GOTO 150
240 J=0:M=M+1
250 POKE S+4,64:POKE S,135:POKE S+1,33:P
OKE S+4,65
```

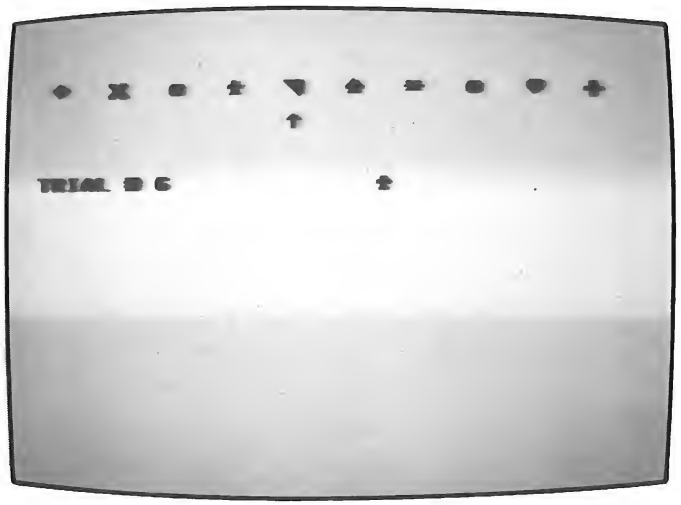
```
260 GET A$:IF A$="" AND J<20*L THEN J=J+
1:GOTO 260
270 IF J=20*L THEN RM=1:GOTO 290
280 IF A$<>"A" AND A$<>"Z" AND A$<>"S" A
ND A$<>" " THEN 260
290 POKE G+RP,B:POKE H+RP,0
300 IF A$="A" THEN RM=1:RD=RD+1:GOSUB 10
00
310 IF A$="Z" THEN RM=1:RD=RD+2:GOSUB 10
00
320 IF A$="S" THEN RM=1:RD=RD-1:GOSUB 10
00
330 IF A$=" " THEN RM=0
340 IF RD=0 THEN RY=RY-RM
350 IF RD=1 THEN RX=RX-RM
360 IF RD=2 THEN RY=RY+RM
370 IF RD=3 THEN RX=RX+RM
380 IF RX=HX AND RY=HY THEN 650
390 IF RX=CX AND RY=CY AND CD<>RD THEN 7
90
400 IF RX<0 OR RX>39 OR RY<0 OR RY>24 TH
EN M=M+1:MC=MC+1:GOTO 440
410 RP=RX+RY*40
420 POKE G+RP,C
430 POKE H+RP,2
440 J=0
450 POKE S+4,64:POKE S,30:POKES+1,25:POK
E S+4,65
460 GET A$:IF A$="" AND J<20*L THEN J=J+
1:GOTO 460
470 IF J=20*L THEN CM=1:GOTO 490
480 IF A$<>CHR$(134) AND A$<>CHR$(135) A
ND A$<>CHR$(136) THEN 460
490 POKE G+CP,B:POKE H+CP,0
500 IF CX=HX AND CY=HY THEN POKE H+CP,4
510 IF A$=CHR$(134) THEN CM=2:GOSUB 1100
520 IF A$=CHR$(135) THEN CM=1:CD=CD+1:GO
SUB 1100
530 IF A$=CHR$(136) THEN CM=0
540 IF CD=0 THEN CY=CY-CM
550 IF CD=1 THEN CX=CX-CM
560 IF CD=2 THEN CY=CY+CM
570 IF CD=3 THEN CX=CX+CM
580 IF RX=CX AND RY=CY AND CD=RD THEN 76
0
590 IF RX=CX AND RY=CY AND CD<>RD THEN 7
90
600 IF CX<0 OR CX>39 OR CY<0 OR CY>24 TH
EN 640
610 CP=CX+CY*40
620 POKE G+CP,C
630 POKE H+CP,6
640 GOTO 230
650 POKE 53280,2:POKE 53281,2
660 PRINT"■■■■■■ROBBERS ESCAPE
670 POKE S+4,64:POKE S+24,0
680 IF MM<M THEN MM=M
690 IF LM>M THEN LM=M
700 PRINT"■■■■■GAME LASTED ";M;" MOVES"
710 PRINT"■■■■LONGEST GAME IS ";MM;" MOV
ES"
720 PRINT"■■■■SHORTEST GAME IS ";LM;" MO
VES"
730 PRINT"■■■■■■■■■PRESS SPACE BAR TO CON
TINUE"
740 GET A$:IF A$<>" " THEN 740
750 GOTO 30
```

```
760 POKE 53280,6:POKE 53280,6
770 PRINT"■■■■■ROBBERS CAPTURED"
780 GOTO 670
790 POKE 53280,2:POKE 53281,6
800 PRINT"■■■■■COPS AND ROBBERS CRASHED
 HEAD-ON"
810 GOTO 670
1000 IF RD>3 THEN RD=RD-4
1010 IF RD<0 THEN RD=RD+4
1020 RETURN
1100 IF CD>3 THEN CD=CD-4
1110 RETURN
```

Variations

You could add sound effects, such as squealing tyres when the cars turn corners, gun-shots when the cars are close, and an enormous crash when the cars collide. Lines 150 and 160 could be altered to position the cars differently (or at random) at the beginning of the game. CX, CY, RX and RY are the variables concerned. The limit on the number of moves (MC) allowed to the robbers off screen can be changed at line 230.

5
Mind Over Electrons



Is it possible to communicate with a computer directly, without using the keyboard or joystick or some other piece of hardware? Can your mind, unaided by material connections, influence what goes on inside the Commodore 64? Most people would say that this is complete nonsense, but there are others who equally strongly assert that it is possible. Here is a program to help put the idea to the test.

How to play

The program is based on choosing one from among ten of the graphics symbols, such as the 'heart', the 'disk' or the 'diamond'. These are displayed across the top of the screen. Before this happens, the computer asks if it is to choose. If you answer 'Y', it chooses a symbol and will then ask you to guess which one it has chosen. Can you guess correctly more often than you should by pure chance? If you can, you are either able to detect what has happened inside the computing circuits, or can predict what symbol will appear on the screen before it has been displayed.

If you answer 'N' to the question it is *you* who does the choosing.

You select a symbol and then try to make the computer select the same symbol. If you concentrate hard while it is choosing, you may be able to force it to choose the one you have decided on.

As soon as you have answered Y or N, the row of symbols is displayed. A grey arrow, below the row, points to the one to be selected. You can move this arrow along from symbol to symbol by pressing the front 'function' key (key F7). Assuming it is the computer which is choosing, try to guess which symbol it has chosen, and move the arrow to point to that symbol. The computer has *already chosen* a symbol and it is displayed on the screen at a point to the right of the message 'TRIAL#1'. You cannot see it because it is coloured light grey and the background is light grey too.

When you have made your guess, press key F5. The colour of the symbol is now changed to black, so that you can see it. If you have guessed correctly, the computer adds to your score.

After a short pause, the symbol disappears and the message changes to 'TRIAL#2'. Choose another symbol if you like, or stay with the same one; then press F5. There are 100 trials altogether, out of which you would expect to get 10 correct by guesswork. If you can score more than 15, you will be doing very well.

After the 100th trial, your score is displayed. The computer also works out how likely it is that you could have got such a score just by chance. If the score is 12 or less this is not unusual, but a score as high as 15 normally occurs in only 1 in 10 (10%) of games. The chances of scoring 18 or more by guesswork are only 1 in 1000. If you score 18 or more it might be taken to show that you really can influence the computer's inner workings.

If you answer 'N', the order of operations is reversed. The computers waits for you to select a symbol. First you press K7 to move the arrow to the symbol you are selecting. Then, you press K5 to tell the computer that this is the selected symbol. The computer then selects and displays a symbol. Concentrate hard and you may be able to 'force' it to choose the same one as you have already chosen.

Keying in

Control characters used are:

CLEAR: lines 40, 60, 350
CRSR DOWN: lines 40(2), 60(2), 90(8), 350(2), 360(2), 430(2), 450(2)

CRSR RIGHT: lines 40(2), 60, 350(2), 360(2), 430(2), 450(2)
CTRL/1: line 60
HOME: line 90
CTRL/2: line 350

Program design

40–50	Deciding who chooses.
60–70	Displaying symbols and the pointer.
100–210	The computer chooses a symbol at random: Q is the random number, T is the screen code of the corresponding symbol.
220	Displaying the symbol invisibly.
230–280	The user selects a symbol.
290	The symbol made visible.
300	Score increased if symbols are the same.
320	Symbol made invisible again, ready for next trial.
350	Displaying the score.
360–440	Working out the probability.
450–480	Inviting repeat play.
1000–1030	Subroutine to move the pointer.

Points of interest

If you play this game with the computer choosing first, the program runs in the order in which it is listed. If you play it with you choosing first, flag F is zero and several routines are taken out of order. It then becomes a terrible example to programmers, with GOTO's sending the computer jumping backward and forward through the listing in a most unbecoming manner!

The probabilities (or rather, percentages) worked out in lines 370 to 420 are based on a standard statistical test, usually known as the 'chi-squared test'.

The program

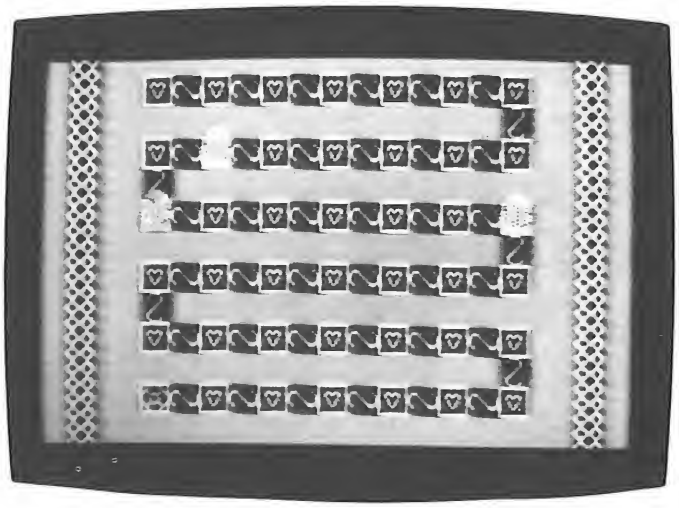
```
10 REM ** MIND OVER ELECTRONS **
20 G=1024:H=55296:S=G+343:R=H+343
30 POKE 53280,15:POKE 53281,15
40 INPUT"■■■■■■COMPUTER CHOOSES (Y/N)";A$
```

```
50 IF A$="Y" THEN F=1
60 PRINT"    X    O    ↑    ◥    ♠    *
  ●    ●    +"
70 P=161:POKE G+P,30:POKE H+P,11
80 FOR J=1 TO 100
90 PRINT"TRIAL #";J
100 IF F=0 THEN 230
110 Q=4*INT(RND(1)*10)+161
120 IF Q=161 THEN T=90
130 IF Q=165 THEN T=86
140 IF Q=169 THEN T=87
150 IF Q=173 THEN T=88
160 IF Q=177 THEN T=95
170 IF Q=181 THEN T=65
180 IF Q=185 THEN T=42
190 IF Q=189 THEN T=81
200 IF Q=194 THEN T=83
210 IF Q=197 THEN T=91
220 POKE R,15:POKE S,T:IF F=0 THEN 290
230 GET A$
240 IF A$="" THEN 230
250 IF A$=CHR$(135) AND F=0 THEN 110
260 IF A$=CHR$(135) THEN 290
270 IF A$=CHR$(136) THEN GOSUB 1000
280 GOTO 230
290 POKE R,0
300 IF P=Q THEN N=N+1
310 FOR K=1 TO 4000:NEXT
320 POKE R,3
330 NEXT
340 POKE 53280,1:POKE 53281,0
350 PRINT"SCORE:";N;"OUT OF 100"
360 IF N<14 THEN PRINT"THIS IS NOT U
NUSUAL":GOTO 440
370 IF N=14 THEN N=20
380 IF N=15 THEN N=10
390 IF N=16 THEN N=5
400 IF N=17 THEN N=2
410 IF N=18 THEN N=1
420 IF N>18 THEN N=.1
430 PRINT"THE PROBABILITY IS";N;"%"
440 N=0
450 PRINT"PRESS SPACE BAR TO REPEAT"
460 GET A$:IF A$<>" " THEN 460
470 GOTO 30
1000 POKE P+G,32
1010 P=P+4:IF P=201 THEN P=161
1020 POKE G+P,30:POKE H+P,11
1030 RETURN
```

Variations

Do not be tempted to reduce the number of trials. A hundred trials is the very minimum that could possibly show any real effect. You could reduce the number of symbols to, say, 4. Some of lines 120 to 210 would need deleting and some would need amending. The first numbers in lines 360 to 420 should be altered to: 31,31,32,33/34, 35, 36,36. By '33/34' we mean that line 390 should be 'IF N=33 OR N=34 THEN N=5'.

6
Nimble Thimble



This is a new version of the traditional game of Nim. It is a game for two players. Over the years, the game has been played in a variety of forms. It can be played with piles of stones on the mud floor of a straw hut, with piles of matches on the living-room table, or with counters on a playing board. This version uses the Commodore 64's sprites to play with coloured thimbles on a delicately embroidered cloth.

How to play

Firstly, the screen displays the playing cloth, on which is marked a meandering embroidered path, leading to a 'home' at the bottom right-hand corner. Three thimbles, coloured cyan (turquoise), purple and green, are placed at random on the path.

The computer tells you whose turn it is to play. The first person plays when the border of the screen has become brown. When the border of the screen becomes pink, it is the second player's turn. When it is your turn, you must move any one (but only one) of the thimbles one step along the path in the direction of home. You make

your move by pressing one of the three keys, '4', '5' or '6' at the top of the keyboard. These have the colours of the thimbles marked on them, so you will know which to press. When you press a key, the computer moves the thimble to the next step on the path.

Two thimbles cannot occupy the same position and thimbles cannot jump over one another. The computer does not allow you to make these illegal moves. When a thimble is moved on to 'home', it is removed from the cloth.

The aim of the game is to be the last person to move a thimble on to 'home'. The name of the winner is displayed at the end of the game.

To play again, press the space bar. It is fairer if players take turns to play first.

Winning tactics

This is a game in which it makes for greater interest if you develop your own tactics. If any more is said on this topic, there is a danger of literally, 'giving the game away'.

Keying in

Line 100 contains (O)s (letter 'oh's), not zeroes.

The control characters used are:

CLEAR: lines 20, 920
HOME: line 130
CTRL/4: line 20
CTRL/2: line 920
CRSR DOWN: line 920(2)

Program design

20–30	Setting initial conditions.
40–50	READing data into RAM for the sprites and user-defined graphics.
60–70	More initialising.
80–200	Displaying the cloth.
210–320	Calculating a table in B%() to control moves.
330–360	Selecting random positions for the thimbles and rejecting the selection if two thimbles are on the same step.

370–400 Displaying the thimbles.
410–500 Asking for input and checking that it is valid.
510–540 Calculating the coordinates of the new position of the moved thimble.
550–610 Move the thimble sprites.
620–660 Detecting a thimble moved to home and preparing for the next turn.
800–860 Move to 'home', made and registered.
870–910 Checking if last thimble.
920–960 Winning display.
970–990 Preparing for re-play.
1000–1010 Subroutine for building display.
1500–1520 Subroutine for switching on sprites which are still in play.
2000–2020 DATA for sprites.
3000–3070 DATA for special graphics.

Points of interest

This program uses multicolour sprite graphics for the thimbles. They all share the same design, and the two 'multicolour' colours, but differ in their individual sprite colours. The pattern of the cloth is built up from 17 special graphics designs, assigned to keys J to Z. This accounts for the letters in D$, E$, F$, G$ (lines 100–120).

The program

```
10 REM ** NIMBLE THIMBLE **
20 PRINT"□◣":POKE 53280,3:POKE 53281,2:P
OKE 649,1
30 POKE 52,48:POKE 56,48:CLR
40 FOR J=12288 TO 12350:READ X:POKE J,X:
NEXT
50 FOR J=12368 TO 12503:READ X:POKE J,X:
NEXT
60 POKE 53272,(PEEK(53272)AND240)+12
70 G=1024:H=55296:V=53248:DIM B%(2,41)
80 FOR J=2 TO 962 STEP 40:POKE G+J,10:PO
KE H+J,3:POKE G+J+1,10:POKE H+J+1,3
90 POKE G+J+34,10:POKE H+J+34,3:POKE G+J
+35,10:POKE H+J+35,3:NEXT
100 D$="OPSTOPSTOPSTOPSTOPSTOPSTOP"
110 E$="QRUVQRUVQRUVQRUVQRUVQRUVQR"
120 F$="WX":G$="YZ"
130 PRINT"■":GOSUB 1000
140 PRINTTAB(31)F$:PRINTTAB(31)G$:GOSUB
1000
```

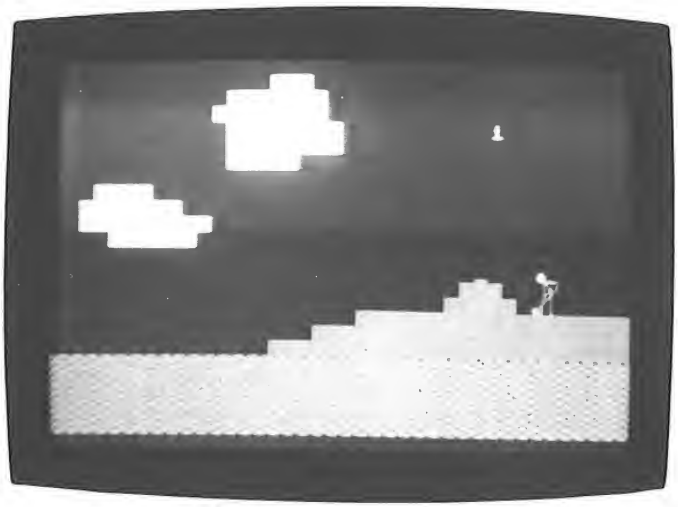
```
150 PRINTTAB(7)F$:PRINTTAB(7)G$:GOSUB 10
00
160 PRINTTAB(31)F$:PRINTTAB(31)G$:GOSUB
1000
170 PRINTTAB(7)F$:PRINTTAB(7)G$:GOSUB 10
00
180 PRINTTAB(31)F$:PRINTTAB(31)G$
190 PRINTTAB(7)"KLSTOPSTOPSTOPSTOPSTOPST
OP"
200 PRINTTAB(7)"MNUVQRUVQRUVQRUVQRUVQRUV
QR"
210 FOR K=0 TO 35 STEP 7
220 FOR L=0 TO 6
230 B%(0,K+L)=16:B%(1,K+L)=53+32*K/7:B%(
2,K+L)=1
240 NEXT:NEXT
250 FOR K=0 TO 28 STEP 14
260 FOR L=0 TO 5
270 B%(0,K+L)=80+32*L:B%(2,K+L)=0
280 NEXT:NEXT
290 FOR K=7 TO 35 STEP 14
300 FOR L=5 TO 0 STEP -1
310 B%(0,K+L+1)=80+32*(5-L):B%(2,K+L+1)=
0
320 NEXT:NEXT
330 FOR J=0 TO 2
340 T(J,0)=INT(RND(1)*29):T(J,1)=1
350 NEXT
360 IF T(0,0)=T(1,0) OR T(1,0)=T(2,0) OR
 T(2,0)=T(0,0) THEN 340
370 POKE 2040,192:POKE 2041,192:POKE 204
2,192
380 FOR TP=0 TO 2:POKE V+TP*2,B%(0,T(TP,
0)):POKE V+TP*2+1,B%(1,T(TP,0))
390 POKE V+16,(PEEK(V+16)AND(255-2↑TP))+
B%(2,T(TP,0))*2↑TP:NEXT:GOSUB 1500
400 POKE V+28,7:POKE V+37,9:POKE V+38,1
:POKE V+39,3:POKE V+40,4:POKE V+41,5
410 P=1:POKE 53280,9
420 GET A$:IF A$="" THEN 420
430 TP=VAL(A$)-4
440 IF TP<0 OR TP>2 THEN 420
450 IF T(TP,1)=0 THEN 420
460 TN=T(TP,0)+1
470 F=0:FOR J=0 TO 2
480 IF T(J,0)=TN AND J<>TP THEN F=1
490 NEXT
500 IF F=1 THEN 420
510 XS=B%(0,T(TP,0))+B%(2,T(TP,0))*256
520 XF=B%(0,TN)+B%(2,TN)*256
530 YS=B%(1,T(TP,0)):YF=B%(1,TN)
540 XD=XF-XS:YD=YF-YS
550 FORJ=1TO20
560 X=XS+XD*J/20
570 Y=INT(YS+YD*J/20)+ABS(10-J)-10
580 Z=INT(X/256):X=INT(X-Z*256)
590 Z=(PEEK(V+16)AND(255-2↑TP))+Z*2↑TP
600 POKE V+TP*2,X:POKE V+16,Z:POKE V+TP*
2+1,Y
610 GOSUB 1500:POKE V+28,7:NEXT
620 T(TP,0)=TN
630 IF TN=41 THEN 900
640 IF P=2 THEN 410
650 P=2:POKE 53280,10
660 GOTO 420
```

```
800 FOR J=1 TO 10
810 T(TP,1)=0:GOSUB 1500
820 FOR K=1 TO 50:NEXT
830 T(TP,1)=1:GOSUB 1500
840 FOR K=1 TO 50:NEXT
850 NEXT
860 T(TP,1)=0:GOSUB 1500
870 F=1:FOR J=0 TO 2
880 IF T(J,1)=1 THEN F=0
890 NEXT
900 T(TP,0)=42
910 IF F=0 THEN 640
920 PRINT"[illegible]":POKE 53272,21
930 PRINTTAB(10)"PLAYER NO. ";P;" WINS"
940 POKE V+16,0
950 POKE V,80:POKE V+1,180:POKE V+2,160:
POKE V+3,180:POKE V+4,240:POKE V+5,180
960 POKE V+23,7:POKE V+29,7:POKE V+21,7
970 GET A$:IF A$<>" " THEN 970
980 POKE V+21,0:POKEV+23,0:POKE V+29,0
990 CLR:GOTO 20
1000 PRINTTAB(7)D$:PRINTTAB(7)E$
1010 RETURN
1220 RETURN
1500 X=T(0,1)+2*T(1,1)+4*T(2,1)
1510 POKE V+21,X
1520 RETURN
2000 DATA 2,128,0,10,160,0,42,168,0,42,1
68,0,26,172,0,54,180,0,29,220,0
2010 DATA 55,116,0,29,220,0,55,116,0,221
,221,0,119,119,0,221,221,0,119,119,0
2020 DATA 221,221,0,183,118,0,173,218,0,
170,170,0,170,170,0,42,168,0,10,160,0
3000 DATA 195,102,60,24,24,60,102,195,25
5,132,136,144,168,197,130,132
3010 DATA 255,33,17,9,21,163,65,33,132,1
30,197,168,144,136,132,255
3020 DATA 33,65,163,21,9,17,33,255,255,1
28,128,128,134,137,136,132
3030 DATA 255,1,1,1,97,145,17,33,132,130
,130,129,128,128,128,255
3040 DATA 33,65,65,129,1,1,1,255,0,0,0,0
,56,68,130,129
3050 DATA 0,0,0,0,4,2,1,1,128,128,64,32,
0,0,0,0
3060 DATA 129,65,34,28,0,0,0,0,3,4,8,0,0
,0,0,0
3070 DATA 192,32,16,16,16,32,64,128,1,2,
4,8,8,8,4,3,0,0,0,0,0,16,32,192
```

Variations

Let the player who moves the last thimble ‘home’ be the loser. It is not too difficult to analyse this game and find a winning strategy. Having done this, there is plenty of room in RAM for you to program the 64 to play against you – and it will win!

7
Sandcastle



This is another game for the younger members of the family. If it is a rainy day and you cannot get to the beach, you can build a sandcastle with the computer. Of course, you must finish the castle before the tide comes in.

How to play

When the computer asks "SPEED?", press one of the keys '1' to '5'. If you have not played this game before, press '1'. Pressing other numbers makes the computer work more quickly. If you press '5', it goes very fast. You will need plenty of practice to keep up with it at speed 5. When you have chosen the speed, press the 'RETURN' key.

Now you will see the beach and soon will hear the sound of the waves as the tide comes in. After the first wave has come in, you will see a child on the beach. This is you, ready to make the sandcastle. Look in the sky above the child's head. Very soon, a number appears there. As quickly as you can, find the key which has the same number and press it, once. If you have been quick enough, the child digs up a spadeful of sand and starts to make the sandcastle. The

amount of sand dug up depends on how big the number is. The bigger the number, the more sand is dug for the castle. But the bigger the number, the quicker you have to be to press the key in time.

As soon as the child has dug up the sand and started to build the castle, the number disappears from the sky. Then you hear another crashing wave, as the tide comes in a bit more. After a short time, another number appears in the sky at the same place. It may be the same number as before, though it is more likely to be different. Find the key which has this number on it, and press it once. If you are in time, the child digs more sand and adds it to the pile.

Each time a number appears and you press the right key, the child digs more sand and the castle gets bigger. Each time the number appears the tide comes in a little bit more. This game is a race to finish the castle before the tide gets to it.

If lots of big numbers (such as '7', '8' and '9') appear in the sky, and if you are quick when you see them, the sandcastle will soon be finished. Then a brightly coloured flag appears on top of the castle and the child stands proudly beside it. This shows you that you have won.

If the tide reaches the castle before it is finished, it is washed away and the game ends. Press any key to start the game again and try to beat the tide.

Winning tactics

The bigger the number, the more sand is added to the castle. Try hard to press the right key quickly when one of the bigger numbers appears in the sky. Take care not to press any key twice, or to press the wrong key by mistake.

Keying in

This is another program for parents, or an elder brother or sister, to key in for the younger family members. It is quite a lot of work, but the animated graphics and sound effects provide compensation for the effort required. The 'A's in lines 160 to 200 are actually re-defined as solid blocks of colour. They appear as a solid block of white, and make up the clouds at the top of the scene.

Control characters used are:

CLEAR: lines 20, 160
CURSOR DOWN: lines 110(2), 160, 180, 210(3)
CTRL/3: line 20
CURSOR RIGHT: line 110(2)
CTRL/2: lines 160, 280
CTRL/8: line 210

Program design

20–80 Placing codes for redefined characters in RAM.
90–100 Setting key variables.
110–130 Requesting speed input.
140–270 The initial display routine.
280–290 The first wave comes in.
300–380 POKEing colour codes, ready for the display of the child.
390–400 POKEing spaces where the castle is to be.
410 to subroutine 1000, to display child digging.
420–440 Displaying a random number, in white.
450–470 Waiting a limited length of time for input.
480 the next wave begins.
490 to subroutine 2000, if answer is correct:
500 to end of game, if castle is complete.
510–540 Clear number and go to subroutine 3000 for next wave.
550–560 Delay, then return for next number.
570 Washing away the castle.
580–590 Waiting to play again.
600–660 Displaying the flag, with child standing beside castle.
1000–1020 Subroutine to display child digging.
1500–1520 A wave begins.
2000–2150 Displaying child throwing sand on castle, then add sand to castle.
3000–3070 Wave display, with crashing sound effect.
4000–4120 DATA for user-defined graphics.

Points of interest

This program has 36 user-defined graphics. These are allotted to keys, as in this list, where they are in order of screen code:

1	(A) solid block of colour
2–7	(B–G) child digging
8–16	(H–P, with G) child throwing sand
17	(Q) flag mast
18–19	(R–S) the flag
20–26	T–Z) child standing by castle
27	( ) block with two wave designs
28	(£) block with one wave design
29	( ) crest of wave
30–36	(–$) spaces, used for sandcastle

The spaces (except for character 32, which is the normal space character and is reserved for use in the main display routines and for clearing away unwanted graphics) are used for the sandcastles. They are defined by placing zeroes in RAM from address 12528 to 12583. The castle is a 'pile' of these spaces so arranged that the ones at the top of the castle are those near the start of this block of memory. Because these begin as spaces, nothing appears on the screen. It retains its blue background colour. It is then only necessary to POKE 255 into successive bytes (line 2050 to 2100) to cause these spaces to be gradually transformed to solid blocks of yellow. As the bytes are changed from '0' to '255', the corresponding area of screen changes little by little from blue to yellow. In this way, the castle is gradually built up.

If the player loses, a series of zeroes are POKEd back into this part of memory (line 570) to convert the blocks back to spaces again, and so 'wash away' the castle.

The program

```
10 REM ** SAND CASTLE **
20 PRINT"■■":POKE 53280,6:POKE 53281,7:P
OKE 52,48:POKE 56,48:CLR
30 POKE 56334,PEEK(56334)AND254:POKE 1,P
EEK(1)AND251
40 FOR J=12296 TO 12303:POKE J,255:NEXT
50 FOR J=12304 TO 12527:READ X:POKE J,X:
NEXT
60 FOR J=12528 TO 12583:POKE J,0:NEXT
70 FOR J=0 TO 79:X=PEEK(53632+J):POKE 12
584+J,X:NEXT
80 POKE 1,PEEK(1)OR4:POKE 56334,PEEK(563
34)OR1
90 G=1024:H=55296:S=54272:SE=G+920:R=125
83
100 FOR J=0 TO 24:POKE S+J,0:NEXT:POKE S
+24,15
110 INPUT"■■■■SPEED (1-5)";L$
```

```
120 L=VAL(L$):IF L<0 OR L>5 THEN PRINT"□
":GOTO 110
130 L=6-L
140 POKE 53272,(PEEK(53272)AND240)+12
150 POKE 53280,2:POKE 53281,6
160 PRINT"□■■";TAB(15)"AAA":PRINTTAB(12)
"AAAAAAA"
170 PRINTTAB(11)"AAAAAAAA":PRINTTAB(12)"
AAAAAAAA"
180 PRINTTAB(12)"AAAAAAAA":PRINTTAB(12)"
AAAAA■"
190 PRINTTAB(3)"AAAA":PRINTTAB(2)"AAAAAA
A"
200 PRINTTAB(2)"AAAAAAAAA":PRINTTAB(4)"A
AAAAA"
210 PRINT"■■■■"
220 FOR J=21 TO 3 STEP -3
230 FOR K=1 TO J
240 PRINT" ";:NEXT
250 FOR K=J+1 TO 40
260 PRINT"A";
270 NEXT:NEXT
280 PRINT"■":GOSUB 1500
290 SC=27:GOSUB 3000
300 FOR J=0 TO 2
310 FOR K=0 TO 3
320 POKEH+512+J+40*K,1
330 NEXT:NEXT
340 POKEH+552,7
350 FOR J=0 TO 4
360 FOR K=0 TO 2
370 POKEH+547+J+40*K,7
380 NEXT:NEXT
390 POKEG+549,30:POKEG+588,33:POKEG+589,
34:POKEG+590,33
400 POKEG+627,31:POKEG+628,35:POKEG+629,
36:POKEG+630,35:POKEG+631,31
410 GOSUB 1000
420 POKEH+190,1
430 N=INT(RND(1)*10)
440 POKEG+190,N+37
450 J=0
460 F=0:GET A$
470 IF A$="" THEN J=J+1:F=1:IF J<(1000*L
)/(N+10) THEN 460
480 GOSUB 1500
490 IF VAL(A$)=N AND F=0 THEN GOSUB 2000
500 IF R<=12528 THEN 600
510 POKEG+190,32
520 IF SC=27 THEN SC=28:SE=SE-40:GOTO 54
0
530 SC=27
540 GOSUB 3000
550 FOR K=1 TO 1000:NEXT
560 IF SE>G+640 THEN 430
570 FOR J=12528 TO 12583:POKE J,0:FOR K=
1 TO 50:NEXT:NEXT
580 GET A$:IF A$<>CHR$(32) THEN 580
590 RESTORE:PRINT"□":POKE 53272,21:GOTO
20
600 POKEG+509,17:POKEH+509,4
610 POKEG+469,18:POKEG+470,19:POKEH+469,
4:POKEH+470,4
620 POKEG+553,32:POKEG+593,32:POKEG+633,
32
630 POKE H+552,1:POKE G+554,32:POKE G+59
4,32:POKE G+634,32
```

```
640 POKE G+511,20:POKE G+512,21:POKE G+5
51,22:POKE G+552,23:POKE H+511,1
650 POKE G+591,24:POKE G+592,25:POKE G+6
32,26
660 POKE S+4,128:POKE G+190,32:GOTO 580
1000 POKE G+553,2:POKE G+554,3:POKE G+59
3,4:POKE G+594,5
1010 POKE G+633,6:POKEG+634,7
1020 RETURN
1500 POKE S+5,196:POKE S+6,196
1510 POKE S,214:POKE S+1,28:POKE S+4,129
1520 RETURN
2000 POKE G+513,8:POKE G+514,9
2010 POKE G+552,10:POKE G+553,11:POKE G+
554,12
2020 POKE G+592,13:POKE G+593,14:POKE G+
594,15
2030 POKE G+633,16
2040 FOR J=1 TO 500:NEXT
2050 FOR K=0 TO N
2060 IF R<12528 THEN 2100
2070 POKE R,255
2080 R=R-1
2090 IF R=12551 THEN R=R-8
2100 NEXT
2110 POKE G+513,32:POKE G+514,32
2120 POKE G+552,32:POKE G+553,2:POKE G+5
54,3
2130 POKE G+592,32:POKE G+593,4:POKE G+5
94,5
2140 POKE G+633,6
2150 RETURN
3000 FOR J=0 TO 38
3010 POKE SE+J,29:POKE H-G+SE+J,1
3020 FOR K=1 TO 80:NEXT
3030 POKE H-G+SE+J,1:POKE SE+J,SC
3040 IF J=30 THEN POKE S+4,128
3050 NEXT
3060 POKE SE+39,SC:POKE H-G+SE+J,1
3070 RETURN
4000 DATA 0,0,28,62,62,62,31,3,0,0,0,0,0
,0,248,4,2,2,2,2,2,3,2,5
4010 DATA 132,68,100,104,176,32,96,160,5
,6,4,12,116,244,244,108
4020 DATA 32,32,32,32,32,32,32,48
4030 DATA 24,60,60,126,126,62,28,4,0,0,0
,0,0,0,56,196
4040 DATA 0,0,0,96,240,120,240,104,7,28,
102,130,66,65,33,33
4050 DATA 4,4,4,4,4,4,24,96,0,0,0,4,31,1
26,252,48,17,23,25,97,130,4,8,16
4060 DATA 128,0,128,128,64,64,64,32,16,8
,8,8,4,4,4,12,48,48,48,48,48,48,48,48
4070 DATA 63,63,63,48,48,63,63,63,252,25
2,252,0,0,252,252,252,0,0,1,3,23,39,39
4080 DATA 39,0,0,128,192,224,224,224,224
,35,32,16,15,0,0,0,0
4090 DATA 192,128,128,240,143,129,129,12
9
4100 DATA 5,21,19,16,16,16,16,48,130,130
,130,130,66,34,18,18,18,18,18,18,19,19
4110 DATA 19,51,231,219,189,126,231,219,
189,126,255,255,255,255,231,219,189,126
4120 DATA 225,219,183,119,251,219,173,11
8
```

Variations

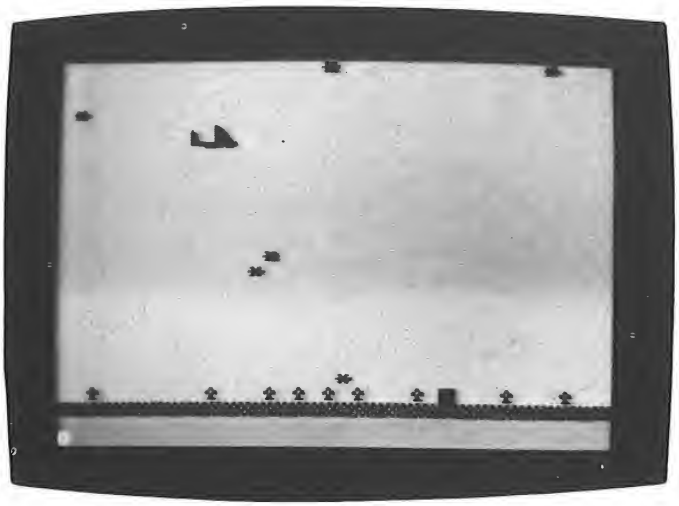
This program originated as a game but there are obvious educational applications. Even as it stands, it could help teach a child to match numerals. A simple extension would be to make it display letters of the alphabet too. It could easily be adapted to display a mathematical expression for evaluation, such as:

```
3 + 6
```

Sand is added to the castle if the child keys in the correct answer.

The range of delays provided by choosing the speed at line 110 may not give enough time for a child to work out the answer. In this event, the value '1000' in line 470 can be increased.

8
Bombing Run



You are on a mission to strike at enemy ground targets. Their territory is heavily defended by anti-aircraft weapons. Your mission continues until you are eventually shot down – but, before that happens, you must hit as many targets as possible. Dropping the bombs at just the right moment is difficult enough. To have to weave in and out of bursts of gunfire as well is liable to make you lose both your concentration and your aim.

How to play

You are first asked at which level you wish to play. The level you choose decides how heavy the anti-aircraft fire will be. At level 1 it is sporadic; you will have ample time to aim your bombs and to take evasive action against most of the bursting shells. At level 10 you will receive an intense barrage of fire. You are not likely to survive long unless you are a particularly skilful pilot.

Press 'RETURN'. The screen clears to display an evening sky, and you hear the drone of the bomber's engines. Soon the bomber appears on the left-hand side of the screen. You can gauge your

speed by watching the trees rush past on the ground below. In the sky above you see the flashes and smoke clouds from bursting shells. If you fly into one of these, you are shot down. You change height by using two of the function keys on the right-hand side of the keyboard. Pressing key F5 makes the bomber climb, while pressing key F7 makes it dive. If you climb to the top of the screen, you avoid almost all of the gunfire, but bombing is more difficult from that height. If you go too low, you will crash into the trees.

When the run first begins you hear a rapid bleeping. This is a sign that no target is in range. Soon the bleeping becomes slower indicating that you are on a bombing run. At the same time a number appears at the bottom left corner of the screen. The number is '9' to begin with and counts down to zero, in time with the bleeps. Just before the tenth bleep, when the number changes to zero, the target appears on the right-hand edge of the screen.

To release a bomb, press key F3. You will hear the whine as it falls and see the flash on the ground just below the bomber when it hits the ground. You also hear the sound of it exploding. If it hits the target, there is a bright flash, all around the screen.

The higher the bomber, is above ground when the bombs are released, the longer they take to fall. They must therefore be released earlier. If the bomber is near the top of the screen you may need to press key F3 when the number is showing '2' or '1'.

Sooner or later you are bound to be shot down. When this happens, the screen clears and you are given the results of your mission.

Winning tactics

Flying high is one way of avoiding the guns, but it is harder to hit the target from a great height. A low-level attack makes aiming easier, but a quick swerve to avoid a bursting shell may crash you into a tree instead.

You can drop bombs at any time. It is worth while trying a few practice runs, aiming at the passing trees. You can then gauge how far ahead the target should be, for any given altitude of the bomber. As you become more proficient, try to put every bomb where it belongs – on an enemy target!

Keying in

This program incorporates two machine code routines for moving the trees, targets and shellbursts past the bomber at high speed. The routines are loaded into memory from the DATA statements in lines 4000 to 4050. It is essential to get every number exactly right, for a minor error can cause all kinds of strange things to happen.

Line 510 is very long. Use the abbreviation for POKE ('P' then 'SHIFT' with 'O') when typing in this line.

Control characters used are:

CLEAR: lines 20, 110, 600
CTRL/2: lines 110, 600
CRSR DOWN: lines 110(2), 600(2), 670(2), 680(2), 690(2), 700(2), 710(5)
CRSR RIGHT: lines 110(2), 600(3), 670(3), 680(3), 690(3), 700(3), 710(2)

Program design

20–30	Initialising the most often used variables.
40–50	Reading sprite details into memory.
60	Reading the machine code routine into memory.
70–80	Setting screen width and memory size.
90–100	Initialising the display.
110–120	Requesting level.
130–140	Setting sound chip registers.
150–170	Intializing the display.
180–190	Setting sprite registers.
200	Setting the clock.
210–520	The main program loop.
600–660	Crash routine with visual and effects.
670–700	Mission details displayed.
710–730	Inviting re-play.
1000–1020	Subroutine for producing sound of explosion.
2000–2040	Subroutine for displaying exploding bombs with sound effects.
3000–3010	DATA statements containing sprite details.
4000–3040	DATA statements containing machine code routine.

Points of interest

The key to the interest of this game is its high speed. One machine code routine takes care of scrolling the screen toward the left. The other clears the column on the right of the screen before it is scrolled. This column is out of sight, once screen width has been reduced to 38 column.

To make the operation of the main loop as rapid as possible, yet keeping it in BASIC, frequently used constants have been assigned to variables. Examples are GB and HB, the video locations for the point on the ground just below the bomber. Two other examples are U for unity (=1) and Z for zero. It is quicker for the micro to refer to its table of variables to find '1' and '0' and other constants than to convert them into their floating-point or integer equivalents every time they are used.

The Commodore 64's *Programmer's Reference Guide* has a section on 'Smooth Scrolling'. This facility is not used in this program as it is too slow. The book states that it is neccessary to use a machine code routine in conjunction with this facility, yet tantalizingly does not include any routines for this purpose. The routine in data lines 4000-4020 can be used in smooth scrolling to the left. As used here, it scrolls the whole of the 38 column screen, except for the bottom 3 rows. To scroll the whole screen, alter the '22' in line 4000 to '25'.

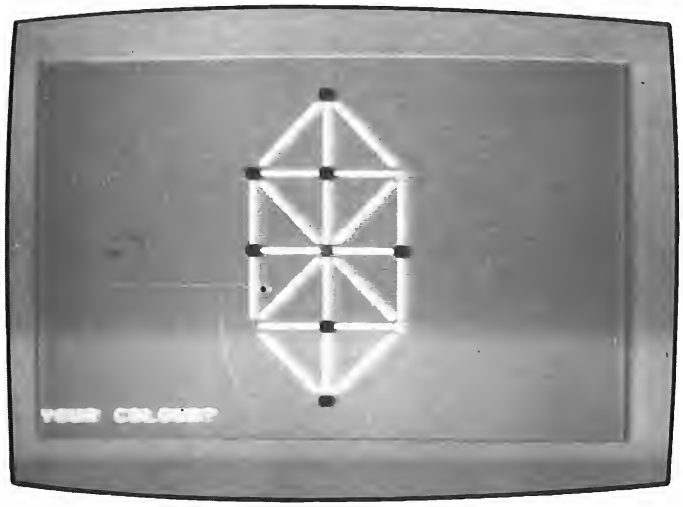
The program

```
10 REM ** BOMBING RUN **
20 PRINT"J":T=0:U=1:Z=0:V=53248:G=1024:H
=55296:B=0:F=0:Y=0
30 GB=G+852:HB=H+852:FF=80:GS=G+879:GT=G
+961:GF=G+80:HF=H+80:BH%=0
40 FOR J=12288 TO 12323:READ X:POKE J,X:
NEXT
50 FOR J=12324 TO 12350:POKE J,0:NEXT
60 FOR J=12544 TO 12617:READ X:POKE J,X:
NEXT
70 POKE 53270,PEEK(53270)AND247:POKE 649
,1:POKE 52,48:POKE 56,48
80 S=54272:FOR J=0 TO 4:POKE S+J,0:NEXT
90 Y=130:FB=0:BT=0:BD=0
100 TT=0:BG=0:A$=""
110 INPUT"J■■■■■LEVEL (1-10)";L$
120 L=VAL(L$):IF L<1 OR L>10 THEN 110
130 POKE S+24,47:POKE S,12:POKE S+1,1:PO
KE S+6,143:POKE S+21,7: POKE S+23,1:POKE
 S+17,1
140 POKE S+4,129:POKE S+14,88:POKE S+15,
115:POKE S+20,240
```

```
150 PRINT"■":POKE 53280,0:POKE 53281,6
160 FOR J=0 TO 1023:POKE H+J,Z:NEXT:POKE
 H+961,U
170 FOR J=1 TO 38:POKE G+880+J,102:NEXT
180 POKE 2040,192:POKE V+39,0
190 POKE V,104:POKE V+1,Y:POKE V+21,1
200 TI$="000000":POKE 650,128:L=L/20
210 E=RND(U):X=PEEK(V+31)
220 POKE HB,Z
230 IF E<L THEN F=RND(U)*760:POKE HF+F,7
:POKE GF+F,42:GOSUB 1000
240 SYS 12586
250 IF E<.3 THEN POKE GS,88
260 IF B=16 THEN POKE GS,160
270 IF E>.95 AND FB=Z THEN B=101:FB=U:TT
=TT+U
280 IF B/10=INT(B/10) THEN POKE S+18,65:
POKE GT,B/10+47
290 POKE S+18,64:IF FB=U THEN B=B-U
300 IF B=Z THEN FB=Z
310 GET A$:IF A$="" THEN 380
320 IF A$=CHR$(134) AND BG=Z THEN BH%=21
0-Y:BG=U:BD=BD+U:POKE S+20,248:GOTO 370
330 IF A$=CHR$(135) THEN Y=Y-8
340 IF A$=CHR$(136) THEN Y=Y+8
350 IF Y<Z THEN Y=Z
360 POKE V+U,Y
370 A$="_"
380 IF BG=U THEN POKE S+18,64:POKE S+15,
130+BH%/2:POKE S+18,65
390 SYS 12544
400 IF BG=Z THEN POKE S+18,64
410 IF PEEK(V+31)=U THEN 600
420 POKE H+F,0
430 BH%=BH%-4
440 IF B=2 AND BH%=Z THEN GOSUB 2000
450 SYS 12586
460 SYS 12544
470 BH%=BH%-4
480 IF BG=U THEN POKE S+18,64:POKE S+15,
130+BH%/2:POKE S+18,65
490 IF PEEK(V+31)=1 THEN 600
500 IF B=2 AND BH%=Z THEN GOSUB 2000
510 IFBG=UANDBH%=ZTHENPOKEHB,1:POKEGB,21
4:POKES+18,64:BG=0:POKES+20,240:GOSUB100
0
520 GOTO 210
600 PRINT"■=■■■■■■■■■■";:POKE 53280,0:POKE 53
281,2:POKE V+21,0
610 FOR J=0 TO 24:POKE S+J,0:NEXT
620 POKE S+24,15:POKE S,200:POKE S+1,2:P
OKE S+5,13:POKE S+6,248:POKE S+4,129
630 FOR K=1 TO 1500:NEXT
640 POKE S+4,128
650 IF Y>206 THEN PRINT"YOU FLEW TOO LOW
":GOTO 670
660 PRINT"YOU WERE SHOT DOWN"
670 PRINT"■■■■■■BOMBS DROPPED:";BD
680 PRINT"■■■■■■TARGETS HIT:";BT
690 PRINT"■■■■■■OUT OF ";TT;"POSSIBLES"
700 PRINT"■■■■■■FLIGHT TIME:";INT(TI/60);
" SECONDS"
710 PRINT"■■■■■■■■PRESS SPACE BAR FOR NEX
T RUN"
720 GET A$:IF A$<>" " THEN 720
730 GOTO 80
```

```
1000 POKE S+7,25:POKE S+8,1:POKE S+12,15
:POKE S+13,240
1010 POKE S+11,129:POKE S+11,128
1020 RETURN
2000 POKE HB,U:POKE GB,214:POKE 53280,U
2010 GOSUB 1000:GOSUB 1000
2020 BT=BT+U
2030 POKE HB,Z:POKE GB,32:POKE 53280,Z
2040 RETURN
3000 DATA 0,4,0,0,6,0,0,7,0,128,7,128,19
2,7,192,224,7,224
3010 DATA 224,7,240,224,7,248,240,7,252,
255,255,255,255,255,255,0,248,12
4000 DATA 169,0,133,251,169,4,133,252,16
0,1,169,22,170,177,251
4010 DATA 136,145,251,202,240,12,24,152,
105,41,168,144,241
4020 DATA 230,252,76,13,49,230,251,169,4
0,197,251,208,219,96
4030 DATA 160,0,169,4,133,254,169,119,13
3,253,162,20,169,32,145,253
4040 DATA 202,208,1,96,24,152,105,40,168
,144,241,230,254,76,54,49
```

9
Breakaway



This is a traditional game with a difference. You play it against the computer. The computer knows the rules of the game, but it does not know how to win. At first it makes its moves entirely at random, so you stand a good chance of winning. But, after you have played against it several times, it makes good moves more often and bad moves less often. It learns from its mistakes and takes note of its successes. You may then find it much more difficult to beat it.

This simple game is a fascinating one in its own right. Played against a computer which is capable of learning to improve its play is a somewhat unusual experience.

How to play

As soon as the program is RUN you see the playing board displayed on the screen. The positions on the board are black when vacant, and coloured when occupied. The lines joining the positions show you the way to move from one position to another. The computer has a single piece, coloured purple. This can be moved one step in any direction at each turn. The computer decides and makes

its own moves. It is programmed not to cheat.

You have three pieces, coloured green, blue and yellow, respectively. When it is your turn to play, you must move one of these, but not more. Your pieces also move one step at a turn, but may move only sideways or toward the top of the screen, not toward the bottom. Pieces cannot jump over each other and cannot occupy the same position. Play always begins with the computer's piece (call it the 64) in the central position, just below the top of the board. Your pieces always start in a row near the bottom.

The aim of the computer is to move the 64 to the very bottom of the board. Your aim is to prevent this and to corner the 64 in the position at the very top of the board.

You are always the first to play. When you see the message "Your colour" decide which piece you wish to move and press the appropriate key:

Green	key 6
Blue	key 7
Yellow	key 8

The colours are marked on the front of each key. Next you are asked 'Your move?'. Reply by pressing one of these keys:

Y	diagonally forward left
U	straight forward (up)
I	diagonally forward right
H	left
J	right

The program does not let you cheat! If you attempt an impossible or illegal move, a message tells you about this and, after a brief pause, you are able to try again.

You win if:

(i) you corner the 64 at the top of the board (cornering it in other positions has no effect – it just misses that turn)

The 64 wins if:

(i) it reaches the bottom of the board

(ii) there is no possible move for you (for example, if all your pieces have reached the top)

(iii) both players repeat the same back and forth move three times in succession.

Winning tactics

Plan your advance carefully as you close in on the 64. You may find that it comes forward, forcing you to side-step for the lack of any alternative move. Then it may escape toward the bottom of the board and there is nothing you can do to catch it. Avoid moving your pieces to the top of the board too quickly. Plenty of sideways moves delay the time when all your pieces have entered the triangle at the top and you lose the game. Even if the 64 escapes, it may make a few careless moves and wander back to the top of the board again. Be ready to catch it then!

Keying in

This program takes up a lot of memory, so the lines in the listing have had to be typed with few spaces. The DATA statements contain the information which direct the pieces as they move from one position to another. If you find the pieces moving where they should not go, check your typing of lines 4000 to 4050.

Control characters used in this program are:

CLEAR: lines 20, 1200, 1260
CTRL/2: lines 20, 1260
CRSR DOWN: lines 80(2), 1200(2), 1230, 1240(4)
CRSR RIGHT: lines 270, 1220(2), 1230(2), 1240(2)
CRSR UP: lines 280, 340, 440, 450, 550
CTRL/REV ON: lines 550, 900, 1170
CTRL/REV OFF: line 550

Unlisted characters:

C= line 1200 (after the Clear)

Do not leave out the spaces in the messages on lines 280, 340, 450, 550, 900, 1170. These are essential for the short messages to completely overprint the longer ones. The symbols used in making up the board (lines 80 to 220) are obtained by using keys 'N', 'M' and '–' with 'SHIFT'. The horizontal line used in H$ (line 30) is a shifted 'C'.

Program design

20–270	Initialising and displaying the board.
280–330	Requesting your colour.
340–410	Requesting your move.
420–430	Finding your piece in array M%( ).
440–450	Checking for a valid move.
460–530	Recording your move in array T%( ), checking for three successive back-and-forth moves (64 wins).
540	Displaying your move.
550–560	Checking if you have no legal move (64 wins).
570–620	64's move; finding 64 in array M%( ); checking if cornered at top (64 loses) or elsewhere (no move).
630–700	Choosing a move and making it.
710–740	Recording move in U%( ).
750	If home (64 wins).
760	Three successive back and forth moves (64 wins).
900–980	Recording successful moves.
1100–1190	Recording unsuccesful moves.
1200–1240	Displaying results.
1250–1410	Preparing for next game.
3000–3110	Subroutine for picking out moves.
4000–4050	Board DATA.

Points of interest

This program uses an array called M%( ). The large numbers in the array (see lines 4000–4050) are the screen locations of each board position. The other elements of the array hold only small numbers. Information about the computer's moves is added to these: '16' for every move made in the current game, '32' for a move in a lost game and '64' for a move in a won game. This information may be extracted again by logical ANDing. Lines 630 to 670, together with subroutine 3000 choose moves at random, but after applying certain criteria (CR) in order to select preferentially:

a previously successful move (CR = 64),
a move not already made in the current game (CR = 0),
a move already made in the current game (CR = 16),
a previously unsuccessful move (CR = 32).

This scheme makes the computer more likely to select good moves

and to avoid bad ones. This short program cannot be expected to cope with all the intricacies of artificial intelligence. There are several ways in which the 64 can win, and when it does win, it does not know why! So it may be trying to win in one way when it ought to be trying a different stategy. Against a strong player, the computer may rarely win, so has no chance to 'learn' good moves. If you want to teach it to win, give it a fair chance of winning first!

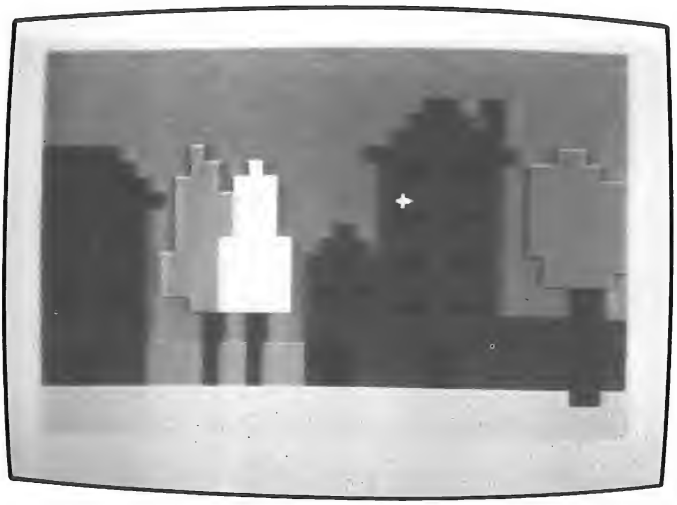
The program

```
10 REM ** BREAKAWAY **
20 PRINT"■■":POKE 53280,6:POKE 53281,8:D
IM M%(9,11),T%(5,2):POKE 649,1
30 G=1024:H=55296:H$="──── ────"
40 FOR K=1 TO 11
50 FOR J=0 TO 9
60 READ M%(J,K)
70 NEXT:NEXT
80 PRINT"■■";TAB(18)"/|\
90 PRINTTAB(17)"/  |  \":PRINTTAB(16)"/   |
  \"
100 PRINTTAB(15)"/    |    \":PRINTTAB(15)
H$
110 PRINTTAB(14)"|\    |    /|"
120 PRINTTAB(14)"| \   |   / |"
130 PRINTTAB(14)"|  \  |  /  |"
140 PRINTTAB(14)"|   \ | /   |":PRINTTAB(1
5)H$
150 PRINTTAB(14)"|   /|\   |"
160 PRINTTAB(14)"|  / | \  |"
170 PRINTTAB(14)"| /  |  \ |"
180 PRINTTAB(14)"|/   |   \|":PRINTTAB(1
5)H$
190 PRINTTAB(15)"\   |   /"
200 PRINTTAB(16)"\  |  /"
210 PRINTTAB(17)"\ | /"
220 PRINTTAB(18)"\|/"
230 FOR K=1 TO 11
240 POKE H+M%(9,K),M%(0,K)
250 POKE G+M%(9,K),81
260 NEXT
270 PRINT"■"
280 PRINT"YOUR COLOUR? ■"
290 C$=""
300 GET C$
310 C=VAL(C$)
320 IF C<6 OR C>8 THEN 290
330 C=C-1
340 PRINT"YOUR MOVE?    ■"
350 GET M$:IF M$="" THEN 350
360 IF M$="Y" THEN M=8:GOTO 420
370 IF M$="U" THEN M=1:GOTO 420
380 IF M$="I" THEN M=5:GOTO 420
390 IF M$="H" THEN M=4:GOTO 420
400 IF M$="J" THEN M=3:GOTO 420
410 GOTO 350
420 K=1
430 IF M%(0,K)<>C THEN K=K+1:GOTO 430
```

```
440 V=M%(M,K)AND15:IF V=0 THEN PRINT"NOT
 POSSIBLE!□":FOR J=1 TO 4000:NEXT:GOTO 2
80
450 V=M%(M,K)AND15:IFM%(0,V)>0 THEN PRIN
T"PLACE TAKEN □":FORJ=0TO4000:NEXT:GOTO
280
460 FOR L=0 TO 2
470 FOR J=5 TO 1 STEP -1
480 T%(J,L)=T%(J-1,L)
490 NEXT
500 T%(0,L)=0
510 NEXT
520 D=C-5:Q=M%(M,K)AND15:T%(0,D)=Q
530 IF T%(2,D)=Q AND T%(4,D)=Q AND T%(1,
D)=T%(3,D) AND T%(3,D)=T%(5,D) THEN 900
540 M%(0,K)=0:POKE H+M%(9,K),0:M%(0,M%(M
,K)AND15)=C:POKEH+M%(9,M%(M,K)AND15),C
550 PRINT"■64'S■ MOVE    □"
560 FOR L=1 TO 2000:NEXT:K=1
570 IF M%(0,K)<>4 THEN K=K+1:GOTO 570
580 IF K=1 AND M%(0,2)>4 AND M%(0,3)>4 A
ND M%(0,4)>4 THEN 1100
590 IF K=5 AND M%(0,2)>4 AND M%(0,6)>4 A
ND M%(0,8)>4 THEN 700
600 IF K=7 AND M%(0,4)>4 AND M%(0,6)>4 A
ND M%(0,10)>4 THEN 700
610 CR=64:GOSUB 3000:IF W>0 THEN 650
620 CR=0:GOSUB 3000:IF W>0 THEN 650
630 CR=16:GOSUB 3000:IF W>0 THEN 650
640 CR=32:GOSUB 3000
650 IF M%(0,NK)>0 THEN 610
660 M%(0,K)=0:POKE H+M%(9,K),0
670 M%(0,NK)=4:POKE H+M%(9,NK),4
680 X=M%(R,K)AND16:IF X=0 THEN M%(R,K)=M
%(R,K)+16
690 FOR J=5 TO 1 STEP -1
700 U%(J)=U%(J-1)
710 NEXT
720 U%(0)=NK
730 IF NK=11 THEN 900
740 IF U%(2)=NK AND U%(4)=NK AND U%(1)=U
%(3) AND U%(3)=U%(5) THEN 900
750 VV=0:FOR K=1 TO 4:VV=VV+M%(0,K):NEXT
:IF VV>18 AND M%(0,1)<>4 THEN 900
760 GOTO 280
900 PRINT"■THE 64 WINS    "
910 FOR J=1 TO 3000:NEXT
920 FOR J=0 TO 4
930 FOR L=1 TO 8
940 V=M%(L,U%(J+1))AND15
950 IF V=U%(J) THEN M%(L,U%(J+1))=V+64
960 NEXT:NEXT
970 VW=VW+1
980 GOTO 1200
1100 FOR J=0 TO 4
1110 FOR L=1 TO 8
1120 V=M%(L,U%(J+1))AND15
1130 IF V=U%(J) THEN M%(L,U%(J+1))=V+32
1140 NEXT
1150 NEXT
1160 VL=VL+1
1170 PRINT"■THE 64 LOSES    "
1180 FOR J=1 TO 3000
1190 NEXT
1200 PRINT"□■■■■■■■"
1210 POKE 53280,2:POKE 53281,13
```

```
1220 PRINT"■■THE COMPUTER HAS WON";VW;"G
AMES"
1230 PRINT"■■■OUT OF";VW+VL
1240 PRINT"■■■■■■PRESS THE SPACE BAR TO
CONTINUE"
1250 GET A$:IF A$<>" " THEN 1250
1260 PRINT"■■":POKE 53280,6:POKE 53281,8
1270 FOR K=1 TO 11
1280 M%(0,K)=0
1290 NEXT
1300 M%(0,3)=4:M%(0,8)=5:M%(0,9)=6:M%(0,
10)=7
1310 FOR J=0 TO 5
1320 U%(J)=0
1330 NEXT
1340 FOR J=1 TO 8
1350 FOR L=1 TO 11
1360 V=M%(J,L)AND112:X=M%(J,L)AND15
1370 IF V=16 THEN M%(J,L)=X
1380 IF V=48 THEN M%(J,L)=X+32
1390 IF V=80 THEN M%(J,L)=64
1400 NEXT:NEXT
1410 GOTO 80
3000 W=0:NK=0:FOR J=1 TO 8
3010 V=M%(J,K)ANDCR
3020 X=M%(J,K)AND15
3030 IF V=CR AND M%(0,X)=0 THEN W=1
3040 NEXT
3050 IF W=0 THEN RETURN
3060 R=INT(RND(1)*8)+1
3070 V=M%(R,K)ANDCR
3080 X=M%(R,K)AND15
3090 IF V=CR AND M%(0,X)=0 THEN NK=M%(R,
K)AND15
3100 IF NK=0 THEN 3050
3110 RETURN
4000 DATA 0,0,3,0,0,0,4,2,0,99,0,0,5,3,0
,1,6,0,0,294
4010 DATA 4,1,6,4,2,0,0,0,0,299,0,0,7,0,
3,0,0,6,1,304
4020 DATA 0,2,8,6,0,0,0,0,0,494,0,3,9,7,
5,4,10,8,2,499
4030 DATA 0,4,10,0,6,0,0,0,0,504,5,5,0,9
,0,6,11,0,0,694
4040 DATA 6,6,11,10,8,0,0,0,0,699,7,7,0,
0,9,0,0,11,6,704
4050 DATA 0,9,0,0,0,10,0,0,8,899
```

10
Snipers



A keen eye and an accurate aim are the prime requirements for beating the snipers. You are in your armoured vehicle stationed beside a row of buildings. Snipers are firing at you from the windows and doorways. There are walls and trees beside the buildings. These provide more hiding places for the snipers. There are snipers in unexpected places, too.

When a sniper fires you see the flash from the rifle. This is your only clue to where the sniper is hiding. If you are quick enough and aim your machine gun at the spot where the flash came from, you can put the sniper out of action. But snipers are cunning and, if you take too long to bring your gun to bear, the sniper will have taken cover, and lives to fire again.

Sooner or later a lucky shot (unlucky for you!) penetrates your armour and you have to give up the fight. Your task before then is to silence all, or as many of the snipers as you can.

How to play

The level you choose decides how many snipers there are, and how

quickly you have to be to put them out of action. As soon as the display has been drawn, the battle begins. The gunsight (a white cross) appears in the area of sky near the top left corner of the screen. Move it by using these keys:

; up
/ down
Z left
X right

The keys have repeat action, so hold any one down to sweep the sight across the screen quickly. You will hear the rifle shots and see the flashes. Move your gunsight to aim it at a sniper. When your sight is aimed at the spot where you saw a sniper, press the space bar, just once. You will hear a burst of machine gun fire. If you fired in time, the sniper will not fire from that spot again. Move quickly on to fire at another sniper, for their shots are being counted by the computer and the next shot may be the one that ends the game.

When the game ends, you are told how many snipers you hit, and the record score so far.

Winning tactics

Go for the snipers in the windows and doorways first, for these are the easiest to see and locate. It is much harder to see the ones in and around the trees. At levels 8 to 10 you have very little time in which to get your sight trained on a sniper. Unless the sight is very close to a flash, it is hardly worth while machine-gunning it. Keep the sight on the building at first, waiting to pounce on any nearby flash you see. Incidentally, the snipers are scattered in a fairly random way, so do not expect too find them in exactly the same places each time you play.

Keying in

As usual, it is the DATA statements which need the most care.

Control characters used:

CLEAR: lines 40, 620, 750, 760
CTRL/2: line 40

CRSR DOWN: lines 70(2), 630(2), 640, 660(3), 670(3), 770(2), 780

CRSR RIGHT: lines 70(2), 630(2), 640(2), 660(2), 670(2), 770(2), 780(2)

Unlisted characters used:

C=/2 line 620, 760

Program design

10–60	Initialising.
70–80	Requesting level.
90–170	READing DATA and using it to display the scene.
180–270	READing DATA to get details of all sniper positions, then inactivating some at random, to leave a random assortment active.
280	The gunsight.
290–370	A sniper fires.
380–520	Moving your sight.
530–610	Firing your machine gun and registering a hit, if any.
620–660	Display of score.
670–750	Inviting re-play.
760–790	Display when all snipers hit.
4000–4130	DATA for display.
5000–5030	DATA for sniper locations.

Points of interest

The display routine shows an easy way of building up complex pictures on the Commodore 64. Solid blocks of colour (reversed 'space', screen code 160) are placed at each screen location by POKEing. The DATA consists of pairs of numbers. The first gives the number of blocks needed (N) and the second their colour (C). The operation is done in a loop (lines 90–170), so 'scanning' the screen and building up the picture.

The program

```
10 REM ** SNIPERS **
20 G=1024:H=55296:S=54272:DIM S%(2,60):P
OKE 649,1:POKE 650,128
30 FOR J=0 TO 24:POKE S+J,0:NEXT:POKE S+
24,15:POKE S,50:POKE S+1,4:POKE S+6,240
40 PRINT"■■":A=G-1:B=H-1:SU=0
50 X=3:XX=3:Y=3:YY=3
60 TI$="000000"
70 INPUT"■■■LEVEL (1-10)";L$
80 L=VAL(L$):IF L<0 OR L>10 THEN 30
90 POKE 53281,1:FOR J=1 TO 209
100 READ N
110 READ C
120 FOR K=1 TO N
130 POKE A+K,160
140 POKE B+K,C
150 NEXT
160 A=A+N:B=B+N
170 NEXT
180 FOR K=1 TO 60
190 READ S%(0,K)
200 NEXT
210 FOR K=1 TO 30-L*2
220 J=INT(RND(1)*61)
230 S%(1,J)=1
240 NEXT
250 FOR J=1 TO 60
260 IF S%(1,J)=0 THEN SU=SU+1
270 NEXT
280 POKEG+X+40*Y,219
290 SP=INT(RND(1)*60)+1
300 IF S%(1,SP)=1 THEN 380
310 S%(2,SP)=TI
320 POKE G+S%(0,SP),174
330 POKE S+4,129
340 FOR K=1 TO 30:NEXT
350 POKE S+4,128
360 POKE G+S%(0,SP),160
370 SS=SS+1
380 T=TI
390 GET A$
400 IF A$=" " THEN 530
410 IF A$="Z" THEN XX=XX-1
420 IF A$="X" THEN XX=XX+1
430 IF XX<0 THEN XX=0
440 IF XX>39 THEN XX=39
450 IF A$=";" THEN YY=YY-1
460 IF A$="/" THEN YY=YY+1
470 IF YY<0 THEN YY=0
480 IF YY>24 THEN YY=24
490 POKE G+X+40*Y,160
500 X=XX:Y=YY
510 POKE G+X+40*Y,219
520 GOTO 600
530 Z=X+40*Y
540 POKE S+1,2:FOR J=1 TO 60
550 POKE S+4,129
560 IF S%(0,J)=Z AND TI-S%(2,J)<500-46*L
 THEN S%(1,J)=1:SN=SN+1
570 POKE S+4,128:POKE S+1,3
580 NEXT
590 IF SN=SU THEN 760
600 IF TI-T>25-L*2 THEN 380
610 IF SS<150 THEN 290
```

```
620 PRINT"■":POKE 53280,5:POKE 53281,15
630 PRINT"■■■THEY GOT YOU AT LAST!"
640 PRINT"■■■BUT YOU HIT";SN;"OF THEM"
650 IF SM<SN THEN SM=SN
660 PRINT"■■■■THE BEST SCORE IS";SM
670 PRINT"■■■■PRESS 'Y' TO CONTINUE"
680 GET A$:IF A$<>"Y" THEN 680
690 FOR J=1 TO 60
700 S%(1,J)=0
710 NEXT
720 RESTORE
730 POKE 53280,14:POKE 53281,6
740 SS=0:SN=0
750 PRINT"■":POKE 53280,14:POKE 53281,6:
GOTO 30
760 PRINT"■":POKE 53280,5:POKE 53281,15
770 PRINT"■■■WELL DONE!"
780 PRINT"■■YOU HIT ALL";SU;"SNIPERS"
790 GOTO 650
4000 DATA 146,6,2,2,2,6,1,2,34,6,4,2,1,6
,1,2,33,6,7,2,9,6,5,11,5,6,1,5,11,6
4010 DATA 10,2,3,6,2,5,3,6,6,11,4,6,2,5,
2,6,1,13,8,6,1,2,2,0,2,2,2,0,1,2,2,6
4020 DATA 5,5,2,6,7,11,2,6,3,5,1,6,3,13,
7,6,8,2,2,6,7,5,8,11,1,6,4,5,3,13,7,6
4030 DATA 8,2,2,6,7,5,7,2,2,6,4,5,3,13,7
,6,1,2,2,0,2,2,2,0,1,2,2,6,7,5,1,2
4040 DATA 2,0,1,2,2,0,1,2,2,6,4,5,3,13,4
,6,1,2,2,6,8,2,2,6,7,5,1,2,2,0,1,2
4050 DATA 2,0,1,2,2,6,3,5,5,13,2,6,3,2,1
,6,8,2,2,6,7,5,7,2,1,6,4,5
4060 DATA 5,13,1,6,2,2,1,0,3,2,2,0,2,2,2
,0,1,2,3,6,6,5,7,2,1,6,4,5,5,13,1,6
4070 DATA 13,2,3,6,5,5,1,6,1,2,2,0,1,2,2
,0,1,2,1,6,4,5,5,13,1,6,1,2,1,0,1,2
4080 DATA 1,0,9,2,5,6,2,9,2,6,1,2,2,0,1,
2,2,0,1,2,3,6,2,5,5,13,1,6,6,2,2,0
4090 DATA 2,2,2,0,1,2,5,6,2,9,2,6,7,2,4,
6,1,9,2,6,1,9,3,6,18,2,2,9,9,2,4,6
4100 DATA 1,9,2,6,1,9,3,6,18,2,2,9,4,2,2
,0,3,2,4,8,1,9,2,8,1,9,3,8,1,2,2,0
4110 DATA 5,2,2,0,8,2,2,9,4,2,2,0,3,2,4,
8,1,9,2,8,1,9,3,8,1,2,2,0,5,2,2,0
4120 DATA 8,2,2,9,4,2,2,0,3,2,4,8,1,9,2,
8,1,9,3,8,1,2,2,0,5,2,2,0,8,2,2,9
4130 DATA 2,2,36,7,2,9,82,7
5000 DATA 189,202,223,231,304,305,308,30
9,330,355,375,398,411,424,425,428,429
5010 DATA 434,445,478,481,502,530,533,54
0,544,545,548,549,555,605,615,619,621
5020 DATA 641,642,645,664,665,668,669,67
3,679,728,735,755,772,779,787,798
5030 DATA 802,803,813,826,850,860,867,91
5,927,988
```

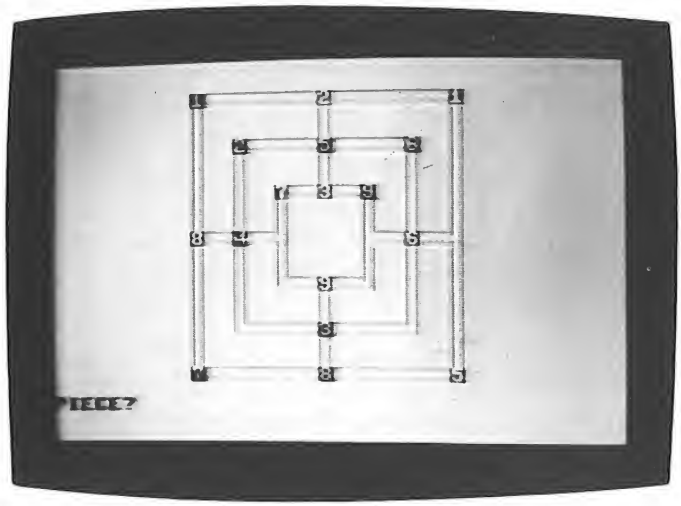
Variations

Altering values in lines 5000–5030 places snipers in different positions. The numbers are screen positions, counting from the top left corner along the rows. If you would prefer a longer battle,

increase the number in line 610 from 150 to, say, 200. If you like the pace of level 10, but can never move your sight quickly enough, alter the 25 in line 600 to 30 or 35.

If you tire of street fighting, why not transfer the skirmish to a jungle or even make it a naval battle? It is easy to redesign the whole display simply by changing DATA lines 4000–4130, and amending lines 5000–5030 to place the snipers in suitable positions.

11
Nine Men Morris



Nine Men Morris is a traditional game. There are two players, each with 9 pieces, or 'men', to move around the curiously designed playing board. Although the rules are simple, winning a game is not always as easy as it might seem.

How to play

The two players are represented by two colours, red and blue. Red plays first. The border of the screen shows the colour of the person whose turn it is to play.

The game has two stages. In the first stage, each player puts 9 pieces on to the board, one at a time. During this stage of the game the screen colour is yellow. As soon as the board is displayed you will see a question mark at the top left corner. This is an invitation to place your first piece on that point. It is your turn to play. Press the '/' key if you want to put a piece on that point. If not, press the space bar, to move the question mark to the next point on the board. You can press the space bar as many times as you like until the question mark is on the position you select. Then press '/'. When you press '/'

a number in your colour appears at the point. This is your piece.

If you accidentally move the question mark past a point you want to select, carry on pressing the space bar to bring the question mark back to the top of the board again and then on to the point you have selected.

During this stage of the game you should try to place three of your own pieces in line. Three pieces are in line when they are filling the three points on the sides of any of the squares, or on any of the lines which run between the squares. When you place pieces in line, the computer asks you to 'TAKE 1'. This means that you must now take one of your opponent's pieces from the board. Do this by pressing the corresponding number key (1–9) at the top of the keyboard.

When each player has placed nine pieces, the game moves on to its next stage. The screen becomes pale green at this stage, but the border is coloured red or blue, by turns, as before.

You have to move your pieces, in turn, from one point to the next vacant point. Pieces cannot jump over each other, and two pieces cannot be on the same point. When it is your turn to play, the border shows your colour and a message 'PIECE'? appears. Select the piece by pressing one of the number keys. If you select a piece that has already been removed, or one that has no moves available to it (because all adjacent points are occupied), your selection is ignored. Select another piece instead. Then the message 'MOVE' appears. Move your piece by pressing one of these keys:

U move up
H move left
J move right
N move down

If by accident you have tried to make an impossible move, you are asked to select a piece and its move again.

You win the game at this stage in one of two ways:

(1) by blocking your opponent so that no moves are possible,
(2) by reducing the number of your opponent's pieces to 2.

As in the first stage of the game, you should try to move three of your pieces into line, so allowing you to take away any one of your opponent's pieces.

When the game ends, a message shows who has won and why. Then press any key to play again.

Winning tactics

In the first stage of the game, try to get your pieces in line as often as possible. Pieces placed in the middle of the side of a square can be used for three-in-line in more than one direction. At the same time try to prevent your opponent from placing three in a line. When you 'take' a piece, avoid taking one that is already in a line of three. Otherwise, your opponent can replace this with another piece at the next turn, scoring three-in-line and taking one of your pieces.

In the second stage you can often move one of your three-in-line pieces out of line and then back into line again at the next turn. This can be a quick way to reduce your opponent's numbers. If you have few pieces left, keep them close together and try for three-in-line as often as you can. If you have many pieces it is usually better to spread them widely over the board, so as to reduce the number of moves available to your opponent and eventually block your opponent completely.

There are some situations, particularly when both players have few pieces left, in which there is a 'stalemate' and neither player can force a win. The computer is not able to detect such a situation, so players must agree to a draw, press the 'RUN/STOP key, and then key in 'RUN' to play again.

Keying in

Control characters used are:

HOME: lines 30, 50
CRSR DOWN: lines 50, 180
CTRL/5: line 60
CTRL/REV ON: lines 60, 70(3), 80(3), 90(5), 100(5), 110(6), 120(2), 140(5), 160(3), 180
CTRL/REV OFF: lines 70(2), 80(2), 90(4), 100(4), 110(5), 120, 140(4), 160(2)
CRSR UP: lines 460, 540, 630, 1080, 1120
CTRL/3: line 900
CTRL/7: line 910

The messages ending in the reverse disk (CRSR UP) are all six characters long, excluding the disk. Note the semi-colons after the messages in lines 900 and 910. Without these the display will be

spoilt. Remember to type the '%' for every occurence of array M%( ). If you forget this, or type '$' by mistake, you will almost certainly get an 'OUT OF MEMORY' error when the program is RUN.

Program design

10–40	Initialising.
50–180	Displaying the board.
190	Re-entry point for each move of 1st stage.
200–320	Placing a piece.
330–350	Looking for 3 in-a-line; return for next move.
360	Re-entry point for each move of 2nd stage.
370–450	Is player completely blocked?
460–530	Which piece to move?
540–660	Requesting and executing move.
670–680	Looking for 3-in-line; return for next move.
690–740	Final routines.
900–910	Subroutine to change player at beginning of each move.
1000–1200	Subroutine looking for 3-in-line, and supervising the 'take'.
2000–2080	Subroutine to find a 3-in-line.
2500–2520	Input subroutine.
4000–4070	DATA for array M%( ).

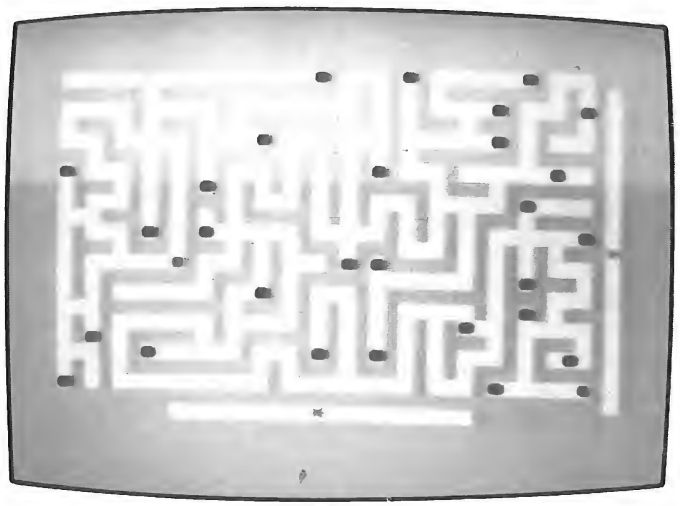
The program

```
10 REM ** NINE MEN MORRIS **
20 G=1024:H=55296:B=53280:S=53281:DIM M%
(6,24),P%(2,9),A$(2):POKE 649,1
30 PRINT"■":POKE B,2:POKE S,7
40 FOR K=1 TO 24:FOR J=0 TO 6:READ M%(J,
K):NEXT:NEXT
50 M=2:PRINT"■■"
60 PRINTTAB(10)"■■┌──────────┬──────────┐"
70 B$="■│■          ■│■          ■│":PRINTTA
B(10)B$:PRINTTAB(10)B$
80 PRINTTAB(10)"■│■   ■├──────────┼──────────┤■   ■│
"
90 C$="■│■   ■│■      ■│■      ■│■   ■│":PRI
NTTAB(10)C$:PRINTTAB(10)C$
100 PRINTTAB(10)"■│■   ■│■   ■├──────┼──────┤■   ■│
■   ■│"
110 D$="■│■   ■│■   ■│■      ■│■   ■│■   ■│":
PRINTTAB(10)D$:PRINTTAB(10)D$
120 PRINTTAB(10)"■├──────┼──────┤■      ■├──────┼──────┤"
130 PRINTTAB(10)D$:PRINTTAB(10)D$
140 PRINTTAB(10)"■│■   ■│■   ■└──────┴──────┘■   ■│
■   ■│"
```

```
150 PRINTTAB(10)C$:PRINTTAB(10)C$
160 PRINTTAB(10)"▒|▬   ▒└─────────┴─────────┘▬   ▒
|"
170 PRINTTAB(10)B$:PRINTTAB(10)B$
180 PRINTTAB(10)"▒└──────────────┴──────────────┘▒"
190 GOSUB 900
200 POKE B,4*M-2
210 IF P=10 THEN P=1
220 K=0
230 K=K+1:IF K=25 THEN K=1
240 IF PEEK(G+M%(5,K))<186 THEN 230
250 POKE G+M%(5,K),191
260 GET A$:IF A$="" THEN 260
270 IF A$=" " THEN POKE G+M%(5,K),M%(6,K
):GOTO 230
280 IF A$="/" THEN 300
290 GOTO 260
300 POKE G+M%(5,K),P+176
310 POKE H+M%(5,K),M*4-2
320 M%(0,K)=P+10*(M-1):P%(M,P)=1
330 GOSUB 1000
340 IF F7=7 THEN 690
350 IF NOT(P=9 AND M=2) THEN 190
360 POKE S,13:GOSUB 900
370 FB=1
380 FOR K=1 TO 24
390 MM=M%(0,K)
400 IF MM=0 OR MM>10 AND M=1 OR MM<10 AN
D M=2 THEN 440
410 FOR J=1 TO 4
420 IFM%(J,K)>0 AND M%(0,M%(J,K))=0 THEN
 FB=0
430 NEXT
440 NEXT
450 POKE B,4*M-2:IF FB=1 THEN 710
460 PRINT"PIECE?⌝"
470 GOSUB 2500
480 IF P%(M,A)<>1 THEN 470
490 K=0
500 IF M%(0,K)<>A+10*(M-1) THEN K=K+1:GO
TO 500
510 FB=1:FOR J=1 TO 4:IF M%(J,K)<>0 AND
M%(0,M%(J,K))=0 THEN FB=0
520 NEXT
530 IF FB=1 THEN 460
540 PRINT"MOVE? ⌝"
550 GET A$:IF A$="" THEN 550
560 IF A$="U" THEN D=1:GOTO 610
570 IF A$="N" THEN D=2:GOTO 610
580 IF A$="J" THEN D=3:GOTO 610
590 IF A$="H" THEN D=4:GOTO 610
600 GOTO 550
610 IF M%(D,K)=0 THEN 460
620 IF M%(0,M%(D,K))<>0 THEN 460
630 PRINT"         ⌝"
640 POKE G+M%(5,K),M%(6,K):POKE H+M%(5,K
),4
650 POKE G+M%(5,M%(D,K)),A+176:POKE H+M%
(5,M%(D,K)),M*4-2
660 M%(0,K)=0:M%(0,M%(D,K))=A+10*(M-1):P
=A
670 GOSUB 1000
680 IF F7<7 THEN 360
690 PRINT"7 REMOVED - YOU WIN"
700 GOTO 720
710 PRINT"BLOCKED - YOU LOSE"
720 A$=""
```

```
730 GET A$:IF A$="" THEN 730
740 P=0:POKE S,7:GOTO 50
889 GOTO889
900 IF M=2 THEN M=1:MX=2:PRINT"■";:P=P+1
:RETURN
910 M=2:MX=1:PRINT"■";:RETURN
1000 FT=0
1010 FOR K=2 TO 23
1020 IF MX(1,K)=0 OR MX(2,K)=0 THEN 1040
1030 PA=1:GOSUB 2000
1040 IF MX(3,K)=0 OR MX(4,K)=0 THEN 1060
1050 PA=3:GOSUB 2000
1060 NEXT
1070 IF FT=0 THEN RETURN
1080 PRINT"TAKE 1□"
1090 GOSUB 2500
1100 IF PX(MX,A)=1 THEN 1120
1110 GOTO 1090
1120 PRINT"       □"
1130 PX(MX,A)=2
1140 FOR K=1 TO 24
1150 IFMX(0,K)=A+10*(MX-1) THEN MX(0,K)=
0:POKE G+MX(5,K),MX(6,K):POKEH+MX(5,K),4
1160 NEXT
1170 F7=0:FOR J=1 TO 9
1180 IF PX(MX,J)=2 THEN F7=F7+1
1190 NEXT
1200 RETURN
2000 IF MX(0,K)=0 OR MX(0,MX(PA,K))=0 OR
 MX(0,MX(PA+1,K))=0 THEN RETURN
2010 FM=0:IF MX(0,K)>10 THEN FM=FM+1
2020 IF MX(0,MX(PA,K))>10 THEN FM=FM+1
2030 IF MX(0,MX(PA+1,K))>10 THEN FM=FM+1
2040 PP=P+10*(M-1)
2050 FP=0:IF MX(0,K)=PP OR MX(0,MX(PA,K)
)=PP OR MX(0,MX(PA+1,K))=PP THEN FP=1
2060 IF M=1 AND FP=1 AND FM=0 THEN FT=1
2070 IF M=2 AND FP=1 AND FM=3 THEN FT=1
2080 RETURN
2500 GET A$:IF A$="" THEN 2500
2510 A=VAL(A$):IF A<0 OR A>9 THEN 2500
2520 RETURN
4000 DATA 0,0,10,2,0,90,240,0,0,5,3,1,99
,242,0,0,15,0,2,108,238
4010 DATA 0,0,11,5,0,213,240,0,2,8,6,4,2
19,219,0,0,14,0,5,225,238
4020 DATA 0,0,12,8,0,336,240,0,5,0,9,7,3
39,241,0,0,13,0,8,342,238
4030 DATA 0,1,22,11,0,450,235,0,4,19,12,
10,453,219,0,7,16,0,11,456,243
4040 DATA 0,9,18,14,0,462,235,0,6,21,15,
13,465,219,0,3,24,0,14,468,243
4050 DATA 0,12,0,17,0,576,237,0,0,20,18,
16,579,242,0,13,0,0,17,582,253
4060 DATA 0,11,0,20,0,693,237,0,17,23,21
,19,699,219,0,14,0,0,20,705,253
4070 DATA 0,10,0,23,0,810,237,0,20,0,24,
22,819,241,0,15,0,0,23,828,253
```

12
Ball Maze



Steer the ball through the maze from beginning to end, without letting it drop into one of the holes. It needs a steady hand and quick reactions but, given a little practice, it can be done.

How to play

The outline of the maze begins to build up on the screen as soon as you RUN the program. Then the holes appear. These are represented by black disks, except for one at the bottom right-hand corner of the maze, and this is coloured blue. This is the hole to which you have to guide the ball. The ball appears as a red disk at the top left-hand corner.

The tilt indicators appear along the right side of the screen, and along the bottom edge. The centre of each indicator is marked with a purple star. The idea of the indicators is that they show you which way the maze is 'tilted' and by how much. This helps you control the direction in which the ball rolls and the speed at which it travels.

The maze is tilted by pressing one of these keys:

Z tilt to left
X tilt to right
; tilt away from you (toward top of screen)
/ tilt toward you (toward bottom of screen)

The keys have repeat action so the longer you hold one down, the greater the amount of tilt.

You need to press the X key to start the ball moving to the right, but do not press it for too long, or the ball may gather speed and fall straight into the first hole. Watch the pointer on the bottom tilt indicator and judge when you should press key Z. One point to note is that when you are trying to run the ball in one direction (say, to the right), the maze must not be tilted strongly in a direction at right angles to this (i.e. up or down). If the ball is not free to move in the direction in which the maze has a steep tilt, it 'jams' against the wall and cannot move.

When the ball falls into any of the holes (except the blue one), you hear a sound effect. A moment later, the ball reappears at the top left corner, ready for another try. The game is being timed, and when you eventually run the ball into the blue hole, you will be told how many minutes you have taken. Press any key to return the ball to the beginning again for another attempt on the speed record.

Winning tactics

Anticipation is the key to success in this game. As the ball gathers speed, tilt the maze in the opposite direction to slow it down and prevent it from running into a hole ahead. You will find that, whenever you alter the direction of tilt, a single key press takes it back to 'zero tilt' immediately and then gives it a slight tilt in the opposite direction.

It is possible to make use of the jamming effect of excessive crossways tilt. Applying this suddenly may often bring the ball to a halt more rapidly then trying to reverse the direction of tilt. It is also useful to be able to make the ball motionless for a few moments, while you plan how you intend to negotiate the next awkward part of the course.

Keying in

The design of the maze is contained in lines 4000 to 4200. Make

certain every '1' and '0' is correct, otherwise the paths of the maze may be closed across, or there may be gaps in the walls, making it too easy.

Control characters are used only in line 40 (CLEAR) and line 720 (HOME).

Program design

20–40	Initialising.
50–350	Displaying the maze and tilt indicators.
360	Starting the clock.
370–380	Re-entry point for each calculation: clearing tilt pointers from screen.
390–460	Reading keys, calculating and displaying tilt.
470–490	Calculating the next position for the ball.
500–570	Finding out if the ball is free to move to this position, and what would happen if it went there.
580	Moving the ball (unless already down a hole).
590–600	Preparing for next move.
610–700	Ball falls down a hole (sound effect and return ball to start).
710–810	Ball falls down final hole (triumphant chords and display time).
820–860	Inviting replay.
4000–4100	DATA for walls of maze.
5000–5010	DATA for positions of holes (except the blue one).

Points of interest

Before the maze is displayed most of the screen is covered with disks (line 70). These are white disks on a white background, so are not visible. To display a hole it is then necessary only to POKE the colour code (for black or blue) to the corresponding address in colour code RAM (lines 320, 340). The ball is made to appear by POKEing the corresponding location with the code for light red (line 350). Moving the ball is easy. All that has to be done is to POKE its current location with the code for white, and its new location with the code for light red (line 580). It disappears from one spot and appears at another, so giving the illusion of motion. The presence of holes and walls ahead of the ball is detected by PEEKing locations in colour code RAM (lines 500 and 520).

The program

```
10 REM ** BALL MAZE **
20 G=1024:H=55296:A=G:B=H:HX=H+939:HY=H+
519:W=160:C=3
30 POKE 650,128:X=1:Y=1:NX=1:NY=1:S=5427
2
40 PRINT"□":POKE 53280,3:POKE 53281,1
50 FOR J=1 TO 37
60 GJ=G+J:HJ=H+J:FOR K=1 TO 21
70 POKE HJ+40*K,1:POKE GJ+40*K,81
80 NEXT
90 POKE HJ,C:POKE GJ,W:POKE HJ+880,C:POK
E GJ+880,W
100 POKE HJ+920,1:POKE GJ+920,30:POKE GJ
+960,W:POKE HJ+960,C
110 NEXT
120 FOR K=0 TO 960 STEP 40
130 POKEH+K,C:POKEG+K,W:POKEH+K+38,C:POK
EG+K+38,W:POKEH+K+39,1:POKEG+K+39,31
140 NEXT
150 FOR J=920 TO 928:POKE H+J,C:POKE G+J
,W:NEXT
160 FOR J=950 TO 959:POKE H+J,C:POKE G+J
,W:NEXT
170 POKE H+39,C:POKE G+39,W:POKE H+79,C:
POKE G+79,W:POKE H+999,C:POKE G+999,W
180 POKE HY,4:POKE G+939,42
190 POKE HX,4:POKE G+519,42
200 FOR J=0 TO 20
210 READ A$
220 N=41
230 FOR K=1 TO 37
240 IF MID$(A$,K,1)="0" THEN 260
250 POKE B+N,C:POKE A+N,W
260 N=N+1
270 NEXT
280 A=A+40:B=B+40
290 NEXT
300 FOR J=1 TO 28
310 READ N
320 POKE H+N,0
330 NEXT
340 POKE H+877,6:POKE G+877,81
350 POKE H+41,10
360 TI$="000000"
370 IF AX<>0 THEN POKE HX+AX,1
380 IF AY<>0 THEN POKE HY+AY*40,1
390 GET A$:IF A$="X" AND AX<0 OR A$="Z"
AND AX>0 THEN AX=0
400 IF A$="X" AND AX<10 THEN AX=AX+1
410 IF A$="Z" AND AX>-10 THEN AX=AX-1
420 IF A$="/" AND AY<0 OR A$=";" AND AY>
0 THEN AY=0
430 IF A$="/" AND AY<10 THEN AY=AY+1
440 IF A$=";" AND AY>-10 THEN AY=AY-1
450 IF AX<>0 THEN POKE HX+AX,4
460 IF AY<>0 THEN POKE HY+AY*40,4
470 VX=VX+AX*2:VY=VY+AY*2
480 IF ABS(VX)>10 THEN NX=X+1*SGN(VX):VX
=0
490 IF ABS(VY)>10 THEN NY=Y+1*SGN(VY):VY
=0
500 CX=PEEK(H+NX+Y*40)AND7
510 CY=PEEK(H+X+NY*40)AND7
520 CZ=PEEK(H+NX+NY*40)AND7
```

```
530 IF CX=0 OR CY=0 OR CZ=0 THEN 610
540 IF CX=3 THEN VX=0:NX=X
550 IF CY=3 THEN VY=0:NY=Y
560 IF CZ=3 THEN VX=0:VY=0:NX=X:NY=Y
570 IF CX=6 OR CY=6 THEN 710
580 POKE H+X+40*Y,1:POKE H+NX+40*NY,10
590 X=NX:Y=NY
600 GOTO 370
610 POKE H+X+40*Y,1
620 IF AX<>0 THEN POKE HX+AX,1
630 IF AY<>0 THEN POKE HY+AY*40,1
640 POKE S+24,15:POKE S+4,17:POKE S+6,15
:POKE S+5,9:POKE S+6,0
650 FOR J=1 TO 50
660 F=4200+J*250:POKE S+1,INT(F/256):POK
E S,F-(256*INT(F/256))
670 NEXT
680 POKE S+24,0:POKE S+4,0
690 AX=0:AY=0:VX=0:VY=0:X=1:Y=1:NX=1:NY=
1
700 GOTO 350
710 T=VAL(MID$(TI$,3,2))
720 PRINT"■WON! - IN";T;"MIN"
730 POKE S+24,15:POKE S+5,0:POKE S+6,240
:POKE S+12,0:POKE S+13,240:POKE S+19,0
740 POKE S+20,240:POKE S+1,17:POKE S,37:
POKE S+8,51:POKE S+7,97:POKE S+15,137
750 POKE S+14,43:POKE S+4,17:POKE S+11,1
7:POKE S+18,17
760 FOR J=1 TO 100:NEXT
770 POKE S+4,0:POKE S+11,0:POKE S+18,0
780 FOR J=1 TO 40:NEXT
790 POKE S+4,17:POKE S+11,17:POKE S+18,1
7
800 FOR J=1 TO 450:POKE S+24,15-J/30:NEX
T
805 FORJ=0TO24:POKES+J,0:NEXT
810 FOR J=0 TO 24:POKE S+J,0:NEXT
820 A$="":GET A$:IF A$="" THEN 820
830 FOR J=0 TO 15
840 POKE H+J,C:POKEG+J,W
850 NEXT
4000 DATA 000000000000000000001000101000
0001000
4010 DATA 111111011111111111111010101011
1111010
4020 DATA 000001000000100000000001010100
0100010
4030 DATA 011101011101011111111101011101
1101110
4040 DATA 000100000100000100001000010000
0101000
4050 DATA 110111010101111101010101011111
1101111
4060 DATA 010001010101000010101010100001
0000000
4070 DATA 011101010101010101010111111011
0111111
4080 DATA 010001010111010001010100010000
0100000
4090 DATA 000111010101011111010101010101
1101111
```

```
4100 DATA 010001010100000001000001010101
0000000
4110 DATA 011101110111010101110111000101
0101011
4120 DATA 010000000100010100010101111101
0101000
4130 DATA 010111111101110101111100000001
0101110
4140 DATA 010100000001000101000101111111
0111000
4150 DATA 000111111111101110101010100000
0101011
4160 DATA 010100000000100000010101010100
0100000
4170 DATA 010101111110101111110101010111
0101110
4180 DATA 000101000000001000010101010100
0101000
4190 DATA 010101111111111010111011101010
1101111
4200 DATA 010100000000000010000000000010001
0000000
5000 DATA 59,65,73,151,157,215,231,281,3
03,315,331,393,447,451,477
5010 DATA 541,543,593,615,673,709,723,76
7,779,783,796,941,871
```

Variations

A new maze can be designed on a sheet of squared paper, 19 squares across and 21 squares deep. This does not include the wall around the maze. Shade in squares to define where the dividing walls are to be. The DATA consists of 21 groups of 19 digits, one group for each row of the design. Where there is a shaded square (wall), the digit is a '1'; where there is a blank square (path) there is a '0'. Replace the listed lines 4000 to 4100 with lines containing details of your own maze. Alter lines 5000 and 5010 to give the positions of the holes in your maze, counting squares along the rows from the top left corner. If you want to start the ball in a different position, or relocate the final hole, amend lines 340 and 350 accordingly.

13
Singalong

Words and music by the Commodore 64! To be truthful, you provide the words, but the computer puts them together in dozens of different ways. The limericks which it invents can be as amusing or as ridiculous as you like, and much more hilarious if you key in the names of your friends and relatives. Play this for fun on your own, or get the party "singing more with the 64"!

How to play

All you need to do is to type in the program and RUN it. You will first hear the theme tune. Then the computer displays a rhyming verse, made up from the words and phrases already included in the program. You are asked 'SINGALONG?' If you want to sing the words to music, press key Y (for 'yes'). The tune is repeated, with the words still on the screen. After this, you are asked 'AGAIN?'. If you press key Y again the tune is repeated.

If you want a new verse, press any other key than Y and your 64 will compose something new.

Keying in

The strings in lines 60 to 80 all begin and end with a space. Also, there are 4 spaces between the question mark and the reversed disk (CRSR UP) in line 170.

Control characters used are:

CLEAR: lines 50, 210
CTRL/2: lines 50, 210
CRSR DOWN: lines 50(3), 480(3)
CRSR UP: lines 170, 480
CTRL/7: lines 170, 480

Program design

10–80	Initialising.
90–150	Playing the tune.
160–190	Inviting repeat of the tune (skipped on first run of program).
200–360	Random selection of words and phrases from DATA lines.
370–470	Assembling and displaying the verse, line by line.
480–500	Inviting 'SINGALONG'.
510–540	Dummy DATA read.
1000–1060	DATA for the tune.
2000–8000	DATA for the verse.

Points of interest

The words are READ directly from the DATA statements each time a new verse is composed. It would have been much easier to have transferred the words to an array, where they could have been more simply handled, but this would have meant holding the words in RAM twice, both in the program and in an array. This is wasteful of memory and would restrict the number of extra words and phrases that the user can add to the program.

Random numbers are used a lot, so a special function is defined for these (line 30).

The program

```
10 REM ** SINGALONG **
20 FOR J=0 TO 24:POKE 54272+J,0:NEXT
30 V=54272:POKE V+5,9:POKE V+6,130:Z=0
40 DEF FNR(X)=INT(RND(1)*X)+1
50 PRINT"⊐▬▯▯▯▯":POKE 53280,5:POKE53281,1
0
60 X$(1)=" CALLED ":X$(2)=" OF "
70 Y$(1)=" ALWAYS ":Y$(2)=" NEVER ":Y$(3
)=" SELDOM "
80 Z$(1)=" DAY ":Z$(2)=" NIGHT "
90 POKE V+24,15:FOR J=1 TO 41
100 READ FH,FL,D
110 POKE V,FL:POKE V+1,FH:POKE V+4,33
120 FOR K=1 TO 150*D
130 NEXT
140 POKE V+4,32
150 NEXT:POKE V+24,0
160 IF Z=0 THEN Z=1:GOTO 200
170 PRINT"▇AGAIN?     ⊐"
180 GET A$:IF A$="" THEN 180
190 IF A$="Y" THEN RESTORE:GOTO 90
200 RX=FNR(2):RY=FNR(3):RZ=FNR(2)
210 PRINT"⊐▬":FOR J=1 TO 7
220 READ Q,S
230 IF J=4 THEN RR=FNR(2):W=(RX-1)*S/Q+(
RQ~1)*Q+RR:GOTO 290
240 RQ=FNR(Q)
250 P=1
260 IF J=3 THEN P=RX
270 IF J=5 THEN P=RZ
280 W=(P-1)*Q+RQ
290 FOR K=1 TO W
300 READ W$(J)
310 NEXT
320 IF K=S+1 THEN 360
330 FOR K=W+1 TO S
340 READ W$
350 NEXT
360 NEXT
370 V$ = "THERE WAS A " + W$(1) + " " +
W$(2) + X$(RX) + W$(3)
380 PRINT:PRINT V$
390 V$ = "WHO" + Y$(RY) + W$(4)
400 PRINT:PRINT V$
410 V$ = "THEN " + W$(5)
420 PRINT:PRINT V$
430 V$ = "ONE " + W$(6) + Z$(RZ)
440 PRINT:PRINT V$
450 V$ = "THAT " + W$(7) + " " + W$(1)+
" " + W$(2) + X$(RX) + W$(3)
460 PRINT:PRINT V$
470 RESTORE
480 PRINT "▯▯▯▯▇SINGALONG?⊐"
490 GET A$:IF A$="" THEN 490
500 IF A$="Y" THEN 90
510 FOR J=1 TO 123
520 READ X
530 NEXT
540 GOTO 200
```

```
1000 DATA 16,195,1,22,96,2,22,96,1,22,96
,1,22,96,2,21,31,1
1010 DATA 18,209,1,21,31,2,25,30,3,0,0,1
,16,195,1,25,30,2
1020 DATA 25,30,1,25,30,1,25,30,2,22,96,
1,21,31,1,22,96,2
1030 DATA 28,49,3,0,0,1,28,49,1,33,135,2
,33,135,1,33,135,1
1040 DATA 33,135,2,0,0,1,28,49,1,29,223,
2,29,223,1,29,223,1
1050 DATA 29,223,2,0,0,1,25,30,1,28,49,2
,33,135,1,28,49,1
1060 DATA 29,223,2,28,49,1,25,30,1,28,49
,2,22,96,2
2000 DATA 4,4,FAT,GREAT,SLY,BRIGHT
3000 DATA 4,4,WRITER,LOAFER,ARTIST,BEAUT
Y
4000 DATA 3,6,NELLY,PARKIN,NEVILLE
4010 DATA LUTON, ELY,NORWICH
5000 DATA 2,12,THREW BRICKS AT THE TELLY
,ATE ICE CREAM AND JELLY
5010 DATA SET ALL THE DOGS BARKING,WAS F
OOLING AND LARKING
5020 DATA SAID 'I'M A REAL DEVIL',WAS QU
ITE ON THE LEVEL
5030 DATA WENT HUNTIN' AND SHOOTIN',PLAY
ED CHESS WITH HIS BOOTS ON
5040 DATA SAID 'IS IT TRUE? - REALLY?',H
AD EYES THAT WERE STEELY
5050 DATA ATE MUSTARD WITH PORRIDGE,LACK
ED VIGOUR AND COURAGE
6000 DATA 3,6,SPENT ALL HIS PAY,FELL IN
THE BAY,JOINED THE AA
6010 DATA DIED OF SHEER FRIGHT,PUT OUT T
HE LIGHT,LOOKED SUCH A SIGHT
7000 DATA 5,5,TERRIBLE,COLD FROSTY,ILL F
ATED,FABULOUS,MEMORABLE
8000 DATA 6,6,STUPID,CLEVER,LAZY,CRAFTY,
EVIL,PRETTY
```

Variations

The tune can be changed by altering lines 1000 to 1060. Each note is stored there as three numbers, representing high frequency, low frequency and duration, in that order. Alter line 90 according to the number of notes in your replacement tune.

The music routine uses only one voice but could readily be expanded to provide a harmonic accompaniment. There are lots of possibilities of imitating different instruments (possibly at random) and adding some percussion effects too.

Most of the fun with this program comes from making up your own words and rhyming phrases. These are typed in as DATA lines, following the scheme adopted in the listing:

Line 2000: A list of words describing a person, each with one syllable, and not beginning with a vowel. Begin the list by typing the

number of words in the list, twice. The numbers and the words must all be separated by commas. Do not type spaces between the words. If you have more words than can go on one DATA line (3 screen lines) continue the list in lines 2010, 2020, 2030, etc. The same applies to the other lists described below. Do not put numbers on the extra lines. In the '2000' range, for example, the numbers of words on all lines is given on line 2000.

Line 3000: A list of 'kinds of person', each consisting of two syllables. It is best for the accent to be on the first syllable, but you can often 'force' a word to fit in, especially when singing along.

Line 4000: A list of names of people, followed by a list of names of places. You must have the same number of names in both lists. The names should all have two syllables, with the accent on the first. Begin line 4000 with two numbers, the first is the number of names in each list, the second is double this.

Line 5000: Phrases which describe what the person did. These rhyme with the names of people and places in line 4000. Their rhythm is 'di-DA-di-di-DA-di'. You can have as many phases to rhyme with each name as you like, but you must have the same number of rhyming lines for each name. Line 5000 begins with the number of rhymes for each name followed by the total number of phrases in all the lines in the '5000' range.

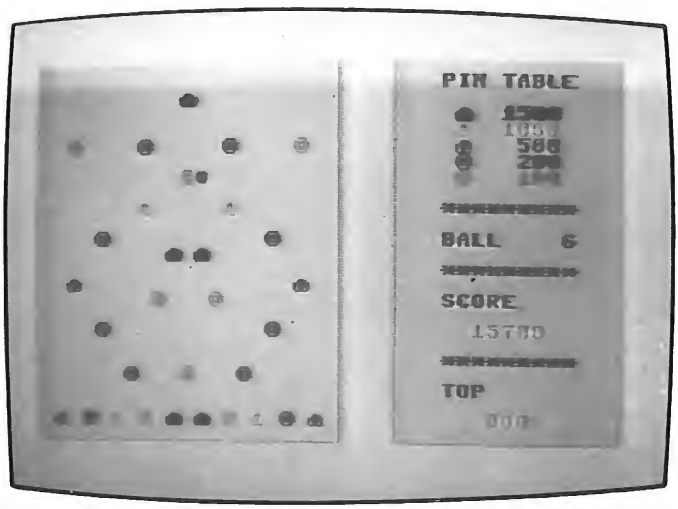
Line 6000: Two sets of rhyming lines about what a person did. The first set rhymes with 'day' while the second set rhymes with 'night'. The rhythm for these is 'DA-di-di-DA'. Line 5000 begins with the number of phrases rhyming with 'day'; the second number is double this.

Line 7000: A list of words describing a day or night; three syllables, with the accent on the first. Begin the line with the number of words, twice.

Line 8000: A list of words describing a person; two syllables, with the accent on the first. Begin the line with the number of words, twice.

If you find the above rather complicated just try substituting a few words and phrases of your own in the listing given, first.

14
Pintable



With a certain amount of skill, and a large amount of luck you can soon amass a huge score on the Pintable. This is a game for all members of the family, even the youngest. After a few turns at the table you will be compelled to try again to break your own record score.

How to play

When you first run the program, it takes a few tens of seconds for the computer to transfer the details of the Pintable into memory. Then the Pintable is displayed on the left of the screen and the score panel on the right. The table has circular 'pins', which are red or green. Every time the ball bounces against one of these you score 100 or 200. The score panel shows you what scores you get. The table also has several pockets into which the ball may fall. The ball may bounce against the outside of the pocket, but there is no score for this. If it falls into a pocket from above, you score 500, 1000 or 1500, depending on the colour of the pocket.

You have 8 balls to play, the number played being shown on the

score panel. When the display appears, the first ball is already bouncing from left to right at the top of the table. When you judge the moment to be right, press any key. The ball is then released and begins to run toward the bottom end of the table. From then on, just watch it bounce on the pins and, with luck, eventually run into a pocket.

As soon as the ball enters a pocket, or reaches the bottom of the table, another ball appears at the top, ready for you to play. When all 8 balls have been played, your record score for the session is displayed. The table is ready for the next game immediately, and the first ball of the next game bounces across the top.

Winning tactics

It is fun to try to release the ball so that it drops straight into the black pocket at the top of the table. But you are likely to get a better score if you let the ball run down the table, bouncing repeatedly against the pins. With practice, you will find that there are certain groups of pins that send the ball bouncing from one to the other many times, giving you a very large score. However, there is a random element built into the game, equivalent to the mechanical variations present in a real pintable, which ensure that the ball cannot bounce to and fro for ever. Sometimes you may have a lucky break, but at other times the bouncing ceases almost as soon as it has begun.

Keying in

Check the DATA lines carefully after you have typed them in. An error there could cause the ball to shoot off in odd directions, and probably spoil the display.

The control characters used are:

CLEAR: line 50
CTRL/5: lines 220, 310, 410, 420
CTRL/7: lines 240, 280, 320, 350, 360, 400, 410, 420
CTRL/1: lines 250, 330, 410
CTRL/6: lines 280(2), 300(2), 360, 390, 410, 420
CTRL/3: lines 280, 290, 310, 320, 340, 370, 380, 390, 410, 420

Lines 230–430 print the display. Some of the lines contain '$', "£",

and "!". On the screen these appear as the symbols for a pin, a pocket, and a solid block of colour, respectively.

It is best to SAVE this program as soon as you have typed it in, before RUNning it.

Program design

20–150	Initializing and transferring special symbols and table to RAM.
160–170	Pointing the VIC-II chip to the special symbols in RAM.
180–210	Displaying the ball in yellow, on every 'square' of the table; the table is yellow, so the ball is invisible.
220–430	Setting up the display.
440–500	Bouncing the ball across the top of the screen, until a key is pressed (making the ball blue, so it can be seen).
510–680	Moving the ball.
690–730	Displaying the score.
800–870	Subroutine for reading and displaying the score (entered at line 810, if the ball has not just fallen into a pocket).
900–940	Subroutine to make sound effects.
1000–1010	DATA for the special symbols.
1120–1260	DATA for the table.

Points of interest

The DATA must be instantly accessible, so cannot be used directly from the DATA statements. As all values are less than 255, the DATA is scored directly in RAM as single bytes. It can then be read quickly by PEEKing. The DATA contains coded information for every location on the table, mainly about how the ball must move, given its direction of travel on arrival. It also holds information about scores for each pin or pocket.

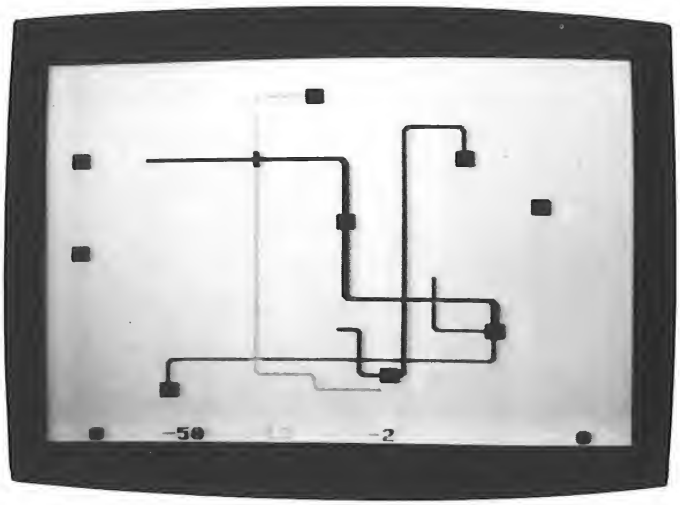
The program

```
10 REM ** PINTABLE **
20 CN=12288:CH=53248:SG=54272:VC=56334
30 FOR L=1 TO 24:POKE SG+L,0:NEXT
40 G=1024:H=55296:POKE SG+5,8:POKE SG+15
,175:POKE SG+24,15
50 PRINT"□";CHR$(124);
60 POKE 53280,4:POKE 53281,7:POKE 52,48:
POKE 56,48
70 POKE VC,PEEK(VC)AND254:POKE 1,PEEK(1)
AND251
80 FOR J=0 TO 463
90 POKE CN+J,PEEK(CH+J)
100 NEXT
110 FOR J=12552 TO 12559
120 POKE J,255
130 NEXT
140 FOR J=12560 TO 12591:READ X:POKE J,X
:NEXT
150 FOR J=12752 TO 13276:READ X:POKE J,X
:NEXT
160 POKE 1,PEEK(1)OR4:POKE VC,PEEK(VC)OR
1
170 POKE 53272,(PEEK(53272)AND240)+12
180 FOR J=0 TO 20
190 FOR K=0 TO 24
200 POKE H+J+40*K,7:POKE G+J+40*K,37
210 NEXT:NEXT
220 D$="■!!!!"
230 PRINTTAB(21)D$
240 PRINTTAB(21)D$;"    ■PIN TABLE"
250 PRINTTAB(10)"■#";SPC(10);D$
260 PRINTTAB(21)D$;"      ■#   1500"
270 PRINTTAB(21)D$;"      #   1000"
280 PRINTTAB(2)"■$";SPC(4);"■$";SPC(5);"
$";SPC(4);"■$";SPC(2);D$;"      ■#    500"
290 PRINTTAB(21)D$;"      ■$    200"
300 PRINTTAB(10)"■$";SPC(10);D$;"     ■$
  100"
310 PRINTTAB(21)D$:PRINTTAB(7)"■#";SPC(5
);"#";SPC(7);D$;"    ■**********"
320 PRINTTAB(21)D$:PRINTTAB(4)"■$";SPC(1
1);"$";SPC(4);D$;"    ■BALL     1"
330 PRINTTAB(9)"■#";SPC(1);"#";SPC(9);D$
340 PRINTTAB(21)D$;"    ■**********"
350 PRINTTAB(2)"■#";SPC(15);"#";SPC(2);D
$
360 PRINTTAB(8)"■$";SPC(3);"$";SPC(8);D$
;"    ■SCORE"
370 PRINTTAB(21)D$:PRINTTAB(4)"■$";SPC(1
1);"$";SPC(4);D$;"        000   "
380 PRINTTAB(21)D$:PRINTTAB(21)D$;"    ■*
**********"
390 PRINTTAB(6)"$";SPC(3);"■$";SPC(3);"■
$";SPC(6);D$
400 PRINTTAB(21)D$;"    ■TOP":PRINTTAB(21
)D$
410 PRINTTAB(1)"■#";SPC(1);"■$";SPC(1);"
■#";SPC(1);"■$";SPC(1);"■#";SPC(1);"#";
420 PRINTSPC(1);"■$";SPC(1);"■#";SPC(1);
"■$";SPC(1);"■#";SPC(1);D$;"        000
"
430 PRINTTAB(21)D$;
440 FOR J=1 TO 8
450 POKE G+476,J+48:VX=1:VY=1:Z=0:NZ=0
```

```
460 NZ=Z+VX:
470 POKE H+Z,7:POKE H+NZ,6:Z=NZ
480 IF Z=20 AND VX=1 THEN VX=-1:GOTO 500
490 IF Z=0 THEN VX=1
500 GET A$:IF A$="" THEN 460
510 Q=Z+12752:P=PEEK(Q)AND15:RN=INT(RND(
1)*6)
520 IF P=12 THEN GOSUB 800:GOTO 680
530 IF P=15 THEN POKE H+INT(Z/21)*40+Z-I
NT(Z/21)*21,7:GOSUB 900:GOTO 680
540 IF P=0 AND VY=-1 AND RN<4 THEN VY=1:
GOTO 630
550 IF P=3 OR P=7 THEN VX=0:GOTO 630
560 IF P=4 AND RN>2 AND VX=0 THEN VX=4-R
N
570 IF P=5 OR P=6 THEN D=1
580 IF D=1 AND VY=1 AND RN>3 THEN VX=0:G
OTO 630
590 R=PEEK(Q)AND1:IF R=1 AND VX=1 OR R=1
 AND VX=0 THEN VX=-1
600 R=PEEK(Q)AND2:IF R=2 AND VX=-1 OR R=
2 AND VX=0 THEN VX=1
610 R=PEEK(Q)AND 4:IF R=4 AND VY=1 THEN
VY=-1
620 R=PEEK(Q)AND8:IF R=8 AND VY=-1 THEN
VY=1
630 NZ=Z+VX+21*VY:GOSUB 810
640 POKE H+INT(Z/21)*40+Z-INT(Z/21)*21,7
650 POKE H+INT(NZ/21)*40+NZ-INT(NZ/21)*2
1,6
660 Z=NZ:D=0
670 GOTO 510
680 NEXT
690 IF SM<S THEN SM=S:S=0:SM$=STR$(SM)
700 FORL=1 TO LEN(SM$):POKE G+948+L,ASC(
MID$(SM$,L,1)):NEXT
710 GOSUB 900:GOSUB 900
720 FOR L=708 TO 716:POKE G+L,32:NEXT
730 GOTO 440
800 POKE H+INT(Z/21)*40+Z-INT(Z/21)*21,7
810 SP=PEEK(Q)AND240:SN=INT(SP*100/16)
820 IF SN>200 AND P<>12 THEN GOSUB 900:R
ETURN
830 IF SN=0 THEN RETURN
840 S=S+SN:S$=STR$(S)
850 FOR L=1 TO LEN(S$):POKE G+708+L,ASC(
MID$(S$,L,1)):NEXT
860 GOSUB 900
870 RETURN
900 POKE 53280,RN+6:POKE SG+1,90+SN/100
910 POKE SG+4,21
920 FOR L=1 TO 200:NEXT
930 POKE 53280,4:POKE SG+4,20
940 RETURN
1000 DATA 0,24,60,189,189,189,153,126,60
,66,189,165,165,189,66,60
1010 DATA 0,60,126,126,126,126,60,0
1020 DATA 10,8,8,8,8,8,8,8,8,8,8,8,8,8,8
,8,8,8,8,8,9
1030 DATA 2,0,0,0,0,0,0,0,0,5,252,6,0,0,
0,0,0,0,0,0,1
1040 DATA 2,0,0,0,0,0,0,0,0,1,0,2,0,0,0,
0,0,0,0,0,1
1050 DATA 2,0,0,0,0,0,0,0,0,9,8,10,0,0,0
,0,0,0,0,0,1
```

```
1060 DATA 2,1,20,22,0,0,37,36,38,0,0,0,3
7,36,38,0,0,21,20,22,1
1070 DATA 2,17,0,18,0,0,33,0,34,0,0,0,33
,0,34,0,0,17,0,18,1
1080 DATA 2,25,24,6,0,0,41,40,42,21,20,2
2,41,40,42,0,0,25,24,26,1
1090 DATA 2,0,0,0,0,0,0,0,0,17,0,18,0,0,
0,0,0,0,0,0,1
1100 DATA 2,0,0,0,0,0,5,172,6,25,24,26,5
,172,6,0,0,0,0,0,1
1110 DATA 2,0,0,0,0,0,1,0,2,0,0,0,1,0,2,
0,0,0,0,0,1
1120 DATA 2,0,0,37,36,38,9,8,10,0,0,0,9,
8,10,37,36,38,0,0,1
1130 DATA 2,0,0,33,0,34,0,0,5,252,7,252,
6,0,0,33,0,34,0,0,1
1140 DATA 2,0,0,41,40,42,0,0,1,0,3,0,2,0
,0,41,40,42,0,0,1
1150 DATA 2,5,92,6,0,0,0,0,9,8,8,8,10,0,
0,0,0,5,92,6,1
1160 DATA 2,1,0,2,0,0,0,21,20,22,0,21,20
,22,0,0,0,1,0,2,1
1170 DATA 2,9,8,10,0,0,0,17,0,18,0,17,0,
18,0,0,0,9,8,10,1
1180 DATA 2,0,0,37,36,38,0,25,24,26,0,25
,24,26,0,37,36,38,0,0,1
1190 DATA 2,0,0,41,40,42,0,0,0,0,0,0,0,0
,0,41,40,42,0,0,1
1200 DATA 2,0,0,31,30,32,0,0,0,0,0,0,0,0
,0,31,30,32,0,0,1
1210 DATA 2,0,0,0,0,37,36,38,0,21,20,22,
0,37,36,38,0,0,0,0,1
1220 DATA 2,0,0,0,0,33,0,34,0,17,0,18,0,
33,0,34,0,0,0,0,1
1230 DATA 2,0,0,0,0,33,0,34,0,17,0,18,0,
33,0,34,0,0,0,0,1
1240 DATA 3,92,39,36,39,172,23,20,23,252
,4,252,23,20,23,172,39,36,39,92,3
1250 DATA 3,0,35,0,35,0,19,0,19,0,3,0,19
,0,19,0,35,0,35,0,3
1260 DATA 15,15,15,15,15,15,15,15,15,15,
15,15,15,15,15,15,15,15,15,15,15
```

15
Rail Runner



This is a game of strategy for 2 to 4 players. Your aim is to be the first to lay a railway line along a route selected by the computer and to run a train along it from start to finish.

How to play

Before RUNning the program the players must decide in which order they are going to play. Each player has a special colour:

1st player – black
2nd player – red
3rd player – purple
4th player – blue

The computer first asks how many persons are playing. Enter a number from 2 to 4 and press 'RETURN'. Now the computer chooses a route at random for each player and displays these. Points on the routes are referred to by the names of well-known towns. Each route begins at one town, passes through another, and ends at a third town. Each player should write down the names of the towns

on his or her route. When everyone has done this, press any key.

The towns appear on the screen as coloured blocks, placed at random within certain areas of the screen:

	LEFT	MIDDLE	RIGHT
TOP	Ayr (purple)	Burnley (green)	Cardiff (blue)
MIDDLE	Derby (yellow)	Everton (orange)	Falkirk (brown)
BOTTOM	Grimsby (pink)	Halifax (dark grey)	Ipswich (light grey)

The colours of towns have nothing to do with the players' colours. The arrangement of towns on the screen is alphabetical, not geographical!

Identify the towns on your route and, when you hear the 'toot-toot', it is the first player's turn. The border of the screen always changes colour to show whose turn it is.

In the first stage of the game you are laying the track. The track begins at the first station and you extend it by pressing one of these keys:

U upward (toward top of screen)
H toward the left
J toward the right
N downward

As soon as you press the key, the track appears on the screen, displayed in your playing colour. The first piece of track laid is straight, coming from one of the four sides of the 'station' block. If you change direction, in a subsequent turn, the computer prints a curved track to take you in the required direction.

Each player builds track by pressing one of the keys when his, or her, colour is displayed on the border of the screen. Here are the rules for track-laying:

(i) On leaving a station, the track is always a straight piece (no corners): the computer is programmed to do this automatically.

(ii) No U-turns allowed: if you try to turn 'up' when the end of your track is directed 'down' for example, the computer gives you a second chance to press a key. Turn to the left (or right) and make a

second turn to the left (or right) when you play again.

(iii) No track can be laid on squares occupied by another player, except that if the other player's track is a straight track running across the way you want to go, the computer will print a '+' crossover for you.

(iv) When your track has reached a station, you need to make another key press in your next turn, to actually enter the station.

(v) Having entered a station, you can leave it in any direction, not already taken by an existing track.

(vi) Not more than one line can go to each of the four sides of a station.

(vii) Do not lay your line through any station except the three on your list. The computer does not prevent you from doing this, but if you go through a wrong station then you will not be able to complete the second part of the game.

Depending on where the stations are located, and the routes allotted to the players, it may happen that a player's route is completely blocked by the lines of other players. The situation is so complex that the computer cannot ensure that this is avoided. Players may agree to stop the game and start again, or may prefer to accept that it is just 'hard luck' on the blocked player, who drops out of the game.

If you make a mistake, or for any other reason want to change your layout, you can remove pieces of track one at a time. When it is your turn, press the 'left-arrow' key. The last portion of track is then removed and the turn passes to the next player. You may take up as many sections of track as you wish, as far back as the last station visited.

Scoring is done automatically. The running scores are displayed at the bottom of the screen, in each player's colour. Scoring is as follows:

For each straight track laid: 1
For crossing another player's track: −10
For laying track to the second station on the route: 10
For laying track to the final station: 20
For being the first train to arrive at its final station: 50

When a player has completed the line to the final station, the next stage of the game begins for that player. In this stage the players train (a circular disk) is automatically moved along the track from start to finish. A random number (from 1 to 4), displayed in the right bottom corner of the screen, decides how far the train moves in each

turn. Note that the train may change colour as it goes over a '+' crossing, but recovers its proper colour on the other side.

The game ends when:

(i) the first train enters it's final station,
or
(ii) two trains collide at a '+' crossing.

The winner is the player with the highest score.

Winning tactics

These apply to the track-laying stage of the game. Long straight runs build up a good score. Plan ahead, to avoid having to lay non-scoring curves. Taking up track wastes time and carries a heavy penalty in points. This is another reason for planning, and also for anticipating where other players are likely to lay their track. When approaching a station, remember that someone else may get there before you, forcing you to carry your track around the station to enter on the far side. The shorter the track, the more quickly will your train reach its destination.

Keying in

Line 530 is too long to be keyed in exactly as listed. When you type it in, use the standard abbreviations for: AND 'A' followed by 'SHIFT' with 'N', THEN 'T' followed by 'SHIFT' with 'H'.

Control characters used are:

CLEAR: lines 80, 260, 330
CTRL/7: lines 80, 280(2), 290, 2030
CRSR down: lines 260(2), 280, 300(2), 1200(2), 1300(2)
CTRL/3: lines 280(2), 290(2), 2020
HOME: lines 1200, 1300, 2010
CTRL/1: line 2020
CTRL/5: line 2020

Unlisted characters used:

C=/2 line 2030

Program design

20–70	Initialising.
80–90	Requesting number of players.
100–250	Setting up arrays and selecting routes.
260–320	Displaying routes.
330–350	Displaying stations and scores (initially zero).
360–390	Re-entry point for beginning of each player's turn; whistle sound-effect.
400–460	Getting input from players.
470–630	Analysing intended move and, if valid, making it.
640–680	Dealing with a move when line reaches a station.
800–960	Routines for taking up track.
1000–1180	Moving trains in 2nd part of game.
1200–1210	End of game; collision.
1300–1320	End of game; 1st train home.
2000–2040	Subroutine for displaying scores and 'dice'.
3000	Subroutine for delay in sound effects.
3500–3610	Subroutine for making whistle sound.
4000	Subroutine for making puffing sound.
5000	DATA for deciding which kinds of track are required.

The program

```
10 REM ** RAIL RUNNER **
20 G=1024:H=55296:S=54272:C=53280
30 S$(0)="AYR":S$(1)="BURNLEY":S$(2)="CA
RDIFF"
40 S$(3)="DERBY":S$(4)="EVERTON":S$(5)="
FALKIRK":S$(6)="GRIMSBY"
50 S$(7)="HALIFAX":S$(8)="IPSWICH"
60 DEF FNR(X)=INT(RND(1)*(X+1))
70 POKE C,6:POKE 53281,7:GOSUB 3000
80 PRINT"■":INPUT"HOW MANY PLAYERS";P$
90 P=VAL(P$):IF P<2 OR P>4 THEN 40
100 DIMD%(3,3),P%(3,10),S%(1,8):FOR K=0
TO 3:FOR J=0 TO 3:READ D%(J,K):NEXT:NEXT
110 FOR J=0 TO S
120 S%(0,J) = 82 + INT(J/3)*280 + (J-INT
(J/3)*3)*13 + FNR(9) + FNR(5)*40
130 NEXT
140 FOR J=0 TO P-1
150 P%(J,0)=FNR(8):IF S%(1,P%(J,0))=4 TH
EN 150
160 S%(1,P%(J,0))=S%(1,P%(J,0))+1
170 P%(J,1)=FNR(8):IF S%(1,P%(J,1))=4 TH
EN 170
180 IF P%(J,1)=P%(J,0) THEN 170
190 S%(1,P%(J,1))=S%(1,P%(J,1))+2
200 P%(J,2)=FNR(8):IF S%(1,P%(J,2))=4 TH
EN 200
```

```
210 IF P%(J,2)=P%(J,1) OR P%(J,2)=P%(J,0
) THEN 200
220 S%(1,P%(J,2))=S%(1,P%(J,2))+1
230 P%(J,3)=S%(0,P%(J,0))
240 P%(J,10)=160
250 NEXT
260 PRINT"■ROUTES:■■"
270 FOR J=0 TO P-1
280 PRINT"■";J+1;TAB(8)"■FROM";TAB(18)"■
VIA";TAB(28)"■TO■"
290 PRINT"■";TAB(8)S$(P%(J,0));"■";TAB(1
8)S$(P%(J,1));"■";
300 PRINTTAB(28)S$(P%(J,2));"■■"
310 NEXT
320 GET A$:IF A$="" THEN 320
330 PRINT"■":POKE C,0:POKE 53281,15
340 FOR J=0 TO 8:POKE H+S%(0,J),J+4:POKE
 G+S%(0,J),160:NEXT
350 GOSUB 2000:PN=-1
360 PN=PN+1
370 IF PN=P THEN PN=0
380 POKEC,PN*2:GOSUB 3500
390 IFP%(PN,7)=1 THEN 1000
400 GET A$:IF A$="" THEN 400
410 IF A$="U" THEN D=0:GOTO 470
420 IF A$="J" THEN D=1:GOTO 470
430 IF A$="N" THEN D=2:GOTO 470
440 IF A$="H" THEN D=3:GOTO 470
450 IF A$="←" THEN 800
460 GOTO 400
470 EL=P%(PN,3):CD=P%(PN,4):IF PEEK(G+EL
)=160 THEN CD=D
480 NP=D%(CD,D):IF NP=0 THEN 400
490 IF CD=0 THEN NL=EL-40
500 IF CD=1 THEN NL=EL+1
510 IF CD=2 THEN NL=EL+40
520 IF CD=3 THEN NL=EL-1
530 IFNL<0ORNL>959ORCD=1ANDNL/40=INT(NL/
40)ORCD=3AND(NL+1)/40=INT((NL+1)/40)THEN
630
540 IF PEEK (G+NL)=160 THEN 640
550 IF PEEK(G+NL)=93 AND NP=64 OR PEEK(G
+NL)=64 AND NP=93 THEN NP=91:GOTO 570
560 IF PEEK(G+NL)<>32 THEN 630
570 POKE H+NL,PN*2:POKE G+NL,NP
580 IF NP=91 THEN SC=-10:GOSUB 2000
590 IF NP=93 OR NP=64 THEN SC=1:GOSUB 20
00
600 IF EL=S%(0,P%(PN,0)) THEN P%(PN,8)=D
610 IF EL=S%(0,P%(PN,1)) THEN P%(PN,9)=D
620 P%(PN,3)=NL:P%(PN,4)=D
630 GOTO 360
640 IF NL=S%(0,P%(PN,1)) AND P%(PN,6)=0
THEN P%(PN,6)=1:SC=10:GOSUB 2000
650 IF NL=S%(0,P%(PN,2)) AND P%(PN,6)=1
AND P%(PN,7)=0 THEN 670
660 GOTO 620
670 P%(PN,7)=1:P%(PN,3)=S%(0,P%(PN,0)):S
C=20:GOSUB 2000
680 GOTO 360
800 EL=P%(PN,3):IF PEEK(G+EL)=160 THEN 3
60
810 IF PEEK(G+EL)=91 THEN 930
820 FOR J=0 TO 3
830 IF D%(J,P%(PN,4))=PEEK(G+EL) THEN D=
J
840 NEXT
```

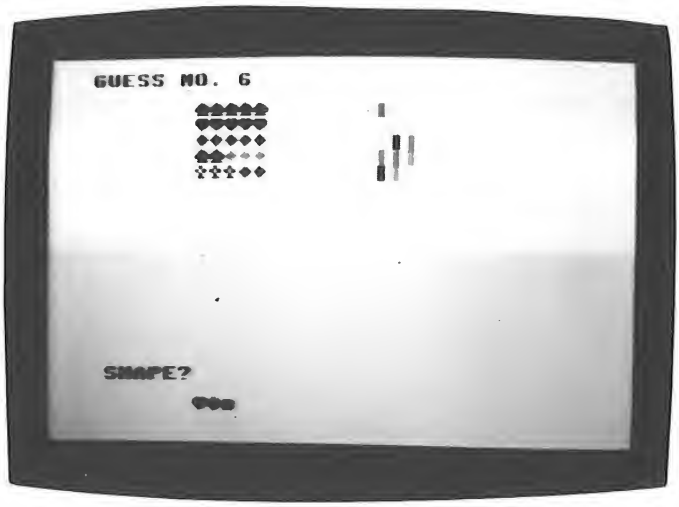
```
850 P%(PN,4)=D
860 POKE G+EL,32:POKE H+EL,1
870 IF D=0 THEN PL=EL+40
880 IF D=1 THEN PL=EL-1
890 IF D=2 THEN PL=EL-40
900 IF D=3 THEN PL=EL+1
910 P%(PN,3)=PL:SC=-10:GOSUB 2000
920 GOTO 360
930 D=P%(PN,4)
940 IF D=1 OR D=3 THEN POKE G+EL,93:POKE
 H+EL,PEEK(H+EL-40)
950 IF D=0 OR D:2 THEN POKE G+EL,64:POKE
 H+EL,PEEK(H+EL-1)
960 GOTO 870
1000 RN=FNR(3)+1:SC=0:GOSUB 2000
1010 FOR J=1 TO RN
1020 D=P%(PN,8):EL=P%(PN,3)
1030 IF EL=S%(0,P%(PN,1))THEN D=P%(PN,9)
1040 IF D=0 THEN NL=EL-40
1050 IF D=1 THEN NL=EL+1
1060 IF D=2 THEN NL=EL+40
1070 IF D=3 THEN NL=EL-1
1080 NP=PEEK(G+NL):IF NP=81 THEN 1200
1090 IF P%(PN,10)<>160 THEN POKE G+EL,P%
(PN,10)
1100 IF NP<>160 THEN POKE G+NL,81
1110 IF NL=S%(0,P%(PN,2)) AND P%(PN,6)=0
 THEN 1300
1120 P%(PN,6)=0:FOR K=0 TO 3
1130 IF D%(D,K)=NP THEN P%(PN,8)=K
1140 NEXT
1150 P%(PN,3)=NL:P%(PN,10)=NP
1160 GOSUB 4000
1170 NEXT
1180 GOTO 360
1200 PRINT"■■■TRAINS COLLIDE"
1210 GOTO 1210
1300 PRINT"■■■FIRST TRAIN ARRIVES"
1310 SC=5:GOSUB 2000
1320 GOTO 1320
2000 P%(PN,5)=P%(PN,5)+SC
2010 PRINT"■";SPC(255);SPC(255);SPC(255)
;SPC(195)
2020 PRINTTAB(2)"■";P%(0,5);TAB(8);"■";P
%(1,5);TAB(14)"■";P%(2,5);
2030 PRINTTAB(22)"■";P%(3,5);TAB(35)"";R
N;
2040 RETURN
3000 FOR Z=0 TO 24:POKE S+Z,0:NEXT:RETUR
N
3500 GOSUB 3000
3510 POKE S+5,46:POKE S+6,0:POKE S+24,15
3520 POKE S,96:POKE S+1,51:POKE S+4,17
3530 POKE S+12,41:POKE S+13,39
3540 POKE S+7,96:POKE S+8,51:POKE S+11,1
29
3550 FOR Z=1 TO 200:NEXT
3560 POKE S+4,16:POKE S+11,128
3570 FOR Z=1 TO 200:NEXT
3580 POKE S+4,17:POKE S+11,129
3590 FOR Z=1 TO 800:NEXT
3600 GOSUB 3000
3610 RETURN
```

```
4000 GOSUB 3000
4010 POKE S+5,132:POKE S+6,128:POKE S+24
,15
4020 POKES,114:POKES+1,11:POKES+4,129
4030 FOR Z=1 TO 100:NEXT
4040 POKE S+1,205
4050 FOR Z=1 TO 200:NEXT
4060 GOSUB 3000
4070 RETURN
5000 DATA 93,75,0,74,85,64,74,0,0,73,93,
85,73,0,75,64
```

Variations

If you want to give the game a more local flavour, alter the names of towns (lines 30–50) to ones of your own choosing.

16
Computer Clues



This is a computer version of what is possibly one of the most popular games ever invented. The computer sets you a problem and you have to find the answer, helped by clues given by the computer. This version allows you to set the level and type of challenge to suit yourself, and to increase and extend it as you become more proficient.

How to play

The idea of the game is that, unseen by you, the computer picks out a set of symbols, or 'pieces'. There can be any number of pieces between 3 and 6. Each piece can be in any one of from 3 to 8 colours. They can also have up to 4 different shapes (spade, heart, diamond, club). When the game begins, you are asked to say how many pieces there are to be, how many colours, and how many shapes. If you decide to have only one shape, this is displayed as a disk. With only 3 pieces of 3 colours and all of them of 1 shape there are 27 possible combinations, but with 6 pieces of 8 colours and 4 shapes the number rises to 1073741800.

The computer now chooses a set of pieces, and your task is to guess the colour and shape of each piece.

After this the screen clears and you are asked to make 'GUESS NO. 1'. The first question asked is 'COLOUR?'. In reply to this, press one of the colour keys (keys 1 to 8) in the top row of the keyboard. The colour is marked on the front of each key except that the 'white' key (No. 2) gives a brown shape. If you have asked for 8 colours, you can press any one of the 8 colour keys, but if, for example, you have asked for only 4 colours, then only the first 4 keys will operate. As soon as you press a key, a disk of the corresponding colour appears near the bottom of the screen. What you do next depends on whether you have asked for 1 shape or more than 1 shape.

If you have asked for 1 shape, the 'COLOUR?' questions remains, and you type in the next colour. Carry on like this until you see the message 'OK? Y/N'. This comes when you have keyed in all the pieces and your set is complete. If the set is as you want it, press key 'Y'. If not, press 'N' and start again. When you press 'Y', the set of symbols disappears from the bottom of the screen, and reappears near the top, on the left. The computer now compares your set with its own set. This takes a second or two, after which you will see a set of coloured stripes at the top right of the screen, level with your set of pieces. The colours of the stripes have these meanings:

Blue: 1 piece of correct colour in the right position.
Red: 1 piece of correct colour.

Note that the stripes are always displayed in the order listed above. This has nothing to do with the position of the pieces in the set. For example, a blue stripe on the extreme left means that one of the pieces (but not necessarily the one on the extreme left) is of the correct colour and is in its correct position in the set. The stripes are the clues as to how correct your guesses have been. If you are really unlucky, you may get no stripes at all, showing that you have chosen shapes and colours completely different to those chosen the computer. If you get one or more blues, you are well on the way toward winning.

Now you are asked to make "GUESS NO. 2". This is done the same way as the previous guess. The computer's set of pieces is still the same as before, so you should use the clues from the first guess to make a better guess this time. When you have keyed in the second set and approved it, it is displayed below the first set, and you see a second set of stripes to the right. You are allowed 16 guesses

altogether. If any of these guesses consists of a set of pieces exactly like that chosen by the computer, you have won. The display informs you of this. If, after 16 turns, you have not managed to find a set exactly like that chosen by the computer, the display tells you that the computer has won. As the game ends, the set chosen by the computer is displayed at the bottom of the screen.

If you decide to have more than 1 colour, the play is a little more complex. At each guess, you are asked to key in the colour, as before. A disk of that colour appears at the bottom of the screen. Then you are asked 'SHAPE?'. Key in any one of the shapes by pressing key A, S, Z, X. The shapes are marked on the front of each of these keys. If you have asked for 2 shapes, the two allowed are 'spade' and 'heart'. If you have asked for 3 shapes, the 'diamond' is added to these. Asking for 4 includes the 'club' too. When you press the shape key, the displayed disk changes to the corresponding shape. Then the question 'COLOUR?' is repeated, for the second piece of the set. In this way you key in both colour and shape for each piece. When you have the full number of pieces you are asked to confirm them by typing 'Y', as usual.

With more than 1 possible shape, the clue stripes can be in one of 6 colours:

Yellow: 1 piece of correct colour and shape in the right position.
Blue: 1 piece of the correct colour (but wrong shape) in the right position.
Green: 1 piece of the correct shape (but wrong colour), in the right position.
Cyan (bluish-green): 1 piece of the correct shape and colour, but in the wrong position.
Red: 1 piece of the correct colour, but wrong in shape and position.
Purple: 1 piece of the correct shape, but wrong in colour and position.

Obviously it takes a lot more thinking to work out how to interpret the clues, but this is all part of the fascination of the colour-plus-shape version. As in the colour-only version, you have 16 guesses, after which the computer displays its chosen set.

Winning tactics

Analysing the clues is a complex task, especially when many shapes and colours are involved. It is too elaborate to go into here. The best

advice is to be systematic, keying in sets all of one colour or shape, then varying the next set according to the clues obtained. Change only one or two pieces at each successive guess, then note how the clues alter. That said, there still remains something in favour of letting your inspiration take over occasionally!

Keying in

There are 8 spaces in the string in line 460. Note the space after the question mark in line 300, and the semi-colons in lines 1000 and 1010.

Control characters uses are:

CLEAR: lines 50, 160
CTRL/7: lines 50, 180, 210, 300
HOME: lines 180, 1000
CRSR DOWN: lines 50, 180, 870, 1010
CRSR RIGHT: line 180(3)
CTRL/3: lines 440, 770
CTRL/1: line 810
CTRL/RVS ON: line 810

Program design

10–40	Initialising.
50–100	Asking for numbers of pieces, colours and shapes.
110–150	Making computer's random choice of pieces.
160–290	Asking for colour.
300–420	Asking for shape.
430–480	Asking set to be confirmed.
490–500	Displaying completed set at top of screen.
510–540	Looking for pieces which match in colour and/or shape in correct positions.
550–620	Looking for pieces which combine correct colour and shape.
630–710	Looking for pieces which are correct in colour or shape.
720–770	Looking for a win by player.
780–790	Clearing arrays and return for next guess.
800–810	Computer wins.
820–910	Clearing arrays, ready for next game.

1000–1030 Subroutine to move cursor.
2000–2030 Subroutine to make up a string of symbols.
3000–3060 Subroutine for clearing arrays.
4000 CHR$ codes for colours and shapes.

The program

```
10 REM ** COMPUTER CLUES **
20 DIM C%(5,2),P%(5,3):G=1024:GC=G+142:H
=55296:HC=H+142
30 FOR J=1 TO 8:READ CC%(J):NEXT:FOR J=0
 TO 4:READ SC%(J):NEXT
40 DEF FNR(X)=INT(RND(1)*X)+1
50 PRINT"■":POKE 53280,6:POKE 53281,1:IN
PUT"■■ PIECES (3-6)";NP$
60 NP=VAL(NP$):IF NP<3 OR NP>6 THEN 50
70 INPUT"■ COLOURS (3-8)";NC$
80 NC=VAL(NC$):IF NC<3 OR NC>8 THEN 50
90 INPUT"■ SHAPES (1-4)";NS$
100 NS=VAL(NS$):IF NS<1 OR NS>4 THEN 50
110 FOR J=0 TO NP-1
120 C%(J,0)=FNR(NC)
130 C%(J,1)=FNR(NS)
140 P%(J,2)=0:P%(J,3)=0
150 NEXT
160 PRINT"■"
170 J=0
180 PRINT"■■■■■■GUESS NO.";J+1
190 FOR K=0 TO NP-1
200 LN=20:GOSUB 1000
210 PRINTTAB(4)"■COLOUR?"
220 GET A$:IF A$="" THEN 220
230 P%(K,0)=ASC(A$)-48
240 IF P%(K,0)<1 OR P%(K,0)>NC THEN 200
250 GOSUB 2000
260 LN=22:GOSUB 1000
270 PRINTTAB(10)P$
280 IF NS=1 THEN 420
290 LN=20:GOSUB 1000
300 PRINTTAB(4)"■SHAPE? "
310 GET A$:IF A$ = "" THEN 310
320 IF A$<>"A" AND A$<>"S" AND A$<>"Z" A
ND A$<>"X" THEN 310
330 IF NS=3 AND A$="X" THEN 310
340 IF NS=2 AND A$="Z" OR NS=2 AND A$="X
" THEN 310
350 IF A$="A" THEN P%(K,1)=1
360 IF A$="S" THEN P%(K,1)=2
370 IF A$="Z" THEN P%(K,1)=3
380 IF A$="X" THEN P%(K,1)=4
390 GOSUB 2000
400 LN=22:GOSUB 1000
410 PRINTTAB(10)P$
420 NEXT
430 LN=20:GOSUB 1000
440 PRINTTAB(4)"■OK? Y/N"
450 GETA$:IF A$="" THEN 450
460 LN=22:GOSUB 1000:PRINTTAB(10)"
  "
470 IF A$="Y" THEN 490
480 FOR K=0 TO NP-1:P%(K,1)=0:NEXT:GOTO
190
```

```
490 LN=J+3:GOSUB 1000
500 PRINTTAB(10)P$
510 FOR K=0 TO NP-1
520 IF P%(K,0)=C%(K,0) THEN P%(K,2)=1:P%
(K,3)=6:C%(K,2)=1
530 IF P%(K,1)=C%(K,1) THEN P%(K,2)=1:P%
(K,3)=P%(K,3) OR 5:C%(K,2)=1
540 NEXT
550 FOR K=0 TO NP-1
560 IF P%(K,2)=1 THEN 620
570 PC=P%(K,0):PS=P%(K,1)
580 M=0
590 IF C%(M,2)=1 THEN 610
600 IF PC=C%(M,0) AND PS=C%(M,1) THEN P%
(K,2)=1:P%(K,3)=3:C%(M,2)=1:GOTO 620
610 M=M+1:IF M<NP THEN 590
620 NEXT
630 FOR K=0 TO NP-1
640 IF P%(K,2)=1 THEN 710
650 PC=P%(K,0):PS=P%(K,1)
660 M=0
670 IF C%(M,2)=1 THEN 700
680 IF PC=C%(M,0) THEN P%(K,2)=1:P%(K,3)
=2:C%(M,2)=1:GOTO 710
690 IF PS=C%(M,1) THEN P%(K,2)=1:P%(K,3)
=4:C%(M,2)=1:GOTO 710
700 M=M+1:IF M<NP THEN 670
710 NEXT
720 X=0:FW=0:FOR K=7 TO 1 STEP -1
730 FOR M=0 TO NP-1
740 IF K=P%(M,3) THEN POKE HC+J*40+X,K:P
OKE GC+J*40+X,225:X=X+1:FW=FW+K
750 NEXT:NEXT
760 FF=7:IF NS=1 THEN FF=6
770 IF FW=FF*NP THEN LN=20:GOSUB 1000:PR
INTTAB(4)"■YOU WIN!!":GOTO 820
780 GOSUB 3000
790 J=J+1:IF J<16 THEN 180
800 LN=20:GOSUB 1000
810 PRINTTAB(4)"■THE COMPUTER WINS"
820 P$=""
830 FOR K=0 TO NP-1
840 IF NS=1 THEN C%(K,1)=0
850 P$=P$ + CHR$(CC%(C%(K,0))) + CHR$(SC
%(C%(K,1)))
860 NEXT
870 PRINT"■";TAB(16)P$
880 GET A$:IF A$="" THEN 880
890 GOSUB 3000
900 FOR K=0 TO NP-1:C%(K,1)=0:NEXT
910 GOTO 50
1000 PRINT"■";:FOR L=1 TO LN
1010 PRINT"■";
1020 NEXT
1030 RETURN
2000 P$="":FOR M=0 TO K
2010 P$=P$ + CHR$(CC%(P%(M,0))) + CHR$(S
C%(P%(M,1)))
2020 NEXT
2030 RETURN
3000 FOR K=0 TO NP-1
3010 FOR M=0 TO 3
3020 P%(K,M)=0
3030 NEXT
3040 C%(K,2)=0
3050 NEXT
3060 RETURN
4000 DATA 144,129,28,159,156,30,31,158,1
13,97,115,122,120
```

17
Spellbound

This is a word game for 2 to 4 players. Your aim is to build up a word from letters which the computer chooses and displays on the screen. The longer the word, the more you score.

How to play

The computer first asks you to key in the number of players, which must be between 2 and 4. As soon as you have done this and pressed 'RETURN', the screen is cleared and set out for the game. At the top, the numbers 1 to 4 show where the letters will appear. The word 'POOL' shows where a special group of letters may appear, as explained later. The scores are displayed across the bottom of the screen. The message 'READY?' in the centre of the screen tells you that the computer is waiting to start. When everyone is ready, press the space bar.

The four sets of letters then appear at the top of the screen, one set below each number. There may be single letters, such as 'A', 'B' or 'H', pairs of letters, such as 'AD' or 'PR', or groups of three letters, such as 'ING' or 'AND'. Your aim is to make up a word using two or

more of these groups. Words made from single groups, such as 'A', 'ME' or 'AND' do not count, though you could make 'ME' from 'M' and 'E', or make 'AND' from 'A', 'N', and 'D'. You may not use the same group of letters more than once in a word. The program ensures that players keep to these rules.

As soon as you spot a word, press the space bar. If two or more players think they have found a word, the first one to press the space bar is the one to play. The screen now displays the message 'GROUPS?'. In reply to this, the player who has found a word keys in the numbers of the groups of letters needed to make up the word. Key them in the order in which they are used. For example the group might be: 1='Y', 2='AND', 3='LA' and 4='H'. To make the word 'HANDY', key in '421'. Then press the space bar to show that the word is finished. Immediately, the word appears in the centre of the screen. Now you are asked 'OK? (Y/N)'. If all players are agreed that the letters do make up a word and that the word is properly spelt, press key 'Y'. Now the computer asks 'WHO?', wanting to know which player made up the word. Key in the player's number, and the score for the word (one point per letter) is added to that players score. Now the groups of letters are replaced by a fresh set and the computer is ready for players to look for a new word.

Sometimes it is not easy to make a word from the letters displayed. If no one has pressed the space bar after 20 seconds, one of the groups of letters is changed. The figures on the right of centre change every second, counting from zero up to 20 seconds, so you can see how much time you have left to make up a word using the existing groups. If everyone agrees that no word is possible from the displayed letters, press function key F7 to change one of the groups, without having to wait the full 20 seconds. Every time a group changes a bonus score is increased by 1. The player to make a word has the bonus added to the score.

If the word is not correctly spelt, or players decide that the word is not to be allowed for any reason, press 'N', when the computer asks 'OK?' The first two letters of the 'word' are then displayed as the pool, and the player gets no score. Now it is possible to use the pool group with groups 1 to 4 to make up words. Press key 'P' along with the number key, when making up a word. If you use the pool group, you score 10 extra points. The pool remains unchanged until it is used, or is replaced by the letters from the next incorrect word.

The game continues until the score of one of the players reaches 100.

Keying in

There are several blank spaces in the following lines: 30(8), 130(1), 280(1), 310(1), 330(4), 350(1), 580(2), 590(4), 680(4) and 3060(3). The semicolons at the ends of lines 80, 190, 280, 680, 1000, 1020, 2030 and 3060 are essential to keep the display tidy.

Control characters used are:

CLEAR: lines 20, 60
CTRL/7: lines 40, 330
CTRL/3: lines 130, 420, 590, 680
CTRL/RVS OFF: line 280
CTRL/5: line 330
CTRL/RVS ON: lines 420, 670, 3060
CTRL/1: lines 670, 3060
HOME: line 1000
CRSR down: line 1020
CRSR left: line 3060(3)

Unlisted characters used are:

C=/2: lines 150 (before Scores), 2030
C=/5: lines 280 (after CTRL/0)

Program design

10–60	Asking number of players.
70–210	Displaying starting screen.
220–250	Displaying first set of groups.
260–310	Waiting for someone to find a word.
320–430	Making up the word.
440–490	Querying correctness of word.
500–570	Scoring.
580–690	Displaying winner, or preparing screen for next turn.
1000–1040	Subroutine for printing at any given line.
2000–2050	Subroutine for displaying scores.
3000–3080	Subroutine for choosing a group of letters at random and displaying it.
4000–4040	DATA containing groups of letters.

The program

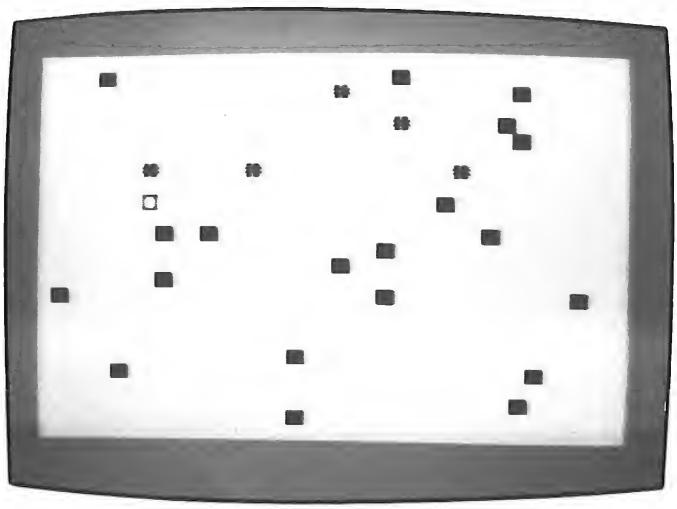
```
10 REM ** SPELLBOUND **
20 PRINT"□":POKE 53280,2:POKE 53281,15
30 B$="          ":FP=1
40 INPUT"■PLAYERS (2-6)";NP$
50 NP=VAL(NP$):IF NP<2 OR NP>6 THEN 20
60 PRINT"□"
70 FOR J=0 TO 3
80 PRINTTAB(8+7*J)J+1;
90 NEXT
100 LN=6:GOSUB 1000
110 PRINTTAB(18)"POOL"
120 LN=11:GOSUB 1000
130 PRINT"■";TAB(6)"READY?   "
140 LN=17:GOSUB 1000
150 PRINTTAB(6)"SCORES:"
160 LN=19
170 GOSUB 1000
180 FOR J=0 TO NP-1
190 PRINTTAB(8+4*J)J+1;
200 NEXT
210 GOSUB 2000
220 GET A$:IF A$="" THEN 220
230 FOR C=1 TO 4
240 GOSUB 3000
250 NEXT:C=0
260 TI$="000000":T$="":NG=0:FOR J=1 TO 4
:F(J)=1:NEXT
270 LN=11:GOSUB 1000
280 PRINT"■";TAB(32)STR$(INT(TI/60));"  "
;
290 GET A$
300 IF A$=CHR$(136) OR TI>1200 THEN C=C+
1:GOSUB 3000:B=B+1:GOTO 260
310 IF A$<>" " THEN 270
320 LN=11:GOSUB 1000
330 PRINT"■";TAB(6)"GROUPS?     ■";
340 GET A$
350 IF A$=" " AND NG>1 THEN 410
360 IF A$="P"ANDFP=0 OR A$="1"AND F(1)=1
 OR A$="2"AND F(2)=1 THEN 390
370 IF A$="3" AND F(3)=1 OR A$="4" AND F
(4)=1 THEN 390
380 GOTO 340
390 IF A$="P" THEN T$=T$+P$:FP=1:NG=NG+1
:PB=10:PRINT"P";:GOTO 340
400 T$=T$+L$(VAL(A$)):F(VAL(A$))=0:NG=NG
+1:PRINT A$;:GOTO 340
410 LN=14:GOSUB 1000
420 PRINT"■■";TAB(13)T$
430 LN=11:GOSUB 1000
440 PRINTTAB(6)"OK? (Y/N)";B$
450 GET A$:IF A$="" THEN 450
460 LN=14:GOSUB 1000:PRINTTAB(13)B$;B$
470 IF A$="Y" THEN 500
480 IF A$="N" THEN 650
490 GOTO 450
500 LN=11:GOSUB 1000:PRINTTAB(6)"WHO?";B
$
510 GET A$:IF A$="" THEN 510
520 IF VAL(A$)<1 OR VAL(A$)>NP THEN 510
530 S(VAL(A$))=S(VAL(A$))+LEN(T$)+PB+B:B
=0
540 GOSUB 2000
550 FOR J=1 TO NP
```

```
560 IF S(J)=>100 THEN WP=J
570 NEXT
580 IF FP=1 THEN LN=8:GOSUB 1000:PRINTTA
B(19)"   "
590 LN=11:GOSUB 1000:PRINT"■";TAB(6)"REA
DY?     ";
600 IF WP=0 THEN 230
610 LN=11:GOSUB 1000
620 PRINTTAB(6)"PLAYER";WP;"WINS"
630 GET A$:IF A$="" THEN 630
640 FOR J=1 TO 4:S(J)=0:NEXT
650 P$=LEFT$(T$,2)
660 LN=8:GOSUB 1000
670 PRINT"■■";TAB(19)P$:FP=0:PB=0
680 LN=11:GOSUB 1000:PRINT"■";TAB(6)"REA
DY?     ";
690 GOTO 230
1000 PRINT"■";
1010 FOR J=1 TO LN
1020 PRINT"■";
1030 NEXT
1040 RETURN
2000 LN=21:GOSUB 1000
2010 FOR J=0 TO NP-1
2020 SC$=STR$(S(J+1))
2030 PRINT"";TAB(8+4*J)SC$;
2040 NEXT
2050 RETURN
3000 N=INT(RND(1)*105)+1:IF C=5 THEN C=1
3010 FOR K=1 TO N
3020 READ X$
3030 NEXT
3040 L$(C)=X$
3050 LN=3:GOSUB 1000
3060 PRINT"■■";TAB(8+7*(C-1))"     ■■■";L$
(C);
3070 RESTORE
3080 RETURN
4000 DATA A,A,AB,AD,AN,AR,AT,AND,AVE,B,B
A,BE,BL,BR,C,CA,CE,CH,D,DA,DE,DO,E,E,E
4010 DATA E,EA,ED,EE,EN,ER,ELL,ENT,EST,F
,FA,FE,FO,G,GE,GR,H,HA,HE,HI,HO
4020 DATA I,I,IN,ING,ISH,IVE,J,L,LA,LD,L
E,LI,LL,LO,M,MA,ME,MI,MO
4030 DATA N,NA,ND,NE,NO,O,O,OT,OND,P,PA,
PE,PR,PLE,PER,R,RA,RE,RM,RT,RAT,RCH,S
4040 DATA SA,SE,SO,SL,ST,SH,T,T,TA,TE,U,
V,VE,W,WA,WO,Y
```

Variations

The selection of letter groups can be varied to suit players of different ages, or other languages. To make the pool more difficult to use, and thus lead to higher scores, the number of letters taken for the pool could be increased to 3 or 4. Alter the '2' in line 650 to '3' or '4'. The score for using pool is '10'; this could be increased by altering the '10' in line 390. It could be helpful to add a routine to sound a bleep shortly before a group of letters is about to be replaced.

18
Minefield



You are an escaped prisoner-of-war trying to make your way back to join your own troops. Between you and safety lies an enemy minefield, which you have to cross on foot. It is a barren area, with only a few trees and bushes for cover. The enemy are aware of your escape and, since you are a high-ranking officer, they are out in force to prevent you from getting back alive.

The enemy has sent armoured vehicles to patrol the minefield, with orders to shoot on sight. Although they are a danger to you, they can also be a help. If you can keep out of sight and watch where the vehicles go, you will be able to discover the un-mined paths across the minefield and so reach safety. But put only one foot wrong and you will be blown sky-high by a mine. When you are in sight, the gunners will shoot you, but they cannot see you when you are hiding in a bush. On the other hand, armoured vehicles do not bother to avoid bushes; they simply drive over them. This could be unfortunate for you, if you happen to be hiding in the flattened bush at the time! Apart from that risk, the bushes are a safe place to seek, for there are no mines in, or under, a bush.

How to play

The game is played at 10 levels, of which the easiest is level 1. The higher the level, the more thickly scattered are the mines. Above level 5 the guns fire further. At the highest levels (8 and over) the vehicles destroy the bushes as they pass over them. If you take too long to cross the minefield, you may find that few places remain in which to hide.

As soon as you have keyed in your chosen level and pressed 'RETURN' the screen clears, the mines are laid (though you cannot see them) and the bushes are displayed in unmined locations. Then the black vehicles of the enemy appear at the top of the screen. You are the red character which is at the top left corner. Your destination is the bottom right corner. The whole of the top row and bottom row are clear of mines so, once you reach the bottom of the screen, it is safe to move along it (though you may still get shot by a passing vehicle).

The vehicles move off into the minefield, travelling down the screen at first. When a vehicle finds a mine ahead, it turns right and then proceeds in a new direction. Vehicles also turn right when one is blocked by another. All the time you will hear the sounds of gunfire, and shells bursting beside you. You move by using these keys:

U move up (toward top of screen)
H move left
J move right
N move down

If you press no key, you stay in the same place. You cannot move off the edge of the screen, but the vehicles can. When you are near one edge of the screen, it is essential to watch the opposite edge, for a vehicle may suddenly appear close to you and shoot you before you have had time to get clear. In levels 4 and lower, the range of the guns is 1 row or column. In other words, the vehicle has to be on the next screen location, vertically or horizontally. At levels 5 and over the range is extended to two rows or columns. When you are hiding in a bush, you cannot be shot but, if a vehicle runs over the bush you are hiding in, you are run over.

The game ends with suitable sound effects, and you are told how long you took to get to safety. If you failed to reach your goal, you are told how long you managed to survive the hazards of the minefield.

Press the space-bar to play again.

Winning tactics

Give the vehicles a few moments to head through the minefield and watch where they go. At the lower levels there is a chance that there is a clear path straight across. Even then, watch out for deflected vehicles cutting across your path and shooting you as they pass. If there is a bush close by, it is worth deviating from your path to hide until the vehicle has passed. Watch out for pairs of vehicles travelling on parallel paths, two or three rows or columns apart. Between them, their guns cover a wide area of the minefield and it is difficult to move out of their way quickly. One of the worst dangers is the vehicle which goes off the screen on one side and instantly reappears on the opposite side. If it goes off the left edge, it reappears one row further up the screen on the right. If it leaves the right edge it reappears one row down on the left. If you are on the extreme edge, it is possible to be shot by a vehicle which approaches the opposite edge. In effect, it catches you unawares by suddenly appearing over the horizon. It is this factor which makes it essential to take care when you have got to the bottom of the screen and are making your final dash for safety.

Keying in

There are no special problems. Control characters used are:

CLEAR: lines 80, 100, 700, 800, 900, 1000
CTRL/2: lines 700
CRSR DOWN: lines 710(3,2), 870(3,2), 980(3,2), 1060(3,3)
CTRL/7: line 1510

Unlisted characters used are:
C=/8: lines 900 (after CLEAR), 1000 (after CLEAR)

Program design

10–50	Initialising variables and sound of vehicles.
60–70	Asking for required level.
80–180	Setting up minefield, with bushes.
190–220	Resetting variables.
230–390	Processing player's move.
400	Processing moves of vehicles, in turn.

610–680 Effect of falling shells.
700–790 'Winning' routine.
800–880 'Mined' routine.
900–990 'Shot' routine.
1000–1050 'Run over' routine.
1500–1510 Making ready for next game.
2000–2010 Subroutine giving duration of notes in victory tune.
3000–3010 Subroutine to clear VICII registers.

Points of interest

The mines are displayed on the screen as asterisks (character code 42) but since they are displayed in light grey (colour code 15, line 90) they are invisible to the player. They are not invisible to the computer, which PEEKs the character code RAM at lines 310 and 460 to find out if the player or a vehicle is about to move on to a mine.

The program

```
10 REM ** MINEFIELD **
20 G=1024:H=55296:DIM M%(60),S%(20)
30 S=54272:CB=53280:CS=53281:GOSUB 3000
40 POKE S+14,8:POKE S+18,32:POKE S+3,2:P
OKE S+24,143:POKE S+6,242:POKE S+4,65
50 POKE CB,5:POKE CS,15:POKE 649,1
60 POKE S+7,100:POKE S+8,1:POKE S+13,240
70 POKE S+11,129:POKE S+24,1
80 PRINT"□":INPUT"LEVEL (1-10)";L$
90 L=VAL(L$):IF L<1 OR L>10 THEN 80
100 PRINT"□"
110 FOR J=1 TO 30+3*L:M%(J)=RND(1)*920+4
0:POKE H+M%(J),15:POKE G+M%(J),42:NEXT
120 FOR J=1 TO 20:S%(J)=RND(1)*920+40
130 FS=0:FOR K=1 TO 30+3*L
140 IF S%(J)=M%(K) THEN FS=1
150 NEXT
160 IF FS=1 THEN 120
170 POKE S%(J)+H,5:POKE S%(J)+G,160
180 NEXT
190 F=0:P=0:PN=0:PD=1:FS=1:POKE G,209:PO
KE H,2:TI$="000000"
200 FOR J=1 TO 5
210 C%(J)=7*J:CN%(J)=C%(J):CD%(J)=2:POKE
 G+C%(J),35:POKE H+C%(J),0
220 NEXT
230 GET A$
240 IF A$="U" THEN PN=P-40
250 IF A$="J" THEN PN=P+1
260 IF A$="H" THEN PN=P-1
270 IF A$="N" THEN PN=P+40
280 IF PN<0 OR PN>999 THEN PN=P
```

```
290 IF A$="H"AND INT(P/40)=P/40 OR A$="J
"AND INT((P+1)/40)=(P+1)/40 THEN PN=P
300 IF PN=999 THEN 700
310 PG=PEEK(G+PN)
320 IF PG=42 THEN 800
330 IF PG=35 THEN PN=P
340 IF PN=P THEN 400
350 POKE G+P,32+FS*128
360 IF FS=1 THEN POKE H+P,5:FS=0
370 SS=PEEK(H+PN)AND7
380 IF SS=5 THEN FS=1
390 P=PN:POKE G+P,209:POKE H+P,2+FS*3
400 J=1
410 IF CD%(J)=0 THEN CN%(J)=C%(J)-40
420 IF CD%(J)=1 THEN CN%(J)=C%(J)+1
430 IF CD%(J)=2 THEN CN%(J)=C%(J)+40
440 IF CD%(J)=3 THEN CN%(J)=C%(J)-1
450 IF CN%(J)<0 OR CN%(J)>999 THEN CN%(J
)=CN%(J)-1000*SGN(CN%(J))
460 PG=PEEK(G+CN%(J))
470 IF PG=42 OR PG=35 OR PG=209 THEN CD%
(J)=CD%(J)+1:CN%(J)=C%(J)
480 IF CD%(J)=4 THEN CD%(J)=0
490 IF FS=1 THEN 530
500 D=ABS(P-CN%(J)):IF D=1 OR D=40 THEN
FG=1
510 IF L>5 AND D=2 OR L>5 AND D=80 THEN
FG=1
520 IF FG=1 THEN 900
530 IF PG=209 THEN 1000
540 IF CN%(J)=C%(J) THEN 600
550 IF L>8 THEN CC%(J)=0
560 POKE G+C%(J),32+128*CC%(J)
570 IF CC%(J)=1 THEN POKE H+C%(J),5:CC%(
J)=0
580 PS=PEEK(H+CN%(J))AND7:IF PS=5 THEN C
C%(J)=1
590 C%(J)=CN%(J):POKE G+C%(J),35:POKE H+
C%(J),0:POKE S+24,1
600 J=J+1:IF J<6 THEN 410
610 IF RND(1)<.8 THEN 230
620 FR=30000+INT(RND(1)*25000)
630 POKE S+4,65:POKE S+11,128
640 POKE S+24,143:FOR J=1 TO 25
650 FQ=FR-PEEK(S+27)*80:HF=INT(FQ/256):L
F=FQ-HF*256
660 POKE S,LF:POKE S+1,HF
670 NEXT
680 POKE S+4,64:POKE S+12,2:POKE S+13,25
2:POKE S+7,20:POKE S+8,2:POKE S+11,129
690 GOTO 230
700 PRINT"■■":POKE CB,6:POKE CS,2:GOSUB
3000
710 PRINTTAB(10)"■■■YOU HAVE ESCAPED!":P
RINTTAB(11)"■■IN";INT(TI/60);"SECONDS"
720 POKE S+24,15:POKE S+5,100:POKE S+6,8
0
730 FL=75:FH=34:D=4:GOSUB 2000:GOSUB 200
0
740 FL=94:FH=32:D=2:GOSUB 2000
750 FL=75:FH=34:GOSUB 2000
760 FL=126:FH=38:GOSUB 2000
770 FL=214:FH=28:D=8:GOSUB 2000
780 FL=177:FH=25:GOSUB 2000
790 POKE S+24,0:GOTO 1500
800 PRINT"■":GOSUB 3000
```

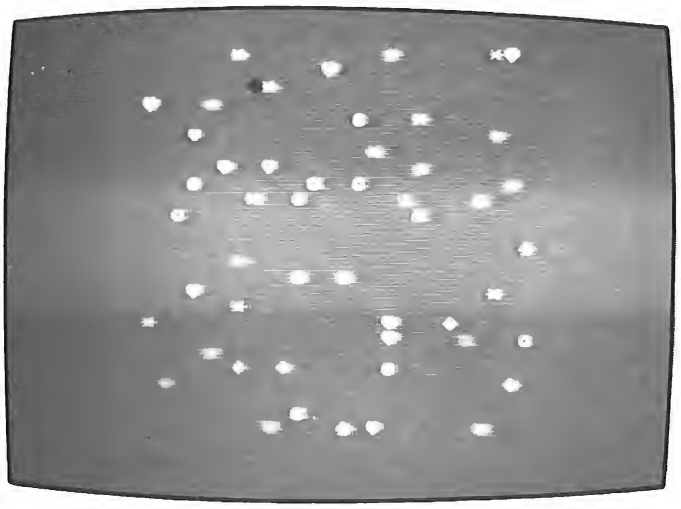
```
810 POKE S+24,15:POKE S+12,0:POKE S+13,2
50:POKE S+7,20:POKE S+8,2:POKE S+11,129
820 POKE CB,0:POKE CS,0
830 FOR K=1 TO 10:NEXT
840 POKE CS,1
850 FOR K=1 TO 5:NEXT
860 POKE S+11,128:POKE CS,0
870 PRINTTAB(12)"■■■YOU WERE MINED":PRIN
TTAB(10)"■■AFTER";INT(TI/60);"SECONDS"
880 GOTO 1500
900 PRINT"■":POKE CB,2:POKE CS,2:GOSUB 3
000
910 POKE S+24,15:POKE S+6,240:POKE S,56:
POKE S+1,27
920 FOR J=1 TO 15
930 POKE S+4,129
940 FOR K=1 TO 5:NEXT
950 POKE S+4,128
960 FOR K=1 TO 20:NEXT
970 NEXT:POKE S+24,0
980 PRINTTAB(13)"■■■YOU WERE SHOT":PRINT
TAB(11)"■■AFTER";INT(TI/60);"SECONDS"
990 GOTO 1500

1000 PRINT"■":POKE CB,2:POKE CS,2
1010 GOSUB 3000
1020 POKE S+24,15:POKE S,144:POKE S+1,23
5:POKE S+5,15:POKE S+6,240:POKE S+4,33
1025 POKE S+7,100:POKE S+8,100:POKE S+13
,240:POKE S+11,129
1030 FOR J=1 TO 1500:NEXT
1040 POKE S+4,32:POKE S+11,128
1050 POKE S+24,0
1060 PRINTTAB(10)"■■■YOU WERE RUN OVER":
PRINTTAB(10)"■■■AFTER";INT(TI/60);"SECON
1500 GET A$:IF A$<>" " THEN 1500
1510 PRINT"■":FG=0:  GOTO 30
2000 POKE S,FL:POKE S+1,FH:POKE S+4,33
2010 FOR K=1 TO 120*D:NEXT
2020 POKE S+4,32
2030 RETURN
3000 FOR J=0 TO 24:POKE S+J,0:NEXT
3010 RETURN
```

Variations

The number of mines planted is 30 plus 3 times the level of play (line 80). To make the game even more difficult, increase the number of mines. In this case line 130 also needs amending to the new value. You can add more vehicles by altering the value of J in line 200. Reduce the initial spacing between them by changing the '7' in line 210 to '6' or '5'. The value in line 600 also needs changing to allow for a greater value of J.

Having more vehicles makes the game slower, although the added time is necessary to give the player time to assess the movements of all the vehicles. As it stands, the game is certainly fast enough but, to increase speed still further, cut out the sound effect routine by deleting lines 610–680.

19
Vibrations



Just for once, the '64' has a chance to be itself. Instead of having to pretend to be a Pintable, an acrobat on a tightrope, or the pieces of a jigsaw puzzle, it can simply be a computer, doing computerish things.

Vibrations is a true computer game, and quite unlike any other game or anything in real life. This is what gives it its fascination. You are pitting your skill and wit against the speed and logic of the computer.

How to play

You are first asked to key in the speed. Speed 1 is slow; it gives you plenty of time to press the keys, though it does not make the solving of the problem any easier. What problem? – this will be explained later. By contrast if you can win at speed 10, your reflexes are precise to within a tenth of a second.

Next, an assortment of symbols of different colours appears on the screen. They start to vibrate. Some move up and down, others move from left to right. After a few seconds, the symbol at the top

left corner of the screen changes its colour to black. None of the other symbols are black, and this black symbol is the one on which you, the player, are riding to begin with.

Now for the problem. You are to ride the symbols, leaping from one to another until you reach the symbol at the bottom right corner of the screen. You can only jump from one symbol to another when they are touching. They must be exactly side-by-side or exactly one above the other on the screen. If you press the correct key at just the right moment, you jump from one to the other. When you make a successful jump the symbol on which you were riding returns to its original colour. The symbol which you jump to becomes black.

To jump, press one of the following keys:

'U' jump up (toward top of screen)
'H' jump to the left
'J' jump to the right
'N' jump downward

It is no good pressing a key and holding it until two symbols are close together. Except when you are playing at the slower speeds, there is not much point in hitting the same key repeatedly. Just watch the rhythmic motions of the symbols and press the key *once* at the instant they come together.

At lower speeds the motion of the symbols appears jerky. This is to be expected, for at speed 1 the computer waits for up to a second for you to press the key, before it moves the symbols to their next positions. At speed 10 you have only a tenth of a second, and their motion is smooth.

Jumping from one symbol to another is only part of the problem. The game can be thought of as a dynamic maze. Although some pairs of symbols seem at first glance to be coming together regularly, closer observation may show that they rarely or perhaps never line up so that you can make the jump. For success, you have to decide your route in advance. Jumping on to certain of the symbols only leads you to a dead end. There is at least one possible passage across the screen, but it makes a bewildering puzzle to find it.

When you reach the symbol in the lower right corner, the game ends. Press the space bar to play again.

The colours and symbols are chosen at random so, even if you have found your way across, it is difficult to recognise the path on subsequent attempts.

Keying in

The usual warnings about keying in the DATA with the greatest care, and SAVEing the program before you RUN it, apply particularly to this program.

Control characters used are:

HOME: lines 30, 700
CRSR down: lines 40(3), 700(3)
CRSR left: lines 40(3), 700(3)
CTRL/2: lines 40, 700

Program design

20–30	Initialising variables.
40–50	Requesting speed.
70–180	Transferring DATA on screen layout to RAM.
190	Transferring machine code program to RAM.
210	Initialising machine code variables.
220–410	Setting up the display.
420–450	Making the symbols oscillate.
460	Putting you on the first symbol.
470–550	Getting and analysing your key press.
560	Move the symbols one step.
570–610	Finding out where your symbol has been moved to.
700–730	Winning routine.
1000–1020	Subroutine for entering data in table.
2000–2020	Subroutine to select symbols and colours at random.
2500–2510	Delay subroutine.
2700–2720	Subroutine to test adjacent area of screen to find where the player's symbol has been moved to.
3000	DATA statement holding codes for symbols.
4000–4030	DATA statements holding details of locations of symbols.
5000–5090	DATA statements holding machine code routine.

Points of interest

The keyboard buffer is POKEd at line 60 (POKE 649,0), so that it cannot hold any input. This makes it impossible to register key

presses before the proper moment. The buffer is opened at line 470 to allow one key press to be registered, and closed again at line 490, immediately after the time for pressing the key is over.

The vibrations rely on horizontal and vertical strips of symbols. Except for one symbol in each strip, they are displayed in red on a red background, so are invisible. At any one time one symbol of each strip is in another colour, and this is the one we see. The machine code routine operates on all the strips in rapid succession. It makes the visible symbol invisible, by changing its colour to red, and makes the adjacent symbol in the strip visible. This gives the illusion of motion to the symbols. If you want to see what the strips actually look like, add this line to the program temporarily:

```
35 POKE 53281,1
```

It spoils the effect, but shows you how the program works.

The program

```
10 REM ** VIBRATIONS **
20 L=0:M=0:P=0:N=0:G=1024:H=55296:B=1228
8
30 PRINT"□":POKE 53280,2:POKE 53281,2:PO
KE 52,48:POKE 56,48
40 INPUT "■■■■■■■■SPEED (1-10)";A$
50 TF=VAL(A$):IF TF<1 OR TF>10 THEN 30
60 PRINT"□":POKE 649,0:FOR J=0 TO 5:READ
 C%(J):NEXT
70 S=1:F=121:X=1:GOSUB 1000
80 S=126:F=251:X=40:GOSUB 1000
90 S=2:F=252:X=0:GOSUB 1000
100 S=4:F=39:X=216:GOSUB 1000
110 S=44:F=64:X=218:GOSUB 1000
120 S=69:F=89:X=217:GOSUB 1000
130 S=94:F=124:X=219:GOSUB 1000
140 S=129:F=149:X=218:GOSUB 1000
150 S=154:F=194:X=216:GOSUB 1000
160 S=199:F=209:X=219:GOSUB 1000
170 S=214:F=254:X=217:GOSUB 1000
180 FOR J=0 TO 250 STEP 5:READ A,M:POKE
B+J,A:POKE B+J+3,M:NEXT
190 FOR J=12546 TO 12732:READ X:POKE J,X
:NEXT
200 RESTORE:FOR J=0 TO 5:READ X:NEXT
210 POKE 251,0:POKE 252,48:POKE 253,0
220 FOR J=1 TO 25
230 GOSUB 2000
240 READ X,Y:IF J>8 THEN X=X+512
250 IF J>13 THEN X=X-256
260 IF J>18 THEN X=X+512
270 IF J=1 THEN P=C
280 POKE H+X,C:POKE G+X,S
290 FOR K=X+1 TO X+Y
300 POKE H+K,2:POKE G+K,S
310 NEXT:NEXT
320 FOR J=1 TO 26
330 GOSUB 2000
```

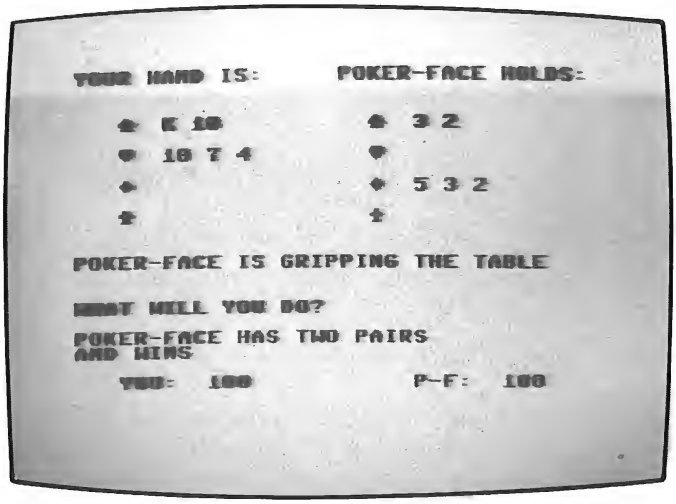
```
340 READ X,Y:X=X+512
350 IF J>5 THEN X=X-512
360 IF J>14 THEN X=X+768
370 IF J>17 THEN X=X-512
380 POKE H+X,C:POKE G+X,S
390 FOR K=X+40 TO X+Y*40 STEP 40
400 POKE H+K,2:POKE G+K,S
410 NEXT:NEXT
420 FOR J=1 TO 50
430 SYS 12546
440 GOSUB 2500
450 NEXT
460 L=H+9:N=PEEK(L)AND7:POKE L,0
470 POKE 649,1:TI$="000000":A$=""
480 GET A$:IF A$="" AND TI<60/TF THEN 48
0
490 POKE 649,0:IF A$="U" AND L>H+39 THEN
 M=L-40
500 IF A$="H" THEN M=L-1
510 IF A$="J" THEN M=L+1
520 IF A$="N" AND L<M+960 THEN M=L+40
530 N=PEEK(M)AND7
540 IF N>2 THEN POKE L,P:POKE M,0:L=M:P=
N
550 IF L=H+989 THEN 700
560 SYS 12546
570 D=-1:GOSUB 2700:IF D=0 THEN 470
580 D=1:GOSUB 2700:IF D=0 THEN 470
590 IF L>H+39 THEN D=-40:GOSUB 2700:IF D
=0 THEN 470
600 IF L<H+960 THEN D=40:GOSUB 2700
610 GOTO 470
700 PRINT"■■■■■■■■■■■■YOU GOT THERE!"
710 POKE 649,1
720 GET A$:IF A$<>" " THEN 720
730 RESTORE:GOTO 30
1000 FOR J=S TO F STEP 5
1010 POKE B+J,X
1020 NEXT:RETURN
2000 S=C%(INT(RND(1)*6))
2010 C=INT(RND(1)*5)+3
2020 RETURN
2500 FOR K=1 TO 200:NEXT
2510 RETURN
2700 X=PEEK(L+D)AND7
2710 IF X=0 THEN L=L+D:D=0
2720 RETURN
3000 DATA 81,87,90,83,65,42
4000 DATA 7,6,228,4,24,6,209,2,57,5,183,
7,95,3,175,6,19,5,229,4,66,4,116,4,96,4
4010 DATA 37,4,254,2,119,5,169,3,151,2,1
,4,215,6,46,4,201,6,53,5,167,4
4020 DATA 101,2,61,4,228,6,115,7,160,5,1
56,7,14,5,250,5,23,3,225,2,31,3,212,7
4030 DATA 182,5,47,5,51,4,0,5,134,2,45,3
,2,2,241,7,48,5
4040 DATA 207,6,53,7,191,6,118,3,5,7,15,
5
5000 DATA 169,254,141,1,49,172,1,49,177,
251,133,254,136,136,177,251,41,7,240,36
5010 DATA 170,136,177,251,141,0,49,136,1
77,251,24,109,0,49,144,2,230,254,202,208
5020 DATA 245,168,177,253,41,7,141,255,4
8,169,2,145,253,76,65,49,136,136,177,251
5030 DATA 76,43,49,172,1,49,177,251,133,
254,136,177,251,141,0,49,136,177,251,41
```

```
5040 DATA 8,208,19,177,251,41,7,205,0,49
,208,20,177,251,24,105,8,145,251,76,114
5050 DATA 49,177,251,41,7,208,4,169,0,14
5,251,177,251,41,8,208,10,177,251,24,105
5060 DATA 1,145,251,76,137,49,177,251,56
,233,1,145,251,177,251,41,7,240,39,170
5070 DATA 136,177,251,141,0,49,136,177,2
51,24,109,0,49,144,2,230,254,202,208
5080 DATA 245,168,173,255,48,145,253,173
,1,49,56,233,5,176,1,96,76
5090 DATA 4,49,136,136,177,251,76,164,49
```

Variations

It should be possible to add sound effects without appreciably slowing down the program. The combination of sliding tones and vibration symbols could prove overpowering!

20
Poker-Face



A melancholy kind of a fellow is Poker-Face, who keeps his thoughts to himself for most of the time. He may occasionally smile in a pensive way. Possibly he is holding a sure-fire poker hand, but probably he is only bluffing. If you think that you can gauge his mood, it could help you to win. If not, you will have to rely on your assessment of the chances of your holding the winning hand. This is a game for one player, based on the rules of five-card poker, with the computer as your poker-faced opponent.

How to play

As soon as you RUN the program, your hand is displayed on a green screen, reminiscent of a green baize card table. You are given 5 cards, and so is Poker-Face, though of course he will not show you his cards until you say 'See you' later in the game.

The cards have been dealt at random from a normal pack of 52 cards with no jokers, and the pack is shuffled between deals. Aces are high. While Poker-Face is analysing his hand, you have time to examine yours. In Poker, there are 10 kinds of scoring hand, which

rank in the order listed below:

Royal straight flush: five consecutive cards of the same suit, from ace down to 10.

Straight flush: five consecutive cards of the same suit, starting with a card other than the ace (e.g. 7,6,5,4,3 of hearts).

Fours: four cards of the same value (e.g. 10 of spades, 10 of hearts, 10 of diamonds, 10 of clubs, plus one other card).

Full House: three cards of the same value, plus a pair of cards of another value (e.g. 4 of spades, 4 of hearts, 4 of clubs, plus a pair of Queens).

Flush: all five cards of the same suit but not of consecutive values (e.g. 3,5,6,J,A of clubs).

Straight: five cards of consecutive value, but not of same suit (e.g. 3 of clubs, 4 of spades, 5 of diamonds, 6 of clubs, 7 of clubs).

Threes: three cards of the same value plus two odd cards.

Two pairs: e.g. a pair of threes and a pair of kings with one odd card.

Pair: e.g. a pair of tens with three odd cards.

Top Card: if a player has none of the hands above, the hand is ranked by the value of its highest card.

By the time you have decided how good your hand is, the computer will have analysed both your hand and that held by Poker-Face. Poker-Face's reactions are then described on the screen. How much you can tell from this is a matter for you to decide. Incidentally, although the computer has analysed your hand too, the part of the program dealing with Poker-Face's reactions does not make use of this information.

You are now asked what you want to do. There are two choices:

'Turn in': press key 'T'. You think that your hand is so worthless by comparison with that held by Poker-Face, that you wish to cut your losses and end the round. If you do this, you do not see what hand Poker-Face actually held. It might have been even worse than yours! Whatever the relative rank of the hands, you lose one chip to Poker-Face. The scores are then displayed at the bottom of the screen. By the way, you each begin the game with 100 chips.

'See You': press key 'S'. The screen then displays Poker-Face's hand, and tells you its ranking. The winner is the player with the higher ranking hand. The score is displayed. If you win, Poker-Face pays you 3 chips. If you lose, you pay him 2 chips.

Press the space bar when you are ready to play the next hand.

Poker players will realize that this program does not attempt to

compare hands that are of the same kind. In this respect a pair of Queens, for example, ranks equally with a pair of twos. In the event of both players holding the same kind of hand, the game is declared a 'null game' and the scores of both players are left unaltered.

Winning tactics

The program is designed so that you cannot reliably guess what hand Poker-Face holds from the reactions he shows. There is a strong random element, which might correspond to a 'change of mood'. There is also an occasional wildly unpredictable response which gives away nothing. Poker-Face can truly bluff!

Playing this game gives you insight into the frequency with which the various hands are obtained. Since the 'deal' is a true simulation of the dealing of a pack of cards, the hands turn up with the same frequencies as would be experienced with a real pack. Apart from setting out the various combinations and working out the probabilities mathematically, the best way to success in poker is to actually play it and get the feel of the game. Gaining experience this way could be less expensive than gaining it at the poker table!

Keying in

Control characters used are:

HOME: line 30
CRSR DOWN: lines 120, 140, 150, 160, 640(2), 720(2), 810(2), 820, 830, 840, 850(7)
CRSR RIGHT: lines 120(2), 640(2), 720(2), 850(2), 880(2), 890(2), 900(2)
CTRL/1: lines 130, 160, 810, 840
CTRL/3: lines 140, 820
HOME: line 810

Unlisted characters used:
C=/2: lines 120 (before CRSR DOWN), 720 (before CRSR DOWN), 910 (before YOU)
C=/4: lines 640 (before CRSR DOWN), 850 (before CRSR DOWN), 910 (before P–F)

The other symbols used in lines 130–160 and 810–840 are the spade,

heart, diamond and club, obtained by typing 'A', 'S', 'Z' or 'X', with the 'SHIFT' key.

Program design

20	Storing initial scores.
30–40	Initialising variables.
50–110	'Dealing' cards.
120–160	Displaying your hand.
170–190	Totalling numbers of cards of same denomination in each hand.
200–220	Totalling numbers of cards of same suit in each hand.
230–360	Analysing hands to pick out various kinds of flush.
370–500	Looking for fours, threes, full house, and pairs.
510–570	Looking for a straight (not flush).
580–630	Finding top card.
640–710	Displaying Poker-Face's reactions.
720–760	Asking for your response.
770–910	Displaying Poker-Face's hand, and scores.
920–930	Preparing for next deal.
1000–1050	Subroutine converting values of cards into 'A', 'K', 'Q', etc. and printing them.
3000	DATA statements holding card denominations.
4000	DATA statements holding Poker-Face's reactions.
5000	DATA statements holding names of hands.

Points of interest

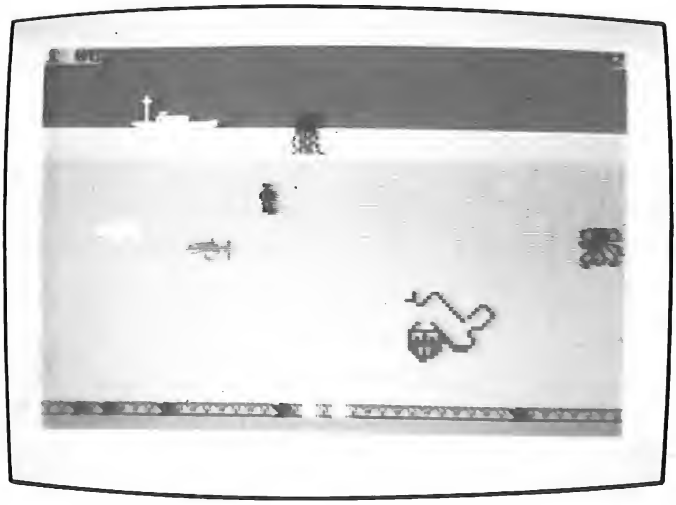
The many arrays and variables are cleared at the end of each hand by using the statement 'CLR' in line 930. To preserve the scores, these are POKEd into two addresses in RAM (251, 252). These addresses are available for use by the programmer and never altered by the computer, so the values may be stored there. When the program begins again, these scores are recovered at line 30.

The program

```
10 REM ** POKER-FACE **
20 POKE 251,100:POKE 252,100
30 PRINT"□";:POKE 53280,13:POKE 53281,13
:PS=PEEK(251):YS=PEEK(252)
40 DIM P%(12,3),H%(12,7),D%(12,1)
50 FOR J=1 TO 10
60 D=INT(RND(1)*13):S=INT(RND(1)*4)
70 IF P%(D,S)=1 THEN 60
80 P%(D,S)=1
90 IF J>5 THEN S=S+4
100 H%(D,S)=1
110 NEXT
120 PRINT"■■■YOUR HAND IS:■■"
130 PRINTTAB(5)"■♠   ";:S=4:GOSUB 1000
140 PRINT"■":PRINTTAB(5)"■♥   ";:S=5:GOSU
B 1000
150 PRINT"■":PRINTTAB(5)"♦   ";:S=6:GOSUB
 1000
160 PRINT"■":PRINTTAB(5)"■♣   ";:S=7:GOSU
B 1000
170 FOR J=0 TO 12:FOR H=0 TO 1:FOR K=0+H
*4 TO 3+H*4
180 IF H%(J,K)=1 THEN D%(J,H)=D%(J,H)+1
190 NEXT:NEXT:NEXT
200 FOR H=0 TO 1:FOR K=H*4 TO 3+H*4:FOR
J=0 TO 12
210 IF H%(J,K)=1 THEN S%(K)=S%(K)+1
220 NEXT:NEXT:NEXT
230 S=0:FOR H=0 TO 1
240 FOR K=H*4 TO 3+H*4
250 IF S%(K)=5 THEN S=K
260 NEXT:IF S=0 THEN 360
270 R=0:FOR J=12 TO 4 STEP -1
280 F=1:FOR L=J TO J-4 STEP -1
290 IF H%(L,S)<>1 THEN F=0
300 NEXT
310 IF F=1 THEN R=J
320 NEXT
330 IF R=122 THEN S(H)=1:GOTO 360
340 IF R>0 AND R<12 THEN S(H)=2:GOTO 360
350 S(H)=5
360 NEXT
370 FOR H=0 TO 1
380 IF S(H)>0 THEN 500
390 FOR J=0 TO 12
400 C%(D%(J,H),H)=1
410 NEXT
420 F=0:FOR J=0 TO 12
430 IF D%(J,H)=2 THEN F=F+1
440 NEXT
450 IF F=2 THEN C%(2,H)=2
460 IF C%(4,H)=1 THEN S(H)=3:GOTO 500
470 IF C%(3,H)=1 AND C%(2,H)=1 THEN S(H)
=4:GOTO 500
480 IF C%(3,H)=1 THEN S(H)=7:GOTO 500
490 IF C%(2,H)>0 THEN S(H)=10-C%(2,H)
500 NEXT
510 FOR H=0 TO 1
520 IF S(H)>0 THEN 570
530 FOR J=0 TO 8
540 F=1:FOR L=J TO J+4:IF D%(L,H)=0 THEN
 F=0
550 NEXT:IF F=1 THEN S(H)=6
560 NEXT
```

```
570 NEXT
580 FOR H=0 TO 1
590 IF S(H)>0 THEN 630
600 F=0:FOR J=12 TO 0 STEP -1
610 IF D%(J,H)=1 AND F=0 THEN S(H)=10+J:
F=1
620 NEXT
630 NEXT
640 PRINT:PRINT"■■■■POKER-FACE IS ";
650 IF RND(1)>.9 THEN M=INT(RND(1)*11):G
OTO 680
660 M=S(0)+INT(RND(1)*3)-2
670 IF M<1 THEN M=1
680 IF M>10 THEN M=10
690 FOR J=0 TO 12:READ D$:NEXT
700 FOR J=1 TO M:READ M$:NEXT
710 PRINTM$
720 PRINT"■■■■WHAT WILL YOU DO?"
730 GET A$:IF A$="" THEN 730
740 IF A$="T" THEN PS=PS+1:YS=YS-1:GOTO
910
750 IF A$="S" THEN 770
760 GOTO 730
770 IF M=10 THEN 790
780 FOR J=M+1 TO 10:READ M$:NEXT
790 M=S(0):IF M>10 THEN M=10
800 FOR J=1 TO M:READ H$:NEXT:RESTORE
810 PRINT"■";TAB(20)"■POKER-FACE HOLDS:■
":PRINT:PRINTTAB(22)"■♠   ";:S=0:GOSUB 10
00
820 PRINT:PRINTTAB(22)"■■♥  ";:S=1:GOSUB
 1000
830 PRINT:PRINTTAB(22)"■♦   ";:S=2:GOSUB
1000
840 PRINT:PRINTTAB(22)"■■♣   ";:S=3:GOSUB
 1000
850 PRINT"■■■■■■■■■":PRINT"■■POKER-FACE HA
S ";
860 PRINTH$
870 IF S(0)>9 AND S(1)>9 THEN S(0)=-S(0)
:S(1)=-S(1)
880 IF S(0)<S(1) THEN PRINT"■■AND WINS":
PS=PS+2:YS=YS-2:GOTO 910
890 IF S(0)>S(1) THEN PRINT"■■AND LOSES"
:PS=PS-2:YS=YS+3:GOTO 910
900 PRINT"■■SO IT'S A NULL HAND"
910 PRINT:PRINTTAB(5)"YOU:  ";YS;TAB(25)"
P-F:  ";PS
920 GET A$:IF A$<>" " THEN 920
930 POKE 251,PS:POKE 252,YS:CLR:GOTO 30

1000 FOR J=12 TO 0 STEP -1
1010 READ D$
1020 IF H%(J,S)<>0 THEN PRINT D$;" ";
1030 NEXT
1040 RESTORE
1050 RETURN
3000 DATA A,K,Q,J,10,9,8,7,6,5,4,3,2
4000 DATA LAUGHING HIS HEAD OFF,GASPING
FOR BREATH,SMILING SERENELY, INSCRUTABLE
4010 DATA OBVIOUSLY PERPLEXED,WATCHING Y
OU,GRIPPING THE TABLE
4020 DATA LOOKING GRIM,TURNING PALE,GASP
ING FOR BREATH
5000 DATA A ROYAL STRAIGHT FLUSH,A STRAI
GHT FLUSH,FOURS,A FULL HOUSE
5010 DATA A FLUSH HAND,A STRAIGHT HAND,T
HREES,TWO PAIRS,A PAIR,HIS TOP CARD
```

21
Snorkel



The shallow waters of the tropical bay of Santa Sesenta-y-Cuatro are a pearl-diver's dream. The bottom of the bay is almost entirely covered with pearl oysters, their pinkish shells shimmering in the sub-aquatic sunlight. But beware, for these tempting waters are hazardous! There are the giant clams, ready to snap shut on the outstretched hand of an unwary diver. The scorpion fish have sharp spines which inflict a painful wound. The Portuguese Man-of-War can paralyse a swimmer who is unlucky enough to come into contact with its dangling tentacles. The octopus darts upward then drifts down, its suckers inflicting painful injuries. Finally, there is the Sea Serpent which appears when least expected, with devastating results should the swimmer happen to be near at the time. As seen from the above, this program makes maximum use of the 64's sprite graphics.

How to play

When you first RUN the program, the screen clears to white. The computer is setting up the display. After about 20 seconds, the screen displays a view of part of the bay, with you in your white

cruiser on the water surface. Also on the surface is a Portuguese Man-of-War, with tentacles trailing in the water. At the bottom of the screen you see the oysters, with pinkish shells. Among them are five giant clams. Two scorpion fish swim continually from right to left at different depths and at different speeds. The huge Sea Serpent is there too, ready to catch you.

Figures at the top left corner of the screen display the time, in minutes and seconds. You have 5 minutes in which to collect as many oysters as you can. There are only two keys to press:

; swim to the surface
/ dive

To start, press '/'. The diver jumps into the water and swims down. There is no need to hold a key down or to press it repeatedly. Once you have pressed '/', you continue to dive until you press ';'. Then you rapidly twist around and head for the surface. Sideways motion comes from the tide, which occasionally carries you toward the right. There is wrap-around on the screen both for the fish and diver so, if you drift off screen at the right you reappear on the left. This is the only way of reaching the oysters which are directly beneath the boat.

If you dive down and touch an oyster, the program automatically picks up the oyster, makes you turn round, swim to the surface and jump into the boat. At the same time, the figure at the top right of screen increases by one, keeping count of how many oysters you have collected. The keys have no effect while this is happening. Note that there is no drift on the way up, so you may head directly into the tentacles of a Man-of-War! It inflicts such serious injury that the game ends. While at the sea floor you may accidentally touch a clam instead of an oyster. If you do, the clam snaps shut, gripping your hand. The keys are out of action while you are struggling to escape. After about 6 seconds you manage to release yourself and immediately return to the boat to tend your wounds. All this takes time which could be better spent in collecting oysters. The scorpion fish are another hindrance, for contact with them makes it necessary to go back to the boat to remove their painful spines from your skin. If you dive down on to a scorpion fish, you immediately turn and swim back to the boat. This too is automatic. Once there, you have to wait for 10 seconds (from the time of contact) before being able to dive again. The border of the screen turns grey at this time (as it does when you are held by a clam) to indicate that you cannot move. When the border goes white, the keys are active once again. After the

game has been in progress for 20–30 seconds, the octopus appears from the right. It drifts slowly down, then darts upwards. Contact with the octopus is very serious; you are forced to surface and to wait in the cruiser for 30 seconds. The Sea Serpent makes occasional suprise visits, and may appear almost anywhere. If you are caught by the Serpent, the game ends. A final score of 4 oysters is good for a beginner, and it would take skilful swimming and a certain amount of luck with the tides to score more than 10.

Winning tactics

Diving as soon as you can is obviously essential, but it is a waste of time to dive straight onto the fish or octopus for you are forced to return to the boat and wait. Try to time your dive so as to go between the fish. Ideally, you should avoid them on the way up, too. By alternately pressing the two keys you can hover in the water waiting for the fish to pass. Be wary when near the sea bed, for a sudden tidal flow could carry you straight on to a clam.

To gain a large score, you will have to drift across the screen until you have reappeared at the left, under the boat. There is greater danger from clams here.

The main point to consider is whether it is better to avoid the fish and octopus or to ignore them. To avoid them, by swimming up and down until they pass, takes extra time. To ignore them adds the risk of making contact and so having to waste time in the boat.

Keying in

Two of the lines (360 and 380) have so much crammed in to them that they cannot be typed in as listed. Use the Commodore abbreviation for the word 'POKE'. Instead of typing the whole word, type 'P' followed by shifted 'O' (press 'SHIFT' and key 'O' at the same time). This requires only 2 characters instead of 4. For 'THEN' type 'T' and shifted 'H'. For 'AND' type 'A' and shifted 'N'. For 'PEEK' type 'P' and shifted 'E'.

Control characters used are:

CLEAR: lines 20, 940, 1030
CTRL/3: line 940
CRSR DOWN: lines 950(2), 1000(3)

CRSR RIGHT: lines 950(2), 960(2), 970(2), 980(2), 990(2), 1000(2)

Program design

20–100	Transferring sprite definitions and special character definitions to RAM.
120	Initializing variables.
130–280	Setting up the display.
290	Starting the clock and turning on the screen.
300–310	Getting input and setting various flag variables.
320–420	Moving the diver.
430–470	Checking for collisions with other sprites, and acting on the results.
480–510	Checking for contact with clams and oysters and acting on the results.
520–540	Detecting 'time up' and updating time display.
550–560	Displaying number of oysters collected.
570	Maximum oysters collected.
580–610	Move octopus.
620–640	Check for contact between diver and octopus.
650–690	Display or clear serpent.
700–720	Check for contact with diver.
730–750	Move Man-of-War.
760–780	Check for contact with diver.
790–810	Move fish 1.
820–840	Check for contact with diver.
850–870	Move fish 2.
890–900	Check for contact with diver.
910	Repeat loop.
920–1000	Displaying final messages.
1010–1060	Preparing for replay.
1200–1200	Subroutine for diving from cruiser.
1500–1500	Subroutine for reading collisions register.
2000–2190	DATA statements holding details of sprites.
2200–2210	DATA statements holding details of special characters.

Points of interest

The display uses all eight sprites and several special characters. The

sprites are: (0) cruiser, (1) octopus, (2) serpent, (3) man-of-war, (4) and (5) fish, (6) diver swimming down, (7) diver swimming up. The cruiser is expanded in the X direction to give it sleeker lines. The serpent is expanded in both directions to make it more fearsome. The last POKE statement in line 20 turns off the video display until the scene is ready. Then the POKE in line 290 turns it on again.

Several frequently-used contacts are assigned to variables (line 120) to help speed the main program loop. These include 'Z' for zero, 'U' for unity ('1'), and SY for '63'.

Contact between the diver and the sea creatures is detected by using the VICII chip's sprite-to-sprite collision register. This tells us which sprites are in collision with other sprites. Unfortunately, if two or more pairs are in collision (e.g. boat with man-of-war and diver with octopus), simply PEEKing the register does not tell which sprite is colliding with the diver. It is essential to check the register after every movement of every sprite – hence all the 'GOSUB 1500's – and find out if the relevant bits have changed since the last PEEK. For this purpose we use PC, the value of the lower 6 bits (i.e. non-diver sprites) at the previous check, as in line 1500. If there is a change, the 'change' flag, CF is set. CC indicates which sprite has collided with the diver.

The program

```
10 REM ** SNORKEL **
20 PRINT"□":POKE 53280,1:POKE 53281,3:PO
KE 53265,PEEK(53265)AND239
30 POKE 56334,PEEK(56334)AND254:POKE 1,P
EEK(1)AND251
40 FOR J=0 TO 79:X=PEEK(53632+J):POKE 14
720+J,X:NEXT
50 POKE 1,PEEK(1)OR4:POKE 56334,PEEK(563
34)OR1
60 POKE 52,48:POKE 56,48:POKE 53272,(PEE
K(53272)AND240)+14
70 POKE 649,1:POKE 650,128
80 FOR J=0 TO 7:POKE 14368+J,255:NEXT
90 FOR J=0 TO 7:POKE 14376+J,0:NEXT
100 FOR J=12288 TO 12734:READ X:POKE J,X
:NEXT
110 FOR J=14334 TO 14365:READ X:POKE J,X
:NEXT
120 G=1024:H=55296:V=53248:VC=V+30:VS=V+
21:VX=V+16:U=1:Z=0:GB=G+170:SY=63:BY=256
130 FOR J=0 TO 199:POKE G+J,4:POKE H+J,6
:NEXT
140 FOR J=200 TO 919:POKE G+J,5:NEXT
150 FOR J=920 TO 959:POKE G+J,3:POKE H+J
,10:NEXT
160 FOR J=960 TO 999:POKE G+J,4:POKE H+J
,7:NEXT
```

```
170 FOR J=1 TO 5:POKE G+920+2↑J,1:POKE H
+920+2↑J,0:NEXT
180 FOR J=0 TO 5:POKE 2040+J,192+J:NEXT
190 FOR J=0 TO 2:POKE 2045+J,196+J:NEXT
200 POKE V+39,1:POKE V+40,0:POKE V+41,6:
POKE V+42,2:POKE V+43,1:POKE V+44,4
210 POKE V+45,0:POKE V+46,0
220 POKE V,72:POKE V+1,74:POKE V+29,5:PO
KE GB,0
230 XP=255:ZP=2:YP=RND(1)*60+106:POKE V+
2,XP:POKE V+3,YP
240 POKE V+4,RND(1)*232+24:POKE V+5,RND(
1)*80+126:POKE V+23,4:SF=600
250 FM=255:FW=-1:POKE V+6,FM:POKE V+7,82
260 F1=344:POKE V+8,88:POKE V+9,RND(1)*4
5+122
270 F2=344:POKE V+10,88:POKE V+11,RND(1)
*45+122
280 POKE VX,50:POKE VS,63
290 TI$="000000":POKE 53265,PEEK(53265)O
R16
300 GOSUB 1500:B=PEEK(V+31):IF TI<TP THE
N POKE 53280,15:GOTO 330
310 TP=Z:POKE 53280,U
320 GET A$:IF A$<>"" THEN A=ASC(A$)
330 IF D<90 THEN FS=Z:FC=Z
340 IF FS=U THEN A$=";":A=59
350 IF A=47 AND D=Z THEN GOSUB 1200
360 IFA=47ANDD>ZANDD<222THEND=D+8:POKEV+
13,D:POKEV+15,D:POKEVS,(PEEK(VS)ANDSY)+6
4
370 IFD<94ANDD<>ZTHEND=Z:POKEVS,PEEK(VS)
AND127:POKEGB,Z:POKEVX,PEEK(VX)ANDSY
380 IFA=59ANDD>93ANDD<230THEND=D-8:POKEV
+13,D:POKEV+15,D:POKEVS,(PEEK(VS)ANDSY)+
128
390 IF A=59 OR D=Z OR RND(U)<.6 OR TP>Z
THEN 430
400 X=X+8:IF X>336 THEN X=24
410 IFX>255THENPOKEVX,(PEEK(VX)ANDSY)+19
2:POKEV+12,X-BY:POKEV+14,X-BY:GOTO430
420 POKE VX,PEEK(VX)ANDSY:POKE V+12,X:PO
KE V+14,X
430 GOSUB 1500
440 IF CF=Z THEN 480
450 IF CC=32 OR CC=16 THEN FS=U:TP=TI+60
0
460 IF CC=8 OR CC=4 THEN 920
470 IF CC=2 THEN FS=U:TP=TI+1800
480 IF D<222 THEN 520
490 BG=G+917+X/8
500 IF PEEK(BG)=U THEN POKE BG,2:FC=U:TP
=TI+600
510 IF PEEK(BG)=3 THEN POKE BG+H-G,3:POK
E BG,4:FS=U:P=P+U
520 IF TI>18000 THEN FT=U:GOTO 920
530 POKE G,ASC(MID$(TI$,4,U)):POKE G+U,5
540 POKE G+2,ASC(MID$(TI$,5,U)):POKE G+3
,ASC(RIGHT$(TI$,U))
550 P$=STR$(P):POKE G+39,ASC(RIGHT$(P$,U
))
560 IF LEN(P$)=2 THEN POKE G+38,ASC(LEFT
$(P$,1))
570 IF P=35 THEN 920
580 XP=XP-2:IF XP<Z AND ZP=2 THEN XP=255
:ZP=Z
590 IF XP<Z AND ZP=Z THEN XP=255:ZP=2
```

```
600 YP=YP+2:IF YP>180 THEN YP=126
610 POKE V+2,XP:POKE VX,(PEEK(VX)AND253)
+ZP:POKE V+3,YP
620 GOSUB 1500
630 IF CF=Z THEN 650
640 IF CC=2 THEN FS=U:TP=TI+1800
650 IF TI<SF THEN 700
660 IF TI>SF AND TI<SF+2400 THEN POKE VS
,PEEK(VS)AND251:GOTO 730
670 IF TI<1800+SF THEN 730
680 POKE V+4,RND(U)*232+24:POKE V+5,RND(
U)*80+126:POKE VS,(PEEK(VS)AND251)+4
690 SF=800+TI
700 GOSUB 1500
710 IF CF=Z THEN 730
720 IF CC=4 THEN 920
730 FM=FM+FW:IF FM=-U THEN FM=Z:FW=U
740 IF FM>255 THEN FM=255:FW=-U
750 POKE V+6,FM
760 GOSUB 1500
770 IF CF=Z THEN 790
780 IF CC=8 THEN 920
790 F1=F1-10:IF F1<10 THEN F1=344
800 IF F1>255 THEN POKE VX,(PEEK(VX)AND2
39)+16:POKE V+8,F1-BY:GOTO 820
810 POKE VX,(PEEK(VX)AND239):POKE V+8,F1
820 GOSUB 1500
830 IF CF=Z THEN 850
840 IF CC=16 THEN FS=U:TP=TI+600
850 F2=F2-20:IF F2<10 THEN F2=344
860 IF F2>255 THEN POKE VX,(PEEK(VX)AND2
23)+32:POKE V+10,F2-BY:GOTO 880
870 POKE VX,(PEEK(VX)AND223):POKE V+10,F
2
880 GOSUB 1500
890 IF CF=Z THEN 910
900 IF CC=32 THEN FS=U:TP=TI+600
910 GOTO 300
920 POKE 53280,2
930 FOR J=1 TO 4000:NEXT
940 PRINT"■":POKE VS,0:POKE 53272,21
950 IF P>10 THEN PRINT"■■■■A GOOD DAY'S
WORK!"
960 IF CC=8 THEN PRINT"■■THE MAN-OF-WAR
STUNG YOU"
970 IF CC=4 THEN PRINT"■■THE SEA-SERPENT
 CAUGHT YOU"
980 IF FC=1 THEN PRINT"■■BADLY INJURED B
Y A CLAM"
990 IF FT=1 THEN PRINT"■■TIME TO REST"
1000 PRINT"■■■■YOU COLLECTED";P;"PEARL
OYSTERS"
1010 GET A$:IF A$<>" " THEN 1010
1020 TP=0:A=0:FS=0:FT=0:P=0:FC=0:D=0
1030 PRINT"■":POKE 53280,1:POKE 53281,3
1040 POKE 53272,(PEEK(53272)AND240)+14
1050 POKE 53265,PEEK(53265)AND235
1060 GOTO 130
1200 D=94:POKE GB,4:X=128:POKE V+12,X:PO
KE V+14,X
1210 POKE V+13,D:POKE V+15,D:POKE VS,PEE
K(VS)+64
1220 RETURN
1500 C=PEEK(VC):CC=(C+PC)ANDSY:PC=CANDSY
1510 CF=Z:IF C>SY THEN CF=U
1520 RETURN
```

```
2000 DATA 8,0,0,8,0,0,28,0,0,8,0,0,8,0,0
,8,0,0,8,0,0,8,0,0
2010 DATA 8,192,0,8,128,0,9,255,0,9,85,0
,9,255,3,255,255,255,127,255,255
2020 DATA 63,255,254,31,255,253,15,255,2
53,0,0,0,0,0,0,0,0,0,0
2030 DATA 4,124,48,56,254,8,65,247,4,65,
242,2,65,255,2,33,255,4,31,254,248
2040 DATA 195,85,0,133,82,240,74,81,12,5
0,144,130,4,136,66,4,136,34
2050 DATA 25,4,17,33,4,8,65,2,8,66,66,8,
34,132,34,17,40,72,72,32,48,48,0,0,0
2060 DATA 0,0,0,33,128,0,34,64,0,228,32,
0,24,16,12,0,8,18,0,4,33,0,2,65
2070 DATA 0,1,2,17,0,4,32,128,56,127,192
,64,68,96,64,85,112,32,127,248,32
2080 DATA 127,207,192,117,199,128,53,128
,0,31,0,0,12,0,0,0,0,0,0
2090 DATA 0,63,0,0,255,0,1,215,0,3,215,0
,3,215,0,3,214,0,3,214,0,7,214,0
2100 DATA 7,255,0,7,255,0,5,37,0,5,37,0,
8,168,128,8,168,128,17,69,0
2110 DATA 17,69,0,8,168,128,8,168,128,5,
20,64,5,20,64,24,168,48,0
2120 DATA 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
0,2,0,0,6,1,0,14,3,63,254,7,103,255,135
2130 DATA 247,255,254,255,255,254,3,255,
206,63,64,199,0,96,135,0,112,7
2140 DATA 0,120,3,0,28,1,0,0,0,0,0,0,0,0
,0,0
2150 DATA 48,0,0,48,0,0,48,0,0,208,0,0,2
08,0,0,208,0,0,80,0,0,120,0,0,24,0,0,24
2160 DATA 0,0,63,0,0,41,0,0,45,0,0,45,0,
0,33,0,0,16,0,0,0,0,0,0,0,0,0,0,0,0,0,0
2170 DATA 0,0,0,0,0,48,0,0,48,0,0,16,0,0,5
6,0,0,92,0,0,154,0,0,154,0,0,154,0,0
2180 DATA 90,0,0,40,0,0,68,0,0,36,0,0,52
,0,0,54,0,0,54,0,0,6,0,0,0,0,0,0,0,0,0
2190 DATA 0,0,0,0,0,0,0,0,0,0
2200 DATA 199,199,231,199,131,255,255,16
,24,12,198,99,51,31,15
2210 DATA 56,124,62,63,31,31,15,15,24,60
,189,219,102,60,60,24
```

Variations

Alter the length of the game by changing the value in line 520 to the required length of time (in seconds) multiplied by 60. There is plenty of scope for programming sound effects, such as the splash of the diver entering the water, the gentle puttering of the engines of the cruiser – not to mention the unimaginable sounds of the sea monsters!

The TI 99/4A

GET MORE FROM THE TI99/4A
Garry Marshall
0 246 12281 1

The VIC 20

GET MORE FROM THE VIC 20
Owen Bishop
0 246 12148 3

THE VIC 20 GAMES BOOK
Owen Bishop
0 246 12187 4

The ZX Spectrum

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

INTRODUCING SPECTRUM MACHINE CODE
Ian Sinclair
0 246 12082 7

SPECTRUM GRAPHICS AND SOUND
Steve Money
0 246 12192 0

THE ZX SPECTRUM How to Use and Program
Ian Sinclair
0 586 06104 5

The ZX81

THE ZX81 How to Use and Program
S. M. Gee and Mike James
0 586 06105 3

Which Computer?

CHOOSING A MICROCOMPUTER
Francis Samish
0 246 12029 0

Languages

COMPUTER LANGUAGES AND THEIR USES
Garry Marshall
0 246 12022 3

EXPLORING FORTH
Owen Bishop
0 246 12188 2

Machine Code

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

6502 MACHINE CODE FOR HUMANS
Alan Tootill and David Barrow
0 246 12076 2

Using Your Micro

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

DATABASES FOR FUN AND PROFIT
Nigel Freestone
0 246 12032 0

SIMPLE INTERFACING PROJECTS
Owen Bishop
0 246 12026 6

INSIDE YOUR COMPUTER
Ian Sinclair
0 246 12235 8

Programming

THE COMPLETE PROGRAMMER
Mike James
0 246 12015 0

PROGRAMMING WITH GRAPHICS
Garry Marshall
0 246 12021 5

HAVE FUN WITH THE 64!

Now you have a machine with such remarkable graphics and sound capabilities you will want to try some games which make full use of them. Here are twenty-one exciting, high quality games which have been fully tested and crash-proofed.

Snorkel, for example, is a scintillating display of the Commodore's 'sprite' graphics, while Pin-Table takes advantage of its immense range of colours and sounds. Even the very youngest of the family will enjoy the race to build the Sandcastle before the tide washes it away, while the fast-moving Black Hole and Bombing Run will appeal to all ages. If you like zapping the enemy and gang shoot-outs, you will love the excitement of games like Sniper, Minefield and Cops'n'Robbers.

Each program comes with instructions showing you how to play the game. The way the game works is explained in detail, with tips on winning tactics. Hints are also given on how to change the games creatively to suit your own ideas.

The Author

Owen Bishop is a freelance technical writer and programmer. He is the author of over thirty books including a number on popular computing. He is a well-known and regular contributor to computing journals.

GRANADA PUBLISHING
Printed in Great Britain 0 246 12258 7

£5.95 net