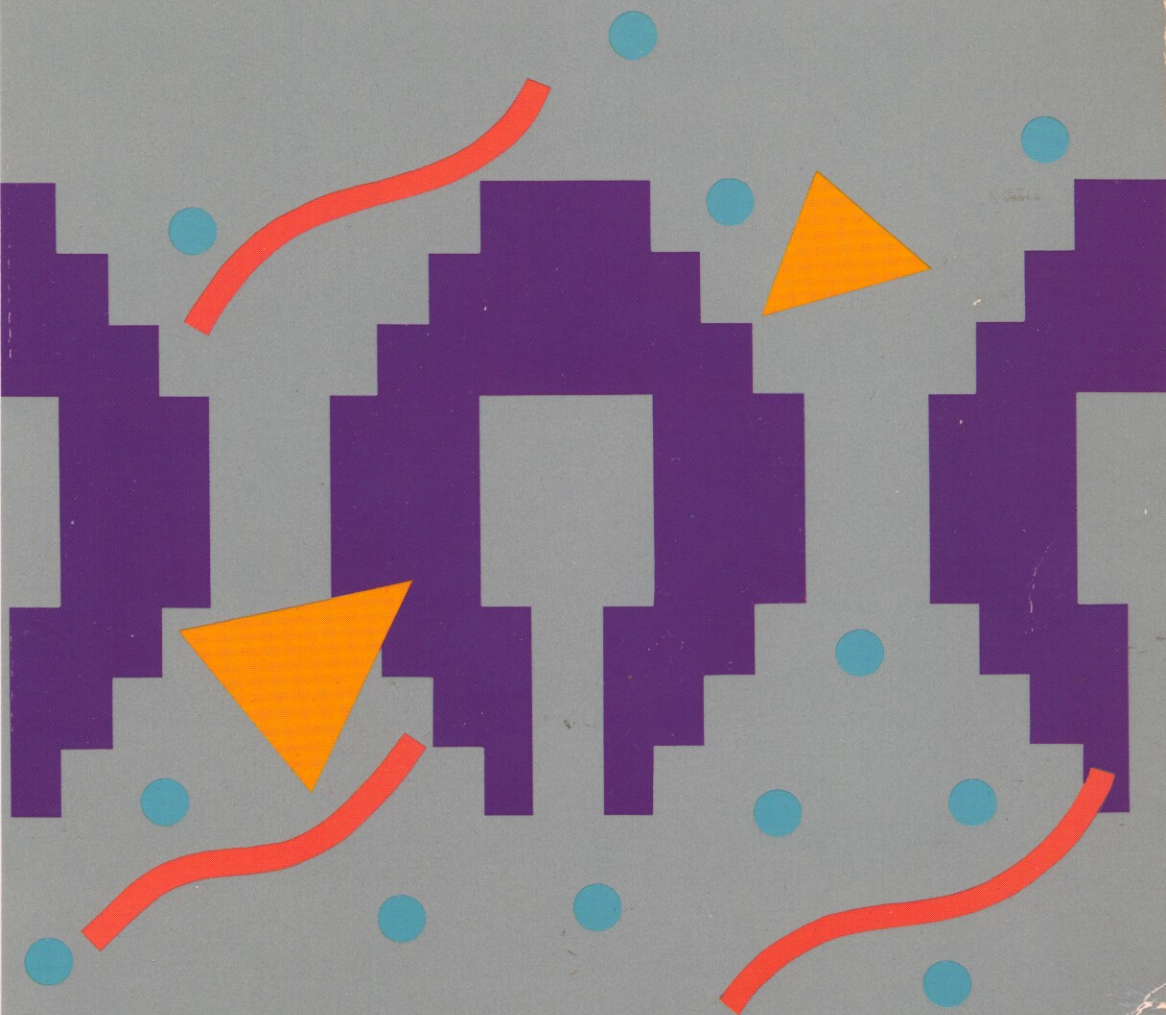
Osborne McGraw-Hill

THE ATARI® BOOK OF GAMES



• Mike James • S. M. Gee • Kay Ewbank

The Atari Book of Games

Mike James, S. M. Gee and Kay Ewbank

Osborne McGraw-Hill
Berkeley, California

Granada Technical Books
Granada Publishing Ltd
8 Grafton Street, London W1X 3LA

British Library Cataloguing in Publication Data
James, Mike
The Atari book of games.
1. Electronic games.
2. Atari (Computer)—Programming
I. Title II. Gee, S.M.
III. Ewbank, K.
794 GV1469.2

1234567890 DODO 8987654

ISBN 0-88134-159-2

First published in Great Britain 1983 by Granada Publishing
Reprinted 1983

First published in the United States by Osborne/McGraw-Hill, 1985.

Typeset by V & M Graphics Ltd., Aylesbury, Bucks

Contents

Introduction

Here is a collection of twenty-one games for you to type into your Atari. They will all run on all Atari models including the Atari 600XL. However, if you have a 16K Atari 400 with disk drives you will need to disconnect the disk in order to have sufficient memory to enter and run the four largest programs in this book – Capture the Quark, Laser Attack, Treasure Island and Smalltalker – and you will need to save these particular programs on cassette tape.

The games are written in BASIC and presented in this form they serve a dual purpose. Typing in games for yourself is a good way to *absorb* BASIC. If you are a beginner you will soon become familiar with its syntax and structure and, if you already have some experience, you will quickly pick up some handy techniques that you can incorporate into your own programs. Atari BASIC is not an easy version of BASIC but, as this book demonstrates, it is possible to do just as much with it as in any other version.

Once you've typed in a game you, and your friends and family, can enjoy playing with it. We've tried to include something for everyone and each one has its own detailed description so that you'll know what to expect before you embark on it. You also have a chance to *see* what to expect as there are samples of the displays produced on your TV screen. Of course these cannot really do justice to many of the programs which use colour graphics – and we cannot find any way of letting you hear the accompanying sound effects.

What's to follow

It's really impossible to indicate the range of programs included in this book as they do not fall into neat categories. Of the twenty-one programs about two-thirds can be described as moving graphics games. Some of these are variations on familiar favourites, for

example Invaders, Attack Squash and Bobsleigh. Others have titles that probably won't ring any bells – Sheepdog Trials, Commando Jump and Across the Ravine – but we hope they will soon become popular once you start to play them. Laser Attack and Mighty Missile are both 'zap-the-enemy' type games with special features that make them very different from others we've played. Treasure Island is another program that is out of the ordinary. It is a game that tests your memory and relies on a variety of interesting graphics techniques. Capture the Quark is a board game in which you play against the computer on an eight-by-eight grid. There are also some programs for traditional pastimes – Magic Dice, Noughts and Crosses, Word Scramble and Mirror Tile all come into this category. Smalltalker is a rather unusual program that enables your Atari to hold a *conversation* with you. If you think we are joking you'll have to try it for yourself.

Improve your programming

Each program is accompanied by an outline of its subroutine structure, details of special programming techniques and suggestions for further improvements. These sections are included for those of you who want to develop your own programming skills. By giving away some of our *trade secrets* we hope that you'll be able to extend your range of techniques.

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

Listing conventions

The programs included have all been extensively tested in the form in which they appear. As a deliberate policy we decided not to renumber the programs once they were fully working as renumbering could have introduced new errors. The programs have been listed in an easy-to-read form, with lines split, where necessary, at points that preserved their syntax and blank lines inserted between subroutines.

All characters that need to be entered in graphics mode have been enclosed in square brackets. So, if you need to type the letter 'R' in graphics mode you will see [R] in the program listing. You'll find this explanation repeated wherever it applies in the 'Typing Tips' section. Although zeros appear without an oblique stroke through them this is unlikely to cause any confusion as we've avoided using capital O. When you do need to type an upper-case letter 'O' it will be pointed out in the 'Typing Tips' section.

Perfect programming

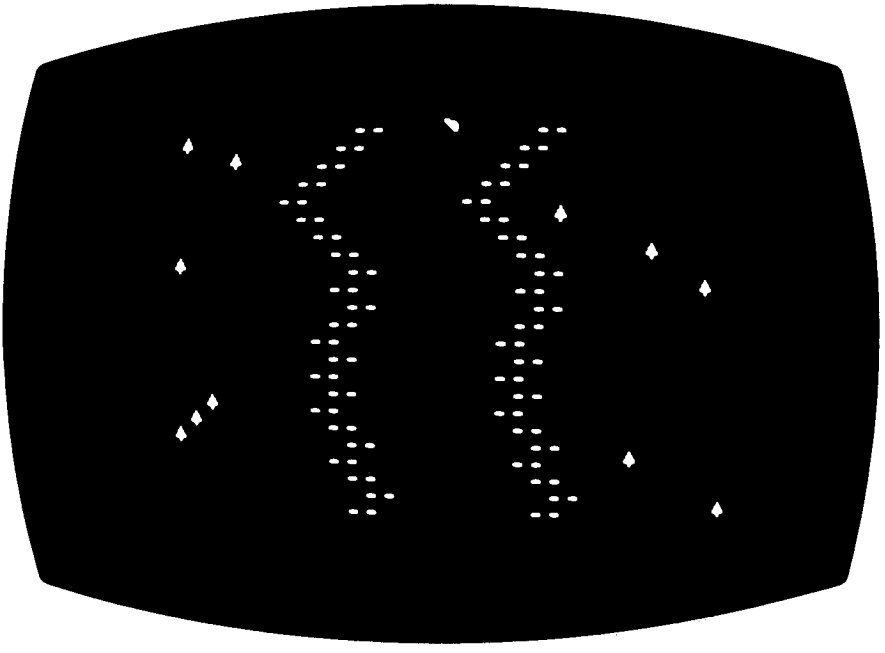
All these programs have been printed out directly from working versions. This means that if you type in exactly what is printed in this book every program should work for you, every time you run it. However, it's a well-known fact that bugs creep in whenever you enter a program – so if a program won't work when you've typed it in check it very carefully against the listing and if it still won't work have a cup of tea and check again – it's all too easy to read what you think should be there rather than what is there! Particular points to look out for are null strings and rows of blank spaces – have you typed the correct number of them. To help you we've included REM statements that tell you how many spaces are needed when its difficult to see by eye. Common sources of errors are confusing full stops and commas or semi-colons and colons, omitting brackets (or putting in extra ones) or mis-reading less than and greater than symbols – getting these round the wrong way will lead to chaotic results. Also, don't be tempted to change the line numbering when you type in the programs as there are far too many pitfalls involved! If there are any special points to look out for when entering a program you'll be alerted to them in the section on 'Typing Tips'. If you do get an error message when you try to RUN a program, don't just give up but use the information it gives you to trace your mistake. The error will not necessarily be in the line whose number is reported but that line will try to use some part of the program with the bug in it. If the line uses a variable, or an array, check to see if it was defined properly. If the line identified goes to another part of the program or calls a subroutine make sure that section is complete.

Cassette Tapes

Typing in long programs can be a very frustrating business. It's not easy to avoid typing mistakes altogether and there is always the risk that you'll lose your program before you've saved it after hours of careful effort – it's far too easy to disconnect your power supply and you can't guard against thunderstorms or other sources of voltage fluctuations without great expense! Many of the programs in this book are indeed long. This is unavoidable as the games concerned include lots of features. But you can avoid having to type them all in for yourselves. The programs, exactly as listed here, are available on a pair of cassette tapes. For full details and an order form send a stamped, self-addressed envelope to:

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

1
Bobsleigh



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

How to play

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

with 1 producing the widest, and therefore easiest, and 5 the narrowest, and most difficult.

Typing tips

You'll find the Atari's editing facilities very useful in entering the blocks of very similar lines in subroutines 1000, 1100 and 1200.

Subroutine structure

20	Sets up graphics characters and disables cursor
150	Prints first part of track
260	Main play loop
380	Scrolls last part of track off
440	Win/lose messages
480	End of game
560	Title frame
670	Moves bobsleigh
770	Prints fir trees
1000	Defines graphics character for bobsleigh
1100	Defines graphics character for fence posts
1200	Defines graphics character for tree
1280	Sets colours
2000	Sound routine

Programming details

The shape of the bobsleigh, course sides and trees are all produced by user-defined characters. Although Atari BASIC doesn't provide any special commands to help define new characters it is not difficult to create your own shapes. First the definition of the entire character set has to be transferred from ROM, where it normally resides, to RAM so that some of the definitions can be modified. In Bobsleigh this transfer is carried out by lines 50 to 90. After this we can change any characters shape by POKEing the memory locations that define it. Bobsleigh uses three redefined characters, the $ which is redefined as the bobsleigh itself in lines 1000 to 1070, the & which is redefined as the course edge marker in lines 1100 to 1170 and the % which is redefined as a tree shape in lines 1200 to 1270. Finally, after moving

the character definition from ROM to RAM and defining the shapes that we want to use, all that is left is to tell the Atari to use the new character definitions in RAM rather than the originals still present in ROM. This is done by line 110.

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

Scope for improvement

You might like to add a tune-playing routine to this program like the one given in 'Atari Ledger'.

Program

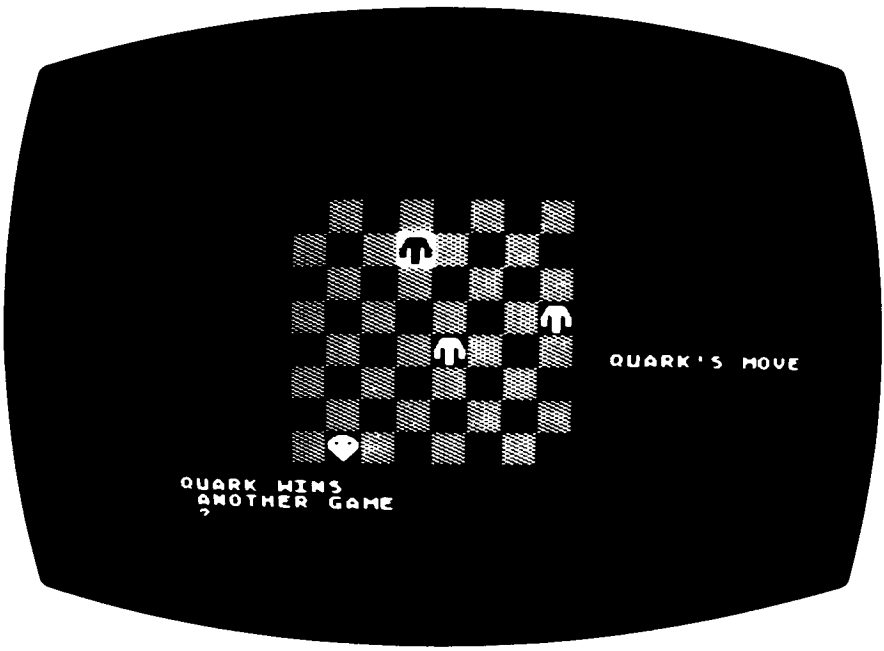
```
 10 REM BOBSLEIGH

 20 GRAPHICS 0
 30 GOSUB 560
 40 DIM A$(1)
 50 CH=(PEEK(106)-8)*256
 60 CHORG=(PEEK(756)*256)
 70 FOR I=0 TO 511
 80 POKE CH+I,PEEK(CHORG+I)
 90 NEXT I
100 GOSUB 1000
110 POKE 756,CH/256
120 POKE 752,1
```

```
150 YB=0
160 X=INT(RND(0)*10)+5
170 XB=INT(X+D/2)
175 PRINT CHR$(125)
180 FOR Y=1 TO 23
190 POSITION X,23:PRINT "&&";:
    POSITION X+D,23:PRINT "&&"
200 GOSUB 770
210 X=X+SGN(RND(0)-0.5)
220 IF X>29 THEN X=29
230 IF X<1 THEN X=1
240 NEXT Y
250 POSITION XB,YB:PRINT "$";

260 FOR Z=1 TO 500:NEXT Z
270 F=0
280 FOR Z=1 TO 300
290 X=X+SGN(RND(0)-0.5)
300 IF X>29 THEN X=29
310 IF X<1 THEN X=1
320 POSITION X,23:PRINT "&&";:
    POSITION X+D,23:PRINT "&&"
330 GOSUB 770
340 GOSUB 670
350 IF F=1 THEN GOTO 460
370 NEXT Z

380 FOR Z=1 TO 30
390 POSITION 19,23:PRINT
400 GOSUB 670
410 IF F=1 THEN GOTO 460
430 NEXT Z

440 POSITION 1,19:PRINT "CONGRATULATIONS
    YOU MADE IT"
450 GOTO 480
460 POSITION 1,19:PRINT "YOU CRASHED"
470 GOSUB 2000

480 POSITION 1,20:PRINT "ANOTHER GAME (Y/N)";
520 INPUT A$
530 IF A$="Y" THEN RUN
540 GRAPHICS 0
550 STOP
```

```
560 PRINT CHR$(125)
580 PRINT "        B O B S L E I G H"
590 POSITION 2,8:PRINT "  You must steer
    your bobsleigh"
595 PRINT

605 PRINT
610 PRINT " the left and right arrow keys."
620 POSITION 2,20:PRINT "Select the
    difficulty level -"
630 PRINT "from 1 (easy) to 5 (difficult) ";
    :INPUT D
640 IF D<1 OR D>5 THEN GOTO 630
650 D=11-D
660 RETURN

670 REM BOB
680 A=PEEK(764)
685 POKE 764,255
710 IF A=6 THEN XB=XB-1
720 IF A=7 THEN XB=XB+1
730 LOCATE XB,YB,Q:IF Q<>32 THEN F=1
740 POSITION XB,YB:PRINT "$"
750 RETURN

770 REM TREE
780 IF RND(0)<0.4 THEN GOTO 860
790 K=SGN(RND(0)-0.5)
800 XT=X+K
810 XT=INT(X+D/2+(K*(RND(0)*5+D)))
820 IF XT<0 OR XT>38 THEN RETURN
840 POSITION XT,23:PRINT "%";
860 RETURN

1000 POKE CH+(ASC("$")-32)*8+0,32
1010 POKE CH+(ASC("$")-32)*8+1,40
1020 POKE CH+(ASC("$")-32)*8+2,232
1030 POKE CH+(ASC("$")-32)*8+3,252
1040 POKE CH+(ASC("$")-32)*8+4,126
1050 POKE CH+(ASC("$")-32)*8+5,62
1060 POKE CH+(ASC("$")-32)*8+6,30
1070 POKE CH+(ASC("$")-32)*8+7,14
```

```
1100 POKE CH+(ASC("&")-32)*8+0,16
1110 POKE CH+(ASC("&")-32)*8+1,16
1120 POKE CH+(ASC("&")-32)*8+2,16
1130 POKE CH+(ASC("&")-32)*8+3,16
1140 POKE CH+(ASC("&")-32)*8+4,16
1150 POKE CH+(ASC("&")-32)*8+5,16
1160 POKE CH+(ASC("&")-32)*8+6,124
1170 POKE CH+(ASC("&")-32)*8+7,124

1200 POKE CH+(ASC("%")-32)*8+0,24
1210 POKE CH+(ASC("%")-32)*8+1,24
1220 POKE CH+(ASC("%")-32)*8+2,60
1230 POKE CH+(ASC("%")-32)*8+3,60
1240 POKE CH+(ASC("%")-32)*8+4,126
1250 POKE CH+(ASC("%")-32)*8+5,126
1260 POKE CH+(ASC("%")-32)*8+6,24
1270 POKE CH+(ASC("%")-32)*8+7,24

1280 SETCOLOR 1,0,0
1290 SETCOLOR 2,0,14
1999 RETURN

2000 SOUND 0,100,10,10
2010 FOR Q=1 TO 50
2020 SOUND 0,0,0,0
2030 RETURN
```

2
Capture the Quark



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

How to play

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

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

Typing tips

If you are using a 16K Atari 400 plus disk drives, disconnect your drives before you start to type in this program, otherwise you will run out of memory. You will need to use a cassette recorder in order to save the program.

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

Subroutine structure

20	Initialises variables
50	Main play loop
1000	Quark move logic
5000	Moves quatlins and validates their moves
6000	Selects which quatlin to move
6500	Inverts quatlin
6600	Returns quatlin to normal
6700	Sound routine
6800	Zeros board
7000	Draws board

7500 Draws initial postions of quatlins and quark and initialises board
8000 Transfers ROM characters to RAM
8050 Defines graphics character for chequered square
8090 Defines graphics characters for quark
8420 Defines graphics characters for quatlin
8920 Selects RAM characters and disables cursor
9000 End of game

Programming details

Although this program runs in black and white the board is drawn up in white and hatched shading which gives an extra tone. This is defined in lines 8050 to 8080.

Program

```
 10 REM CAPTURE THE QUARK

 15 OPEN #2,4,0,"K:"
 20 DIM D(10,10)
 30 DIM X(4),Y(4)
 40 DIM A(4),B(4)
 45 GOSUB 6800
 48 DIM A$(10)

 50 GOSUB 8000
 55 SETCOLOR 1,0,15
 56 SETCOLOR 2,0,0
 60 GOSUB 7000
 70 H=INT(RND(0)*3)+2:GOSUB 7500
130 GOSUB 6000
140 GOSUB 1000
150 GOTO 130
```

```
1000 POSITION 26,12:PRINT #6;"QUARK'S MOVE"
1010 IF D(QI+2,QJ+2) AND D(QI+2,QJ) AND
     D(QI,QJ+2) AND D(QI,QJ) THEN GOTO 9000
1020 M=1
1030 GOSUB 3000
1035 IF QI+N<1 OR QI+N>8 THEN GOTO 1100
1040 IF D(QI+N+1,QJ+M+1) THEN GOSUB 2500
1050 IF D(QI+N+2,QJ+M+2)=1 AND D(QI+N,QJ+M+2)
     =1 AND QJ<7 AND QJ>1 THEN M=-M
1100 IF D(QI+N+1,QJ+M+1) THEN GOSUB 2500
1110 IF OI=QI AND OJ=QJ THEN M=-M:
     N=SGN(RND(1)-0.5):GOTO 1035
1120 OI=QI:OJ=QJ
2000 POSITION QX,QY:PRINT #6;"  ";:
     POSITION QX,QY+1:PRINT #6;"  ";
2010 QX=QX+N*2
2020 QY=QY+M*2
2030 POSITION QX,QY:PRINT #6;"23";:POSITION
     QX,QY+1:PRINT #6;"45";
2040 D(QI+N+1,QJ+M+1)=2
2050 D(QI+1,QJ+1)=0
2060 QI=QI+N
2070 QJ=QJ+M
2080 IF QJ=8 THEN GOTO 9500
2090 RETURN
2500 N=-N
2510 IF D(QI+N+1,QJ+M+1)<>1 THEN RETURN
2520 M=-M
2525 IF QJ<4 THEN N=1
2530 IF D(QI+N+1,QJ+M+1)<>1 THEN RETURN
2540 N=-N
2550 RETURN
3000 N=(QI<5)-(QI>=5)
3005 R=RND(1)
3010 IF QJ>6 THEN RETURN
3020 IF R>0.5 AND QI>3 THEN GOTO 3050
3025 IF QI>7 THEN GOTO 3035
3030 IF (D(QI+3,QJ+3)=0 OR D(QI+2,QJ+3)=0)
     AND D(QI+2,QJ+2)=0 THEN N=-1:RETURN
3035 IF QI<4 OR R>0.5 THEN RETURN
3050 IF (D(QI-3,QJ+3)=0 OR D(QI-2,QJ+3)=0)
     AND D(QI-2,QJ+2)=0 THEN N=1:RETURN
3060 IF R>0.5 THEN GOTO 3025
3070 RETURN
```

```
5000 A(HM)=A(HM)+M
5010 B(HM)=B(HM)-1
5020 IF D(A(HM)+1,B(HM)+1)<>0 THEN GOTO 5100
5050 D(A(HM)-M+1,B(HM)+2)=0
5060 D(A(HM)+1,B(HM)+1)=1
5070 GOTO 5500
5100 A(HM)=A(HM)-M
5110 B(HM)=B(HM)+1
5120 GOSUB 6700
5130 GOTO 6000
5500 POSITION X(HM),Y(HM):PRINT #6;"  ";:
     POSITION X(HM),Y(HM)+1:PRINT #6;"  ";
5510 Y(HM)=Y(HM)-2
5520 X(HM)=X(HM)+M*2
5530 POSITION X(HM),Y(HM):PRINT #6;"68";:
     POSITION X(HM),Y(HM)+1:PRINT #6;"79";
5540 GOSUB 6500
5550 RETURN

6000 GOSUB 6700
6007 POSITION 26,12:PRINT #6;"YOUR MOVE   ";
6010 GET #2,M
6025 IF M=82 THEN GOTO 6100
6030 IF M=43 THEN M=-1:GOTO 5000
6040 IF M=42 THEN M=1:GOTO 5000
6050 GOSUB 6200
6060 GOTO 6000
6100 POSITION 27,14:PRINT #6;"RESIGN Y/N?"
6105 GET #2,A
6110 IF A=89 THEN GOTO 9500
6120 POSITION 27,14:PRINT #6;"            ":
     REM 12 SPACES
6130 GOTO 6000
6200 GOSUB 6600
6210 HM=HM+1
6220 IF HM>H THEN HM=1
6230 GOSUB 6500
6240 RETURN

6500 POSITION X(HM),Y(HM):PRINT #6;
     CHR$(54+128);CHR$(56+128);
6510 POSITION X(HM),Y(HM)+1:PRINT #6;
     CHR$(55+128);CHR$(57+128)
6520 RETURN
```

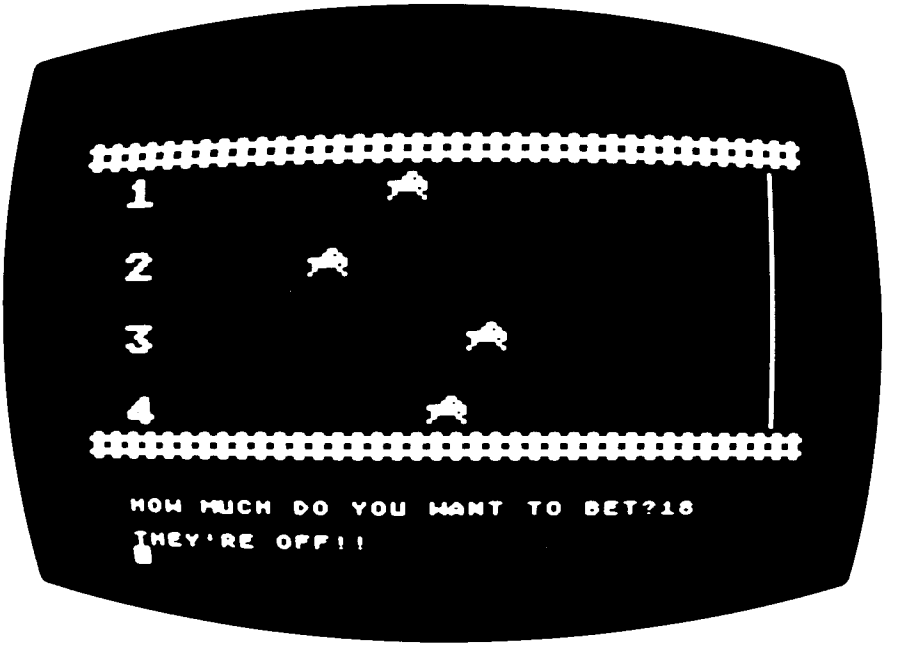
```
6600 POSITION X(HM),Y(HM):PRINT #6;"68";:
     POSITION X(HM),Y(HM)+1:PRINT #6;"79";
6610 RETURN

6700 SOUND 0,80,10,10
6710 FOR I=1 TO 50
6720 NEXT I
6730 SOUND 0,0,0,0
6740 RETURN

6800 FOR I=1 TO 10
6810 FOR J=1 TO 10
6820 D(I,J)=0
6830 NEXT J
6840 NEXT I
6850 RETURN

7000 POSITION 8,3
7005 FOR I=1 TO 4
7010 FOR J=1 TO 8
7020 PRINT "  $$";
7030 IF J/4=INT(J/4) THEN PRINT :
     PRINT "      ";:REM 6 SPACES
7040 NEXT J
7050 FOR J=1 TO 8
7060 PRINT "$$  ";
7070 IF J/4=INT(J/4) THEN PRINT :
     PRINT "      ";:REM 6 SPACES
7080 NEXT J
7090 NEXT I
7100 RETURN
```

```
7500 FOR I=1 TO H
7510 X=I*4+6
7520 POSITION X,17:PRINT #6;"68";:
     POSITION X,18:PRINT #6;"79";
7530 D(I*2+1,9)=1
7540 X(I)=X
7550 Y(I)=17
7560 HM=1
7570 A(I)=I*2
7580 B(I)=8
7590 NEXT I
7600 GOSUB 6500
7610 QI=5
7620 QJ=1
7630 QX=QI*2+6
7640 QY=3
7650 POSITION QX,QY:PRINT #6;"23":
     POSITION QX,QY+1:PRINT #6;"45";
7660 FOR I=1 TO 10
7670 D(I,1)=1
7680 D(1,I)=1
7690 D(10,I)=1
7700 D(I,10)=1
7710 NEXT I
7720 D(QI+1,QJ+1)=2
7730 OI=0
7740 OJ=0
7750 RETURN

8000 GRAPHICS 0
8005 CH=(PEEK(106)-8)*256
8010 CHORG=(PEEK(756)*256)
8020 FOR I=0 TO 511
8030 POKE CH+I,PEEK(CHORG+I)
8040 NEXT I

8050 FOR Q=0 TO 7 STEP 2
8060 POKE CH+(ASC("$")-32)*8+Q,51
8070 POKE CH+(ASC("$")-32)*8+Q+1,204
8080 NEXT Q
```

```
8090 POKE CH+(ASC("2")-32)*8+0,0
8100 POKE CH+(ASC("2")-32)*8+1,0
8120 POKE CH+(ASC("2")-32)*8+2,7
8130 POKE CH+(ASC("2")-32)*8+3,31
8140 POKE CH+(ASC("2")-32)*8+4,63
8150 POKE CH+(ASC("2")-32)*8+5,115
8160 POKE CH+(ASC("2")-32)*8+6,115
8170 POKE CH+(ASC("2")-32)*8+7,127
8180 POKE CH+(ASC("3")-32)*8+0,0
8190 POKE CH+(ASC("3")-32)*8+1,0
8200 POKE CH+(ASC("3")-32)*8+2,224
8210 POKE CH+(ASC("3")-32)*8+3,248
8220 POKE CH+(ASC("3")-32)*8+4,252
8230 POKE CH+(ASC("3")-32)*8+5,206
8240 POKE CH+(ASC("3")-32)*8+6,206
8250 POKE CH+(ASC("3")-32)*8+7,254
8260 POKE CH+(ASC("4")-32)*8+0,127
8270 POKE CH+(ASC("4")-32)*8+1,63
8280 POKE CH+(ASC("4")-32)*8+2,31
8290 POKE CH+(ASC("4")-32)*8+3,15
8300 POKE CH+(ASC("4")-32)*8+4,7
8310 POKE CH+(ASC("4")-32)*8+5,3
8320 POKE CH+(ASC("4")-32)*8+6,1
8330 POKE CH+(ASC("4")-32)*8+7,0
8340 POKE CH+(ASC("5")-32)*8+0,254
8350 POKE CH+(ASC("5")-32)*8+1,252
8360 POKE CH+(ASC("5")-32)*8+2,248
8370 POKE CH+(ASC("5")-32)*8+3,240
8380 POKE CH+(ASC("5")-32)*8+4,224
8390 POKE CH+(ASC("5")-32)*8+5,192
8400 POKE CH+(ASC("5")-32)*8+6,128
8410 POKE CH+(ASC("5")-32)*8+7,0
```

```
8420 POKE CH+(ASC("6")-32)*8+0,0
8430 POKE CH+(ASC("6")-32)*8+1,0
8440 POKE CH+(ASC("6")-32)*8+2,15
8450 POKE CH+(ASC("6")-32)*8+3,31
8460 POKE CH+(ASC("6")-32)*8+4,31
8470 POKE CH+(ASC("6")-32)*8+5,63
8480 POKE CH+(ASC("6")-32)*8+6,55
8490 POKE CH+(ASC("6")-32)*8+7,115
8500 POKE CH+(ASC("7")-32)*8+0,115
8510 POKE CH+(ASC("7")-32)*8+1,115
8520 POKE CH+(ASC("7")-32)*8+2,115
8530 POKE CH+(ASC("7")-32)*8+3,115
8540 POKE CH+(ASC("7")-32)*8+4,3
8550 POKE CH+(ASC("7")-32)*8+5,3
8560 POKE CH+(ASC("7")-32)*8+6,3
8570 POKE CH+(ASC("7")-32)*8+7,0
8580 POKE CH+(ASC("8")-32)*8+0,0
8590 POKE CH+(ASC("8")-32)*8+1,0
8600 POKE CH+(ASC("8")-32)*8+2,240
8610 POKE CH+(ASC("8")-32)*8+3,248
8620 POKE CH+(ASC("8")-32)*8+4,248
8630 POKE CH+(ASC("8")-32)*8+5,252
8650 POKE CH+(ASC("8")-32)*8+6,220
8660 POKE CH+(ASC("8")-32)*8+7,206
8670 POKE CH+(ASC("9")-32)*8+0,206
8680 POKE CH+(ASC("9")-32)*8+1,206
8690 POKE CH+(ASC("9")-32)*8+2,206
8700 POKE CH+(ASC("9")-32)*8+3,206
8710 POKE CH+(ASC("9")-32)*8+4,192
8720 POKE CH+(ASC("9")-32)*8+5,192
8730 POKE CH+(ASC("9")-32)*8+6,192
8740 POKE CH+(ASC("9")-32)*8+7,0

8920 PRINT CHR$(225)
8930 POKE 756,CH/256
8940 POKE 752,1
8950 RETURN
9000 POSITION 1,20:PRINT "YOU WIN"
9010 GOTO 9900
9500 POSITION 1,20:PRINT "QUARK WINS"
9900 PRINT "ANOTHER GAME"
9910 INPUT A$
9930 IF A$="Y" THEN RUN
9940 GRAPHICS 0
9950 PRINT "BYE"
```

3
Atari Ledger



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

How to play

At the beginning of the game you are allocated a hundred chips. You have to bet on which horse will come in first and must decide how much to stake (the odds are five to one, so if you win, having placed 20, you will receive 100). The Atari keeps a tally of your winnings and losses and will tell you if you go broke. During the race, your horse is the white one but as the race is run automatically, and at random, there's nothing you can do to make your horse win.

Typing tips

Pay very careful attention to the stops and commas in the DATA statements in subroutine 6000. If you make mistakes in typing in this subroutine, the program may still run but the tune may be unrecognisable or sound discordant! The character between quotes in line 1220 is the hash symbol (SHIFT and 3).

Subroutine structure

200	Defines graphics character for horse
600	Calls routines for music and title frame
610	Initialises graphics characters and variables
1070	Prints race course
1560	Runs race
2000	Title frame
3000	Betting routine
4000	Moves horses
5000	Plays whole tune
6000	Plays one note of tune
7000	Moves horses' legs
8000	Win/lose routine
9000	Another race or end of game

Programming details

This is the only program in this collection that plays a tune, so it is worth drawing your attention to the details of lines 6000–6080. Think of these DATA statements as being made up of a pair of values. The first in each pair relates to the pitch of the note and the second to the length of time it is sounded for. This second value is typically 1 or .5 or .25 – representing a minim, a crochet or a semiquaver. The number −99 crops up instead of a pitch value every so often. The effect of this is to cause a *rest*, or pause, in the music. Line 6080 signals the end of the tune. After detecting 99,99 the program resets the DATA statements so that next time round the tune starts playing from the beginning. At the beginning of the game the tune is produced by calling subroutine 5000 which plays all the notes one after the other. However, because the Atari cannot do two things at once – it can't play a note and move the horses – the notes of

the tune are produced by a call directly to subroutine 6000 which plays one note in the tune and then moves the pointer to the next note so that the next time it is called the next note is played. To synchronise the sound and the movement, the horses are moved, then subroutine 6000 is called to play a note, the horses are moved again, another note is played and so on. The result is a sequence of notes with longer than normal rests between them but, because it does not take much time to move the horses, you still get the impression of a tune being played. You can use this technique in other games but, if the amount of calculation that you have to do between calling each note becomes too long, you'll no longer be able to hear the tune.

Scope for improvement

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

Program

```
 10 REM ATARI  LEDGER

200 GRAPHICS 2+16
210 CH=(PEEK(106)-8)*256
220 CHORG=(PEEK(756)*256)
230 FOR I=0 TO 511
240 POKE CH+I,PEEK(CHORG+I)
250 NEXT I
400 FOR I=0 TO 7
410 POKE CH+(ASC("&")-32)*8+I,128
420 NEXT I
500 POKE CH+(ASC("$")-32)*8+0,0
510 POKE CH+(ASC("$")-32)*8+1,12
520 POKE CH+(ASC("$")-32)*8+2,26
530 POKE CH+(ASC("$")-32)*8+3,255
540 POKE CH+(ASC("$")-32)*8+4,125
550 POKE CH+(ASC("$")-32)*8+5,66
560 POKE CH+(ASC("$")-32)*8+6,129
570 POKE CH+(ASC("$")-32)*8+7,0
580 POKE 756,CH/256
```

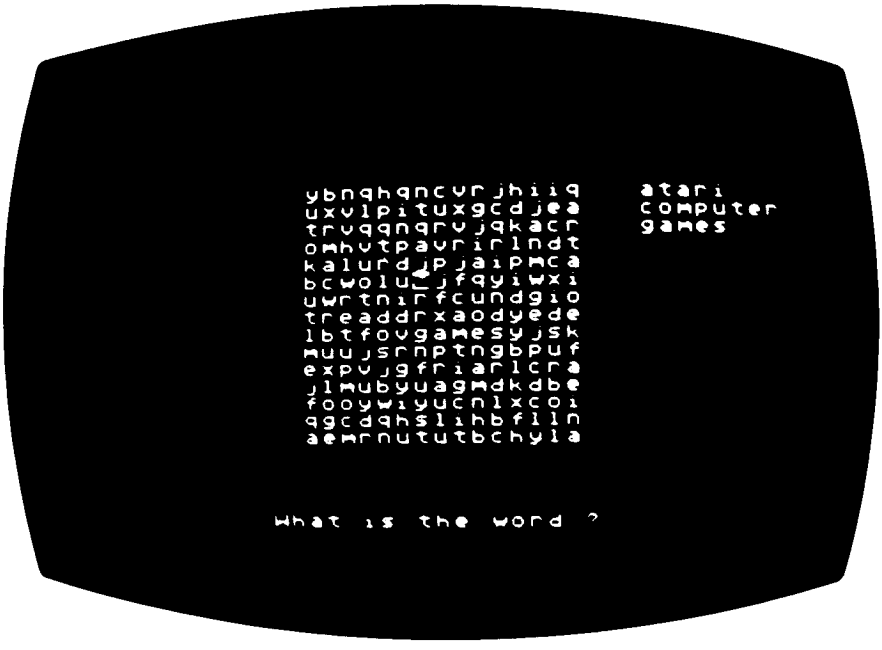
```
 600 GOSUB 5000

 610 TOTAL=100
 620 FLAG=0
1000 DIM X(5),Y(5),A$(5)
1005 GRAPHICS 2
1010 SETCOLOR 0,0,0
1020 SETCOLOR 1,0,14
1030 SETCOLOR 2,12,4
1040 SETCOLOR 3,3,5
1050 SETCOLOR 4,12,4
1060 POKE 756,CH/256

1070 RESTORE
1200 FOR X=0 TO 18
1220 POSITION X,0:PRINT #6;"#"
1225 POSITION X,8:PRINT #6;"#"
1230 NEXT X
1300 FOR Y=1 TO 7
1310 POSITION 18,Y:PRINT #6;"&"
1320 NEXT Y
1500 FOR Y=1 TO 4
1520 X(Y)=2
1530 Y(Y)=Y*2-1
1540 POSITION X(Y)-1,Y(Y):PRINT #6;Y;
     CHR$(132)
1550 NEXT Y

1560 GOSUB 3000
1570 TEMPO=50
1710 GOSUB 4000
1720 GOTO 1710

2000 SETCOLOR 0,0,0
2005 SETCOLOR 3,3,5
2010 SETCOLOR 4,0,14
2020 POSITION 3,2
2050 PRINT #6;"A T A R I"
2060 POSITION 5,4
2070 PRINT #6;"L E D G E R"
2080 RETURN
```

```
3000 PRINT "WHICH HORSE DO YOU WANT TO BET
     ON ";:INPUT B
3010 IF B<1 OR B>4 THEN PRINT "NO SUCH
     HORSE":GOTO 3000
3020 PRINT "YOU HAVE ";TOTAL;" POUNDS"
3030 PRINT "HOW MUCH DO YOU WANT TO BET";:
     INPUT M
3040 IF TOTAL-M<0 THEN PRINT "YOU DON'T HAVE
     ENOUGH MONEY!":GOTO 3020
3050 TOTAL=TOTAL-M
3060 PRINT :PRINT "THEY'RE OFF!!"

4000 REM MOVE HORSES
4010 Z=INT(RND(0)*4)+1
4050 POSITION X(Z),Y(Z):PRINT #6;" "
4110 X(Z)=X(Z)+(1+RND(0)*0.4)
4120 COL=132
4130 IF Z=B THEN COL=4
4140 POSITION X(Z),Y(Z):PRINT #6;CHR$(COL)
4145 GOSUB 7000
4160 IF X(Z)>16.5 THEN GOTO 8000
4170 GOSUB 6000
4180 IF T=99 THEN RESTORE
4190 RETURN

5000 GOSUB 2000
5005 TEMPO=50
5010 I=1
5015 GOSUB 6000
5020 IF T=99 THEN POSITION I+1,9:PRINT ;" ":
     RESTORE :RETURN
5030 POSITION I,9:PRINT #6;" ";CHR$(132)
5032 GOSUB 7000
5035 FOR Q=0 TO 10:NEXT Q
5040 I=I+0.3
5060 GOTO 5015
```

```
6000 DATA 72,.5,72,.5,72,.5,85,.5,72,.5,64,.5
6005 DATA 72,.5,85,.5,-99,.5,85,.5,96,1.5,85,
     .5,96,1
6010 DATA 72,.5,72,.5,85,.5,72,.5,64,.5,72,
     .5,85,.5
6020 DATA -99,.5,85,.25,96,.25,108,.25,96,.25
6025 DATA 85,.5,96,.5,108,1.5
6030 DATA -99,1,108,.75,108,.25,85,.5,72,.5,
     53,1.5
6040 DATA -99,.5,64,.75,64,.25,53,.5,64,.5,
     72,1.5
6050 DATA 85,.25,81,.25,72,.5,72,.5,85,.25,
     85,.25
6060 DATA 72,.25,72,.25,64,.5,72,.5,85,1
6070 DATA 96,.5,85,.5,81,.25,85,.5,96,.25,96,
     .25,108,1.5
6080 DATA 99,99
6090 READ P,T
6100 IF T=99 THEN RETURN
6110 T=T*TEMPO
6120 IF P=-99 THEN FOR Q=1 TO T:NEXT Q:RETURN
6130 SOUND 0,P,10,8
6140 FOR Q=1 TO T:NEXT Q
6150 SOUND 0,P,10,0
6160 RETURN

7000 IF FLAG=1 THEN GOTO 7500
7010 FLAG=1
7020 POKE CH+(ASC("$")-32)*8+6,40
7030 POKE 756,CH/256
7040 RETURN
7500 FLAG=0
7510 POKE CH+(ASC("$")-32)*8+6,129
7520 POKE 756,CH/256
7530 RETURN

8000 IF Z=B THEN PRINT "YOU WIN ";:
     TOTAL=TOTAL+INT(5*M):PRINT INT(5*M)
8010 IF Z<>B THEN PRINT "YOU LOSE ";M
8020 IF TOTAL<=0 THEN PRINT "YOU'RE
     BROKE    ":GOTO 9020
```

```
9000 PRINT "YOU HAVE ";TOTAL;" ANOTHER
     RACE";:INPUT A$
9005 RESTORE
9010 IF A$(1)="Y" THEN GOTO 1005
9020 FOR I=1 TO 100:NEXT I
9030 END
```

4
Word Scramble



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

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

position in the square is indicated by an inverted character which you can use as a pointer to identify the *first* letter of each word you find. Move this pointer using the cursor control keys. When you think you have found the first letter of a word type a 'w'. If you are correct you score a point.

How to play

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

Typing tips

When playing this game remember to use lower case letters only. There is a single space between double quotes in lines 410, 600, 800, 810, 930, 1400 and 1420 so don't type in a null string instead. A null string (two sets of double quotes without any space between) does, however, occur in line 1090.

Subroutine structure

15	Initialisation routine
80	Calls finding words routine
90	Asks for words to be input
280	Gives error message if more than 10 letters input
330	Finds longest word left in list
380	Constructs puzzle
760	Prints puzzle
870	Allows preview and fills up puzzle with random letters
1010	Main play loop
1240	Asks for guess of word
1320	Word correct routine
1500	Checks for word
1650	Sets score to zero
1670	Resign routine
1710	End of game
2000	Sound tone routine

Programming details

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

One of the problems with Atari BASIC is its lack of string arrays. This is something that can be overcome however using the very simple idea of *packing*. For example, the fifteen by fifteen grid of letters in this program is stored like a two-dimensional array where every element is a single character using the string D$. The letter at I,J is given by D$(I+(J−1)*15,I+(J−1)*15).

Scope for improvement

If you have a printer, you could add a routine to this program to enable the completed puzzle to be printed out so that you take it away to be solved. To make the game more difficult you could allow it to accept more words. Notice, however, that the more words there are, the longer it will take to be set up initially. To make the game easier you could remove words from the word list, or mark them in some way, once they had been found.

Program

```
 10 REM WORD SCRAMBLE

 15 OPEN #2,4,0,"K:"
 20 DIM W$(30)
 40 GOSUB 90
 50 GOSUB 380
 70 GOSUB 870

 80 GOSUB 1010

 90 GRAPHICS 0
100 DIM L$(100)
105 FOR I=1 TO 100:L$(I)=" ":NEXT I
110 DIM C(10)
120 LST=0
130 PRINT CHR$(125)
140 POSITION 5,2
150 PRINT "How many words ";:INPUT W
155 PRINT
160 IF W<2 OR W>10 GOSUB 2000:GOTO 140
170 PRINT "Enter words (lower case only)"
180 FOR I=1 TO W
190 POSITION 5,5+I:PRINT "Word Number ";I;
    "=";
200 INPUT W$
210 IF LEN(W$)>10 THEN GOSUB 280
220 IF LEN(W$)<1 THEN GOTO 200
230 L$((I-1)*10+1,(I-1)*10+LEN(W$))=W$
240 C(I)=LEN(W$)
250 NEXT I
270 RETURN
```

```
280 GOSUB 2000
290 POSITION 5,5+I:PRINT "maximum of ten
    letters!";
300 INPUT W$
310 POSITION 5,5+I:PRINT "Word Number ";I;
    "=";W$;"
320 RETURN

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

```
380 REM FIT
390 DIM D$(225)
400 FOR J=1 TO 225
410 D$(J)=" "
420 NEXT J
450 PRINT CHR$(125)
460 F=0
470 FOR I=1 TO W
480 GOSUB 330
490 L=C(J)
500 C(J)=0
510 X=INT(RND(0)*(15-L))+1
520 Y=INT(RND(0)*(15-L))+1
530 V=INT(RND(0)*3)-1
540 U=INT(RND(0)*3)-1
550 IF U=0 AND V=0 THEN GOTO 530
560 A=X:B=Y
570 IF V<0 THEN A=A+L
580 IF U<0 THEN B=B+L
590 FOR K=1 TO L
595 W$(1,1)=L$((J-1)*10+K,(J-1)*10+K)
600 IF D$(A+(B-1)*15,A+(B-1)*15)<>" " AND
    D$(A+(B-1)*15,A+(B-1)*15)<>W$(1,1) THEN
    F=1:K=L:GOTO 630
610 A=A+V
620 B=B+U
630 NEXT K
640 IF F=1 THEN F=0:GOTO 510
650 PRINT "Working"
660 A=X:B=Y
670 IF V<0 THEN A=A+L
680 IF U<0 THEN B=B+L
690 FOR K=1 TO L
700 D$(A+(B-1)*15,A+(B-1)*15)=L$((J-1)*10+K,
    (J-1)*10+K)
710 A=A+V
720 B=B+U
730 NEXT K
740 NEXT I
750 RETURN
```

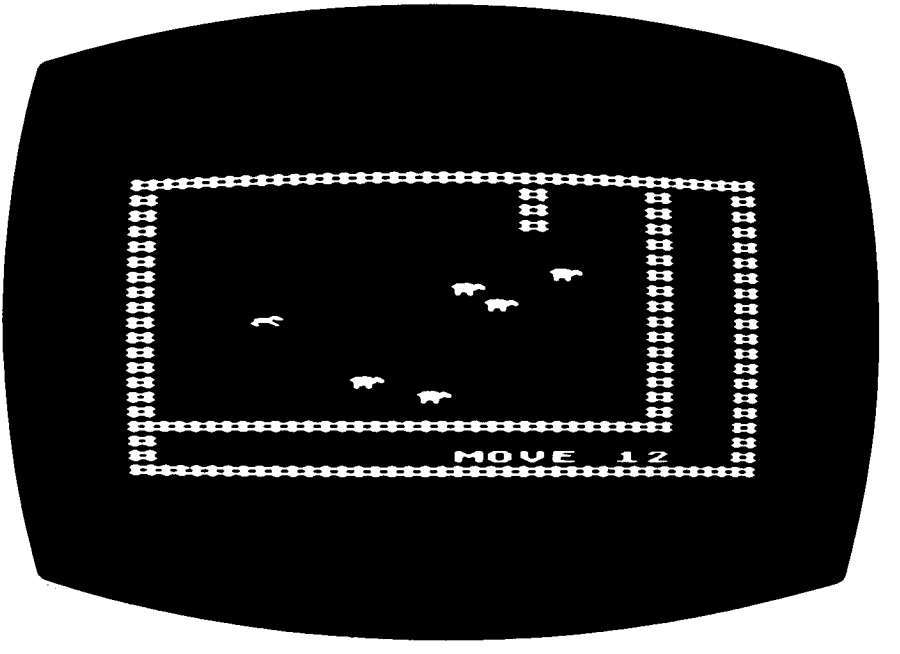
```
 760 REM PRINT
 765 POKE 752,1
 770 PRINT CHR$(125)
 780 FOR M=1 TO 15
 790 FOR N=1 TO 15
 800 IF D$((M-1)*15+N,(M-1)*15+N)=" " THEN
     PRINT ".";
 810 IF D$((M-1)*15+N,(M-1)*15+N)<>" " THEN
     PRINT D$((M-1)*15+N,(M-1)*15+N);
 820 NEXT N
 830 IF LST=0 OR M>10 THEN PRINT
 840 IF LST=1 AND M<=10 THEN POSITION 20,M:
     PRINT L$((M-1)*10+1,M*10)
 850 NEXT M
 860 RETURN

 870 REM PREVIEW
 880 DIM A$(1)
 890 PRINT "Do you want to preview the
     puzzle y/n ":INPUT A$
 900 IF A$="y" THEN GOSUB 760
 910 FOR I=1 TO 15
 920 FOR J=1 TO 15
 930 IF D$(I+(J-1)*15,I+(J-1)*15)=" " THEN
     D$(I+(J-1)*15,I+(J-1)*15)=
     CHR$(INT(RND(0)*25)+97)
 940 NEXT J
 950 NEXT I
 960 PRINT "Do you want to display"
 962 PRINT "the words beside the puzzle
     y/n ":INPUT A$
 970 IF LEN(A$)=0 THEN GOTO 960
 980 IF A$="y" THEN LST=1
 990 GOSUB 760
1000 RETURN
```

```
1010 X=2
1020 Y=1
1050 LOCATE X,Y,Q
1055 POSITION X,Y:PRINT CHR$(Q+128);
1080 GET #2,A
1090 IF A$="" THEN GOTO 1080
1100 IF A=119 THEN GOTO 1230
1110 POSITION X,Y:PRINT CHR$(Q);
1120 IF A=43 AND X>2 THEN X=X-1
1130 IF A=42 AND X<16 THEN X=X+1
1140 IF A=45 AND Y>1 THEN Y=Y-1
1150 IF A=61 AND Y<15 THEN Y=Y+1
1160 IF A=114 THEN GOSUB 1670
1220 GOTO 1050
1230 POSITION 0,20
1235 W$="          ":REM TEN SPACES

1240 PRINT "What is the word ";:INPUT W$
1250 IF LEN(W$)=0 OR LEN(W$)>15 THEN
     GOSUB 2000:GOTO 1230
1260 GOSUB 1450
1270 POSITION 0,20:PRINT "
                ":REM 30 SPACES
1280 IF MATCH=0 THEN GOSUB 2000:GOTO 1080
1290 FOR U=-1 TO 1
1300 FOR V=-1 TO 1
1310 IF U=0 AND V=0 THEN GOTO 1340
```

```
1320 GOSUB 1500
1330 IF MATCH=1 THEN V=1:U=1:GOTO 1340
1340 NEXT V
1350 NEXT U
1360 IF MATCH=1 THEN GOTO 1390
1370 GOSUB 2000
1380 GOTO 1080
1390 SCORE=SCORE+1
1400 POSITION 10,18:PRINT "Score= ";SCORE;" "
1410 GOSUB 2000
1420 L$(10*(WORD-1)+1,10*(WORD-1)+1)=" "
1430 IF SCORE=W THEN GOTO 1780
1440 GOTO 1080
1450 MATCH=0
1460 FOR I=1 TO W
1470 IF W$=L$(1+(I-1)*10,LEN(W$)+(I-1)*10)
     THEN MATCH=1:WORD=I:I=W
1480 NEXT I
1490 RETURN

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

1650 SCORE=0
1660 RETURN
```

```
1670 POSITION 0,19
1680 PRINT "Are you sure that you can"
1685 PRINT "find no more words y/n ";
     INPUT A$
1690 POSITION 0,19
1695 FOR I=1 TO 180:PRINT " ";:NEXT I
1700 IF A$<>"y" THEN RETURN

1710 PRINT CHR$(125)
1720 POSITION 10,8:PRINT "Final score= ";
     SCORE
1730 POSITION 8,10
1740 PRINT "Another game y/n ";:INPUT A$
1750 IF A$="y" THEN RUN
1760 IF A$<>"n" THEN GOTO 1730
1770 STOP
1780 POSITION 0,19:PRINT "You got them all!"
1790 GOTO 1740

2000 SOUND 0,80,10,8
2010 FOR T=1 TO 50
2020 NEXT T
2030 SOUND 0,0,0,0
2040 RETURN
```

5
Sheepdog Trials



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

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

How to play

The object of the game is to herd all five sheep into the pen at the top

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

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

Typing tips

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

Subroutine structure

20	Sets up graphics characters and arrays
120	Prints fences
280	Prints sheep
340	Prints dog
390	Moves dog
520	Moves sheep and checks for end of game
550	Move logic for sheep
780	Prints sheep
860	Prints messages and end of game
990	Scatters sheep
2000	Defines graphics character for sheep
2100	Defines graphics character for dog
2200	Change to RAM character set
2220	Set colours

Programming details

The arrow keys used to control the position of the dog are read using the GET command. Before this can get data from the keyboard, however, the keyboard must be OPENed as an input device. This is done by line 30 which OPENs the keyboard as file number 2. Notice that GET waits until a key is actually pressed before allowing the program to continue.

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

Scope for improvement

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

Program

```
10 REM SHEEPDOG TRIAL
```

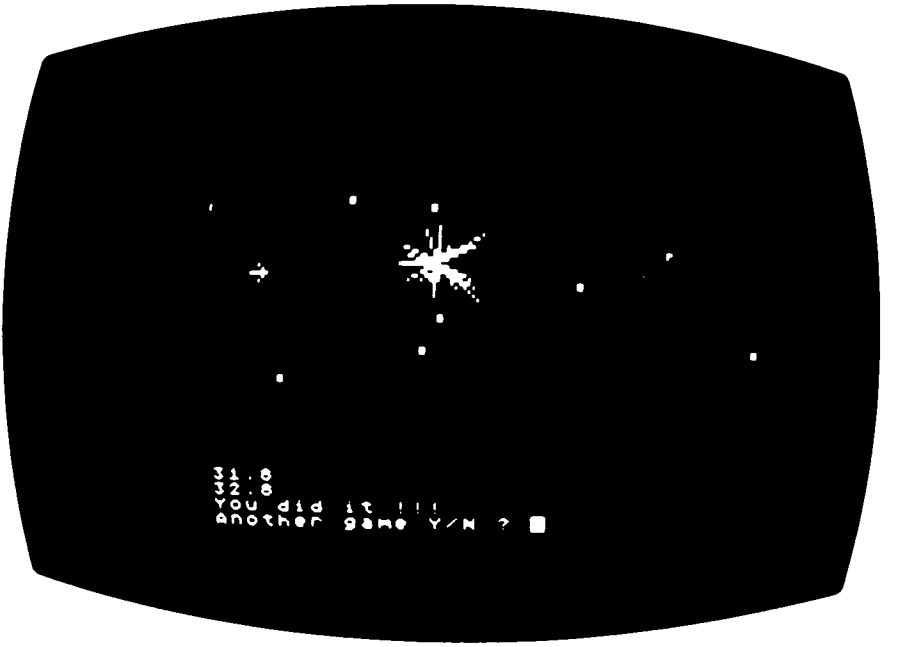
```
 20 GRAPHICS 1+16
 30 OPEN #2,4,0,"K:"
 40 CH=(PEEK(106)-8)*256
 50 CHORG=(PEEK(756)*256)
 60 FOR I=0 TO 511
 70 POKE CH+I,PEEK(CHORG+I)
 80 NEXT I
 90 GOSUB 2000
100 DIM Y(5):DIM X(5)
110 M=0

120 FOR X=0 TO 15
130 POSITION X,16:PRINT #6;"*"
140 NEXT X
150 FOR Y=0 TO 16
160 POSITION 16,Y:PRINT #6;"*"
170 NEXT Y
180 FOR Y=0 TO 19
190 POSITION 0,Y:PRINT #6;"*";:
    POSITION 19,Y:PRINT #6;"*"
200 NEXT Y
210 FOR X=0 TO 19
220 POSITION X,0:PRINT #6;"*";:
    POSITION X,19:PRINT #6;"*"
230 NEXT X
240 FOR Y=1 TO 3
250 POSITION 12,Y:PRINT #6;"*"
260 NEXT Y

280 FOR S=1 TO 5
290 Y(S)=5+INT(RND(0)*10)
300 X(S)=4+INT(RND(0)*6)
310 POSITION X(S),Y(S):PRINT #6;CHR$(4);
320 NEXT S

340 YD=1+INT(RND(0)*3)
350 XD=1+INT(RND(0)*3)
360 POSITION XD,YD:PRINT #6;CHR$(5+160)
```

```
390 GET #2,D
400 IF D=0 THEN GOTO 390
410 IF XD=12 AND YD=4 AND D=45 THEN GOTO 390
420 IF XD=11 AND YD<4 AND D=42 THEN GOTO 390
430 IF XD=13 AND YD<4 AND D=43 THEN GOTO 390
440 POSITION XD,YD:PRINT #6;" "
450 IF D=43 AND XD>1 THEN XD=XD-1
460 IF D=42 AND XD<15 THEN XD=XD+1
470 IF D=61 AND YD<15 THEN YD=YD+1
480 IF D=45 AND YD>1 THEN YD=YD-1
490 POSITION XD,YD:PRINT #6;CHR$(5+160)
500 M=M+1
510 POSITION 10,18:PRINT #6;"MOVE ";M

520 GOSUB 550
530 IF F=0 THEN GOTO 860
540 GOTO 390

550 F=0
570 FOR S=1 TO 5
580 Y=Y(S)
590 X=X(S)
600 IF (ABS(X(S)-XD)<2 AND ABS(Y(S)-YD)<2)
    THEN GOSUB 990
610 IF RND(0)<0.01 THEN GOSUB 990
620 IF ABS(X(S)-XD)>2+RND(0)*2 THEN
    GOTO 780
630 IF ABS(Y(S)-YD)>2+RND(0)*2 THEN
    GOTO 780
640 X(S)=X(S)+SGN(X(S)-XD)
650 Y(S)=Y(S)+SGN(Y(S)-YD)
660 IF X(S)<13 AND X(S)>11 AND Y(S)<4 THEN
    X(S)=X
670 FS=0
680 FOR Z=1 TO 5
690 IF Z=S THEN GOTO 710
700 IF (X(S)=X(Z)) AND (Y(S)=Y(Z)) THEN FS=1
710 NEXT Z
720 IF FS=1 THEN GOTO 640
730 IF X(S)=XD AND Y(S)=YD THEN GOTO 640
740 IF X(S)<1 THEN X(S)=1
750 IF X(S)>15 THEN X(S)=15
760 IF Y(S)<1 THEN Y(S)=1
770 IF Y(S)>15 THEN Y(S)=15
```

```
 780 POSITION X,Y:PRINT #6;" "
 790 POSITION X(S),Y(S):PRINT #6;CHR$(4);
 800 IF X(S)>12 AND (Y(S)>0 AND Y(S)<4) THEN
     GOTO 820
 810 F=1
 820 NEXT S
 830 RETURN

 860 FOR I=1 TO 100:NEXT I
 865 POSITION 2,22
 870 IF M<40 THEN PRINT #6;"SUPER
     SHEPHERD!!":GOTO 920
 880 IF M<60 THEN PRINT #6;"GOOD DOG!!":
     GOTO 920
 890 IF M<90 THEN PRINT #6;"KEEP
     PRACTISING":GOTO 920
 900 IF M<120 THEN PRINT #6;"BETTER LUCK
     NEXT TIME":GOTO 920
 910 PRINT #6;"HAND IN YOUR CROOK !!"
 920 PRINT "YOU TOOK  ";M;" MOVES"
 930 DIM A$(1)
 940 PRINT "ANOTHER GAME Y/N ":INPUT A$
 950 IF A$="Y" THEN RUN
 960 GRAPHICS 0
 980 END

 990 XT=X(S):YT=Y(S)
1000 X(S)=X(S)+(SGN(RND(0)-0.5)*(2+RND(0)*2))
1010 IF X(S)<1 THEN X(S)=1
1020 IF X(S)>15 THEN X(S)=15
1030 Y(S)=Y(S)+(SGN(RND(0)-0.5)*(2+RND(0)*2))
1035 Y(S)=INT(Y(S)):X(S)=INT(X(S))
1040 IF Y(S)<1 THEN Y(S)=1
1050 IF Y(S)>15 THEN Y(S)=15
1060 IF X(S)=12 AND Y(S)<3 THEN GOTO 1000
1070 IF XT=X(S) AND YT=Y(S) THEN GOTO 1000
1080 RETURN

2000 POKE CH+(ASC("$")-32)*8+0,0
2010 POKE CH+(ASC("$")-32)*8+1,0
2020 POKE CH+(ASC("$")-32)*8+2,122
2030 POKE CH+(ASC("$")-32)*8+3,255
2040 POKE CH+(ASC("$")-32)*8+4,125
2050 POKE CH+(ASC("$")-32)*8+5,120
2060 POKE CH+(ASC("$")-32)*8+6,72
2070 POKE CH+(ASC("$")-32)*8+7,72
```

```
2100 POKE CH+(ASC("%")-32)*8+0,0
2110 POKE CH+(ASC("%")-32)*8+1,0
2120 POKE CH+(ASC("%")-32)*8+2,6
2130 POKE CH+(ASC("%")-32)*8+3,123
2140 POKE CH+(ASC("%")-32)*8+4,120
2150 POKE CH+(ASC("%")-32)*8+5,132
2160 POKE CH+(ASC("%")-32)*8+6,66
2170 POKE CH+(ASC("%")-32)*8+7,0

2200 POKE 752,1
2210 POKE 756,CH/256

2220 SETCOLOR 0,0,0
2230 SETCOLOR 1,0,14
2240 SETCOLOR 2,2,5
2250 SETCOLOR 4,13,8
2500 RETURN
```

6
Laser Attack



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

How to play

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

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

Typing tips

This game only just fits into the memory of a 16K Atari 400, so if by any chance you have a disk drive attached to this model you will need to disconnect it before entering this program and have a cassette recorder available for saving it.

Subroutine structure

20	Main play loop
1000	Fires or rotates direction of arrow
2000	Laser zap logic
2235	Explosion routine
2500	Detect hit logic
3000	Moves arrow
4000	Moves target
5000	Blanks arrow
5100	Draws arrow
5200	Sound routine
5300	Draws target
5500	Zap sound for laser fire
7000	Title frame
7500	Gets and prints time
7900	Plots stars
8000	Initialises variables and set colours
9000	End of game

Programming details

This is a complicated program and one that illustrates a number of

novel programming methods. The cartwheel and the arrow are both produced by collections of PLOT and DRAWTO commands. The arrays V and W are used to hold the horizontal and vertical velocities that determine the movement of both the cartwheel and the arrow. They also determine the direction in which the lines that make up the moving shapes are drawn. This technique ensures that the arrow keeps pointing the same way it is moving.

Scope for improvement

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

Program

```
  10 REM LASER ATTACK

  20 GOSUB 7000
  30 GOSUB 8000
  40 GOSUB 7500
  50 IF INT(T)>1000 THEN GOTO 9000
  60 GOSUB 1000
  70 GOSUB 3000
  80 IF H=1 THEN GOTO 9000
  90 GOSUB 4000
 100 GOTO 40

1000 I=PEEK(764)
1010 IF I=255 THEN RETURN
1015 POKE 764,255
1020 IF I=33 THEN GOTO 2000
1030 K=K+1
1040 IF K>8 THEN K=1
1050 RETURN
```

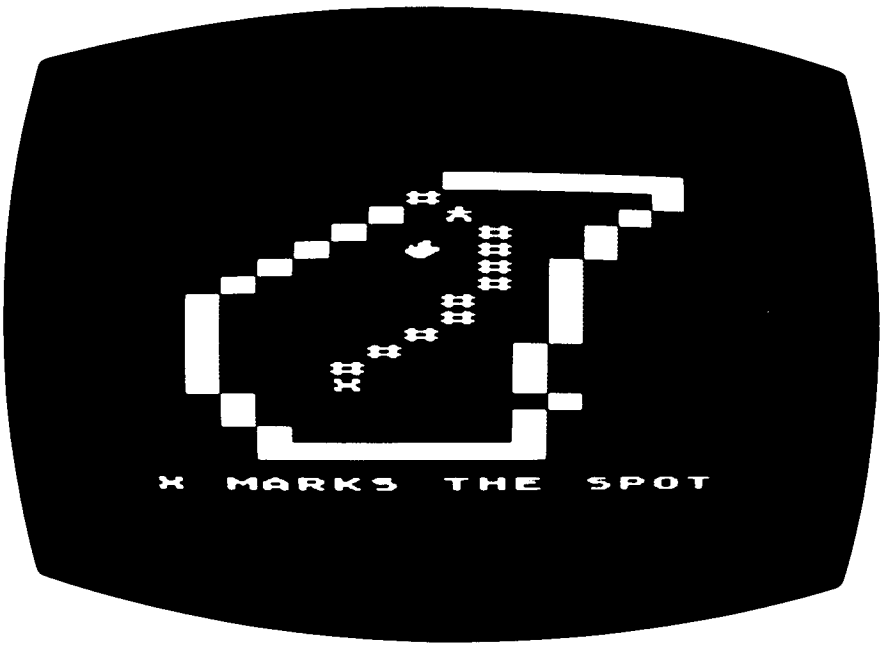
```
2000 XL=X+4*V(K)
2010 YL=Y+4*W(K)
2015 GOSUB 2500
2050 PLOT XL,YL
2060 DX=0
2070 IF V(K)=1 THEN DX=144-XL
2080 IF V(K)=-1 THEN DX=6-XL
2090 DY=0
2100 IF W(K)=1 THEN DY=72-YL
2120 IF W(K)=-1 THEN DY=6-YL
2130 IF V(K)*W(K)=0 THEN GOTO 2200
2140 IF ABS(DX)<ABS(DY) THEN
     DY=ABS(DX)*SGN(DY):GOTO 2200
2150 DX=ABS(DY)*SGN(DX)
2200 COLOR 1:DRAWTO XL+DX,YL+DY
2210 COLOR 0:PLOT XL,YL
2215 GOSUB 5500
2220 COLOR 0:DRAWTO XL+DX,YL+DY
2230 IF H=0 THEN RETURN

2235 MX=B+2:MY=A-2
2240 FOR I=1 TO INT(RND(0)*5)+20
2250 COLOR 1:PLOT MX,MY
2260 DX=10-RND(0)*20
2270 IF MX+DX>159 OR MX+DX<0 THEN GOTO 2330
2280 DY=10-RND(0)*20
2290 IF MY+DY>79 OR MY+DY<0 THEN GOTO 2330
2300 PLOT MX,MY
2310 DRAWTO MX+DX,MY+DY
2320 SOUND 0,INT(RND(0)*100),4,8
2330 NEXT I
2340 RETURN

2500 H=0
2510 DY=A-Y-4*W(K)
2520 DX=B-X-4*V(K)
2530 IF W(K)*DX<>V(K)*DY THEN RETURN
2540 IF ABS(V(K))*SGN(DX)<>V(K) OR
     ABS(W(K))*SGN(DY)<>W(K) THEN RETURN
2550 H=1
2560 RETURN
```

```
3000 IF NB1=0 THEN GOSUB 5000
3010 X=X+6*V(K)
3020 Y=Y+6*W(K)
3030 IF X<6 THEN X=144
3040 IF X>144 THEN X=6
3050 IF Y<6 THEN Y=72
3060 IF Y>72 THEN Y=6
3065 LOCATE X+V(K)*4,Y+W(K)*4,Q:IF Q<>0 THEN
     GOSUB 5200:NB1=1:GOTO 1030
3070 GOSUB 5100
3080 NB1=0
3090 RETURN

4000 IF RND(0)>1.05-DF/20 THEN Z=Z+1
4010 IF Z>8 THEN Z=1
4020 IF NB2=0 THEN GOSUB 5300
4030 A=A+6*V(Z)
4040 B=B+6*W(Z)
4050 IF B<6 THEN B=144
4060 IF B>144 THEN B=6
4070 IF A<6 THEN A=72
4080 IF A>72 THEN A=6
4085 LOCATE B+2,A+2,Q:IF Q<>0 THEN
     GOSUB 5200:NB2=1:Z=Z+1:GOTO 4010
4090 R=R+1:IF R>8 THEN R=1
4100 GOSUB 5400
4110 NB2=0
4120 RETURN

5000 COLOR 0
5010 K=K-1
5020 GOSUB 5110
5030 K=K+1
5040 GOTO 5110

5100 COLOR 1
5110 PLOT X,Y
5120 DRAWTO X+4*V(K),Y+4*W(K)
5130 I=K+3:I=I-INT(I/8)*8
5140 DRAWTO X+4*V(K)+2*V(I),Y+4*W(K)+2*W(I)
5150 PLOT X+4*V(K),Y+4*W(K)
5160 I=K+5:I=I-INT(I/8)*8
5170 DRAWTO X+4*V(K)+2*V(I),Y+4*W(K)+2*W(I)
5180 RETURN
```

```
5200 SOUND 0,80,10,10
5210 FOR S=1 TO 10
5220 NEXT S
5230 SOUND 0,0,0,0
5240 RETURN

5300 COLOR 0
5310 GOTO 5410
5400 COLOR 1
5410 PLOT B-2*V(R),A-2*W(R)
5420 DRAWTO B+V(R)*2,A+W(R)*2
5430 I=R+2:I=I-INT(I/8)*8
5440 PLOT B-V(I)*2,A-W(I)*2
5450 DRAWTO B+V(I)*2,A+W(I)*2
5460 RETURN

5500 FOR S=255 TO 0 STEP 20
5510 SOUND 0,S,10,10:SOUND 0,S,10,14
5520 NEXT S
5530 SOUND 0,0,0,0
5540 RETURN

7000 GRAPHICS 0
7040 POSITION 8,2:
     PRINT "L a s e r   A t t a c k"
7045 POSITION 5,5
7050 PRINT " You are in control of an"
7060 POSITION 4,7:PRINT "advanced laser
     attack ship in"
7070 POSITION 6,9:PRINT "pursuit of an enemy
     craft"
7080 POSITION 4,13:PRINT "Shoot it down
     before your time"
7090 POSITION 14,15:PRINT "is up !!!!"
7100 POSITION 2,22:PRINT "Select the
     difficulty level - "
7105 PRINT " 1 (easy) to 10 (difficult) ";
7110 INPUT DF
7120 IF DF<1 OR DF>10 THEN GOTO 7100
7130 RETURN

7500 T=(PEEK(20)+PEEK(19)*256)/5
7510 PRINT INT(T)/10;"  "
7520 RETURN
```

```
7900 REM STARS
7910 XS=INT(RND(0)*159)
7920 YS=INT(RND(0)*60)
7930 COLOR 1
7940 PLOT XS,YS:PLOT XS+1,YS:PLOT XS,YS+1:
     PLOT XS+1,YS+1
7950 RETURN

8000 DIM W(8),V(8),A$(1)
8010 GRAPHICS 7
8020 DATA -1,1,0,1,1,1,1,0,1,-1,0,-1,-1,-1,
     -1,0,-1,1
8030 FOR I=0 TO 8
8040 READ W,V:W(I)=W:V(I)=V
8050 NEXT I
8060 K=1
8070 X=3*6+6
8080 Y=4*6+6
8090 V=V(K)
8100 W=W(K)
8105 SETCOLOR 1,0,14:SETCOLOR 2,7,8
8106 SETCOLOR 4,7,8:SETCOLOR 0,0,14
8110 FOR J=1 TO 10+RND(0)*DF:GOSUB 7900:
     NEXT J
8120 POKE 18,0
8130 POKE 19,0
8140 POKE 20,0
8150 B=INT(RND(0)*10)*6+6
8160 A=INT(RND(0)*10)*6+6
8900 RETURN

9000 IF H<>1 THEN GOTO 9500
9010 PRINT "You did it !!!"
9020 GOTO 9600
9500 PRINT "Your time is up"
9600 SOUND 0,0,0,0
9610 PRINT "Another game Y/N ";:INPUT A$
9620 IF A$="Y" THEN RUN
```

7
Treasure Island



Find the hidden treasure before the pirate ship reaches the island. This game has all the ingredients of high adventure. A desert island, peopled by natives, both hostile and friendly, gold buried at the spot marked 'X' on the map, quicksands that spell danger to the unlucky treasure-seeker and even long John Silver's parrot. The game is displayed in colour graphics and has sound effects as well.

How to play

The treasure is buried at the spot marked 'X' on the map that is briefly flashed onto the screen at the start of the game. The path is also indicated. You have to follow the path exactly. If you stray there are three possible outcomes. If you are lucky Long John Silver's parrot will guide you back to the path – you will see the parrot hovering over the next position on the path; if you are

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

Typing tips

As this game only just fits into the memory of a 16K Atari 400, it can only be entered and run if there is no disk drive attached.

Subroutine structure

20	Transfers characters from ROM to RAM
100	Defines graphics character for man
200	Defines graphics character for quicksand
300	Defines graphics character for parrot
400	Defines graphics character for natives
500	Defines graphics character for part of ship
600	Defines graphics character for part of ship
700	Defines graphics character for solid block
800	Selects RAM character set and supresses cursor
820	Initialises variables and arrays
1000	Main play loop
1200	Tests for whether in quicksand
1300	Logic for natives' attack
2000	Logic for parrot's help
3000	Prints map
3500	Moves man
3700	Prints quicksands on map
4000	Logic for quicksands
4500	Sound routine
4600	Clears screen and sets up graphics

5000 Constructs island
5500 Constructs path
6000 Moves and prints pirate ship
7000 Prints island
8000 Help routine
9000 Treasure found routine
9990 End of game

Programming details

This is a very long program with lots of different ingredients. It therefore appears rather complicated whereas, in fact, it is quite straightforward. One technique to notice is the way the island is constructed at random by subroutine 5000. There the use of SGN function results in the counter of the island first going out and then coming in at random, giving an island-like shape.

Scope for alteration

If you want to alter the length of time the map is displayed for when you ask to see it, you can alter the length of the delay provided by the FOR loop in line 8020. You could also alter the amount that the pirate ship moves at each step by setting a different value for 'R' in line 6020. Currently it's set randomly to a value between 0 and 2.

Program

```
10 REM TREASURE

20 GRAPHICS 1+16
30 OPEN #2,4,0,"K:"
40 CH=(PEEK(106)-8)*256
50 CHORG=(PEEK(756)*256)
60 FOR I=0 TO 511
70 POKE CH+I,PEEK(CHORG+I)
80 NEXT I
```

```
100 POKE CH+(ASC("$")-32)*8+0,24
110 POKE CH+(ASC("$")-32)*8+1,24
120 POKE CH+(ASC("$")-32)*8+2,126
130 POKE CH+(ASC("$")-32)*8+3,24
140 POKE CH+(ASC("$")-32)*8+4,60
150 POKE CH+(ASC("$")-32)*8+5,102
160 POKE CH+(ASC("$")-32)*8+6,102
170 POKE CH+(ASC("$")-32)*8+7,0

200 POKE CH+(ASC("@")-32)*8+0,12
210 POKE CH+(ASC("@")-32)*8+1,28
220 POKE CH+(ASC("@")-32)*8+2,92
230 POKE CH+(ASC("@")-32)*8+3,127
240 POKE CH+(ASC("@")-32)*8+4,255
250 POKE CH+(ASC("@")-32)*8+5,254
260 POKE CH+(ASC("@")-32)*8+6,124
270 POKE CH+(ASC("@")-32)*8+7,56

300 POKE CH+(ASC("&")-32)*8+0,32
310 POKE CH+(ASC("&")-32)*8+1,54
320 POKE CH+(ASC("&")-32)*8+2,62
330 POKE CH+(ASC("&")-32)*8+3,28
340 POKE CH+(ASC("&")-32)*8+4,30
350 POKE CH+(ASC("&")-32)*8+5,39
360 POKE CH+(ASC("&")-32)*8+6,64
370 POKE CH+(ASC("&")-32)*8+7,0

400 POKE CH+(ASC("%")-32)*8+0,1
410 POKE CH+(ASC("%")-32)*8+1,25
420 POKE CH+(ASC("%")-32)*8+2,217
430 POKE CH+(ASC("%")-32)*8+3,255
440 POKE CH+(ASC("%")-32)*8+4,217
450 POKE CH+(ASC("%")-32)*8+5,25
460 POKE CH+(ASC("%")-32)*8+6,37
470 POKE CH+(ASC("%")-32)*8+7,37

500 POKE CH+(ASC("[")-32)*8+0,255
510 POKE CH+(ASC("[")-32)*8+1,126
520 POKE CH+(ASC("[")-32)*8+2,165
530 POKE CH+(ASC("[")-32)*8+3,195
540 POKE CH+(ASC("[")-32)*8+4,195
550 POKE CH+(ASC("[")-32)*8+5,165
560 POKE CH+(ASC("[")-32)*8+6,126
570 POKE CH+(ASC("[")-32)*8+7,255
```

```
 600 POKE CH+(ASC("]")-32)*8+0,16
 610 POKE CH+(ASC("]")-32)*8+1,255
 620 POKE CH+(ASC("]")-32)*8+2,255
 630 POKE CH+(ASC("]")-32)*8+3,126
 640 POKE CH+(ASC("]")-32)*8+4,126
 650 POKE CH+(ASC("]")-32)*8+5,126
 660 POKE CH+(ASC("]")-32)*8+6,60
 670 POKE CH+(ASC("]")-32)*8+7,60

 700 POKE CH+(ASC("'")-32)*8+0,255
 710 POKE CH+(ASC("'")-32)*8+1,255
 720 POKE CH+(ASC("'")-32)*8+2,255
 730 POKE CH+(ASC("'")-32)*8+3,255
 740 POKE CH+(ASC("'")-32)*8+4,255
 750 POKE CH+(ASC("'")-32)*8+5,255
 760 POKE CH+(ASC("'")-32)*8+6,255
 770 POKE CH+(ASC("'")-32)*8+7,255

 800 POKE 756,CH/256
 810 POKE 752,1

 820 F=0
 830 XS=1:YS=1
 840 MES=0
 850 DIM V(11)
 860 DIM U(11)
 870 DIM L(21)
 880 DIM R(21)
 890 DIM X(21)
 900 DIM A$(10)

1000 GOSUB 5000
1010 XM=X(T+1)
1070 YM=T+1
1080 GOSUB 3000
1090 FOR Q=1 TO 300+INT(RND(0)*200):NEXT Q
1100 GOSUB 7000
1110 GOSUB 3600
1120 GOSUB 3500
1130 IF XM=XT AND YM=YT THEN GOTO 9000
1140 F=F+1
1150 IF INT(F/5)=F/5 THEN GOSUB 6000
1160 IF X(YM)=XM AND INT(F/5)=F/5
     THEN GOTO 1100
1170 IF X(YM)=XM THEN GOTO 1120
```

```
1200 GOSUB 4500
1210 GOSUB 6000
1220 FOR Q=1 TO 10
1230 IF V(Q)=XM AND U(Q)=YM THEN GOSUB 4000
1240 NEXT Q
1250 IF RND(0)<=0.4 THEN GOTO 2000

1300 GOSUB 7000
1310 POSITION 1,19
1320 PRINT #6;"HOSTILE NATIVES"
1325 PRINT #6;" AHEAD!"
1330 FOR N=1 TO 3
1340 R=INT(RND(0)*3)
1350 IF YM+R>=B THEN R=0
1360 POSITION XM,YM+R
1370 PRINT #6;"Z"
1380 NEXT N
1390 YM=YM-3
1400 IF YM<=T+1 THEN YM=T+1
1410 XM=X(YM)
1420 MES=1
1430 POSITION XM,YM
1440 PRINT #6;"$";
1450 GOTO 1120

2000 GOSUB 7000
2010 GOSUB 3600
2020 POSITION 1,19
2030 PRINT #6;"FOLLOW LONG JOHN"
2035 PRINT #6;"SILVERS PARROT"
2040 YJ=YM+1
2050 IF YJ>P THEN YJ=P
2060 XJ=X(YJ)
2070 POSITION XJ,YJ
2080 PRINT #6;CHR$(6)
2090 MES=1
2100 GOTO 1120
```

```
3000 GOSUB 4600
3020 FOR X=L(T) TO R(T)
3030 POSITION X,T
3040 PRINT #6;CHR$(7)
3050 NEXT X
3060 FOR Y=T TO B
3070 POSITION L(Y),Y
3080 PRINT #6;CHR$(7)
3090 POSITION R(Y),Y
3100 PRINT #6;CHR$(7)
3120 NEXT Y
3130 FOR X=L(Y-1) TO R(Y-1)
3140 POSITION X,Y
3150 PRINT #6;CHR$(7)
3160 NEXT X
3300 FOR Y=T+1 TO P
3310 POSITION X(Y),Y
3320 PRINT #6;CHR$(3+128)
3330 NEXT Y
3340 POSITION 1,20
3350 PRINT #6;"X MARKS THE SPOT".
3360 POSITION X(P),P
3370 PRINT #6;"X";
3380 POSITION XM,YM
3390 PRINT #6;"$";
3400 RETURN

3500 GOSUB 4500
3510 GET #2,A
3520 IF A=-1 THEN GOTO 3510
3530 POSITION XM,YM
3540 PRINT #6;CHR$(7+128)
3550 IF A=72 THEN GOSUB 8000:RETURN
3560 IF A=43 THEN XM=XM-1:GOTO 3590
3570 IF A=42 THEN XM=XM+1:GOTO 3590
3580 IF A<>61 THEN GOTO 3500
3590 YM=YM+1
3600 POSITION XM,YM
3610 PRINT #6;"$"
3620 IF MES=0 THEN RETURN
3630 POSITION 0,19
3635 FOR I=1 TO 40
3640 PRINT #6;" ";
3650 NEXT I
3660 MES=0
3670 RETURN
```

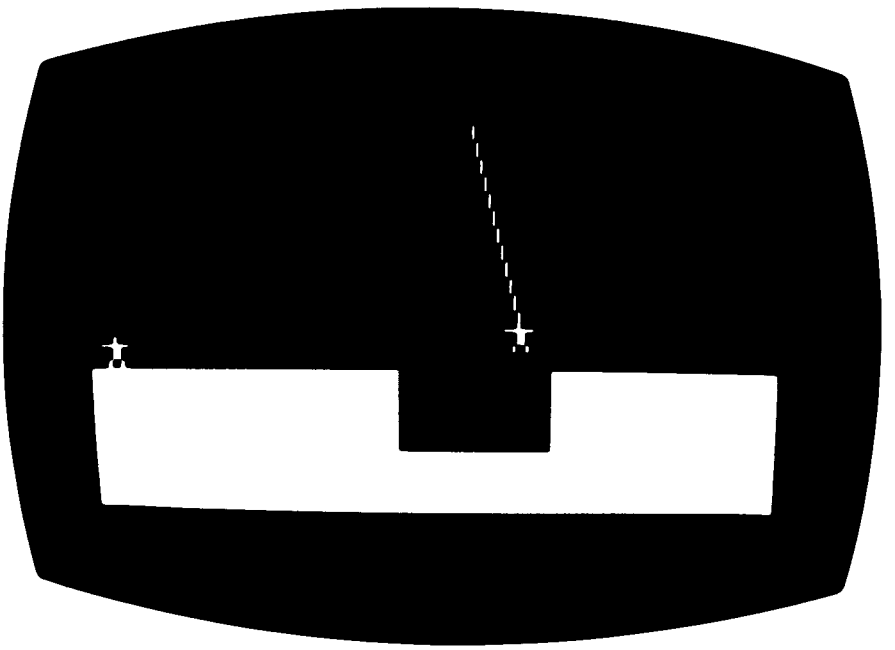
```
3700 FOR Q=1 TO 10
3710 POSITION V(Q),U(Q)
3720 PRINT #6;CHR$(32+64)
3730 NEXT Q
3740 XS=XS+1
3750 YS=YS+1
3760 RETURN

4000 GOSUB 7000
4010 POSITION XM,YM
4020 PRINT #6;CHR$(32+64)
4030 POSITION XM+1,YM+1
4050 FOR I=60 TO 40 STEP -2
4060 SOUND 0,80-I,10,8
4065 FOR D=1 TO 10:NEXT D
4070 NEXT I
4075 SOUND 0,0,0,0
4080 POSITION 1,19
4090 PRINT #6;"IN THE QUICKSAND"
4095 FOR Q=1 TO 500:NEXT Q
4100 IF RND(0)>0.5 THEN GRAPHICS 1:GOTO 9990
4120 PRINT #6;"YOU WERE PULLED OUT"
4130 FOR Q=1 TO 500:NEXT Q
4140 RETURN

4500 SOUND 0,80,10,8
4510 FOR D=1 TO 20
4520 NEXT D
4530 SOUND 0,0,0,0
4540 RETURN

4600 GRAPHICS 1+16
4610 POKE 756,CH/256
4620 SETCOLOR 4,0,0
4630 SETCOLOR 2,0,0
4640 SETCOLOR 0,3,8
4650 SETCOLOR 1,14,8
4660 SETCOLOR 3,12,8
4670 RETURN
```

```
5000 L=INT(RND(0)*3)+7
5010 T=INT(RND(0)*2)+2
5020 W=6+INT(RND(0)*3)
5030 B=INT(RND(0)*2)+17
5040 FOR Y=T TO B
5050 L(Y)=L
5060 R(Y)=L+W
5070 L=L-(SGN(10-Y)*INT(RND(0)*2))
5080 W=W+(SGN(10-Y)*INT(RND(0)*2))
5090 IF L(Y)<2 THEN L(Y)=2
5100 IF R(Y)>18 THEN R(Y)=18
5110 NEXT Y

5500 X(T+1)=L(T+1)+INT(RND(0)*3)
5510 K=T+2
5520 FOR P=K TO B-1-INT(RND(0)*3)
5530 X(P)=X(P-1)+INT(RND(0)*3)-1
5540 IF X(P)>=R(P) THEN X(P)=R(P)-1
5550 IF X(P)<=L(P) THEN X(P)=L(P)+1
5560 NEXT P
5570 P=P-1
5580 XT=X(P)
5590 YT=P
5700 FOR Q=1 TO 10
5710 D=INT(RND(0)*(P-T-2))+T+1
5720 U(Q)=D
5730 V(Q)=X(D)+(SGN(RND(0)-0.5)*2)
5740 IF V(Q)<=L(D) THEN V(Q)=V(Q)+3
5750 IF V(Q)>=R(D) THEN V(Q)=V(Q)-3
5760 NEXT Q
5770 RETURN
```

```
6000 GOSUB 4600
6020 R=INT(RND(0)*3)
6030 XS=XS+R
6040 YS=YS+R
6050 POSITION 18,18
6060 PRINT #6;CHR$(32+64)
6070 POSITION XS,YS
6080 PRINT #6;"[";
6090 POSITION XS,YS+1
6100 PRINT #6;"]";
6110 FOR Q=1 TO 200:NEXT Q
6120 IF YS<18 THEN RETURN
6121 GRAPHICS 1
6122 SETCOLOR 2,0,0
6130 POSITION 2,2
6140 PRINT #6;"THE PIRATES HAVE"
6150 POSITION 3,4
6160 PRINT #6;"LANDED YOU ARE"
6165 POSITION 3,6
6166 PRINT #6;"C A P T U R E D"
6170 FOR I=1 TO 100:NEXT I
6180 GOTO 9990

7000 GOSUB 4600
7020 FOR X=L(T) TO R(T)
7030 POSITION X,T-1
7040 PRINT #6;CHR$(7+128)
7050 NEXT X
7060 FOR Y=T TO B
7070 POSITION L(Y)-1,Y
7080 PRINT #6;CHR$(7+128)
7090 FOR X=L(Y) TO R(Y)
7100 POSITION X,Y
7110 PRINT #6;CHR$(7+128)
7120 NEXT X
7130 POSITION X,Y
7140 PRINT #6;CHR$(7+128)
7150 NEXT Y
7160 FOR X=L(Y-1) TO R(Y-1)
7170 POSITION X,Y
7180 PRINT #6;CHR$(7+128)
7190 NEXT X
7200 RETURN
```

```
8000 GOSUB 3000
8010 GOSUB 3700
8020 FOR Q=1 TO 100+INT(RND(0)*100)
8030 NEXT Q
8040 GOSUB 6000
8050 GOSUB 7000
8060 GOSUB 3600
8070 RETURN

9000 GRAPHICS 1
9005 FOR C=15 TO 1 STEP -1
9010 SETCOLOR 4,C,8:SETCOLOR 2,C,8
9020 PRINT CHR$(125)
9030 SOUND 0,C+66,10,8
9040 FOR Q=1 TO 20:NEXT Q
9045 NEXT C
9046 SOUND 0,0,0,0
9048 SETCOLOR 0,0,0
9050 POSITION 1,10
9060 PRINT #6;"YOU FOUND THE GOLD"

9990 POSITION 1,11
9991 PRINT #6;"ANOTHER GAME Y/N";
9992 INPUT A$
9993 IF A$="Y" THEN RUN
9999 GRAPHICS 0
```

8
Across the Ravine



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

How to play

This game is all a matter of timing – you have to judge when to start each run for the rope. It is vital to catch the rope on the downswing and each man can jump approximately his own width. You can wait

as long as you like before making a run and when you are ready, press any key to jump.

Subroutine structure

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

Programming details

There are several interesting features in this program. The first point to note is that the co-ordinates of the positions of the swinging rope are first calculated by subroutine 2000 and stored in two arrays, X and Y. These positions are then used repeatedly in the plotting of the swinging rope. This saves having to recalculate them each time they are needed and so speeds the whole program up. This technique can be applied to any situation where anything is moving rhythmically or periodically. The second interesting technique is in subroutine 4000 which plots the man using high resolution graphics. This means that he can appear anywhere on the screen and therefore move smoothly rather than just be printed at set positions which would result in jerky movement.

Program

```
10 REM ACROSS THE RAVINE
```

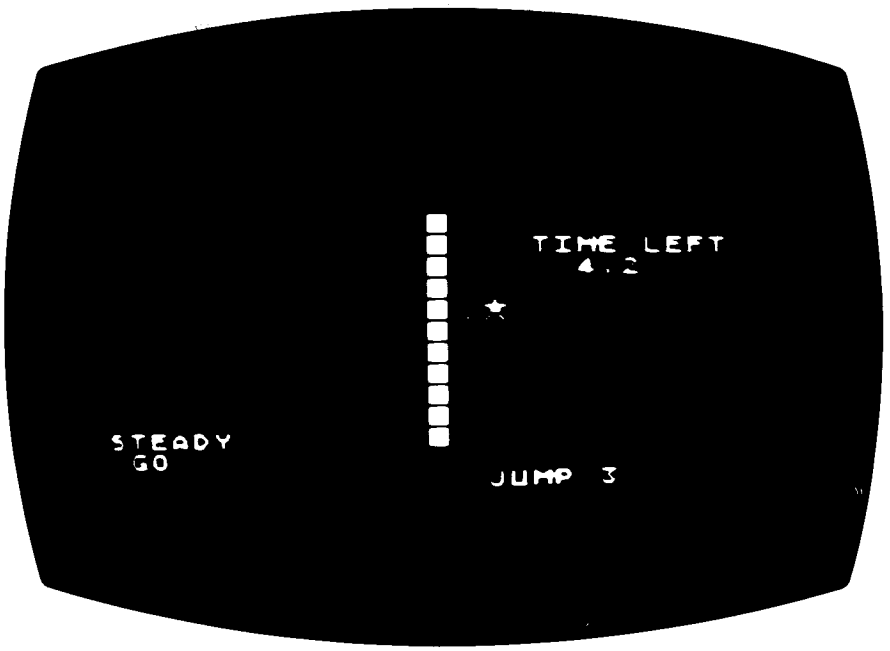
```
  20 DIM X(16)
  30 DIM Y(16)
  35 DIM A$(1)
  40 GOSUB 3000
  50 GOSUB 2000
  60 GOSUB 7500
  70 GOSUB 6000
  75 POKE 764,255
  80 GOSUB 1500
  85 POKE 764,255
  90 IF MEN<=0 THEN MEN=0:POKE 764,255:
     GOTO 9000
 100 GOTO 80

1500 FOR T=1+R TO N-R
1510 COLOR 2:PLOT 85,0
1520 DRAWTO 85+X(T),Y(T)
1525 S=1:COLOR 1:GOSUB 4000
1530 COLOR 0:PLOT 85,0
1540 DRAWTO 85+X(T),Y(T)
1550 IF J=1 THEN S=2:COLOR 0:GOSUB 4000
1560 NEXT T
1565 IF MEN=0 THEN RETURN
1570 IF C=1 THEN GOTO 1680
1600 FOR T=N-R TO 1+R STEP -1
1610 COLOR 2:PLOT 85,0
1620 DRAWTO 85+X(T),Y(T)
1625 S=3:COLOR 1:GOSUB 4000
1630 COLOR 0:PLOT 85,0
1640 DRAWTO 85+X(T),Y(T)
1650 IF J=1 THEN S=4:COLOR 0:GOSUB 4000
1660 NEXT T
1670 RETURN
1680 ACROSS=ACROSS+1
1690 DX=(ACROSS-1)*10+5
1700 DY=52
1710 COLOR 1:GOSUB 4020
1720 C=0
1730 MEN=MEN-1
1735 IF MEN=0 THEN GOTO 6050
1740 GOSUB 6000
1750 RETURN
```

```
2000 N=0
2010 FOR T=-PI/6 TO PI/6 STEP 0.1
2020 N=N+1
2030 X(N)=-INT(50*SIN(T))
2040 Y(N)=INT(50*COS(T))
2050 NEXT T
2060 RETURN

3000 GRAPHICS 7
3020 COLOR 2
3030 PLOT 159,79
3040 DRAWTO 159,60
3050 DRAWTO 105,60
3060 POSITION 105,79
3070 POKE 765,2
3080 XIO 18,#6,0,0,"S:"
3090 PLOT 70,80
3100 DRAWTO 70,60
3110 DRAWTO 0,60
3120 POSITION 0,80
3130 XIO 18,#6,0,0,"S:"
3140 J=0:C=0
3150 PI=4*ATN(1)
3999 RETURN

4000 IF C=0 THEN GOTO 5000
4010 DX=85+X(T):DY=Y(T)
4020 PLOT DX,DY
4030 PLOT DX,DY+1
4040 PLOT DX-3,DY+2
4050 DRAWTO DX+3,DY+2
4060 PLOT DX,DY+3
4070 DRAWTO DX+2,DY+7
4080 PLOT DX+1,DY+3
4090 DRAWTO DX-1,DY+7
4100 RETURN
```

```
5000 A=PEEK(764):IF A=255 AND J=0 THEN
      FOR I=1 TO 50:NEXT I:RETURN
5005 POKE 764,255
5006 IF MEN=0 THEN RETURN
5010 J=1
5020 DY=MY
5030 DX=MX
5040 COLOR 0:GOSUB 4020
5050 MX=MX-10
5060 DY=MY
5070 DX=MX
5080 COLOR 1:GOSUB 4020
5090 IF ABS(MX-85-X(T))<10 AND S=2 THEN
     C=1:COLOR 0:GOTO 4020
5200 IF MX<105 THEN GOTO 7000
5210 RETURN

6000 MY=52
6010 MX=140
6020 DY=MY
6030 DX=MX
6040 J=0
6045 COLOR 1
6050 GOSUB 4020
6055 PRINT
6060 PRINT "MEN LEFT","ACROSS","LOST"
6070 PRINT MEN,ACROSS,LOST
6080 RETURN

7000 COLOR 0:GOSUB 4020
7010 MY=MY+2
7020 DY=MY
7030 COLOR 1:GOSUB 4020
7040 SOUND 0,10,MY/2,8
7050 COLOR 0:GOSUB 4020
7060 IF MY<78 THEN GOTO 7010
7070 SOUND 0,80,10,8:FOR I=1 TO 50:NEXT I:
     SOUND 0,0,0,0
7080 LOST=LOST+1
7090 C=0
7100 J=0
7110 MEN=MEN-1
7120 IF MEN=0 THEN POKE 764,255:GOTO 6060
7130 GOSUB 6000
7140 RETURN
```

```
7500 MEN=5
7510 LOST=0
7520 ACROSS=0
7530 J=0
7540 C=0
7550 R=INT(RND(0)*4)
7560 FOR I=52 TO 59
7570 FOR Q=0 TO 60
7580 COLOR 0:PLOT Q,I
7590 NEXT Q
7595 NEXT I
7596 RETURN

9000 PRINT "YOU LOST ";LOST;"  ANOTHER
     GAME ";:INPUT A$
9010 IF A$="Y" THEN GOTO 60
9020 GRAPHICS 0
```

9
Commando Jump



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

How to play

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

away you have to press the up and down arrow keys alternately to get the man over the wall. The man will climb one brick higher for every five pairs of key presses (but see note in 'Scope for Alteration') – so the more rapidly you press the more quickly he will climb. If you keep your finger on a key you will hear a bleeping sound this is because only complete key presses, i.e. press and release, count. If the man is not over within the time limit he will slither back down the wall and you have another try. In all you are given ten attempts. Even if you are very slow off the mark, do press a key – until you do so, you cannot move on to the next try. If you hit a key just before the "GO" signal, the computer will accuse you of cheating and you will lose that turn.

Typing tips

You'll find the Atari's editing facilities useful in entering lines which are similar to one another such as those in lines 8100 to 8170.

Subroutine structure

20	Main play loop
1000	Countdown
1500	Cheat routine
2000	Prints wall
2060	Jump logic
2500	Fall down wall
2600	Prints man over wall
7000	Zeroes time
7100	Gets time
8000	Sets up game and defines brick graphic
8100	Defines man graphics character
8200	Changes to RAM character set and supresses cursor
9000	Win/lose messages
9600	End of game

Programming details

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

the Atari's frame counter is used as a reaction timer and a countdown device. The clock is zeroed by routine 7000 and read by routine 7100. You could use these routines in programs of your own. Another interesting point to note is the way that a test for a key press is made. Memory location 764 holds the internal code corresponding to the last key pressed. If no key has been pressed in contains 255 and to reset the location 255 is POKEd into it after a key press has been detected. Notice the use of line 8210 to remove the Atari's normal cursor so that it does not interfere in this game.

Scope for alteration

If you are using an Atari 400 you may find this game impossibly difficult because you cannot press the keys quickly enough. In this case try increasing the value in line 2210 which govern the number of pairs of presses needed to climb by one brick. It is currently set at 0.2 which means that the man moves up after five pairs. If you change this value to 0.5 he will move after two pairs of presses and so on.

Program

```
10 REM COMMANDO JUMP

20 GOSUB 8000
30 GOSUB 2000
50 GOTO 9000
```

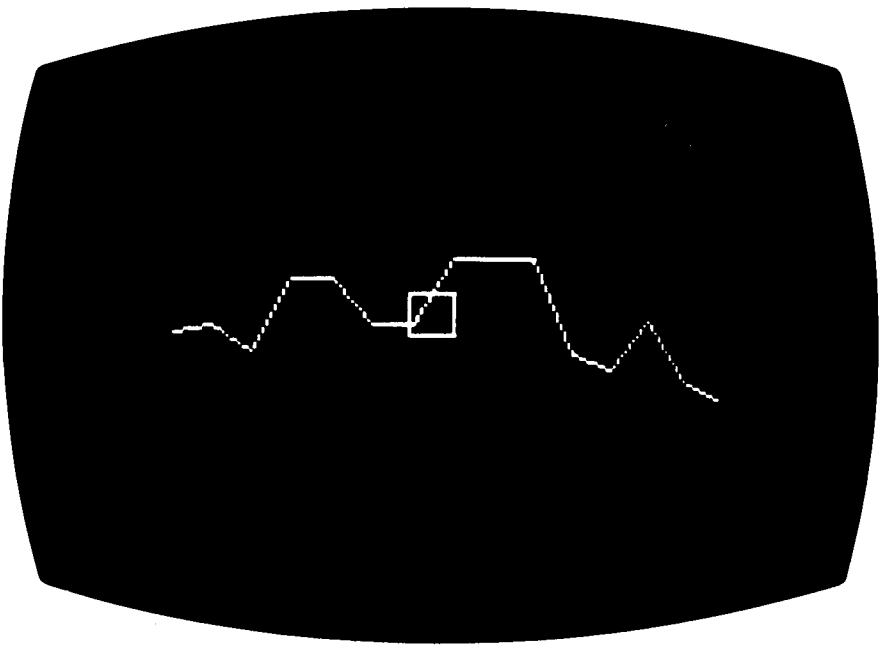
```
1000 POSITION 1,18:
     PRINT #6;"      ":REM SIX SPACES
1004 PRINT #6;"  ":REM TWO SPACES
1005 POSITION 1,18:PRINT #6;"READY";
1010 FOR I=1 TO RND(0)*200+200
1020 NEXT I
1025 POKE 764,255
1030 POSITION 1,18:PRINT #6;"STEADY"
1040 FOR I=1 TO RND(0)*200+200
1050 NEXT I
1060 IF PEEK(764)<>255 THEN GOTO 1500
1070 GOSUB 7000
1080 PRINT #6;"GO"
1085 SOUND 0,100,10,10
1090 IF PEEK(764)=255 THEN GOTO 1090
1095 SOUND 0,0,0,0
1100 GOSUB 7100
1110 RETURN

1500 POSITION 1,10:PRINT #6;"CHEAT"
1510 SOUND 0,150,2,8
1515 FOR D=1 TO 1000:NEXT D:
     SOUND 0,0,0,0
1520 POSITION 1,10:
     PRINT #6;"     ":REM 5 SPACES
1530 T=5
1540 POKE 764,255
1550 RETURN

2000 PRINT CHR$(125):JUMP=1
2010 H=10+INT(RND(0)*5)
2020 FOR I=18 TO 19-H STEP -1
2030 POSITION 15,I:PRINT #6;"%"
2040 NEXT I
2050 POSITION 15,18-H:PRINT #6;"%"
```

```
2060 POSITION 18,20:PRINT #6;"JUMP ";JUMP
2070 POSITION 18,18:PRINT #6;"$"
2100 GOSUB 1000
2110 FOR I=18 TO 18-H+INT(T*20) STEP -1
2120 POSITION 18,I:PRINT #6;" "
2130 POSITION 18,I-1:PRINT #6;"$"
2140 SOUND 0,80+I,10,10
2145 FOR D=1 TO 10:NEXT D:SOUND 0,0,0,0
2150 NEXT I
2160 J=I:L=INT(I)
2170 GOSUB 7100
2175 IF T>5 THEN GOTO 2500
2180 POSITION 20,9:PRINT #6;"TIME LEFT":
     POSITION 22,10:PRINT #6;
     INT((5-T)*10)/10;"  "
2190 IF PEEK(764)=255 THEN GOTO 2170
2195 POKE 764,255
2200 POSITION 18,L:PRINT #6;" "
2210 J=J-0.2
2215 L=INT(J)
2220 POSITION 18,L:PRINT #6;"$"
2230 IF L<=17-H THEN POSITION 18,L+1:
     PRINT #6;" ":GOTO 2600
2340 IF PEEK(764)<>15 THEN GOTO 2340
2350 GOTO 2170

2500 FOR I=L TO 18
2510 POSITION 18,I-1:PRINT #6;" "
2520 POSITION 18,I:PRINT #6;"$"
2530 SOUND 0,80+I,10,10
2535 FOR D=1 TO 5:NEXT D:SOUND 0,0,0,0
2540 NEXT I
2550 JUMP=JUMP+1
2560 POSITION 25,10:
     PRINT #6;"    ":REM 4 SPACES
2570 IF JUMP<=10 THEN GOTO 2060
2580 RETURN
```

```
2600 FOR I=18 TO 10 STEP -1
2610 POSITION I+1,L:PRINT #6;" "
2620 POSITION I,L:PRINT #6;"$"
2630 FOR K=1 TO 10:NEXT K
2640 NEXT I
2650 FOR I=L TO 18
2660 POSITION 10,I-1:PRINT #6;" "
2670 POSITION 10,I:PRINT #6;"$"
2675 SOUND 0,80+I*2,10,10
2676 FOR D=1 TO 20:NEXT D:SOUND 0,0,0,0
2690 NEXT I
2700 RETURN

7000 POKE 18,0
7010 POKE 19,0
7020 POKE 20,0
7030 RETURN

7100 T=(PEEK(20)+256*PEEK(19))/50
7110 RETURN

8000 GRAPHICS 1+16
8005 POKE 764,255
8010 CH=(PEEK(106)-8)*256
8015 PRINT "ONE MOMENT WHILE I SET UP THE GAME"
8020 CHORG=(PEEK(756)*256)
8030 FOR I=0 TO 511
8040 POKE CH+I,PEEK(CHORG+I)
8050 NEXT I
8060 POKE CH+(ASC("%")-32)*8+0,0
8070 FOR I=1 TO 7
8080 POKE CH+(ASC("%")-32)*8+I,255
8090 NEXT I

8100 POKE CH+(ASC("$")-32)*8+0,24
8110 POKE CH+(ASC("$")-32)*8+1,24
8120 POKE CH+(ASC("$")-32)*8+2,255
8130 POKE CH+(ASC("$")-32)*8+3,60
8140 POKE CH+(ASC("$")-32)*8+4,60
8150 POKE CH+(ASC("$")-32)*8+5,36
8160 POKE CH+(ASC("$")-32)*8+6,102
8170 POKE CH+(ASC("$")-32)*8+7,195
```

```
8200 POKE 756,CH/256
8210 POKE 752,1
8220 T=0
8230 DIM A$(10)
8240 RETURN

9000 IF JUMP<=10 THEN GOTO 9500
9010 POSITION 10,0:PRINT #6;"YOU FAILED!!!"
9020 GOTO 9600
9500 POSITION 10,0:
     PRINT #6;"YOU TOOK ";JUMP;" JUMPS TO"
9510 POSITION 14,1:PRINT #6;"CLEAR THE WALL"

9600 POKE 764,255
9610 POSITION 15,2:
     PRINT "ANOTHER GAME ";:INPUT A$
9620 IF A$="Y" THEN GOTO 30
9640 GRAPHICS 0
```

10
Guideline



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

How to play

At the beginning of the game you have to select the level of difficulty for the game. This determines the size of the square that has to be guided along the wire. Once the square appears there is a short delay

and then it automatically starts to move left. Press the up and down arrow keys to change direction. Keeping your finger down on an arrow key will keep the square moving in the same direction.

Subroutine structure

15	Main play loop
110	End of game
1000	Initialises variables and prints title frame
2000	Plots wire
5000	Moves and plots square and re-plots wire
5570	Checks for being on target
6000	Sound routine

Programming details

High resolution graphics are used in this program to give the smooth movement needed to test the players' skill. Subroutine 2000 is responsible for plotting the line for the wire and the square is constructed by a collection of DRAWTO statements at the beginning of subroutine 5000. The LOCATE function is used in subroutine 5570 to test whether the square ring is actually on target around the wire.

Program

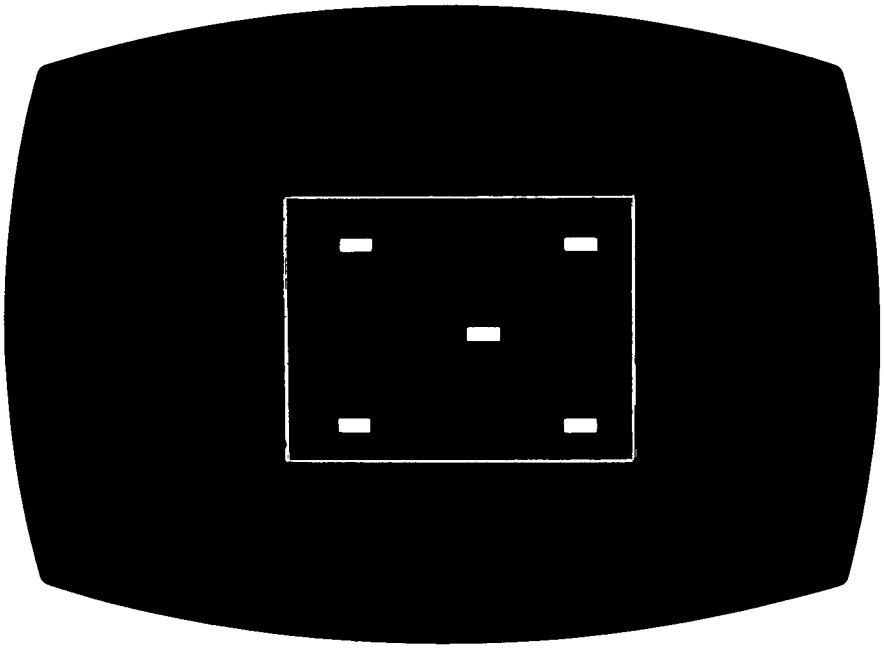
```
 10 REM GUIDELINE

 15 DIM A$(10),C(15)
 20 GOSUB 1000
 30 GOSUB 2000
 40 FOR Z=1 TO 100:NEXT Z
 60 GOSUB 6000
 70 Y=35:X=10
 80 B=Y:R=12-DF
 90 R2=INT(R/2):V=2
100 GOSUB 5000
```

```
 110 PRINT "YOU WERE ON TARGET ";
     INT(HIT/(HIT+MISS)*10000)/100;"% OF
     THE TIME"
 120 PRINT "ANOTHER GAME ";:INPUT A$
 130 IF A$="Y" THEN RUN
 140 IF A$<>"N" THEN GO TO 120
 160 PRINT CHR$(125)
 170 STOP

1000 X=10
1010 Y=40
1020 D=30
1030 P=0
1070 PRINT CHR$(125)
1080 POSITION 10,2:PRINT "G U I D E L I N E"
1090 POSITION 5,5:PRINT "You must guide a
     ring along the"
1100 POSITION 5,6:PRINT "wavy 'wire' using
     the up and"
1110 POSITION 5,7:PRINT "down arrow keys"
1120 POSITION 5,10
1130 PRINT "You will be marked on how"
1140 POSITION 5,11:PRINT "accurate you are"
1150 POSITION 5,21:PRINT "SELECT THE
     DIFFICULTY LEVEL"
1160 POSITION 5,22:PRINT "FROM 1-EASY TO
     5-DIFFICULT";:INPUT DF
1170 IF DF<1 OR DF>5 THEN GO TO 1150
1180 PRINT CHR$(125)
1190 GRAPHICS 7+16:COLOR 1
1210 POKE 752,1
1220 COLOR 1
1999 RETURN
```

```
2000 PLOT X,Y
2005 C(0)=Y
2010 FOR I=1 TO 14
2020 R=D-INT(RND(0)*2*D)
2030 Y=Y+R:X=X+10
2040 IF Y>60 THEN Y=Y-R
2050 IF Y<20 THEN Y=Y-R
2060 DRAWTO X,Y
2065 C(I)=Y
2070 NEXT I
2080 HIT=0
2090 MISS=0
2100 RETURN

5000 PLOT X-R2,Y-R2
5010 DRAWTO X-R2,R+Y-R2
5020 DRAWTO X-R2+R,R+Y-R2
5030 DRAWTO X-R2+R,Y-R2
5040 DRAWTO X-R2,Y-R2
5050 PLOT 10,C(0)
5060 FOR I=1 TO 14
5070 DRAWTO 10+I*10,C(I)
5080 NEXT I
5100 M=PEEK(764)
5105 POKE 764,255
5110 IF M=14 AND Y-R2>2*V THEN B=B-2*V
5120 IF M=15 AND Y+R<80-2*V THEN B=B+2*V
5500 COLOR 0
5510 PLOT X-R2,Y-R2
5520 DRAWTO X-R2,R+Y-R2
5530 DRAWTO X-R2+R,R+Y-R2
5540 DRAWTO X-R2+R,Y-R2
5550 DRAWTO X-R2,Y-R2
5560 X=X+V
5561 COLOR 1
5565 IF X>145 THEN RETURN
```

```
5570 Y=B
5575 H=0
5580 FOR I=-R2 TO R2
5585 LOCATE X,Y+I,Q
5590 IF Q<>0 THEN HIT=HIT+1:FOR Z=1 TO 20:
     NEXT Z:I=R-1:H=1
5600 NEXT I
5605 IF H=1 THEN GOTO 5000
5610 MISS=MISS+1
5700 GOSUB 6000
5710 GOTO 5000

6000 SOUND 0,80,10,8
6010 FOR Q=1 TO 10
6020 NEXT Q
6030 SOUND 0,0,0,0
6040 RETURN
```

11
Atari Dice



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

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

Of course, your Atari Dice can become the centre of a game. You can devise gambling games to play against the computer or against

other people. After all, dice have been around for thousands of years so there must be plenty of ideas about how to use them.

However you choose to use the program, it will give you a large clear display on the TV screen – colourful too if you run it on a colour set – and you'll be impressed by the way it even sounds like a dice rolling over a wooden surface and actually slows down before it comes to a final halt.

How to use the program

Using this program is simplicity itself. Type RUN and, when your Atari prompts, just press any key in order to start the dice rolling. It then carries on rolling for a random number of turns and slows down and stops when it is ready. When you've finished with the program press the BREAK key to stop it running.

Subroutine structure

10	Main play loop
1000	Prints and unprints dots
2000	Draws yellow square for dice
5000	Sets colours and opens keyboard as input device
6000	Emits click sound
7000	Transfers ROM characters to RAM
7060	Defines dot graphics character
7090	Initialises variables
7200	Selects RAM characters

Programming details

The essence of a dice program is in generating random numbers. In fact, this program uses random numbers in two ways. Firstly, randomness is used in the conventional way, to determine which face of the dice will show at the next turn – this is done in line 140, which uses the RND function to select 'R', a number between one and six. This information is then used in the printing subroutine (starting at line 1000). The program goes to one of the six line numbers 1100, 1200, 1300, 1400, 1500, 1600, according to the value of 'R'. At 1100 one dot is printed, at 1200 two dots are printed and so on.

The other use of the random number generator is to give the Atari Dice realistic suspense. When a human throws a dice it will turn just a few times or quite a number of times and, before it actually stops, it will slow down. Your Atari Dice copies both the features by incorporating lines 90 to 110. Another random number, 'T', with a value between 5 and 15 is selected. This governs the number of turns the dice makes and each time it rolls over the pause before the dots reappear lengthens.

Scope for improvement

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

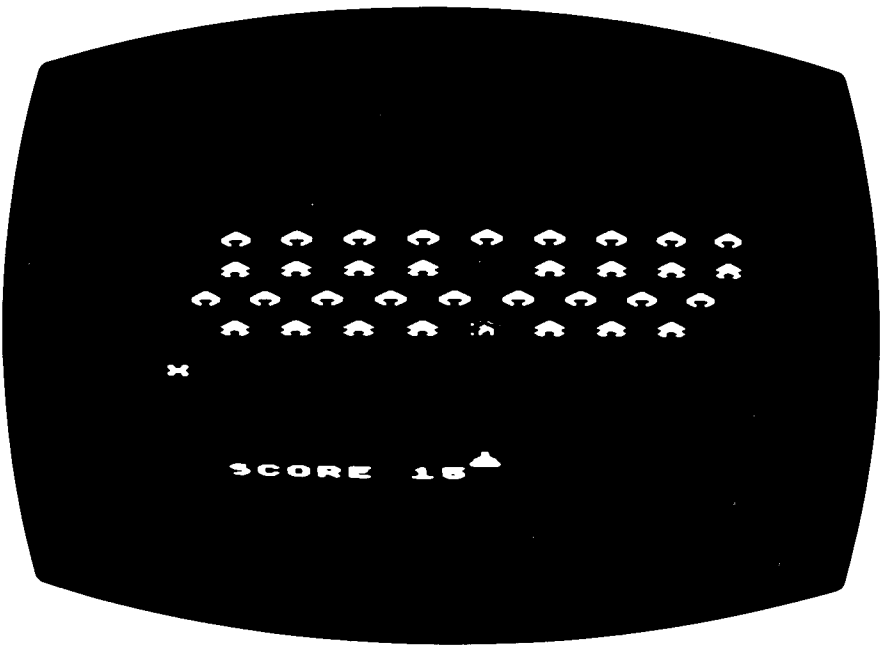
Program

```
  5 REM ATARI DICE

 10 GOSUB 7000
 20 GOSUB 5000
 30 GOSUB 2000
 40 POSITION 0,23
 50 PRINT #6;"  PRESS ANY KEY"
 60 GET #2,B
 90 T=RND(0)*5+5
100 FOR I=1 TO T
110 FOR Q=1 TO 20*I:NEXT Q
120 D$=B$
130 GOSUB 1000
140 R=INT(RND(0)*6)+1
150 D$=A$
160 GOSUB 1000
170 NEXT I
180 GOTO 60
```

```
1000 GOSUB 6000
1010 GOTO 1000+R*100
1100 POSITION 8,11
1110 PRINT #6;D$
1120 RETURN
1200 POSITION 5,5
1210 PRINT #6;D$
1220 POSITION 12,17
1230 PRINT #6;D$
1240 RETURN
1300 GOSUB 1100
1310 GOTO 1200
1400 POSITION 5,17
1410 PRINT #6;D$
1420 POSITION 12,5
1430 PRINT #6;D$
1440 GOTO 1200
1500 GOSUB 1400
1510 GOTO 1100
1600 POSITION 5,11
1610 PRINT #6;D$
1620 POSITION 12,11
1630 PRINT #6;D$
1640 GOTO 1400

2000 FOR I=1 TO 16
2010 POSITION 4,I+2
2020 PRINT #6;B$
2030 NEXT I
2040 R=1
2050 D$=A$
2060 GOTO 1000

5000 REM COLOURS
5010 SETCOLOR 0,0,0
5020 SETCOLOR 1,12,12
5030 SETCOLOR 2,3,6
5040 SETCOLOR 4,8,10
5050 OPEN #2,4,0,"K"
5060 RETURN

6000 SOUND 0,100,12,8
6010 SOUND 0,0,0,0
6020 RETURN
```

```
7000 GRAPHICS 1+16
7010 CH=(PEEK(106)-8)*256
7020 CHORG=(PEEK(756)*256)
7030 FOR I=0 TO 511
7040 POKE CH+I,PEEK(CHORG+I)
7050 NEXT I

7060 FOR I=0 TO 7
7070 POKE CH+(ASC("$")-32)*8+I,255
7080 NEXT I

7090 DIM A$(1),B$(10),D$(1)
7100 A$=CHR$(164)
7110 FOR I=1 TO 10
7120 B$(I)=CHR$(4)
7130 NEXT I

7200 POKE 756,CH/256
7999 RETURN
```

12
Alien Invaders



Two types of alien ships are heading towards earth and you have to defend civilisation as we know it. Your task is daunting. You have to ensure that none of the aliens get within firing range of earth – and they are coming at an alarming speed. The point of no return is marked with an 'X' on the left of the screen. Once any of the advancing ships pass this point the game is over – you will have lost. Your only hope is to wipe out the aliens with your missiles. Every missile that hits its target increases your chance of saving the world. If you play this game on a colour TV you'll see that the ships are red and yellow.

How to play

You can move your missile launcher to left and right using the appropriate arrow keys. Press the up arrow key to fire a missile. You

score points for every alien you hit, the ones further away from you counting for more points than the nearer ones. The game is over when you have destroyed all the invaders or when the nearest remaining ones reach the point marked by the 'X'.

Subroutine structure

20	Sets up screen
140	Initialises variables
220	Initialises strings
280	Main play loop
580	End of game
740	Moves and fires missile
890	Fire routine
1250	Tests whether alien hit
1360	Moves invaders to left
1380	Moves invaders to right
2000	Defines graphics character for alien ship one
2100	Defines graphics character for missile launcher
2200	Defines graphics character for alien ship two
2300	Defines graphics character for explosion
2400	Change to RAM character set
2500	Sound routine
2600	Move invaders arms

Programming details

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

The way that the four rows of ships move is interesting as it increases the difficulty of the game. The front row moves to the right, while the second row moves to the left and the back two rows

oscillate from side to side. Line 1260 detects when an alien invader is hit. When this happens its position in the string is replaced by a blank space. This manipulation is carried out in line 1270 which divides the string at the appropriate point, inserts a space in place of the destroyed ship and then rejoins the two halves of the string. You can learn a lot about the way that you can handle strings in Atari BASIC by studying lines 1360 to 1398.

Scope for improvement

You might like to add a routine to make the aliens shoot back at random so that the missile launcher faced the added problem of dodging enemy fire.

Program

```
 10 REM ALIEN INVADERS

 20 GRAPHICS 1+16
 30 OPEN #2,4,0,"K"
 40 CH=(PEEK(106)-8)*256
 50 CHORG=(PEEK(756)*256)
 60 FOR I=0 TO 511
 70 POKE CH+I,PEEK(CHORG+I)
 80 NEXT I
 90 SETCOLOR 0,3,6
100 SETCOLOR 1,0,14
110 SETCOLOR 2,13,8
130 GOSUB 2000

140 DIM A$(20):DIM B$(20)
145 DIM C$(20):DIM D$(20)
146 DIM E$(20):DIM Q$(20)
148 DIM T$(20)
150 Y=1
160 XL=10
170 YM=0
180 T=0
190 S=0
200 J=0
210 K=0
```

```
220 FOR I=0 TO 8
225 A$(1+I*2)=CHR$(164)
226 A$(2+I*2)=" "
228 NEXT I
230 B$=" & & & & & & & & &"
240 C$=A$
250 D$=B$
260 E$="                    ":REM 18 SPACES

280 POSITION 0,14:PRINT #6;"X";
290 J= NOT J
300 IF K=1 THEN GOTO 600
310 IF T>30+INT(RND(0)*15) THEN
    POSITION 0,Y:PRINT #6;E$:Y=Y+2:T=0:
    GOSUB 2500
315 Q$=A$
320 IF J THEN GOSUB 1360:A$=Q$:GOTO 340
330 GOSUB 1380:A$=Q$
340 POSITION 1,Y:PRINT #6;A$
350 SOUND 1,10,50+Y*8,10
360 GOSUB 740
365 Q$=B$
370 IF J THEN GOSUB 1360:B$=Q$:GOTO 390
380 GOSUB 1380:B$=Q$
390 POSITION 1,Y+2:PRINT #6;B$
400 SOUND 1,10,129-Y*8,10
410 GOSUB 740
420 Q$=C$:GOSUB 1360:C$=Q$
440 POSITION 1,Y+4:PRINT #6;C$
450 SOUND 1,5,121-Y*8,5:SOUND 0,0,0,0
460 GOSUB 740
470 Q$=D$:GOSUB 1380:D$=Q$
490 POSITION 1,Y+6:PRINT #6;D$
500 SOUND 1,10,129-Y*8,10
510 GOSUB 740
520 IF Y>8 AND D$<>E$ THEN GOTO 580
530 IF Y>10 AND C$<>E$ THEN GOTO 580
540 IF Y>12 AND B$<>E$ THEN GOTO 580
550 IF Y>14 AND A$<>E$ THEN GOTO 580
560 T=T+1
565 GOSUB 2600
570 GOTO 280
```

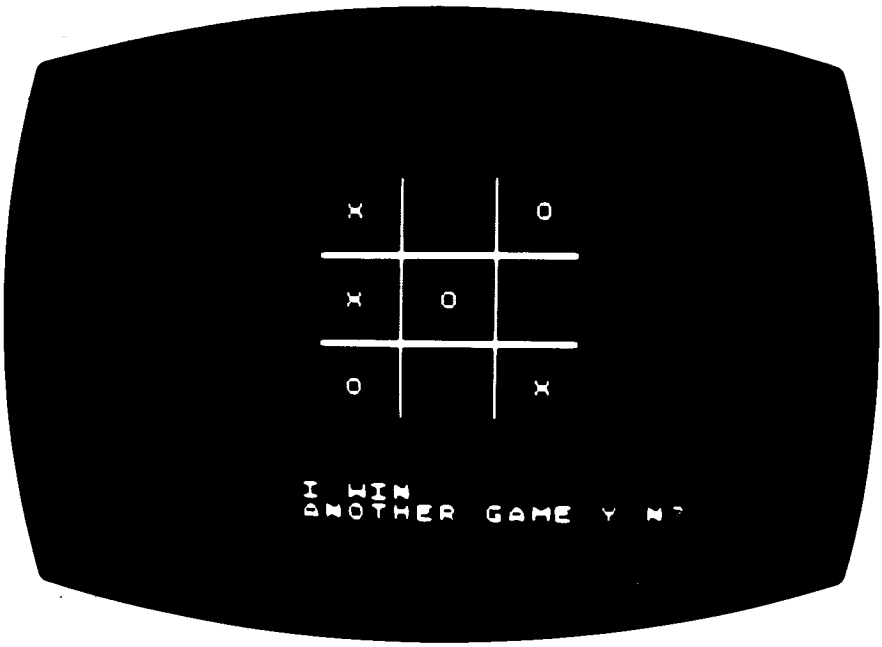
```
580 POSITION 1,19:
    PRINT #6;" THEY GOT YOU!"
590 GOTO 610
600 POSITION 1,19:PRINT #6;
    "WELL DONE YOU SAVED THE WORLD!"
610 SOUND 0,1,80,10
620 FOR D=1 TO 1000:NEXT D
650 IF K=0 THEN SOUND 0,10,4,10
655 SOUND 0,0,0,0
660 PRINT "ANOTHER GAME Y/N":INPUT A$
680 IF A$="Y" THEN RUN
730 END

740 A=PEEK(764)
750 POKE 764,255
770 T=T+1
790 POSITION XL,20:PRINT #6;CHR$(5)
800 IF A=255 THEN RETURN
810 POSITION XL,20:PRINT #6;" "
820 IF A=6 AND XL>1 THEN XL=XL-1
830 IF A=7 AND XL<17 THEN XL=XL+1
850 POSITION XL,20:PRINT #6;CHR$(5)
860 IF A=14 THEN GOSUB 890
880 RETURN
```

```
 890 REM FIRE
 900 FOR M=18 TO Y+6 STEP -1
 910 POSITION XL,M:PRINT #6;":";
 920 POSITION XL,M+1:PRINT #6;" "
 930 NEXT M
 940 POSITION XL,M+1:PRINT #6;" "
 950 F=0
 960 Q$=D$
 970 R=6
 980 GOSUB 1250
 990 D$=Q$
1000 IF F=1 THEN GOTO 1220
1020 POSITION XL,Y+5:PRINT #6;".";:
     POSITION XL,Y+5:PRINT #6;" ";
1025 POSITION XL,Y+4:PRINT #6;".";:
     POSITION XL,Y+4:PRINT #6;" ";
1030 Q$=C$
1040 R=4
1050 GOSUB 1250
1060 C$=Q$
1070 IF F=1 THEN GOTO 1220
1080 POSITION XL,Y+3:PRINT #6;".";:
     POSITION XL,Y+3:PRINT #6;" ";
1090 POSITION XL,Y+2:PRINT #6;".";
     POSITION XL,Y+2:PRINT #6;" "
1100 Q$=B$
1110 R=2
1130 GOSUB 1250
1140 B$=Q$
1150 IF F=1 THEN GOTO 1220
1160 POSITION XL,Y+1:PRINT #6;".";:
     POSITION XL,Y+1:PRINT #6;" ";
1165 POSITION XL,Y:PRINT #6;".";:
     POSITION XL,Y:PRINT #6;" ";
1170 Q$=A$
1180 R=0
1200 GOSUB 1250
1210 A$=Q$
1220 IF A$=E$ AND B$=E$ AND C$=E$ AND D$=E$
     THEN K=1
1230 IF Q$=E$ THEN POSITION 1,Y:
     PRINT #6;E$:Y=Y+2
1240 RETURN
```

```
1250 REM HIT
1260 IF Q$(XL,XL)=" " THEN RETURN
1270 Q$(XL,XL)=" "
1280 F=1
1290 S=S+10-Y
1310 POSITION XL,Y+R:PRINT #6;"@"
1320 SOUND 1,15,4,3:SOUND 0,0,0,0
1330 POSITION 2,21:PRINT #6;"SCORE ";S;"    "
1340 T=T-RND(0)*3
1350 RETURN

1360 QL=LEN(Q$):Q$(QL+1)=Q$(1,1):
     Q$=Q$(2,QL+1)
1370 RETURN

1380 T$=Q$(LEN(Q$))
1390 T$(2)=Q$(1,LEN(Q$)-1)
1395 Q$=T$
1398 RETURN

2000 POKE CH+(ASC("$")-32)*8+0,24
2010 POKE CH+(ASC("$")-32)*8+1,60
2020 POKE CH+(ASC("$")-32)*8+2,126
2030 POKE CH+(ASC("$")-32)*8+3,255
2040 POKE CH+(ASC("$")-32)*8+4,195
2050 POKE CH+(ASC("$")-32)*8+5,195
2060 POKE CH+(ASC("$")-32)*8+6,102
2070 POKE CH+(ASC("$")-32)*8+7,36

2100 POKE CH+(ASC("%")-32)*8+0,24
2110 POKE CH+(ASC("%")-32)*8+1,24
2120 POKE CH+(ASC("%")-32)*8+2,24
2130 POKE CH+(ASC("%")-32)*8+3,60
2140 POKE CH+(ASC("%")-32)*8+4,126
2150 POKE CH+(ASC("%")-32)*8+5,126
2160 POKE CH+(ASC("%")-32)*8+6,255
2170 POKE CH+(ASC("%")-32)*8+7,255
```

```
2200 POKE CH+(ASC("&")-32)*8+0,24
2210 POKE CH+(ASC("&")-32)*8+1,60
2220 POKE CH+(ASC("&")-32)*8+2,126
2230 POKE CH+(ASC("&")-32)*8+3,255
2240 POKE CH+(ASC("&")-32)*8+4,60
2250 POKE CH+(ASC("&")-32)*8+5,102
2260 POKE CH+(ASC("&")-32)*8+6,195
2270 POKE CH+(ASC("&")-32)*8+7,102

2300 POKE CH+(ASC("@")-32)*8+0,40
2310 POKE CH+(ASC("@")-32)*8+1,132
2320 POKE CH+(ASC("@")-32)*8+2,145
2330 POKE CH+(ASC("@")-32)*8+3,40
2340 POKE CH+(ASC("@")-32)*8+4,28
2350 POKE CH+(ASC("@")-32)*8+5,52
2360 POKE CH+(ASC("@")-32)*8+6,164
2370 POKE CH+(ASC("@")-32)*8+7,164

2400 POKE 756,CH/256
2410 POKE 752,1
2420 FLAG=0
2499 RETURN

2500 SOUND 1,4,50,8:SOUND 0,0,0,0
2510 RETURN

2600 IF FLAG=1 THEN GOTO 2650
2610 POKE CH+(ASC("$")-32)*8+7,129
2620 POKE CH+(ASC("&")-32)*8+7,195
2630 FLAG=1
2640 RETURN
2650 POKE CH+(ASC("$")-32)*8+7,36
2660 POKE CH+(ASC("&")-32)*8+7,102
2670 FLAG=0
2680 RETURN
```

13
Noughts and Crosses



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

How to play

This game is played on a simple three-by-three grid in the traditional way. You have the 'X' and play first. To make your move you have to specify which square to place your mark on. Type in the column number first and then the row number but do not leave a space

between them. For example, type 1 1 to place your nought in the top, lefthand corner. If you type a number in the wrong format, for example 1 1 (i.e. with a space between numbers), or a number that does not correspond to a position on the grid, for example 4,1, the Atari won't accept it and will beep at you. If you type the number of a position that is already occupied a message to that effect will be displayed. Once you've made your move the computer replies with its 'O' and you make your next move. At the end the Atari will display "I win" if it has been successful, "You win" if you've been successful and "Drawn" if it's stalemate. In the case of a draw you have to fill the board for the game to finish.

Typing tips

The grid display for this game uses some of the Atari's graphics characters. These are indicated in the listing by the use of square brackets around the letter keys you need to press. When you encounter the first of the pair of square brackets in lines 8190 and 8400 press CTRL and CAPS LOWER together to get into graphics mode and when you get to the second one leave graphics mode by pressing CAPS LOWER with SHIFT. The symbol in lines 8200 to 8390 is the vertical bar which is to be found on the same key as the down arrow and the equals sign. Press this key and SHIFT to produce the character you require. The player's cross is an "X" (line 7020) and capital "O" is used for the Atari's nought (line 7030).

Subroutine structure

20	Main play loop
4000	Evaluates computers move
5000	Tries each move
6000	Gets player's move
7000	Prints board
8000	Sets up arrays and variables
8140	Prints frame
8500	Sound routine
9000	End of game

Programming details

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

Program

```
10 REM NOUGHTS AND CROSSES

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

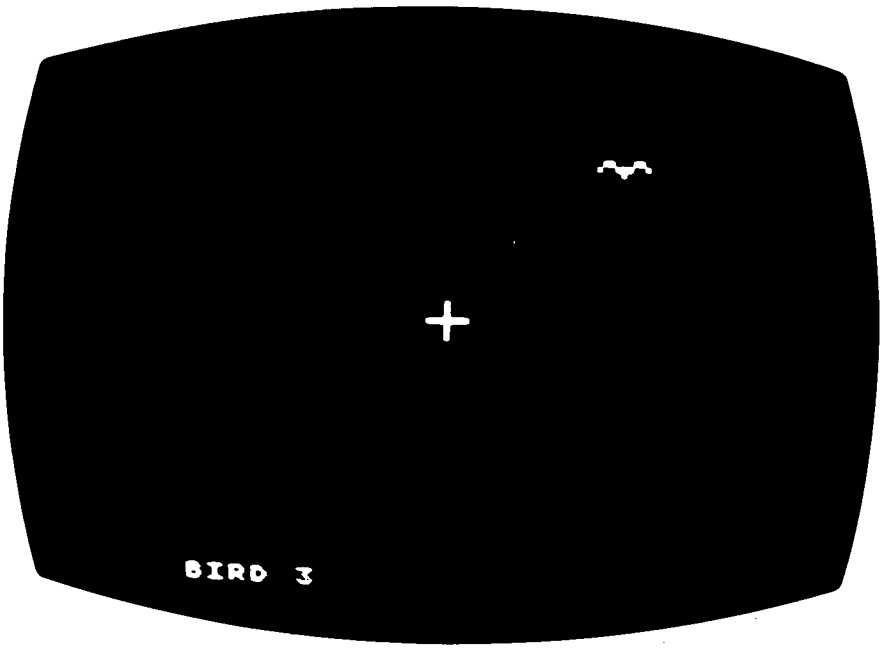
```
4000 FOR Q=1 TO 4
4010 X(Q)=0:Y(Q)=0
4015 NEXT Q
4020 FOR L=1 TO 3
4030 S=0
4040 T=0
4050 FOR K=1 TO 3
4060 IF A(L,K)=1 THEN S=S+1
4070 IF B(L,K)=1 THEN T=T+1
4080 NEXT K
4090 IF S=0 THEN Y(T+1)=Y(T+1)+1
4100 IF T=0 THEN X(S+1)=X(S+1)+1
4105 NEXT L
4110 FOR L=1 TO 3
4120 T=0
4125 S=0
4130 FOR K=1 TO 3
4140 IF A(K,L)=1 THEN S=S+1
4150 IF B(K,L)=1 THEN T=T+1
4160 NEXT K
4170 IF S=0 THEN Y(T+1)=Y(T+1)+1
4180 IF T=0 THEN X(S+1)=X(S+1)+1
4185 NEXT L
4190 GOSUB 4300
4200 GOSUB 4400
4210 IF X(4)=1 THEN FIN=1:RETURN
4215 IF Y(4)=1 THEN FIN=2
4220 E=128*Y(4)-63*X(3)+31*Y(3)-15*X(2)+
     7*Y(2)
4230 RETURN
4300 T=0
4310 S=0
4320 FOR K=1 TO 3
4330 T=T+A(K,K)
4340 S=S+B(K,K)
4350 NEXT K
4360 IF S=0 THEN X(T+1)=X(T+1)+1
4370 IF T=0 THEN Y(S+1)=Y(S+1)+1
4380 RETURN
4400 T=0
4410 S=0
4420 FOR K=1 TO 3
4430 T=T+A(4-K,K)
4440 S=S+B(4-K,K)
4450 NEXT K
4460 IF S=0 THEN X(T+1)=X(T+1)+1
4470 IF T=0 THEN Y(S+1)=Y(S+1)+1
4480 RETURN
```

```
5000 M=-256:DR=1
5005 FOR J=1 TO 3
5010 FOR I=1 TO 3
5015 IF A(I,J)=1 OR B(I,J)=1 THEN GOTO 5040
5016 DR=0:B(I,J)=1
5020 GOSUB 4000
5025 IF FIN=1 THEN RETURN
5030 IF E>M THEN M=E:A=I:B=J
5035 B(I,J)=0
5040 NEXT I
5050 NEXT J
5060 B(A,B)=1
5070 RETURN

6000 POSITION 8,19:PRINT "YOUR MOVE
     (COL ROW) ";:INPUT A$
6005 IF LEN(A$)<>2 THEN GOSUB 8500:GOTO 6000
6010 J=VAL(A$(1,1)):I=VAL(A$(2,2))
6020 IF I<1 OR I>3 THEN GOSUB 8500:GOTO 6000
6030 IF J<1 OR J>3 THEN GOSUB 8500:GOTO 6000
6040 IF A(I,J)=1 THEN GOTO 6100
6050 IF B(I,J)=1 THEN GOTO 6100
6060 A(I,J)=1
6070 POSITION 8,20:PRINT "
           ":REM 25 SPACES
6080 POSITION 8,19:PRINT "
                  ":REM 31 SPACES
6090 RETURN
6100 POSITION 8,20:PRINT "POSITION ALREADY
     OCCUPIED"
6110 GOSUB 8500
6120 GOTO 6000
```

```
7000 FOR J=1 TO 3
7010 FOR I=1 TO 3
7020 IF A(I,J)=1 THEN POSITION J*4+8,I*4+3:
     PRINT "X";
7030 IF B(I,J)=1 THEN POSITION J*4+8,I*4+3:
     PRINT "O";
7040 IF A(I,J)+B(I,J)=0 THEN POSITION
     J*4+8,I*4+3:PRINT " ";
7050 NEXT I
7060 PRINT
7070 NEXT J
7080 RETURN

8000 DIM A(3,3)
8005 DIM B(3,3)
8010 DIM A$(2)
8020 DIM X(4)
8030 DIM Y(4)
8040 FOR Q=1 TO 3
8050 FOR W=1 TO 3
8060 A(Q,W)=0
8070 B(Q,W)=0
8080 NEXT W
8090 NEXT Q
8120 FIN=0
8130 DR=0
```

```
8140 PRINT CHR$(125)
8150 POKE 752,1
8190 POSITION 11,9:PRINT "[RRRSRRRSRRR]"
8200 POSITION 14,6:PRINT "|";
8210 POSITION 18,6:PRINT "|";
8220 POSITION 14,8:PRINT "|";
8230 POSITION 18,8:PRINT "|";
8240 POSITION 14,7:PRINT "|";
8250 POSITION 18,7:PRINT "|";
8260 POSITION 14,10:PRINT "|";
8270 POSITION 18,10:PRINT "|";
8280 POSITION 14,11:PRINT "|";
8290 POSITION 18,11:PRINT "|";
8300 POSITION 14,12:PRINT "|";
8310 POSITION 18,12:PRINT "|";
8320 POSITION 14,14:PRINT "|";
8350 POSITION 18,14:PRINT "|";
8360 POSITION 14,15:PRINT "|";
8370 POSITION 18,15:PRINT "|";
8380 POSITION 14,16:PRINT "|";
8390 POSITION 18,16:PRINT "|";
8400 POSITION 11,13:PRINT "[RRRSRRRSRRR]"
8410 RETURN

8500 SOUND 0,100,10,10
8510 FOR D=1 TO 100
8520 NEXT D
8530 SOUND 0,0,0,0
8540 RETURN

9000 POSITION 10,20:PRINT "I WIN            "
9010 GOTO 9600
9100 POSITION 10,20:PRINT "DRAW             "
9110 GOTO 9600
9500 POSITION 10,20:PRINT "YOU WIN          "
9600 POSITION 10,21:PRINT "ANOTHER GAME
     Y/N";:INPUT A$
9610 IF A$="Y" THEN RUN
9620 GRAPHICS 0
```

14
Pot Shot



All the target practice you could ever want and the magic bird lives to fly on – however many times you score a direct hit. The elements of this game are simple – a blue sky with a bird winging its way from left to right and your rifle sight. The object is straightforward – to line up the sight with the bird and shoot it. But in practice its not that simple. For one thing, every time you fire, your rifle 'kicks' to one side or the other so you have to realign your sight for another to have perfect aim. Unless your sight is directly over the bird you will not score a hit. A total of five birds fly across the sky and there is no limit to the number of hits you can score – your total for each bird and the whole game are displayed at the end. There is a distinctive sound every time you fire your rifle and another sound when you hit the bird. The word 'hit' also appears to confirm a good shot.

How to play

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

Subroutine structure

20	Main play loop
1000	Plots cross
2000	Moves sight
3000	Sets up game
4000	Plots bird
5000	Shoots, tests for hit and recoils sight
6000	Sound routine for hit
6500	Sound routine for gunshot
8000	Title frame and calculates path of bird
9000	End of game

programming details

This is a high resolution dynamic graphics game. The rifle sight is plotted as a high resolution cross, using PLOT and DRAWTO, in subroutine 1000. The bird is also plotted as a collection of high resolution points, in subroutine 4000. The use of high resolution graphics means that the range and smoothness of both the bird and the rifle sight is better than could be achieved with low resolution graphics. The function LOCATE is used in subroutine 5000 to discover if the target has been hit. Notice the way that the gunfire sound is made by subroutine 6500. You may like to use this in your own programs. It is also interesting to note the way the bird's flight path is calculated and stored in the array B for use later in the program.

Program

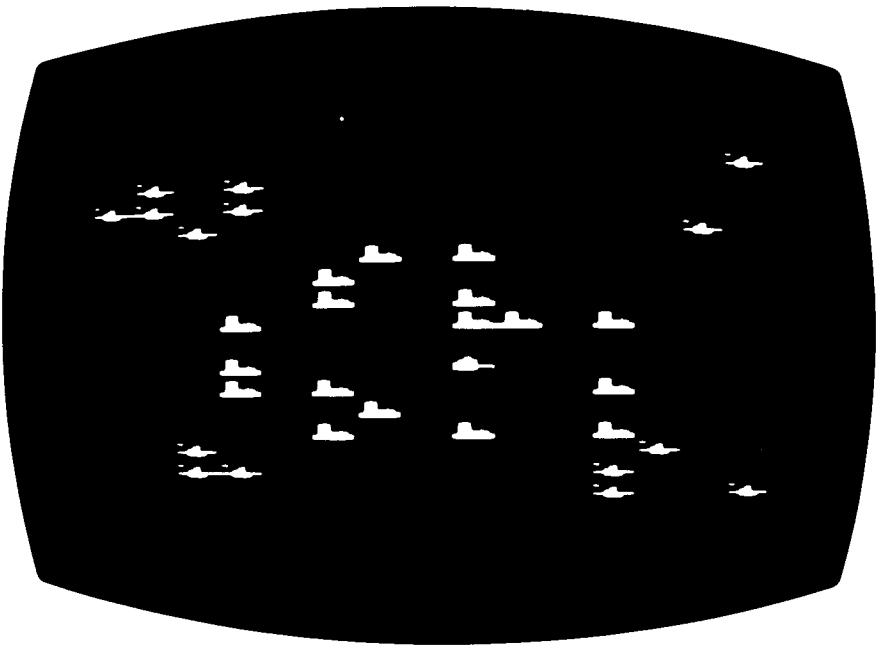
```
10 REM POT SHOT
```

```
  20 GOSUB 8000
  30 FOR G=1 TO 5
  35 PRINT CHR$(125)
  38 GOSUB 8190
  40 GOSUB 3000
  45 PRINT "BIRD ";G;
  50 GOSUB 2000
  60 NEXT G
  70 GOTO 9000
  90 STOP

1000 PLOT X-3,Y
1010 DRAWTO X+3,Y
1020 PLOT X,Y-3
1030 DRAWTO X,Y+3
1060 RETURN

2000 A=PEEK(764)
2005 POKE 764,255
2006 J=B(I):COLOR 0:GOSUB 4000
2008 COLOR 0:GOSUB 1000
2010 IF A=6 AND X>4 THEN X=X-2
2020 IF A=14 AND Y>4 THEN Y=Y-2
2030 IF A=15 AND Y<76 THEN Y=Y+2
2040 IF A=7 AND X<146 THEN X=X+2
2050 I=I+1:J=B(I):COLOR 2:GOSUB 4000
2060 IF A=33 THEN GOSUB 5000
2065 IF X<5 THEN X=5
2066 IF X>146 THEN X=146
2070 COLOR 1:GOSUB 1000
2080 IF FIN=1 THEN FIN=0:RETURN
2100 GOTO 2000

3000 FIN=0
3010 X=INT(RND(0)*30)+20
3020 Y=35
3030 I=6
3040 GOSUB 1000
3050 J=B(I):GOSUB 4000
3060 RETURN
```

```
4000 PLOT I,J-1
4005 PLOT I-3,J-2
4010 PLOT I-2,J-2
4020 PLOT I-1,J-1
4030 PLOT I,J
4040 PLOT I+1,J-1
4050 PLOT I+2,J-2
4060 PLOT I+3,J-2
4070 PLOT I-4,J-1
4080 PLOT I+4,J-1
4090 IF I>146 THEN FIN=1
4100 RETURN

5000 GOSUB 6500
5010 LOCATE X,Y,BI:IF BI<>2 THEN
     X=X+10-INT(RND(0)*20):RETURN
5020 X=X+10-INT(RND(0)*20)
5030 GOSUB 6000
5040 PRINT " HIT  ";
5050 H(G)=H(G)+1
5060 RETURN

6000 SOUND 0,50,10,8
6010 FOR D=1 TO 10
6020 NEXT D
6030 SOUND 0,0,0,0
6040 RETURN

6500 FOR P=15 TO 0 STEP -2
6510 SOUND 0,80,0,P
6515 FOR D=1 TO 15-P:NEXT D
6520 NEXT P
6530 SOUND 0,0,0,0
6540 RETURN
```

```
8000 HIT=0
8005 GRAPHICS 0
8010 POKE 752,1:PRINT CHR$(125)
8020 POSITION 12,2:PRINT "F O T  S H O T"
8030 POSITION 10,10:PRINT "ONE MOMENT PLEASE"
8060 DIM B(160)
8070 FOR I=1 TO 160
8080 B(I)=(75-I)*(75-I)/150+10
8100 NEXT I
8130 DIM H(5)
8140 FOR H=1 TO 5
8150 H(H)=0
8160 NEXT H
8170 DIM A$(1)
8190 GRAPHICS 7
8200 POKE 752,1
8210 SETCOLOR 4,7,8
8230 SETCOLOR 1,0,14
8240 SETCOLOR 0,0,0
8250 SETCOLOR 2,7,8
8260 RETURN

9000 GRAPHICS 0
9010 T=0
9020 FOR I=1 TO 5
9030 POSITION 5,5+I
9040 PRINT "BIRD ";I;" HIT ";H(I)
9050 T=T+H(I)
9060 NEXT I
9070 POSITION 10,12:PRINT "TOTAL HITS= ";T
9080 POSITION 10,15:PRINT "ANOTHER GAME Y/N";
9090 INPUT A$
9100 IF A$="Y" THEN RUN
```

15
Save The Whale



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

How to play

At the beginning of the game you can select the difficulty level for your turn. Your selection governs the starting positions of the icebergs and so makes the game easier or harder to play. You move the whale first by pressing any of the arrow keys. The kayaks then move and a tone signals when it's time to move again. If any kayak runs into an iceberg the kayak vanishes, if the whale runs into an iceberg the iceberg vanishes this of course reduces his

protection so it is not advisable except in extreme circumstances – and if an eskimo reaches the whale he harpoons the whale and kills him. The game is over when all the eskimos have been removed from play or when the whale is dead.

Subroutine structure

20	Sets up graphics characters and arrays
160	Title frame
250	Sets up screen
280	Prints kayaks
360	Prints icebergs
420	Prints whale
450	Main play loop
520	Checks for game over
540	Move whale routine
680	Move kayaks routine
800	End of game
950	Transfer ROM characters to RAM
1000	Defines graphics character for whale
1100	Defines graphics character for eskimos
1200	Defines graphics character for icebergs
1280	Changes RAM character set and switches cursor off
1300	Sets colours
1600	Sound tone routine

Programming details

The initial positions of the kayaks are set at random within a band at the edges of the screen (lines 290 and 300). The initial positions of the icebergs are set in a similar fashion (lines 370 and 380) but account is also taken of the difficulty factor, 'D' input at 230. The LOCATE command is used in line 735 to detect whether a kayak has hit an iceberg in which case that is the end of the kayak. The alternative method of simply comparing co-ordinates is used in line 750 to detect whether a kayak has harpooned the whale in which case that is the end of the whale (and the game).

Program

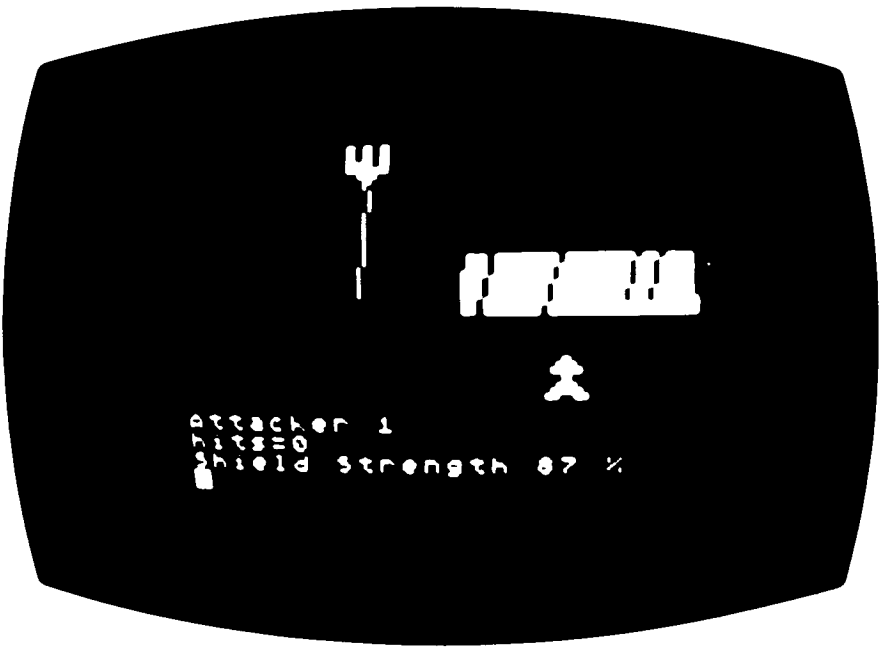
```
 10 REM SAVE THE WHALE

 20 GRAPHICS 0
 30 OPEN #2,4,0,"K:"
 40 DIM X(20)
 50 DIM Y(20)
 60 DIM U(20)
 70 DIM V(20)

160 POSITION 6,2:PRINT "S A V E  T H E
    W H A L E"
170 POSITION 4,8:PRINT "In this game, you,
    the whale"
180 POSITION 4,10:PRINT "must outwit the
    eskimos hunting"
190 POSITION 4,12:PRINT "you in their
    kayaks"
200 POSITION 4,14:PRINT "by luring them
    onto the icebergs"
210 POSITION 1,18:PRINT "Which difficulty
    level do you wish to"
220 POSITION 15,19:PRINT "play at"
230 POSITION 0,21:PRINT "(1) Expert,(2)
    Intermediate,(3) Novice ";:GET #2,D
236 D=D-48
240 IF D<1 OR D>3 THEN GOTO 230

250 GRAPHICS 1+16:GOSUB 950

280 FOR C=1 TO 20
290 X=SGN(RND(0)-0.5)*INT(RND(0)*4+5)+9
300 Y=SGN(RND(0)-0.5)*INT(RND(0)*4+6)+9
310 POSITION X,Y:PRINT #6;CHR$(5);
320 X(C)=X
330 Y(C)=Y
340 NEXT C
```

```
360 FOR C=1 TO 20
370 U(C)=SGN(RND(0)-0.5)*INT(RND(0)*4+3-D)+9
380 V(C)=SGN(RND(0)-0.5)*INT(RND(0)*4+3-D)+9
390 POSITION U(C),V(C):PRINT #6;"@";
400 NEXT C

420 X=INT(RND(0)*2+10)
430 Y=INT(RND(0)*2+10)
440 POSITION X,Y:PRINT #6;CHR$(4+160)

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

520 IF F=0 THEN GOTO 830
530 GOTO 450

540 GOSUB 1600
550 Z=X:V=Y
570 GET #2,A
580 IF A=0 THEN GOTO 570
590 IF A=43 AND X>1 THEN X=X-1
600 IF A=42 AND X<19 THEN X=X+1
610 IF A=61 AND Y<19 THEN Y=Y+1
620 IF A=45 AND Y>0 THEN Y=Y-1
640 POSITION Z,V:PRINT #6;" "
660 POSITION X,Y:PRINT #6;CHR$(4+160)
670 RETURN

680 POSITION X(C),Y(C):PRINT #6;" "
690 E=0
700 E=SGN(X(C)-X)
710 X(C)=INT(X(C)-E)
720 E=SGN(Y(C)-Y)
730 Y(C)=INT(Y(C)-E)
735 LOCATE X(C),Y(C),D
740 IF D=64 THEN X(C)=0:GOSUB 1600:GOTO 780
750 IF X(C)=X AND Y(C)=Y THEN GOTO 800
770 POSITION X(C),Y(C):PRINT #6;CHR$(5);
780 RETURN
```

```
 800 POSITION X(C),Y(C):PRINT #6;" "
 810 POSITION 1,15:PRINT #6;" YOU WERE
     KILLED"
 820 GOTO 840
 830 POSITION 1,15:PRINT #6;"YOU ESCAPED"
 835 POSITION 1,16:PRINT #6;"THIS TIME!"
 840 DIM A$(1)
 845 FOR Q=1 TO 500:NEXT Q
 846 GRAPHICS 0
 850 POSITION 10,10:PRINT "ANOTHER GAME
     (Y/N)";:INPUT A$
 860 IF A$="Y" THEN RUN
 870 GRAPHICS 0
 900 STOP

 950 CH=(PEEK(106)-8)*256
 960 CHORG=(PEEK(756)*256)
 970 FOR I=0 TO 511
 980 POKE CH+I,PEEK(CHORG+I)
 990 NEXT I

1000 POKE CH+(ASC("$")-32)*8+0,0
1010 POKE CH+(ASC("$")-32)*8+1,0
1020 POKE CH+(ASC("$")-32)*8+2,48
1030 POKE CH+(ASC("$")-32)*8+3,120
1040 POKE CH+(ASC("$")-32)*8+4,248
1050 POKE CH+(ASC("$")-32)*8+5,255
1060 POKE CH+(ASC("$")-32)*8+6,249
1070 POKE CH+(ASC("$")-32)*8+7,0

1100 POKE CH+(ASC("%")-32)*8+0,0
1110 POKE CH+(ASC("%")-32)*8+1,200
1120 POKE CH+(ASC("%")-32)*8+2,88
1130 POKE CH+(ASC("%")-32)*8+3,56
1140 POKE CH+(ASC("%")-32)*8+4,255
1150 POKE CH+(ASC("%")-32)*8+5,60
1160 POKE CH+(ASC("%")-32)*8+6,8
1170 POKE CH+(ASC("%")-32)*8+7,0
```

```
1200 POKE CH+(ASC("@")-32)*8+0,0
1210 POKE CH+(ASC("@")-32)*8+1,32
1220 POKE CH+(ASC("@")-32)*8+2,112
1230 POKE CH+(ASC("@")-32)*8+3,116
1240 POKE CH+(ASC("@")-32)*8+4,116
1250 POKE CH+(ASC("@")-32)*8+5,126
1260 POKE CH+(ASC("@")-32)*8+6,255
1270 POKE CH+(ASC("@")-32)*8+7,255

1280 POKE 756,CH/256
1290 POKE 752,1

1300 SETCOLOR 0,0,14
1310 SETCOLOR 1,3,4
1320 SETCOLOR 2,0,0
1340 SETCOLOR 4,7,8
1500 RETURN

1600 SOUND 0,60,10,6
1610 FOR D=1 TO 50
1620 NEXT D
1630 SOUND 0,0,0,0
1640 RETURN
```

16
Mighty Missile



Your weapon can destroy anything anything that it actually hits. So the only problem in this game is to ensure that the missile finds its target, quickly and accurately. The enemy ships sweep in from the left and the right firing relentlessly. Your missile is only vulnerable if its protective shield is eroded away and then it can be easily blasted in its home base. Otherwise, it is impervious to enemy fire, even outside its base. If it hits an enemy it will explode on contact but if it fails to find its target it will disintegrate as it reaches the upper atmosphere. You can launch ten missiles and there are ten enemy ships. Each ship maintains a stable orbit until you actually take a shot at it, so you can wait in the base while deciding which side to fire from and when to fire except that with every orbit more of your shield is blasted away by enemy fire and once there is only 30 per cent of it left you will no longer have any protection from the enemies' lasers.

How to play

In this game it is important to notice how much 'Shield strength' you have left since when the figure displayed drops to 30% you will be vulnerable to attack. Once the shield is so eroded the enemy laser is able to home in and destroy you wherever they hit you, including inside the missile base. The object of the game is to score as many hits as possible so it is worth watching each new enemy ship's orbit at least once or twice before you try to shoot it down. To fire you have to leave your base. Do this by pressing the right or left arrow key. This will take you to a fixed position on the right or left of the screen and launch the missile. Remember to take into account the time it will take for your missile to reach the enemy ship which will continue on its path! At the end of the game your score is displayed and you are given the option of another game.

Typing tips

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

Subroutine structure

30	Set up
70	Main play loop
310	End of game
380	Prints attacker and fire laser
500	Checks to see if player has activated missile
620	Calculates shield strength
710	Moves and fires missile and tests for hit or miss
780	Explosion graphics and sound
940	Sets up attack orbit
1070	Prints shield
1200	Clears screen and sets colours
2000	Plots explosion
2200	Unplots missile
2300	Plots missile
2400	Plots attacker
2600	Sound routine

Programming details

As this program has a clear structure and uses separate routines for most of its major elements, you should find it relatively easy to follow. One interesting point to note is the way in which the path of the attacking ship is calculated in subroutine 940 and stored in a pair of arrays X and Y to be used repeatedly for the various orbits used during the game. A second point of interest is the way that the shield is whittled away by the laser beam passing through it. The laser beam is simply a high resolution line in colour 2 which is then blanked out using colour 0. This blanking out will, of course, blank out anything that the laser passes through.

The strength of the shield is estimated, in subroutine 620, by the number of black points left in the shield, using the LOCATE command. LOCATE returns a 0 if the point at x,y is a background point.

Program

```
10 REM MIGHTY MISSILE

30 GOSUB 1200
40 GOSUB 940
50 GOSUB 1070
60 GOSUB 620
```

```
70 FOR A=1 TO 5
80 DIR=SGN(RND(0)-0.5)
90 PRINT "Attacker ";A
100 PRINT "hits=";HIT
110 R=7-INT(RND(0)*14)
120 IF R<0 THEN S=3-R:E=36
130 IF R>=0 THEN S=3:E=36-R
140 IF DIR=-1 THEN T=S:S=E:E=T
150 FOR I=S TO E STEP DIR
160 GOSUB 380
170 GOSUB 500
180 IF F=1 THEN GOSUB 710
190 NEXT I
200 GOSUB 2000
210 IF F=1 THEN FIN=1
220 IF FIN=2 THEN A=6:GOTO 300
230 GOSUB 620
240 IF F=1 THEN F=0:GOSUB 2100
250 GOSUB 2200
260 MX=74:MY=76
270 GOSUB 2300
280 IF FIN=0 THEN GOTO 150
290 FIN=0
300 NEXT A

310 IF FIN=2 THEN PRINT "They got you"
320 PRINT "hits=";HIT
330 PRINT "You hit ";HIT
340 PRINT "Another game";:INPUT A$
350 IF A$="Y" THEN RUN
360 GRAPHICS 0
370 STOP
```

```
380 REM ENEMY
390 GOSUB 2000
400 GOSUB 2400
410 IF RND(0)<0.3 THEN RETURN
420 IF C<=30 AND ABS(MX-X(I+DIR)+4)<10 THEN
    FIN=2:GOTO 840
430 COLOR 2:PLOT X(I+DIR)+4,Y(I+R+DIR)
440 D=5-INT(RND(0)*10)
450 DRAWTO X(I+DIR)+D,Y(I+R+DIR)+20
460 GOSUB 2600
470 COLOR 0:PLOT X(I+DIR)+4,Y(I+R+DIR)
480 DRAWTO X(I+DIR)+D,Y(I+R+DIR)+20
490 RETURN

500 REM PROCFIRE
510 IF F=1 THEN RETURN
520 Z=PEEK(764)
530 POKE 764,255
540 IF Z=255 THEN RETURN
550 GOSUB 2200
560 IF Z=6 THEN MX=MX-40:GOTO 590
570 IF Z=7 THEN MX=MX+40:GOTO 590
580 RETURN
590 GOSUB 2300
600 F=1
610 RETURN

620 REM STREN
630 C=0
640 J=54
650 FOR I=60 TO 100
655 LOCATE I,J,P
660 C=C+P
670 NEXT I
680 C=INT(C/41*100)
690 PRINT "Shield Strength ";C;" % "
700 RETURN

710 REM GUIDE
720 GOSUB 2200
730 MY=MY-4
740 IF MY<12 THEN F=0:FIN=1:GOTO 860
750 GOSUB 2300
760 IF ABS(MX-X(I+DIR))>7 THEN RETURN
770 IF ABS(MY-Y(I+R+DIR))>5 THEN RETURN
```

```
 780 FIN=1
 790 GOSUB 2200
 800 MX=X(I+DIR)
 810 MY=Y(I+R+DIR)
 820 HIT=HIT+1
 830 F=0
 840 COLOR 2:PLOT X(I+DIR)+3,Y(I+R+DIR)
 850 DRAWTO MX+3,MY
 860 GOSUB 2100
 890 IF FIN=2 THEN GOTO 920
 900 GOSUB 2000
 910 X=X(I+DIR):Y=Y(I+R+DIR):COLOR 0:
     GOSUB 2430
 920 I=E+DIR
 930 RETURN

 940 REM PATH
 950 DIM X(50),Y(50)
 960 X=0:Y=0
 970 N=39
 980 FOR I=1 TO N
 990 X=X+4
1000 Y=40-INT(((80-X)*(80-X))/200)
1010 X(I)=X
1020 Y(I)=Y
1030 NEXT I
1040 I=1
1050 HIT=0
1060 COLOR 1

1070 REM BLOCK
1090 FOR I=50 TO 60
1100 PLOT 60,I:DRAWTO 100,I
1110 NEXT I
1120 MX=74
1130 MY=76
1160 GOSUB 2300
1170 F=0
1180 FIN=0
1190 RETURN
```

```
1200 REM INIT
1210 GRAPHICS 7
1220 SETCOLOR 4,8,8
1230 SETCOLOR 2,8,8
1240 SETCOLOR 1,0,14
1250 SETCOLOR 0,4,5
1260 DIM A$(1)
1320 RETURN

2000 COLOR 0
2010 X=X(I)
2020 Y=Y(I+R)
2030 GOTO 2430
2100 COLOR 2
2110 FOR Q=1 TO 20
2120 PLOT MX+INT(RND(0)*5),MY-INT(RND(0)*5)
2130 NEXT Q
2140 COLOR 0
2150 FOR Q=0 TO 7
2160 FOR Z=0 TO 7
2170 PLOT MX+Z,MY-Q
2180 NEXT Z
2190 NEXT Q
2195 RETURN

2200 COLOR 0
2210 GOTO 2310

2300 COLOR 1
2310 PLOT MX,MY:DRAWTO MX+2,MY
2320 PLOT MX+5,MY:DRAWTO MX+7,MY
2330 PLOT MX+1,MY-1:DRAWTO MX+6,MY-1
2340 PLOT MX+2,MY-2:DRAWTO MX+5,MY-2
2350 PLOT MX+3,MY-3:PLOT MX+4,MY-3
2360 PLOT MX+3,MY-4:PLOT MX+4,MY-4
2370 PLOT MX+1,MY-5:DRAWTO MX+6,MY-5
2380 PLOT MX+2,MY-6:DRAWTO MX+5,MY-6
2390 PLOT MX+3,MY-7:PLOT MX+4,MY-7
2395 RETURN
```

```
2400 COLOR 1
2410 X=X(I+DIR)
2420 Y=Y(I+R+DIR)
2430 PLOT X+3,Y:PLOT X+4,Y
2440 PLOT X+3,Y-1:PLOT X+4,Y-1
2450 PLOT X+2,Y-2:DRAWTO X+5,Y-2
2460 PLOT X,Y-3:DRAWTO X+7,Y-3
2470 PLOT X,Y-4:DRAWTO X+7,Y-3
2480 FOR Y1=Y-4 TO Y-7 STEP -1
2490 PLOT X,Y1
2500 PLOT X+1,Y1
2510 PLOT X+3,Y1
2520 PLOT X+4,Y1
2530 PLOT X+6,Y1
2540 PLOT X+7,Y1
2550 NEXT Y1
2560 RETURN

2600 SOUND 0,100,10,10
2610 FOR Q=1 TO 4
2620 NEXT Q
2630 SOUND 0,0,0,0
2640 RETURN
```

17
Nine Hole Golf

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

How to play

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

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

Subroutine structure

20	Initialises variables
2000	Sets up and plays each hole
4900	Reports score for each hole
4930	End of game
5000	Effectiveness of hit routine
5500	Plots balls' flight
6500	Detects overshoot
7000	Displays swinging club
8000	Putting routine
9000	Sets colours
9200	Draws flag
9300	Draws man
9400	Undraws man
9500	Sound routine
9600	Routine for short delay
9700	Routine for long delay

Programming details

An interesting feature of this game is the use of high resolution graphics to make the player appear to swing his club. This is done in subroutine 7000 which draws a line which is the continually shifting radius of a circle. The flight of the ball is also plotted using high resolution graphics. The path tht the ball appears to follow (subroutine 5000) is a distorted parabola that always carries the ball in the direction of the flag.

Scope for improvement

You may have noticed that the score card at the end of the game does

not total your score nor compare it with any ideal *par* for the course. You might like to add both these features. You will need to play the game a few times to discover what figure to set as the par.

Program

```
   10 REM GOLF

   20 DIM A$(1)
  790 DIM T(9)
  810 XH=0:XC=0
  820 YH=0:HT=0:YC=0

 2000 FOR H=1 TO 9
 2010 GOSUB 9000
 2020 T(H)=0
 2040 PRINT "HOLE NUMBER ";H
 2100 REM drive section
 2120 X=INT(RND(0)*25)+5
 2130 Y=INT(RND(0)*10+60)
 2150 XT=INT(RND(0)*60+85)
 2170 YT=INT(RND(0)*30)+6
 2180 D=SGN(XT-X)*SQR((XT-X)*(XT-X)+(YT-Y)*
      (YT-Y))
 2190 GOSUB 9200
 2200 GOSUB 9300
 2300 PRINT "DISTANCE TO NEXT HOLE IS ";INT(D)
 2310 PRINT "WHICH CLUB (1 TO 8)";:INPUT C
 2315 IF C<1 OR C>8 THEN GOTO 2310
 2320 C=10-C
 2325 GOSUB 7000
 2326 B=0
 2330 GOSUB 5000
 2335 IF B=1 THEN B=0:GOTO 4920
 2340 D=SGN(XT-XHT)*SQR((XT-XHT)*(XT-XHT)+
      (YT-YHT)*(YT-YHT))
 2360 IF D<-10 THEN PRINT "YOU OVERSHOT-
      TRY ANOTHER HOLE":GOSUB 9700:GOTO 2010
 2370 IF D<5 THEN PRINT "ON THE GREEN":
      GOSUB 9700:GOTO 8000
 2380 GOSUB 9400
 2390 X=XHT
 2400 Y=YHT
 2500 GOTO 2190
```

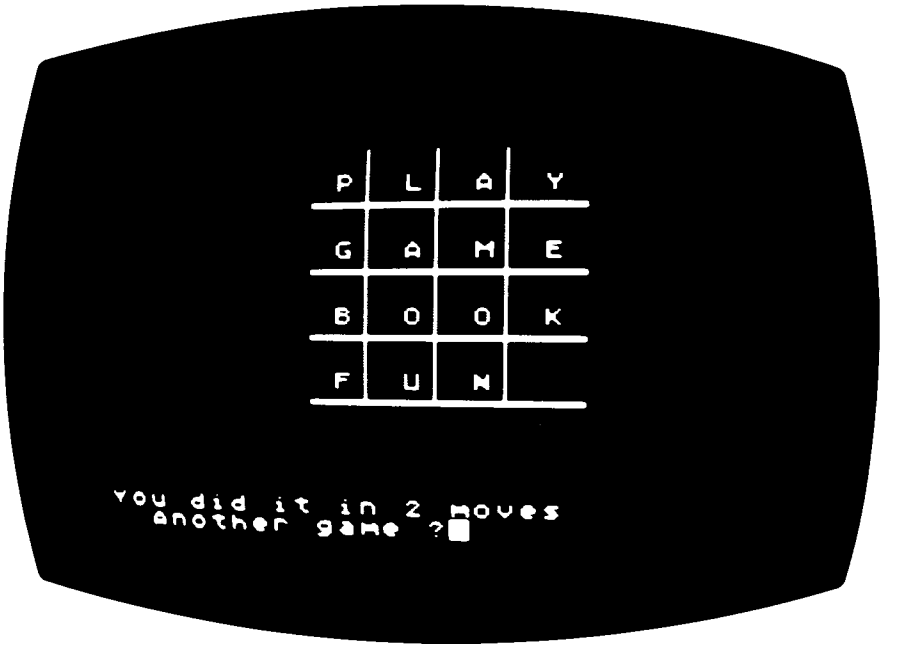
```
4900 PRINT "YOU TOOK ";T(H);" STROKES"
4910 GOSUB 9700
4920 NEXT H

4930 GRAPHICS 0
4940 POSITION 10,2:PRINT "THIS ROUND"
4945 PRINT
4950 FOR I=1 TO 9
4960 PRINT "    HOLE";I;
4970 IF T(I)=-1 THEN PRINT " LOST BALL":
     GOTO 4990
4980 PRINT " ";T(I);" STROKES"
4990 NEXT I
4995 PRINT :PRINT "ANOTHER ROUND ";:INPUT A$
4996 IF A$="Y" THEN RUN
4999 GRAPHICS 0:STOP

5000 REM hit routine
5100 VT=(C*(1+RND(0)*0.1))+1
5160 HT=0
5170 XH=0

5500 REM plot ball
5530 Q=(Y-YT)/(XT-X)
5600 VV=-VT*(SIN(45*3.14159/180))
5610 XC=X+1
5620 YC=Y
5630 VH=VT*(COS(45*3.14159/180))
5650 HT=HT+VV
5660 YH=Q*XH
5670 VV=VV+1
5800 XH=XH+VH
5810 YH=-Q*XH
5820 IF XH+XC>159 THEN YH=0:YT=0:YC=0:XH=0:
     XC=0:GOTO 6500
5830 IF YH+HT+YC<0 THEN YH=0:YT=0:YC=0:XH=0:
     XC=0:GOTO 6500
5850 IF HT>=0 THEN GOTO 5900
5860 COLOR 2:PLOT XH+XC,YH+HT+YC
5865 GOSUB 9600
5880 COLOR 0:PLOT XH+XC,YH+HT+YC
5890 GOTO 5650
5900 XHT=XH+XC
5910 YHT=YH+HT+YC
5915 RETURN
6000 PLOT XH+XC,YH+HT+YC
6010 RETURN
```

```
6500 PRINT "YOU'VE LOST YOUR BALL !!"
6510 GOSUB 9500
6520 HT=0:T(H)=-1
6530 GOSUB 9700
6540 B=1:RETURN

7000 REM swing
7010 T(H)=T(H)+1
7100 XS=X+3
7110 YS=Y:YS=YS-3
7200 FOR S=-30 TO 5 STEP 2
7220 A=S/30*3.14159
7230 SX=6*SIN(A):SY=6*COS(A)
7240 COLOR 2:PLOT XS,YS:DRAWTO XS+SX,YS+SY
7250 IF S=0 THEN GOSUB 9500
7260 GOSUB 9600
7270 COLOR 0:PLOT XS,YS:DRAWTO XS+SX,YS+SY
7275 GOSUB 9300
7280 NEXT S
7500 RETURN
```

```
8000 REM putting
8010 GOSUB 9000
8030 XG=INT(RND(0)*15)+15
8040 YG=70
8050 XH=INT(RND(0)*45)+100
8060 YH=70
8070 D=XH-XG
8080 IF D<0 THEN D=ABS(D)
8100 XT=XH:YT=YH:GOSUB 9200
8110 X=XG:Y=YG:GOSUB 9300
8120 PRINT "DISTANCE TO HOLE IS ";D;"    "
8130 PRINT "WHICH CLUB (1 TO 8)";:INPUT C
8135 IF C<1 OR C>8 THEN GOTO 8130
8140 T(H)=T(H)+1
8145 H1=9-C+INT(RND(0)*2)
8150 H1=H1*5
8160 D=D-H1
8165 IF D<0 THEN H1=XH-XG
8170 FOR Z=XG+1 TO XG+H1
8180 COLOR 2:PLOT Z,YH
8200 COLOR 0:PLOT Z,YH
8210 NEXT Z
8220 GOSUB 9400
8230 XG=XH-D
8240 IF ABS(XG-XH)<4 THEN GOTO 8300
8250 IF D<0 THEN GOSUB 9000:XG=XH+2*D:
     GOTO 8070
8260 GOTO 8100
8300 REM in the hole
8310 GOTO 4900

9000 GRAPHICS 7
9010 SETCOLOR 0,0,0
9020 SETCOLOR 4,12,9
9030 SETCOLOR 1,0,14
9040 SETCOLOR 2,12,9
9050 RETURN

9200 COLOR 1
9210 PLOT XT,YT
9220 DRAWTO XT,YT-5
9230 DRAWTO XT+2,YT-4
9240 DRAWTO XT,YT-3
9250 RETURN
```

```
9300 COLOR 2
9310 PLOT X,Y:PLOT X+5,Y
9320 PLOT X+1,Y-1:PLOT X+4,Y-1
9330 PLOT X+1,Y-2:PLOT X+4,Y-2
9340 PLOT X+2,Y-3:PLOT X+3,Y-3
9350 PLOT X+1,Y-4:DRAWTO X+4,Y-4
9360 PLOT X,Y-5:PLOT X+2,Y-5:PLOT X+3,Y-5:
     PLOT X+5,Y-5
9370 PLOT X+1,Y-6:DRAWTO X+4,Y-6
9380 PLOT X+2,Y-7:PLOT X+3,Y-7
9390 RETURN

9400 COLOR 0
9410 GOTO 9310

9500 SOUND 0,80,10,8
9510 GOSUB 9600
9520 SOUND 0,0,0,0
9530 RETURN

9600 RETURN :FOR Q=1 TO 15
9610 NEXT Q
9620 RETURN

9700 FOR Q=1 TO 500
9710 NEXT Q
9720 RETURN
```

18
Mirror Tile



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

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

How to play

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

These same rules apply to the Atari version of the game. You can move any letter that is directly to the left or right of the hole or just above or below it. To indicate your choice you type in the number (1 to 16) of the square containing the piece you want to move. If you try to make a wrong move the Atari won't let you. Instead it will be helpful and number each square for you, in case your mistake was due to typing in the wrong number for your choice. If you want to see these numbers displayed, type any letter key – in fact any key other than one for a legal move.

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

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

Typing tips

Notice the use of the vertical bar in lines 2020 and 2050. This is entered by pressing SHIFT and the key with the equals sign (=) on it. There are two spaces before the bar in line 2020. This program also uses some of the Atari's special graphics characters. As it is

impossible to print these out they are indicated in the listing by square brackets around the key that has to be pressed once you are in graphics mode. To get into graphics mode press CTRL and CAPS LOWER at the same time. So for example, in line 2080 you get into graphics mode, then press R twice followed by S and then leave graphics mode by pressing CAPS LOWER with SHIFT. The same convention applies in lines 2130, and 2180 to 2220. In line 3520 there are two blank spaces between the quotation marks.

Subroutine structure

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

Programming details

This program is complicated both because of its length and also because it involves a lot of logic. However, its subroutine structure is very clear, so if you follow it through section-by-section you should be able to see what happens at every step.

Scope for improvement

Adding to a program that is already as long as this one may seem to

be a tall order. However there is actually scope for improvement. A routine that reminded the player of the target arrangement might be a very useful extra.

Program

```
  5 REM MIRROR TILE

 10 DIM B(20)
 20 DIM B$(20)
 30 DIM W$(20)
 35 DIM M$(5)
 36 DIM A$(10)

 40 GOSUB 5500
 50 GOSUB 1000
 60 IF WR=1 THEN GOSUB 6500
 70 MOVE=0:ERROR=0
 85 PRINT CHR$(125)
 90 GOSUB 2000
100 GOSUB 5000
110 GOSUB 2500

120 GOSUB 4000
130 GOSUB 500
140 MOVE=MOVE+1
150 IF FIN<>0 THEN GOTO 120
160 POSITION 0,20:
    PRINT ;"You did it in ";MOVE;" moves"
170 PRINT "Another game ";:INPUT A$
180 IF A$="Y" THEN RUN
200 PRINT CHR$(125)
210 STOP

500 FIN=0:K=0
510 FOR I=1 TO 4
520 FOR J=1 TO 4
525 K=K+1
530 IF B$(B(I*4+J),B(I*4+J))<>W$(K,K)
    THEN FIN=1
540 NEXT J
550 NEXT I
560 RETURN
```

```
1000 K=0
1010 FOR I=1 TO 4
1020 FOR J=1 TO 4
1030 K=K+1
1040 B$(LEN(B$)+1)=CHR$(64+K)
1050 B(I*4+J)=K
1060 NEXT J
1070 NEXT I
1080 B$(16)=" "
1090 W$=B$
1100 RETURN

2000 FOR I=0 TO 3
2010 FOR J=0 TO 3
2020 POSITION 10+J*3,4+I*3:PRINT "   |";
2030 NEXT J
2040 FOR J=0 TO 3
2050 POSITION 10+J*3,5+I*3:PRINT " ";
     B$(B((I+1)*4+J+1),B((I+1)*4+J+1));:
     PRINT "|";
2060 NEXT J
2070 FOR J=0 TO 3
2080 POSITION 10+J*3,6+I*3:PRINT "[RRS]";
2090 NEXT J
2100 NEXT I
2110 FOR I=0 TO 11
2120 POSITION 21,4+I:PRINT " ";
2130 IF INT((I+1)/3)=(I+1)/3 THEN
     POSITION 21,4+I:PRINT "[R]";
2170 NEXT I
2180 POSITION 12,15:PRINT "[X]";
2190 POSITION 15,15:PRINT "[X]";
2200 POSITION 18,15:PRINT "[X]"
2210 RETURN
```

```
2500 IS=4:JS=4:IY=0:JY=0
2510 FOR D=1 TO S
2520 I=IS:J=JS
2530 IF RND(0)>0.5 THEN GOTO 2570
2540 I=IS+INT(RND(0)*2)*2-1
2550 IF I>4 OR I<1 THEN I=IS:GOTO 2570
2560 GOTO 2590
2570 J=JS+INT(RND(0)*2)*2-1
2580 IF J>3 OR J<1 THEN J=JS:GOTO 2520
2590 IF I=IY AND J=JY THEN GOTO 2520
2600 IY=IS:JY=JS
2610 GOSUB 6000
2620 NEXT D
2630 RETURN

3000 K=0
3010 FOR I=0 TO 3
3020 FOR J=0 TO 3
3030 K=K+1
3040 POSITION 10+J*3,4+I*3:PRINT K
3050 NEXT J
3060 NEXT I
3070 RETURN

3500 FOR L=0 TO 3
3510 FOR P=0 TO 3
3520 POSITION 10+P*3,4+L*3:PRINT "  "
3530 NEXT P
3540 NEXT L
3555 IF ERROR=0 THEN RETURN
3560 POSITION 0,18
3570 FOR L=1 TO 40*5-4
3580 PRINT " ";
3590 NEXT L
3600 ERROR=0
3610 RETURN
```

```
4000 POSITION 2,19:PRINT "What is your
     move ";:INPUT M$
4002 POSITION 2,19:PRINT "
           ":REM 25 SPACES
4005 IF M$="" THEN GOTO 4000
4006 IF LEN(M$)<2 THEN A$(1)=M$(1):
     M$(1)="0":M$(2)=A$(1)
4010 IF M$(1,1)<"0" OR M$(1,1)>"9"
     OR M$(2,2)<"0" OR M$(2,2)>"9" THEN
     GOSUB 3000:GOTO 4000
4020 M=VAL(M$)
4030 IF M>0 AND M<17 THEN GOTO 4070
4035 ERROR=1
4040 POSITION 2,20:PRINT "A move must be a
     number between "
4050 POSITION 2,21:PRINT "1 and 16 as shown"
4060 GOSUB 3000
4065 GOTO 4000
4070 I=INT((M-1)/4)
4080 J=M-I*4
4090 I=I+1
4100 IF ABS(I-IS)+ABS(J-JS)=1 THEN GOTO 6000
4110 GOSUB 7000
4115 ERROR=1
4120 POSITION 2,20:PRINT "You can only move
     a tile next to"
4130 POSITION 2,21:PRINT "the space
          ":REM 9 SPACES
4140 GOSUB 3000
4150 GOTO 4000

5000 K=0
5005 FOR L=0 TO 3
5010 FOR P=1 TO 4
5015 K=K+1
5020 IF B(P+L*3)=16 THEN IS=L:JS=P:MS=K
5030 NEXT P
5040 NEXT L
5050 RETURN
```

```
5500 PRINT CHR$(125)
5510 POSITION 10,1
5520 PRINT "M i r r o r   T i l e"
5530 POSITION 6,10:PRINT "Do you want to
     input your"
5540 POSITION 6,11:PRINT "own set of words ";
5550 INPUT A$
5555 IF A$="" THEN GOTO 5550
5560 IF A$="Y" THEN WR=1:GOTO 5600
5570 IF A$="N" THEN WR=0:GOTO 5600
5580 GOTO 5550
5600 POSITION 6,15:PRINT "How many shuffles";
5610 INPUT S
5620 IF S<1 THEN GOTO 5600
5640 RETURN

6000 GOSUB 3500
6070 GOSUB 8000
6080 POSITION 11+(J-1)*3,5+(I-1)*3:
     PRINT B$(B(I*4+J),B(I*4+J));
6090 POSITION 11+(JS-1)*3,5+(IS-1)*3:PRINT
     B$(B(IS*4+JS),B(IS*4+JS))
6140 RETURN
```

```
6500 PRINT CHR$(125)
6510 POSITION 2,4:PRINT "Choose 3 four-letter
     words"
6520 PRINT "and 1 three-letter word."
6530 PRINT "Type the first four-letter word ";
     :INPUT A$
6540 IF LEN(A$)<>4 THEN GOTO 6530
6550 W$=A$
6560 PRINT "First word= ";A$
6570 PRINT "Type the second four-letter word";
     :INPUT A$
6580 IF LEN(A$)<>4 THEN GOTO 6570
6590 W$(LEN(W$)+1)=A$
6600 PRINT "Second word= ";A$
6610 PRINT "Type the third four-letter word ";
     :INPUT A$
6620 IF LEN(A$)<>4 THEN GOTO 6610
6630 W$(LEN(W$)+1)=A$
6640 PRINT "Third word= ";A$
6650 PRINT "Type the three-letter word ":
     INPUT A$
6660 IF LEN(A$)<>3 THEN GOTO 6650
6670 W$(LEN(W$)+1)=A$
6680 PRINT "Fourth word= ";A$
6690 FOR Q=1 TO 100:NEXT Q
6700 W$(16)=" "
6710 B$=W$
6720 RETURN

7000 SOUND 0,80,10,8
7010 FOR F=1 TO 50
7020 NEXT F
7030 SOUND 0,0,0,0
7040 RETURN

8000 B(IS*4+JS)=B(I*4+J)
8030 B(I*4+J)=16
8040 T=IS:IS=I:I=T
8050 T=J:J=JS:JS=T
8060 RETURN
```

19
Fruit Machine

Here's a way of playing the fruit machine without spending a penny - your Atari gives you 100 pence to start with, takes 10 pence for every go, and awards you a sum between 5 pence and 50 pence every time you come up with a winning combination. You can give up while you are winning or carry on playing until you are broke.

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

How to play

There are four symbols in the display – cherries, banana, apple and bell. All the winning combinations are displayed on the screen while

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

Subroutine structure

20	Initialisation
50	Main play loop
170	Sets starting points of drum
220	Spins drum
410	Pay out routine
470	Transfers ROM characters to RAM
540	Title frame
630	Data for graphics characters
740	Jackpot routine
800	Signals when broke
1000	Defines graphics characters
1060	Selects RAM characters, suppresses cursor and opens keyboard as an input device
2000	Sound routine

Programming details

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

Scope for improvement

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

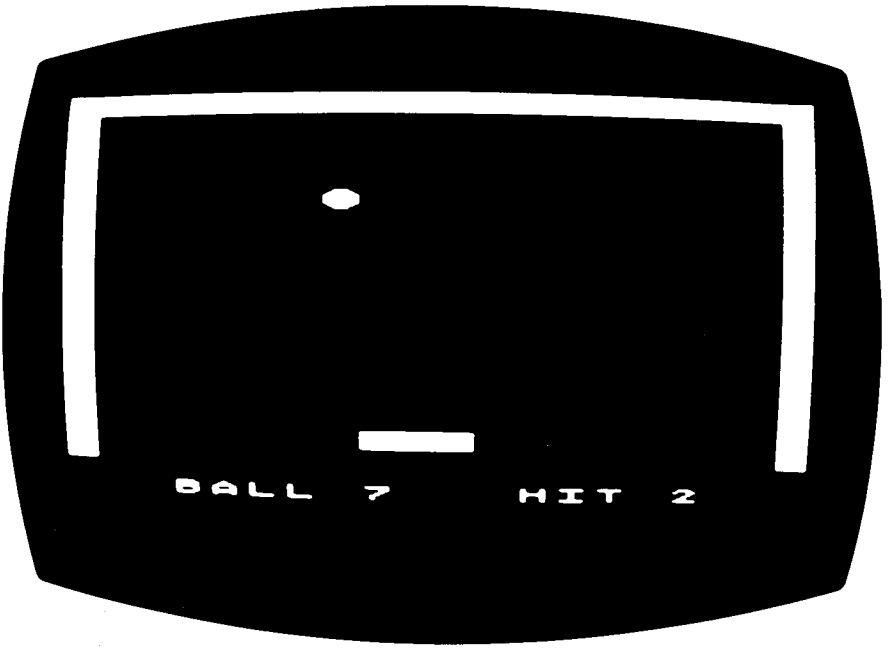
Program

```
 10 REM FRUIT MACHINE

 20 GOSUB 630
 30 GOSUB 470
 40 M=100

 50 GOSUB 170
 60 M=M-10
 70 GOSUB 220
 80 GOSUB 410
 90 IF M<=0 THEN GOTO 800
100 POSITION 1,18:PRINT #6;"YOU HAVE ";M;
    " P   "
110 POSITION 1,19:PRINT #6;"ANOTHER SPIN ?"
120 GET #2,A
130 IF A=32 THEN GOTO 120
140 IF A=89 THEN GOTO 50
150 POSITION 1,20:PRINT #6;"YOU TAKE
    HOME ";M;" P"
160 FOR I=1 TO 1000:NEXT I
165 STOP

170 REM RESULT
180 X=INT(RND(0)*4)*10+1
190 Y=INT(RND(0)*4)*10+1
200 Z=INT(RND(0)*4)*10+1
210 RETURN
```

```
220 REM SPIN
230 S=INT(RND(0)*2)+1
240 FOR I=0 TO S*10
250 FOR Q=0 TO 7
260 POKE CH+(ASC("$")-32)*8+Q,C(Q+X)
270 POKE CH+(ASC("%")-32)*8+Q,C(Q+Y)
280 POKE CH+(ASC("&")-32)*8+Q,C(Q+Z)
290 NEXT Q
310 POSITION 7,16:PRINT #6;"$ % &"
330 IF X=40 THEN X=0
340 IF Y=40 THEN Y=0
350 IF Z=40 THEN Z=0
360 X=X+1:Y=Y+1:Z=Z+1
370 NEXT I
380 X=X-1:Y=Y-1:Z=Z-1
390 RETURN

410 REM PAYOUT
420 IF X=1 AND Y=1 AND Z=1 THEN
    GOSUB 740:RETURN
430 IF (X=11)+(Y=11)+(Z=11)=2 THEN M=M+25:
    GOSUB 2000
440 IF (X=21)+(Y=21)+(Z=21)=2 THEN M=M+10:
    GOSUB 2000
450 IF (X=1)+(Y=1)+(Z=1)=1 THEN M=M+5:
    GOSUB 2000
460 RETURN

470 GRAPHICS 1+16
480 CH=(PEEK(106)-8)*256
490 CHORG=(PEEK(756)*256)
500 FOR I=0 TO 511
510 POKE CH+I,PEEK(CHORG+I)
520 NEXT I
530 GOSUB 1000

540 GRAPHICS 1+16:POKE 756,CH/256
550 POSITION 5,1:PRINT #6;"F R U I T"
560 POSITION 4,2:PRINT #6;"M A C H I N E"
570 POSITION 3,4:PRINT #6;"YOU HAVE 1.00
    TO        GAMBLE"
580 POSITION 3,6:PRINT #6;"EACH SPIN
    COSTS          .10"
590 POSITION 5,9:PRINT #6;"! ! ! WINS 50P"
600 POSITION 5,10:PRINT #6;"@ @ - WINS 25P"
610 POSITION 5,11:PRINT #6;"- [ [ WINS 10P"
620 POSITION 5,12:PRINT #6;"! - - WINS 5P"
625 RETURN
```

```
 630 DATA 06,10,20,36,68,207,239,230,0,0
 640 DATA 02,12,28,56,56,28,12,02,0,0
 650 DATA 24,60,60,60,126,255,24,24,0,0
 660 DATA 12,24,124,255,255,255,126,60,0,0
 670 DATA 06,10,20,36,68,207,239,230,0,0
 680 DIM C(48)
 690 FOR I=1 TO 48
 700 READ C:C(I)=C
 710 NEXT I
 730 RETURN

 740 REM JACK
 750 FOR I=50 TO 200 STEP 8
 760 GOSUB 2000
 770 NEXT I
 780 M=M+50
 790 RETURN

 800 REM BUST
 810 POSITION 0,20:PRINT #6;"YOU ARE BROKE"
 820 FOR I=1 TO 1000:NEXT I
 825 STOP

1000 FOR Q=0 TO 7
1010 POKE CH+(ASC("$")-32)*8+Q,C(Q+1)
1015 POKE CH+(ASC("!")-32)*8+Q,C(Q+1)
1020 POKE CH+(ASC("%")-32)*8+Q,C(Q+10)
1025 POKE CH+(ASC("@")-32)*8+Q,C(Q+10)
1030 POKE CH+(ASC("&")-32)*8+Q,C(Q+20)
1035 POKE CH+(ASC("[")-32)*8+Q,C(Q+20)
1050 NEXT Q

1060 POKE 756,CH/256
1070 POKE 752,1
1080 OPEN #2,4,0,"K:"
1090 RETURN

2000 SOUND 0,80,10,8
2010 FOR T=1 TO 50
2030 NEXT T
2040 SOUND 0,0,0,0
2050 RETURN
```

20
Attack Squash



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

How to play

At the start of the game the bat is at the bottom of the screen and in the centre. You control the left and right movement of the bat by pressing the appropriate arrow keys. Every time you hit the ball the position of the bat changes – it moves nearer to the top of the screen

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

Subroutine structure

15	Sets up variables and graphics characters
150	Sets up court and balls
200	Main play loop
450	Draws court
620	Bounce routine
770	Moves bat up screen
890	End of game – prints final score and offers another game
1010	Subroutine to move bat
1200	Blanks left of bat
1250	Blanks right of bat
1500	Defines graphics character for solid block
1600	Defines graphics character for ball
1700	Change to RAM character set
2000	Sound routine for hit
2500	Sound routine for miss

Programming details

This program is a fairly straightforward application of the Atari's user-defined graphics. Only two new characters are defined – the solid block for the court sides and the round ball shape so you should be able to follow the details quite easily. Another feature to note is the way the sound is used to make the bounce of the ball seem more positive. If you want to see the effect of leaving the sound out change line 2000 to RETURN.

Program

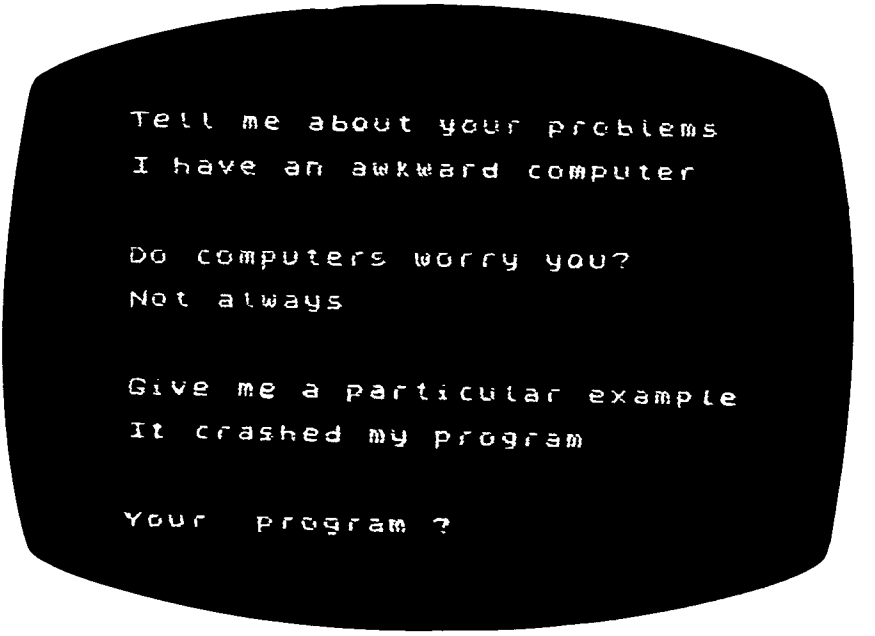
```
 10 REM ATTACK SQUASH

 15 DIM B$(4)
 20 GRAPHICS 1+16
 40 CH=(PEEK(106)-8)*256
 50 CHORG=(PEEK(756)*256)
 60 FOR I=0 TO 511
 70 POKE CH+I,PEEK(CHORG+I)
 80 NEXT I
 90 GOSUB 1500
100 H=0
110 HT=0
120 D=19
130 BALL=0
140 C=2

150 GOSUB 450
190 X=10

200 BALL=BALL+1
210 IF BALL>10 THEN GOTO 890
220 A=10+INT(RND(0)*6)
230 B=2
240 V=1
250 W=1
260 Y=D
270 POSITION 3,21
280 PRINT #6;"BALL ";BALL;
290 GOSUB 1010
300 POSITION X,Y:PRINT #6;B$
310 POSITION 12,21
320 PRINT #6;"HIT ";HT
340 GOSUB 620
360 IF B+W<>Y THEN Y=D:GOTO 290
370 GOSUB 2500
380 POSITION X,Y
390 PRINT #6;"   ";:REM 3
400 GOSUB 620
410 POSITION A,B:PRINT #6;" "
420 IF D<19 THEN D=D+1
430 H=0
440 GOTO 200
```

```
450 REM COURT
480 FOR I=0 TO 19
490 POSITION I,0:PRINT #6;"$";
500 NEXT I
510 FOR I=0 TO 19
520 POSITION 0,I
530 PRINT #6;"$";
540 POSITION 19,I
550 PRINT #6;"$";
560 NEXT I
570 RETURN

620 REM BOUNCE
650 POSITION A,B
660 PRINT #6;" "
670 A=A+V
680 B=B+W
690 IF A=18 OR A=1 THEN V=-V:GOSUB 2000
700 IF B=1 THEN W=-W:GOSUB 2000
710 IF B+W=Y THEN GOTO 770
740 POSITION A,B
750 PRINT #6;CHR$(5+128)
760 RETURN

770 R=A-X
780 IF R<0 OR R>2 THEN GOTO 740
790 W=-W
800 GOSUB 2000
810 H=H+1
820 HT=HT+1
830 IF H<>1 THEN GOTO 720
840 H=0
850 IF D>3 THEN D=D-1
860 POSITION X,Y
870 PRINT #6;"   ";:REM 3
880 GOTO 760
```

```
 890 FOR Z=1 TO 1000:NEXT Z
 900 GRAPHICS 0:POSITION 10,10
 910 PRINT ;"You Scored ";HT
 950 PRINT "ANOTHER GAME Y/N ";
 960 DIM A$(1)
 970 INPUT A$
 980 IF A$="Y" THEN RUN
 990 PRINT ""
1000 END

1010 K=PEEK(764)
1040 IF K=7 AND X<16 THEN GOTO 1200
1050 IF K=6 AND X>1 THEN GOTO 1250
1060 RETURN

1200 POSITION X,Y
1210 PRINT #6;" ";
1220 X=X+1
1240 RETURN

1250 POSITION X+2,Y
1260 PRINT #6;" ";
1270 X=X-1
1290 RETURN

1500 POKE CH+(ASC("$")-32)*8+0,255
1510 POKE CH+(ASC("$")-32)*8+1,255
1520 POKE CH+(ASC("$")-32)*8+2,255
1530 POKE CH+(ASC("$")-32)*8+3,255
1540 POKE CH+(ASC("$")-32)*8+4,255
1550 POKE CH+(ASC("$")-32)*8+5,255
1560 POKE CH+(ASC("$")-32)*8+6,255
1570 POKE CH+(ASC("$")-32)*8+7,255

1600 POKE CH+(ASC("%")-32)*8+0,60
1610 POKE CH+(ASC("%")-32)*8+1,126
1620 POKE CH+(ASC("%")-32)*8+2,255
1630 POKE CH+(ASC("%")-32)*8+3,255
1640 POKE CH+(ASC("%")-32)*8+4,255
1650 POKE CH+(ASC("%")-32)*8+5,255
1660 POKE CH+(ASC("%")-32)*8+6,126
1670 POKE CH+(ASC("%")-32)*8+7,60
```

```
1700 POKE 756,CH/256
1710 POKE 752,1
1720 B$="$$$"
1730 RETURN

2000 SOUND 0,80,10,8
2010 FOR T=1 TO 10:NEXT T
2020 SOUND 0,0,0,0
2030 RETURN

2500 FOR I=1 TO 10
2510 SOUND 0,160,10,10
2520 GOSUB 2000
2530 NEXT I
2540 RETURN
```

21
Smalltalker



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

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

How to use this program

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

Typing tips

Both when typing this program in and when using it, do be careful about your spelling. If you type in either the initial program or subsequent responses with mis-spellings the computer won't recognise your messages and you won't receive any sensible answers. The apostrophe is the only punctuation mark that should be used in dialogues with the Atari. Type in lower case with capital letters at the beginning of your sentences.

This is by far the longest program in this collection and if you are using a 16K Atari 400 it will fit only as long as there are no disk drives interfaced to the computer.

Subroutine structure

20	Main program loop
1000	Initial message and set up
2000	Input human's sentence
3000	Divides sentences into words
3600	Changes tense/pronouns
3800	Tense/pronouns data
5000	Finds keywords in sentence
6000	Keywords data
7000	Keyword responses
9800	Prints computer's response
9900	Requests sensible input

Programming details

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

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

Scope for improvement

If you wish to add to the list of keywords that the computer recognises, you need to notice how, in subroutine 6000, the keywords are paired with the line number of the subroutine that responds to them. It is also important to be aware of the priorities assigned to each keyword. If two keywords are present in a sentence then the one first in the list will be acted upon.

Program

```
 10 REM SMALLTALKER

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

```
1000 PRINT CHR$(125)
1005 OPEN #2,4,0,"K:"
1010 POSITION 2,1:PRINT "Hi there!"
1030 PRINT
1040 PRINT "I would like you to talk to me"
1050 PRINT "but I don't have ears so will"
1060 PRINT "you type sentences on my"
1070 PRINT "keyboard"
1074 PRINT
1075 PRINT "Don't use any punctuation apart"
1076 PRINT "from apostrophies which are"
1078 PRINT "important"
1080 PRINT
1110 PRINT "When you have finished typing"
1115 PRINT "press RETURN"
1120 POSITION 2,18:PRINT "Tell me about your
     problems"
1125 DIM P$(100),R$(50),M$(50),D$(50),C$(50)
1130 R$=""
1140 M$=""
1150 D$=""
1160 DIM N$(3*32)
1165 FOR I=1 TO 3*32:N$(I)=" ":NEXT I
1170 N$(1,32)="Please go on"
1180 N$(33,64)="I'm not sure I understand you"
1190 N$(65,96)="Tell me more"
1200 DIM I$(3*40)
1205 FOR I=1 TO 3*32:I$(I)=" ":NEXT I
1210 I$(1,40)="Let's talk some more about your"
1220 I$(41,80)="Earlier you spoke of your "
1230 I$(81,120)="Does that have anything to
     do with your "
1235 DIM J$(2*32)
1236 FOR I=1 TO 2*32:J$(I)=" ":NEXT I
1240 J$(1,32)="Are you just being negative"
1250 J$(33,64)="I see"
1260 DIM A$(100),B$(100),Z$(100)
1280 DIM T$(50)
1290 DIM W(20,2)
1990 RETURN
```

```
2000 A$=" "
2005 POKE 752,1
2010 GET #2,B
2030 IF B=155 THEN GOTO 2200
2035 IF B=126 AND LEN(A$)=1 THEN A$=" ":
     GOTO 2090
2040 IF B=126 AND LEN(A$)>1 THEN
     A$=A$(1,LEN(A$)-1):GOTO 2090
2050 IF B<32 OR B>123 THEN GOTO 2000
2060 A$(LEN(A$)+1)=CHR$(B)
2090 POSITION 2,20:PRINT A$;" ";
2100 GOTO 2010
2200 IF A$=" " THEN GOTO 2000
2201 IF A$(LEN(A$),LEN(A$))=";" THEN
     B$=A$(1,LEN(A$)-1):A$=B$:GOTO 2200
2202 IF A$=R$ THEN POSITION 0,21:PRINT
     "You're repeating  yourself!":PRINT :
     PRINT :GOTO 2000
2203 R$=A$
2204 IF A$=" " THEN GOTO 2000
2215 IF ASC(A$(2,2))<97 THEN A$(2,2)=
     CHR$(ASC(A$(2,2))+32)
2220 POSITION 0,21:PRINT
2230 RETURN

3000 FOR I=1 TO 20
3002 W(I,1)=0:W(I,2)=0
3004 NEXT I
3005 N=1
3010 B=0
3020 FOR I=1 TO LEN(A$)
3040 IF (A$(I,I)=" " OR A$(I,I)=",") AND B=0
     THEN B=1
3050 IF (A$(I,I)<>" " AND A$(I,I)<>",") AND
     B<=1 THEN W(N,1)=I:B=2
3060 IF (A$(I,I)=" " OR A$(I,I)=",") AND B=2
     THEN W(N,2)=I-1:N=N+1:B=0
3070 NEXT I
3080 W(N,2)=LEN(A$)
3085 A$(LEN(A$)+1)="   ":REM 3 SPACES
```

```
3600 FOR I=1 TO N
3605 RESTORE 3800
3610 READ B$
3620 IF B$="s" THEN GOTO 3690
3630 IF B$<>A$(W(I,1),W(I,2)) THEN GOTO 3680
3640 READ C$
3650 Z$=A$(1,W(I,1)-1):Z$(LEN(Z$)+1)=C$:
     Z$(LEN(Z$)+1)=A$(W(I,2)+1)
3654 A$=Z$
3655 W(I,2)=W(I,2)+LEN(C$)-LEN(B$)
3656 FOR J=I+1 TO N
3660 W(J,2)=W(J,2)+LEN(C$)-LEN(B$)
3664 W(J,1)=W(J,1)+LEN(C$)-LEN(B$)
3665 NEXT J
3670 GOTO 3690
3680 READ B$
3685 GOTO 3610
3690 NEXT I
3700 RETURN

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

```
5000 RESTORE 6000
5010 READ B$,NUM
5020 IF B$="s" THEN GOTO 5710
5025 FOR I=1 TO N
5030 IF A$(W(I,1),W(I,2))<>B$ THEN GOTO 5700
5040 T$=A$(W(I,2)+1)
5050 RETURN
5700 NEXT I
5705 GOTO 5010
5710 NUM=0
5720 IF M$<>"" THEN GOTO 5800
5730 Z=INT(RND(0)*3):P$=N$(Z*32+1,Z*32+32)
5740 RETURN
5800 Z=INT(RND(0)*3):P$=I$(Z*32+1,X*32+32)
5805 P$(LEN(P$)+1)=M$
5900 RETURN

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

```
7000 P$="Do computers worry you?"
7010 RETURN
7100 P$="In what way ?"
7110 RETURN
7200 P$="Why talk of possibilities"
7210 RETURN
7300 P$="Really "
7305 P$(LEN(P$)+1)=B$
7306 P$(LEN(P$)+1)=" ?"
7310 RETURN
7400 IF I=N THEN GOTO 7450
7410 I=I+1
7420 IF A$(W(I,1),W(I,2))="one" THEN
     B$(LEN(B$)+1)=" one":GOTO 7300
7450 Z=INT(RND(0)*2):P$=J$(Z*32+1,Z*32+32)
7460 RETURN
7500 IF I=N THEN GOTO 9900
7510 I=I+1
7515 IF I>N THEN GOTO 5720
7520 IF A$(W(I,1),W(I,2))<>"you*" THEN
     GOTO 7550
7530 P$="What if you were "
7535 P$(LEN(P$)+1)=A$(W(I,2)+1)
7536 P$(LEN(P$)+1)=" ?"
7540 RETURN
7550 IF A$(W(I,1),W(I,2))<>"I*" THEN GOTO 5720
7560 P$="Would you like to believe I was "
7565 P$(LEN(P$)+1)=A$(W(I,2)+1)
7570 RETURN
7600 I=I+1
7605 IF I>N THEN GOTO 7450
7610 IF A$(W(I,1),W(I,2))="Mother" THEN
     GOTO 7700
7620 IF A$(W(I,1),W(I,2))="Father" THEN
     GOTO 7700
7630 IF A$(W(I,1),W(I,2))="sister" THEN
     GOTO 7700
7640 IF A$(W(I,1),W(I,2))="brother" THEN
     GOTO 7700
7650 IF A$(W(I,1),W(I,2))="wife" THEN
     GOTO 7700
7660 IF A$(W(I,1),W(I,2))="husband" THEN
     GOTO 7700
7670 IF A$(W(I,1),W(I,2))="children" THEN
     GOTO 7700
```

```
7680 IF LEN(T$)>10 THEN M$=T$
7690 P$="your "
7692 P$(LEN(P$)+1)=T$
7694 P$(LEN(P$)+1)=" ? "
7695 RETURN
7700 P$="Tell me more about your family"
7710 RETURN
7800 P$="Give me a particular example"
7810 RETURN
7900 I=I+1
7905 IF I>N THEN P$="Am I what ?":RETURN
7920 P$="Why are you interested in whether
     I am "
7924 P$(LEN(P$)+1)=A$(W(I,1))
7926 P$(LEN(P$)+1)=" or not?"
7930 RETURN
8000 P$="Do you think you are "
8010 P$(LEN(P$)+1)=A$(W(I,2))
8030 RETURN
8100 P$="Why do you ask ?"
8110 RETURN
8150 P$="Tell me about any other reasons"
8160 RETURN
```

```
8200 I=I+1
8205 IF I>N THEN P$="What ?":RETURN
8210 IF A$(W(I,1),W(I,2))="I*" THEN
     P$="Do you believe I can":P$(LEN(P$)+1)=
     A$(W(I,2)+1):RETURN
8220 IF A$(W(I,1),W(I,2))="you*" THEN P$=
     "Do you believe you can ":P$(LEN(P$)+1)=
     A$(W(I,2)+1):RETURN
8230 GOTO 5720
8250 P$="You seem very positive"
8260 RETURN
8300 P$="Pleased to meet you - let's talk
     about your problems"
8310 RETURN
8350 P$="Could you try to be more positive"
8360 RETURN
8370 P$="Why are you concerned about my"
8375 P$(LEN(P$)+1)=T$
8380 RETURN
8400 P$="You don't have to apologise to me"
8410 RETURN
8450 P$="Some questions are difficult to
     answer..."
8460 RETURN
8500 I=I+1
8505 IF I>N THEN GOTO 5720
8510 P$="I am sorry to hear that you are":
     P$(LEN(P$)+1)=A$(W(I,1),W(I,2))
8520 IF A$(W(I,1),W(I,2))="sad" THEN RETURN
8530 IF A$(W(I,1),W(I,2))="unhappy" THEN
     RETURN
8540 IF A$(W(I,1),W(I,2))="depressed" THEN
     RETURN
8550 IF A$(W(I,1),W(I,2))="sick" THEN RETURN
8560 P$="How have I helped you to be ":
     P$(LEN(P$)+1)=A$(W(I,1),W(I,2))
8570 IF A$(W(I,1),W(I,2))="happy" THEN RETURN
8580 IF A$(W(I,1),W(I,2))="elated" THEN RETURN
8590 IF A$(W(I,1),W(I,2))="glad" THEN RETURN
8600 IF A$(W(I,1),W(I,2))="better" THEN RETURN
8610 P$="Is it because you are "
8615 P$(LEN(P$)+1)=A$(W(I,1))
8616 P$(LEN(P$)+1)=" you would like to talk
     to me"
8620 RETURN
```

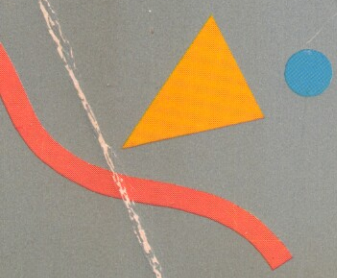
```
8650 IF I=1 THEN GOTO 8655
8654 IF A$(W(I-1,1),W(I-1,2))="are*" THEN
     GOTO 8000
8655 I=I+1
8656 IF I>N THEN GOTO 9900
8660 IF A$(W(I,1),W(I,2))="are*" THEN GOTO 8500
8670 P$="what would it mean if you got ":
     P$(LEN(P$)+1)=A$(W(I,2)+1)
8675 IF A$(W(I,1),W(I,2))="want" OR
     A$(W(I,1),W(I,2))="need" THEN RETURN
8680 P$="How do you know you can't":
     P$(LEN(P$)+1)=A$(W(I,2)+1)
8685 IF A$(W(I,1),W(I,2))="can't" OR
     A$(W(I,1),W(I,2))="cannot" THEN RETURN
8690 IF A$(W(I,1),W(I,2))="feel" THEN P$=
     "tell me more about hou you feel":RETURN
8700 GOTO 5720
8800 IF I-1<1 THEN GOTO 8805
8804 IF A$(W(I,1),W(I,2))="am" THEN GOTO 7790
8805 I=I+1
8810 IF I>=N THEN P$="What am I ?":RETURN
8820 IF A$(W(I,1),W(I,2))="am" THEN
     P$="Why do you think so ?":RETURN
8830 P$="Is that what you think of me !"
8840 RETURN

9800 FOR J=1 TO LEN(P$)
9810 IF P$(J,J)<>"*" THEN PRINT P$(J,J);
9820 NEXT J
9830 POSITION 0,23:PRINT :PRINT :PRINT :PRINT
9840 RETURN

9900 P$="Please talk sensibly !"
9910 RETURN
```

THE ATARI® BOOK OF GAMES

Here are twenty-one exciting, challenging games especially written for all ATARI models, in full color and sound. Each game is listed in its entirety with a description of the object of the game and details of how to play it. Moving graphics are featured in practically all the games. Nine Hole Golf, Treasure Island and Save the Whale all use clever graphics, while Rainbow Squash, Bobsleigh and Laser Attack exploit the ATARI's speed. Guideline and Commando Jump test your skill and reaction times and if you want to try outsmarting the ATARI, try Capture the Quark. Tell your ATARI all about your problems with Smalltalker and it will answer you back.

These games are written in BASIC and show how much can be achieved without resorting to machine code. Not only does each program come with an explanation of how to play the game and how it works, but you are also given tips on how to personalize it for your own special use or change it as your skill improves.

About the Authors—
Mike James, S M Gee and Kay Ewbank have collaborated on many programming projects and have written many successful books between them. All of them are regular contributors to *Electronics and Computing Monthly* and other popular magazines.



ISBN 0-88134-159-2