THE DRAGON 32 BOOK OF GAMES

MIKE JAMES,
S.M.GEE AND KAY EWBANK

The Dragon 32 Book of Games

The Dragon 32 Book of Games

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

GRANADA
London Toronto Sydney New York

Granada Publishing Limited - Technical Books Division
Frogmore, St Albans, Herts AL2 2NF
and
36 Golden Square, London W1R 4AH
515 Madison Avenue, New York, NY 10022, USA
117 York Street, Sydney, NSW 2000, Australia
60 International Boulevard, Rexdale, Ontario R9W 6J2, Canada
61 Beach Road, Auckland, New Zealand

British Library Cataloguing in Publication Data
James, M.
The Dragon 32 book of games
1. Electronic games 2. Dragon 32 (Computer)—Programming
I. Title II. Gee, S. M. III. Ewbank, K.
794 GV1469.2

ISBN 0-246-12102-5

First published in Great Britain 1983 by Granada Publishing

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

Contents

Introduction

This book is a collection of twenty-one games written to be played on your Dragon 32. (They are also suitable for use on a Tandy Color Computer.) Every game is complete in itself, so you can turn to whichever one takes your fancy, type it in and play it. We've tried to include something for everyone and each one has its own detailed description so that you'll know what to expect before you embark on it. You also have a chance to *see* what to expect as there are samples of the displays produced on your TV screen. Of course these cannot really do justice to many of the programs which use colour graphics – and we cannot find any way of letting you hear the accompanying sound effects.

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, Rainbow Squash and Bobsleigh. Others have titles that probably won't ring any bells – Sheepdog Trials, Commando Jump and Across the Ravine, but we hope they will soon become popular once you start to play them. Laser Attack and Mighty Missiles 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. Corner the Dragon 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. Dragontalk is a rather unusual program that enables your Dragon to hold a *conversation* with you. If you think we are joking you'll have to try it for yourself.

Improve your programming

This book isn't however intended as just another collection of programs. As well as hoping to provide programs that you can have hours of fun with, we also hope to cater for all Dragon owners by presenting a book that can be used in more than one way. True, you can use it simply as a source of exciting games programs, but on the other hand you can use it to further improve your own knowledge of Dragon 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 is because we would like Dragon owners to be able to experiment with programming that the games (with one exception) use 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 cartridges 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. Bobsleigh does incorporate some machine code within its BASIC program and, as you may like to use similar routines in your own programs, details of the machine code instructions used are given.

Once you've run some of the programs in this book, you may be surprised to discover what can actually be achieved from BASIC. These programs do manage to produce some surprisingly good effects. So if you have been struggling to make your own games fast moving or graphically interesting then try gleaning some useful tips from the listings that follow.

Perfect programming

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

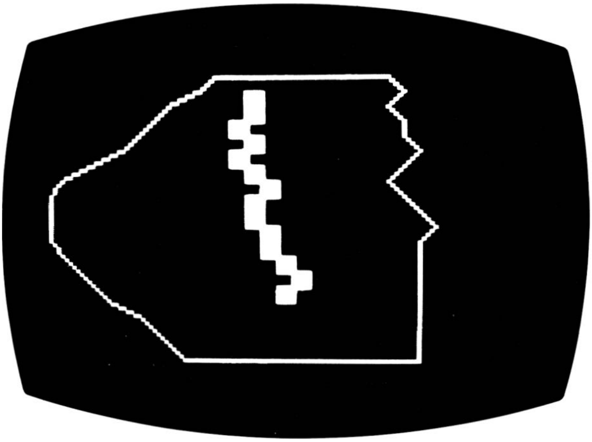
However, it's a well-known fact that bugs creep in whenever you enter a program – so if a program won't work when you've typed it in check it very carefully against the listing and if it still won't work have a cup of tea and check again – it's all too easy to read what you think should be there rather than what is there! If there are any particular points to look out for when entering a program you'll be alerted to them in the section on 'Typing tips'. To help you we've sometimes included REM statements that tell you how many spaces are needed when it's difficult to see by eye. Common sources of possible 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.

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 many 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.

I
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 hostile natives, gold buried at a concealed location and even Long John Silver's parrot. The great danger however, is that the pirate ship is sailing towards the island at great speed and once the pirates land you are certain to be captured. If you do find the gold you are rewarded by a fanfare and a display of colours culminating in gold itself. The game is displayed in colour graphics and has sound effects as well.

How to play

At the beginning of the game you are shown the map of the island. Your position is on the first square of the path and the gold is hidden at the last square. You have to follow the path exactly. If you stray there are two 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 two paces back from where you left it. 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 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, and comes nearer every time you stray from the path – so you need to be accurate. The pirate ship advances by a random distance each time it moves, so its difficult to predict how soon it will land. To move along the path use the right, left and forward arrow keys – you cannot move backwards.

Subroutine structure

20	Set-up routine
160	Main play loop
250	Logic for natives' attack
450	Draws native
490	Logic for parrot's help
640	Draws parrot
660	Prints map
850	Moves man
980	Draws man
1030	Calculates island
1210	Calculates path
1350	Draws pirate ship approaching
1490	Pirates arrive routine
1530	Draws island (without map)
1660	Help routine
1740	Treasure found routine
1820	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 1030. There the use of SGN function results in the contour of the island first going out and then coming in at random, giving an island-like shape. Three separate graphics screen displays are used in this game. One is used for the map, another is used to show the man on the island, while the third shows the pirate ship approaching the island – the island being at the centre of the display. The ability to swap between screen displays saves a great deal of time as it means that the island shape only has to be PAINTed once.

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 loop in line 1680. You might also chose to alter the amount that the pirate ship moves at each step by setting a different value for R in line 1370. Currently it is set randomly to a value between 1 and 10.

Program

```
 10 REM TREASURE ISLAND
 20 PCLEAR 8
 30 F=0
 40 XS=8
 50 YS=16
 60 MS=0
 70 DIM X(40)
 80 GOSUB 1030
 90 Q=1
100 YM=T+16
110 XM=X(1)
120 GOSUB 660
130 GOSUB 1530
140 SCREEN 1,0
150 GOSUB 980
```

```
160 GOSUB 850
170 IF XM=XT AND YM=YT THEN GOTO 1740
180 F=F+1
190 IF INT(F/5)=F/5 THEN GOSUB 1350
200 IF ABS(X((YM-T-8)/8)-XM)<8 THEN
    GOTO 160
210 SOUND 100,5
220 GOSUB 1350
230 PMODE 1,3
240 SCREEN 3,0

250 REM NATIVE OR PARROT
260 IF RND(0)<=.5 THEN GOTO 490
270 R=0
280 IF YM>YT THEN R=RND(3)*8+16
290 W$=STR$(YM+8)
300 Q$=STR$(X((YM-T-R)/8))
310 C$="C2"
320 GOSUB 460
330 CM$="C1"
340 GOSUB 990
350 YM=YM-24-R
360 IF YM<=T+16  THEN YM=T+16
370 XM=X((YM-T-8)/8)
380 CM$="C4"
390 GOSUB 990
400 FOR Q=1 TO 500:NEXT Q
410 C$="C1"
420 GOSUB 460
430 MS=1
440 GOTO 160

450 SCREEN 1,0
460 DRAW "BM"+Q$+","+W$+C$+"E4F4H4U2R
    2L2U1D1L4U3D7"
470 FOR Q=1 TO 500:NEXT Q
480 RETURN
```

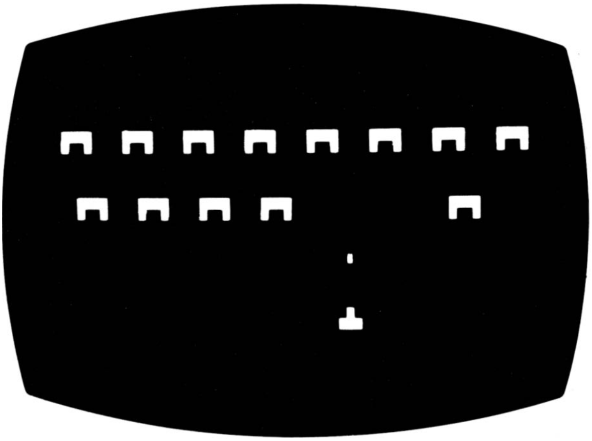
```
490 REM PARROT
500 GOSUB 980
510 YJ=YM+8
520 IF YJ>P THEN YJ=P
530 XJ=X((YJ-T-8)/8)
540 CJ$="C2"
550 YJ$=STR$(YJ)
560 XJ$=STR$(XJ)
570 GOSUB 640
580 MS=1
590 GOSUB 850
600 CJ$="C1"
610 GOSUB 640
620 GOSUB 980
630 GOTO 170

640 DRAW "BM"+XJ$+","+YJ$+CJ$+"E6G2H2F4"
650 RETURN

660 PMODE 1,1:SCREEN 1,0:PCLS 3
670 COLOR 2,1
680 LINE(L(1),T)-(R(1),T),PSET
690 S=0
700 FOR Y=T TO B-8 STEP 8
710 S=S+1
720 LINE(L(S),Y)-(L(S+1),Y+8),PSET
730 LINE(R(S),Y)-(R(S+1),Y+8),PSET
740 NEXT Y
750 LINE(L(S+1),B)-(R(S+1),B),PSET
760 PAINT(L(S+1)+2,B-2),1,2
770 S=0
780 FOR Y=T+16 TO P STEP 8
790 S=S+1
800 LINE(X(S),Y)-(X(S)+8,Y-8),PSET,BF
810 NEXT Y
820 COLOR 2,1
830 FOR Q=1TO 4000:NEXT Q
840 RETURN
```

```
 850 SOUND 100,5
 860 A$=INKEY$
 870 IF A$="" THEN GOTO 860
 880 CM$="C1"
 890 GOSUB 990
 900 IF A$="H" THEN GOSUB 1660:RETURN
 910 IF A$=CHR$(8) THEN XM=XM-8:GOTO 960
 920 IF A$=CHR$(9) THEN XM=XM+8:GOTO 960
 930 IF A$<>CHR$(10) THEN GOTO 850
 940 CM$="C1"
 950 GOSUB 990
 960 YM=YM+8
 970 Q=Q+1

 980 CM$="C4"
 990 YM$=STR$(YM)
1000 XM$=STR$(XM)
1010 DRAW "BM"+XM$+","+YM$+CM$+
     "E4F4H4U2R2L2U1D1L2R2D2"
1020 RETURN

1030 L=RND(3)+10
1040 L=L*8
1050 T=RND(3)*8
1060 W=80+RND(10)
1070 B=RND(2)+20
1080 B=B*8
1090 DIM L((B-T)/8+1)
1100 DIM R((B-T)/8+1)
1110 S=0
1120 FOR Y=T TO B STEP 8
1130 S=S+1
1140 L(S)=L
1150 R(S)=L+W
1160 L=L-(SGN(90-Y)*RND(2)*8)
1170 W=W+(SGN(90-Y)*RND(2)*8)
1180 IF L(S)<10 THEN L(S)=10
1190 IF R(S)>250 THEN R(S)=250
1200 NEXT Y
```

```
1210 K=T+8
1220 S=1
1230 X(S)=L(T/8)+RND(5)*8+8
1240 FOR P=K TO B-RND(4)*8 STEP 8
1250 S=S+1
1260 X(S)=X(S-1)+(2-RND(3))*8
1270 IF X(S)>=R(S)-16 THEN X(S)=R(S)-16
1280 IF X(S)<=L(S)+8 THEN X(S)=L(S)+8
1290 P1=P
1300 NEXT P
1310 P=P1-8
1320 XT=X((P-T-8)/8)
1330 YT=P
1340 RETURN

1350 PMODE 1,5
1360 PCLS 3
1370 R=RND(10)
1380 XS=XS+R:XS$=STR$(XS)
1390 YS=YS+R:YS$=STR$(YS)
1400 CIRCLE(125,90),2,2
1410 DRAW "BM"+XS$+","+YS$+
     "C2R5E5L15F5BR7U3"
1420 LINE(XS-3,YS-13)-(XS+7,YS-8),PSET,B
1430 LINE(XS+1,YS-12)-(XS+5,YS-9),PSET
1440 SCREEN 1,0
1450 FOR Q=1 TO 1000:NEXT Q
1460 PMODE 1,3
1470 SCREEN 1,0
1480 IF YS<90 THEN RETURN

1490 CLS
1500 PRINT @ 100,"THE PIRATES HAVE LANDED"
1510 PRINT:PRINT"YOU ARE CAPTURED"
1520 GOTO 1820
```

```
1530 PMODE 1,3
1540 PCLS 3
1550 COLOR 2,1
1560 LINE(L(1),T)-(R(1),T),PSET
1570 S=0
1580 FOR Y=T TO B-8 STEP 8
1590 S=S+1
1600 LINE(L(S),Y)-(L(S+1),Y+8),PSET
1610 LINE(R(S),Y)-(R(S+1),Y+8),PSET
1620 NEXT Y
1630 LINE(L(S+1),B)-(R(S+1),B),PSET
1640 PAINT(L(S+1)+2,B-2),1,2
1650 RETURN

1660 PMODE 1,1
1670 SCREEN 1,0
1680 FOR Q=1 TO 500+RND(200):NEXT Q
1690 GOSUB 1350
1700 PMODE 1,3
1710 SCREEN 1,0
1720 GOSUB 980
1730 RETURN

1740 CLS:SCREEN 0,0
1750 FOR C=1 TO 8
1760 CLS C
1770 FOR Q=1 TO 50:NEXT Q
1780 SOUND C*10,5
1790 NEXT C
1800 PRINT @ 100,"YOU FOUND THE GOLD"
1810 FOR Q=1 TO 500:NEXT Q

1820 INPUT "ANOTHER GAME Y/N";A$
1830 IF LEFT$(A$,1)="Y" THEN RUN
1840 CLS
1850 END
```

2
Alien Invaders



Two ranks 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. Once the back row of advancing ships pass the point of no return the game is over – you will be anihilated. Your only hope is to wipe out the aliens with your missiles. The problem is that the aliens move constantly from side to side and are therefore very difficult targets to hit. 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 yellow on a black background. This is however a combination that shows up well in black and white.

How to play

You can move your missile launcher to left and right using the arrow keys. Press the up arrow to fire a missile. To win this game you need to decide quickly whether the missile you have launched will score a hit. If you decide it's going to miss altogether you should re-fire. If you fire another missile before the one already in flight reaches its target, the first missile will abort and the second missile will be active. The game is over when you have destroyed all the invaders or when the second row gets too close to you.

Subroutine structure

20	Main play loop
150	Set-up invader string
260	Set-up initial screen
330	Move and fire logic
420	Fires missile
480	Moves missile and tests for hit
540	Explosion routine
630	Removes hit invader
720	Prints invader string
740	Moves invaders from side to side
830	End of game

Programming details

This game relies entirely on low resolution graphics in order to make it possible to program in BASIC. The invaders are held in one long string, L$, and every time one is hit the string is manipulated so that the old invader shape is replaced by blanks. The tension of this game is intensified by the constant throbbing noise which is programmed in lines 770 to 780. The explosion routine in subroutine 540 is an interesting use of low resolution graphics that you may like to utilise in your own programs. Random coloured blocks are displayed in place of the destroyed invader and the explosion is also marked by a random sound effect.

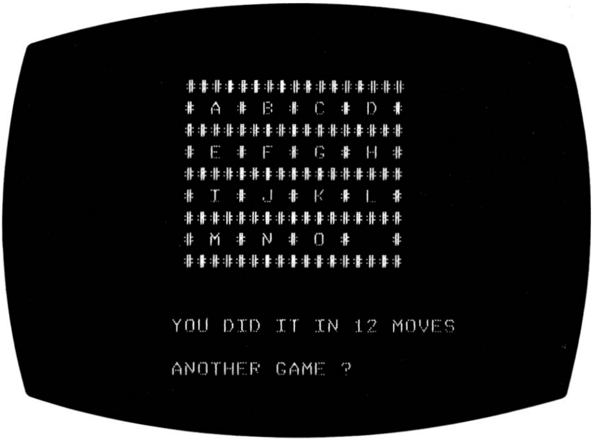
Program

```
 10 REM INVADERS
 20 CLEAR 2000
 30 GOSUB 150
 40 GOSUB 260
 50 GOSUB 330
 60 GOSUB 280
 70 IF F=1THEN GOSUB 480 ELSE FOR Q=1 TO 20:
    NEXT Q
 80 IF H=1 THEN GOSUB 540
 90 M=M+1
100 IF M>4 THEN M=0:GOSUB 740
110 IF RND(0)>.98 THEN PRINT @L,B$;:
    L=L+32:GOSUB 720
120 IF L>319 THEN GOSUB 800
130 IF L$=B$ THEN GOTO 890
140 GOTO 50

150 I$=CHR$(158)+CHR$(157)+CHR$(128)
160 FOR I=1 TO 8
170 L$=L$+I$+CHR$(128)
180 NEXT I
190 E=0
200 L$=L$+STRING$(65,128)+LEFT$(L$,LEN(L$)-2)
210 X=31
220 B$=STRING$(127,128)
230 K=0
240 L=32
250 RETURN

260 CLS 0
270 PRINT @L,L$;
280 SET(X,31,2)
290 SET(X+1,31,2)
300 SET(X+2,31,2)
310 SET(X+1,30,2)
320 RETURN
```

```
330 X$=INKEY$
340 IF X$="" THEN RETURN
350 IF X$=CHR$(94) THEN GOTO 420
360 RESET(X,31)
370 RESET(X+1,30)
380 RESET(X+2,31)
390 IF X$=CHR$(8)  AND X>2THEN X=X-1
400 IF X$=CHR$(9) AND X<60 THEN X=X+1
410 RETURN

420 RESET(XM,YM)
430 XM=X+1
440 YM=29
450 SET(XM,YM,0)
460 F=1
470 RETURN

480 RESET(XM,YM)
490 YM=YM-1
500 IF YM=0 THEN F=0:RETURN
510 IF POINT(XM,YM)=2 THEN H=1:RETURN
520 SET(XM,YM,0)
530 RETURN

540 FOR Q=1 TO 10
550 SET(XM+2-RND(4),YM+1-RND(2),RND(5))
560 SOUND RND(255),1
570 NEXT Q
580 FOR I=XM-2 TO XM+1
590 FOR J=YM-1 TO YM
600 RESET(I,J)
610 NEXT J
620 NEXT I

630 IF E=1 THEN RETURN
640 F=0
650 H=0
660 D=INT(XM/2)+INT(YM/2)*32-L
670 IF D=1 THEN D=2
680 L$=LEFT$(L$,D-2)+CHR$(128)+CHR$(128)+CHR$
    (128)+CHR$(128)+RIGHT$(L$,LEN(L$)-D-2)
690 GOSUB 720
700 YM=30
710 RETURN
```

```
720 PRINT @L,L$;
730 RETURN

740 IF K THEN L$=RIGHT$(L$,LEN(L$)-2)+
    CHR$(128)+CHR$(128)
750 IF NOT K THEN L$=CHR$(128)+CHR$(128)+
    LEFT$(L$,LEN(L$)-2)
760 K=NOT K
770 IF K THEN SOUND 50,1
780 IF NOT K THEN SOUND 25,1
790 GOTO 720
800 XM=X:YM=31
810 E=1
820 GOSUB 540

830 PRINT @5,"YOU LOSE ";
840 INPUT "ANOTHER GAME ";A$
850 IF A$="Y" THEN RUN
860 IF A$<>"N" THEN GOTO 840
870 CLS
880 END
890 PRINT @5,"YOU WIN ";
900 GOTO 840
```

3
Mirror Tile



```
#####################
# A # B # C # D #
#####################
# E # F # G # H #
#####################
# I # J # K # L #
#####################
# M # N # O #   #
#####################


YOU DID IT IN 12 MOVES

ANOTHER GAME ?
```

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

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

How to play

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

These same rules apply to the Dragon 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 Dragon 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 N. If you prefer to select your own starting arrangement you will be asked to type in three four-letter words, and one three-letter word. Of course, this means you can vary the difficulty of the puzzle. If you choose words that repeat some letters the game will actually be easier. For example, typing in ROOF, CATS, RENT and TENT would give a fairly simple game. The most difficult puzzle is one where every letter is different, e.g. HOME, CART, WING, SKY.

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

Typing tips

Be very careful to type in the correct number of hashes. If you type in too few or too many you will fail to display the board correctly. Notice that there are three spaces between each hash in lines 420, 440, 460 and 480. It is also vitally important not to omit the semi-colons at the end of 'PRINT@' statements (for example, line 520). Leaving these out will cause the remainder of the line to blank out.

Subroutine structure

20	Defines arrays and main play loop
160	End of game
210	Tests for end of game
290	Sets up default (alphabetic) board
410	Prints board and characters
550	Shuffle routine
690	Prints number overlay
800	Overprints numbers and wipes out any error messages
900	Gets move and tests validity
1120	Locates empty space
1310	Performs move
1360	Own words input routine
1700	Move routine continued

Programming details

This program is complicated 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

```
 10 REM MIRROR TILE
 20 DIM B(4,4)
 30 DIM B$(16)
 40 DIM W$(16)
 50 GOSUB 1190
 60 GOSUB 290
 70 IF WR=1 THEN GOSUB 1360
 80 MV=0:ER=0
 90 GOSUB 410
100 GOSUB 1120
110 GOSUB 550
120 GOSUB 900
130 GOSUB 210
140 MV=MV+1
150 IF ED<>0 THEN GOTO 120

160 PRINT @ 448,"YOU DID IT IN ";MV;
    " MOVES"
170 INPUT "ANOTHER GAME ";A$
180 IF LEFT$(A$,1)="Y" THEN RUN
190 CLS
200 STOP

210 ED=0:K=0
220 FOR I=1 TO 4
230 FOR J=1 TO 4
240 K=K+1
250 IF B$(B(I,J))<>W$(K) THEN ED=1
260 NEXT J
270 NEXT I
280 RETURN

290 K=0
300 FOR I=1 TO 4
310 FOR J=1 TO 4
320 K=K+1
330 B$(K)=CHR$(96+K)
340 W$(K)=B$(K)
350 B(I,J)=K
360 NEXT J
370 NEXT I
380 B$(16)=" "
390 W$(16)=" "
400 RETURN
```

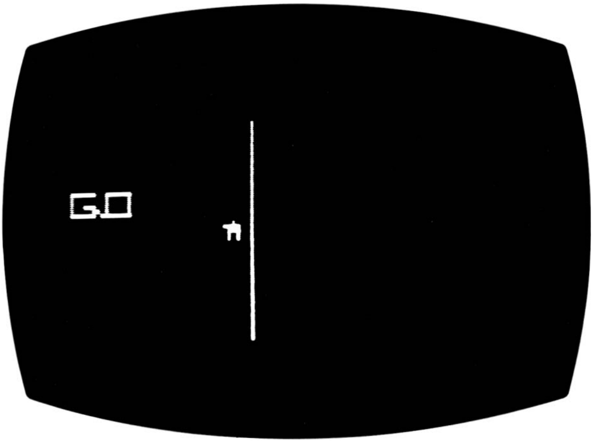
```
410 PRINT  @ 42,"#################";
420 PRINT @  74,"#   #   #   #   #";
430 PRINT @ 106,"#################";
440 PRINT @ 138,"#   #   #   #   #";
450 PRINT @ 170,"#################";
460 PRINT @ 202,"#   #   #   #   #";
470 PRINT @ 234,"#################";
480 PRINT @ 266,"#   #   #   #   #";
490 PRINT @ 298,"#################";
500 FOR I=1 TO 4
510 FOR J=1 TO 4
520 PRINT @ I*4+J*64+8,B$(B(J,I));
530 NEXT J,I
540 RETURN

550 IS=4:JS=4:IO=0:JO=0
560 FOR D=1 TO S
570 I=IS:J=JS
580 IF RND(0)>.5 THEN GOTO 620
590 I=IS+RND(2)*2-3
600 IF I>4 OR I<1 THEN I=IS:GOTO 620
610 GOTO 640
620 J=JS+RND(2)*2-3
630 IF J>4 OR J<1 THEN J=JS:GOTO 570
640 IF I=IO ANDJ=JO THEN GOTO 570
650 IO=IS:JO=JS
660 GOSUB 1310
670 NEXT D
680 RETURN

690 K=0
700 PRINT @ 10,"#################";
710 FOR I=1 TO 4
720 FOR J=1 TO 4
730 K=K+1
740 K$=STR$(K)
750 IF K<10 THEN K$=K$+" "
760 PRINT @ J*4+I*64-25,K$;
770 NEXT J
780 NEXT I
790 RETURN
```

```
 800 FOR L=1 TO 4
 810 FOR P=1 TO 4
 820 PRINT @ L*4+P*64-26,"####";
 830 NEXT P,L
 840 PRINT @ 10,"                    ";
 850 IF ER=0 THEN RETURN
 860 PRINT @ 384," "
 870 PRINT :PRINT
 880 ER=0
 890 RETURN

 900 PRINT @ 448,;
 910 INPUT "WHAT IS YOUR MOVE ";M$
 920 IF M$="" THEN GOTO 900
 930 IF LEN (M$)<2 THEN M$="0"+M$
 940 IF LEFT$(M$,1)<"0" OR LEFT$(M$,1)>"1"
     OR RIGHT$(M$,1)<"0" OR RIGHT$(M$,1)>"9"
     THEN GOSUB 690:GOTO 900
 950 M=VAL (M$)
 960 IF M>0 AND M<17 THEN GOTO 1020
 970 ER=1
 980 PRINT @ 384,"A MOVE MUST BE A NUMBER
     BETWEEN"
 990 PRINT "1 AND 16 AS SHOWN"
1000 GOSUB 690
1010 GOTO 900
1020 I=INT((M-1)/4)
1030 J=M-I*4
1040 I=I+1
1050 IF ABS(I-IS)+ABS(J-JS)=1 THEN GOTO 1310
1060 SOUND 50,3
1070 ER=1
1080 PRINT @ 384,"YOU CAN ONLY MOVE A
     TILE NEXT "
1090 PRINT "TO THE SPACE         "
     :REM 9 SPACES
1100 GOSUB 690
1110 GOTO 900
```

```
1120 K=0
1130 FOR L=1 TO 4
1140 FOR P=1 TO 4
1150 K=K+1
1160 IF B(L,P)=16 THEN IS=L:JS=P:MS=K
1170 NEXT P,L
1180 RETURN
1190 CLS
1200 PRINT @ 36, " M I R R O R  T I L E"
1210 PRINT:PRINT "DO YOU WANT TO INPUT YOUR"
1220 INPUT"OWN SET OF WORDS ";A$
1230 IF A$="" THEN GOTO 1220
1240 IF LEFT$(A$,1)="Y" THEN WR=1:GOTO 1270
1250 IF LEFT$(A$,1)="N" THEN WR=0:GOTO 1270
1260 GOTO 1220
1270 PRINT:INPUT "HOW MANY SHUFFLES ";S
1280 IF S<1 THEN GOTO 1270
1290 CLS
1300 RETURN

1310 GOSUB 800
1320 GOSUB 1700
1330 PRINT @ J*4+I*64+8,B$(B(I,J));
1340 PRINT @ JS*4+IS*64+8,B$(B(IS,JS));
1350 RETURN
```

```
1360 CLS
1370 PRINT:PRINT "CHOOSE 3 FOUR LETTER WORDS"
1380 PRINT:PRINT "AND 1 THREE LETTER WORD"
1390 PRINT:INPUT "FIRST FOUR LETTER
     WORD ";A$
1400 IF LEN (A$)<>4 THEN GOTO 1390
1410 FOR I=1 TO 4
1420 W$(I)=MID$(A$,I,1)
1430 NEXT I
1440 PRINT "FIRST WORD= ";A$
1450 INPUT "SECOND FOUR LETTER WORD ";A$
1460 IF LEN (A$)<>4 THEN GOTO 1450
1470 FOR I=1 TO 4
1480 W$(I+4)=MID$(A$,I,1)
1490 NEXT I
1500 PRINT "SECOND WORD= ";A$
1510 INPUT "THIRD FOUR LETTER WORD ";A$
1520 IF LEN (A$)<>4 THEN GOTO 1510
1530 PRINT "THIRD WORD= ";A$
1540 FOR I=1 TO 4
1550 W$(I+8)=MID$(A$,I,1)
1560 NEXT I
1570 INPUT "THREE LETTER WORD";A$
1580 IF LEN (A$)<>3 THEN GOTO 1570
1590 PRINT "FOURTH WORD= ";A$
1600 FOR I=1 TO 3
1610 W$(I+12)=MID$(A$,I,1)
1620 NEXT I
1630 W$(16)=" "
1640 FOR I=1 TO 15
1650 B$(I)=CHR$(ASC(W$(I))+32)
1660 W$(I)=B$(I)
1670 NEXT I
1680 CLS
1690 RETURN

1700 B(IS,JS)=B(I,J)
1710 B(I,J)=16
1720 T=IS:IS=I:I=T
1730 T=J:J=JS:JS=T
1740 RETURN
```

4
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 "1 . . . 2 . . . 3 . . . GO" appears on the left of the screen and on the word "GO" the man has to jump as high as possible and then scrabble up the remainder of the wall. 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 is dependent on the delay between the signal appearing and your key press. The quicker you react, the higher the man will jump. While the rest of your available time (2.2 seconds) ticks away you must keep on pressing any key to get the man over the wall. The man will climb one position higher for every complete key press – the more rapidly you press the more quickly he will climb. 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 at the wall. 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

Take care when typing in the long strings of letters and digits in the lines dealing with the high resolution graphics characters, line 230 for example. It is very easy to make mistakes with such long and repetitive strings but if you do you'll end up with a very strange display.

Subroutine structure

20	Play loop
40	Countdown
230	Cheat routine
310	Prints wall
360	Jump logic
650	Fall down wall routine
750	Prints man across wall
820	Prints man going down far side
910	Zeroes time
930	Gets time
950	Draws man
970	End of game

Programming details

One interesting feature of this game is the way that the Dragon's internal clock is used as a reaction timer and a countdown device. The clock is zeroed by routine 910 and read by routine 930. You could use these routines in programs of your own. Also notice the way that the numbers 1,2,3 and the words "GO" and "CHEAT" are displayed in high resolution graphics. Finally, it is worth mentioning the sound effects in this game. The pitch of the note sounded increases as the man scales the wall (line 450) and every time he falls down the wall a descending scale is sounded (line 700). This scale has a different rhythm when the man has successfully climbed over the wall (line 870). And if you try to cheat you'll certainly hear about it.

Program

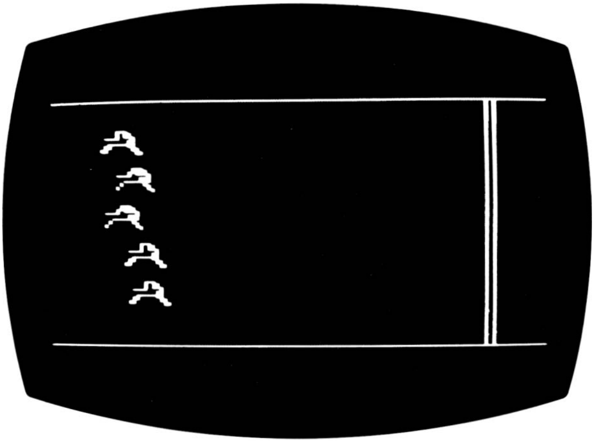
```
 10 REM COMMANDO JUMP
 20 GOSUB 310
 30 GOTO 970

 40 IF INKEY$<>"" THEN GOTO 40
 50 DRAW "BM20,100C0S8U5"
 60 FOR I=1 TO RND(500)+500:NEXT I
 70 DRAW "BM20,100C1S8U5"
 80 DRAW "BM20,100S8C0R5L5U3R5U2L5"
 90 FOR I=1 TO RND(500)+500
100 NEXT I
110 DRAW "BM20,100S8C1R5L5U3R5U2L5"
120 A$=INKEY$
130 DRAW "BM20,100S8C0R5U3L2R2U2L5"
140 FOR I=1 TO 500+RND(500):NEXT
150 DRAW "BM20,100S8C1R5U3L2R2U2L5"
160 IF INKEY$<>"" THEN GOTO 230
170 GOSUB 910
180 DRAW "BM20,100S8C0U5R5L5D5R5U2L2M+5,
    +2U5R5D5L5"
190 IF INKEY$="" THEN GOTO 190
200 GOSUB 930
210 DRAW "BM20,100S8C1U5R5L5D5R5U2L2M+5,+2U5R5D5L5"
220 RETURN

230 C$="S8R5L5U5R5M+2,+5U5D2R5U2D5M+2,
    +0R5L5U3R5L5U2R5M+2,+5U5R5D5U3L5M+7,
    +3U5L2R4"
240 DRAW "BM20,100C0"+C$
250 SOUND 100,16
260 FOR I=1 TO 500:NEXT I
270 DRAW "BM20,100C1"+C$
280 JM=JM+1
290 IF JM>10 THEN GOTO 970
300 GOTO 40
```

```
310 PCLEAR 8
320 PMODE1,1:SCREEN 1,1
330 PCLS:JM=1
340 H=85+RND(25)
350 LINE(100,160)-(100,160-H),PSET

360 CM$="C0"
370 XM$="90"
380 YM$="160"
390 GOSUB 950
400 GOSUB 40
410 CM$="C1"
420 YM$=STR$(160)
430 GOSUB 950
440 FOR I=160 TO 160-H+INT(T*90) STEP -8
450 SOUND 180-I,1
460 NEXT I
470 CM$="C0"
480 YM$=STR$(I)
490 GOSUB 950
500 J=I:L=INT(I)
510 GOSUB 930
520 IF T>2.2 THEN GOTO 650
530 IF INKEY$="" THEN GOTO 520
540 CM$="C1":YM$=STR$(L)
550 GOSUB 950
560 CM$="C0":YM$=STR$(L-8)
570 GOSUB 950
580 J=J-8
590 L=J
600 CM$="C1":YM$=STR$(L)
610 GOSUB 950
620 IF L<=160-H THEN CM$="C1":
    YM$=STR$(L+8):GOSUB 950:GOTO 750
630 IF INKEY$<>"" THEN SOUND 100,4:
    GOTO 630
640 GOTO 510
```

```
650 FOR I=L TO 160 STEP 8
660 CM$="C1":YM$=STR$(I-8)
670 GOSUB 950
680 CM$="C0":YM$=STR$(I)
690 GOSUB 950
700 SOUND 180-I,4
710 NEXT I
720 JM=JM+1
730 IF JM<=10 THEN GOTO 360
740 GOTO 970

750 FOR I=120 TO 128 STEP 8
760 CM$="C1":XM$=STR$(I-8)
770 GOSUB 950
780 CM$="C0":XM$=STR$(I)
790 GOSUB 950
800 FOR Q=1 TO 50:NEXT Q
810 NEXT I

820 FOR I=L+8 TO 160 STEP 8
830 CM$="C1":YM$=STR$(I-8)
840 GOSUB 950
850 CM$="C0":YM$=STR$(I)
860 GOSUB 950
870 SOUND 180-I,4
880 FOR Q=1 TO 50:NEXT Q
890 NEXT I
900 GOTO 970

910 TIMER=0
920 RETURN

930 T=TIMER/50
940 RETURN

950 DRAW "BM"+XM$+","+YM$+CM$+
    "S4U4R4D4U4L2U2R3L3U2D2L3"
960 RETURN
```

```
 970 IF JM<=10 THEN GOTO 1010
 980 CLS
 990 PRINT @ 100,"YOU FAILED"
1000 GOTO 1030
1010 CLS:PRINT @ 96," YOU TOOK ";JM;
     " JUMPS TO"
1020 PRINT "  CLEAR THE WALL"
1030 INPUT "ANOTHER GAME";A$
1040 IF LEFT$(A$,1)="Y" THEN RUN
1050 CLS
1060 STOP
```

5
Derby Day



This is a very simple betting game with an impressive and convincing horse race display plus an appropriate musical introduction. 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 Dragon 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 it you win having placed 20 you will receive 100). The Dragon keeps a tally of your winnings and losses and will tell you if you go broke. During the race, your horse is the yellow one – all the others are red – 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 strings in lines 600 to 670, including the hashes and full stops. If you make mistakes in typing in this subroutine, the program may still run but the tune may be unrecognisable or sound discordant!

Subroutine structure

20	Set-up routine
120	Main play loop
140	Draws race course
260	Prints title
290	Betting routine
410	Runs race
530	Draws horses
560	Start of game
590	Plays tune
700	Win/lose routine and 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 lines 590 to 690. The notes for the tune are stored in the array Q$ in lines 600 to 670. The length of time each note is sounded for is set by TM$ in line 590 and the tune is actually played by line 680.

You might think that it would be very difficult to draw a shape that looks like a horse using the Dragon's graphics, but once you've seen this program run you may change your mind. The horses are drawn at line 530 using the DRAW command in high resolution graphics. The horses are moved in subroutine 410 using two random components. Line 450 selects which horse is to be moved and the distance it moves is determined in line 470.

Scope for improvement

You may prefer to substitute another tune in this program – or even add extra tunes so that a different one is played between races. It is quite easy to convert any tune you have sheet music for to play on the Dragon – or you could even compose your own.

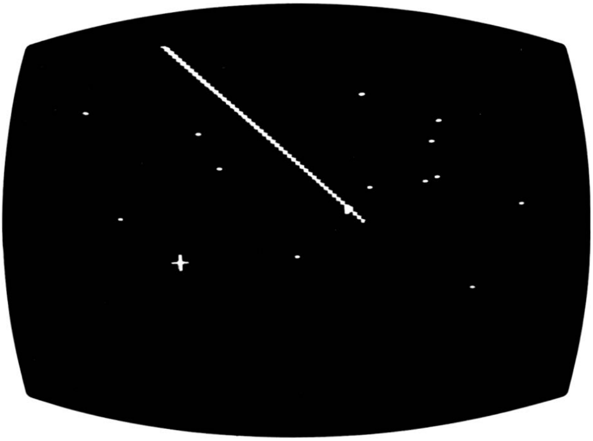
Program

```
 10 REM DERBY DAY
 20 CLEAR 1000
 30 DIM X(5)
 40 DIM Y(5)
 50 PMODE 3,1
 60 PCLS
 70 TT=100
 80 GOSUB 560
 90 GOSUB 290
100 TM$="T30"
110 GOSUB 140

120 GOSUB 410
130 GOTO 120

140 REM DRAW COURSE
150 LINE (0,10)-(250,10),PSET
160 LINE (0,140)-(250,140),PSET
170 LINE(220,10)-(220,140),PSET
180 LINE(225,10)-(225,140),PSET
190 FOR Y=1 TO 5
200 X(Y)=2
210 Y(Y)=Y*20+10
220 IF B=Y THEN C$="C2" ELSE C$="C4"
230 GOSUB 530
240 NEXT Y
250 RETURN

260 CLS 1
270 PRINT @ 102,"D E R B Y   D A Y"
280 RETURN
```

```
290 CLS
300 FOR I=1 TO 5
310 PRINT @ I*32+2,I
320 NEXT I
330 PRINT "WHICH HORSE DO YOU"
340 INPUT "WANT TO BET ON ";B
350 IF B<1 OR B>5 THEN PRINT @ 390,
    "NO SUCH HORSE":SOUND 100,8:GOTO 290
360 PRINT "YOU HAVE",TT
370 INPUT "HOW MUCH DO YOU WANT TO BET";M
380 IF TT-M<0 THEN PRINT @ 390,
    "YOU DONT HAVE THAT MUCH":SOUND 100,8:
    GOTO 370
390 TT=TT-M
400 RETURN

410 PMODE 3,1
420 SCREEN 1,0
430 PCLS 1
440 GOSUB 140
450 Y=RND(5)
460 C$="C1":GOSUB 530
470 X(Y)=X(Y)+5+RND(2)
480 IF B=Y THEN C$="C2" ELSE C$="C4"
490 GOSUB 530
500 IF X(Y)>195THEN GOTO 700
510 GOTO 450
520 RETURN

530 M$="BM"+STR$(X(Y))+","+STR$(Y(Y))
540 DRAW C$+M$+"R6U2R1U2R4D1L1D2R2U2R1D
    2R2D4L1U2L2D1F6L1H5L8G4L1E4"
550 RETURN

560 GOSUB 260
570 GOSUB 590
580 RETURN
```

```
590 TM$="T10"
600 Q$="L2AAAF#ABAF#P2L2F#L1.EL2F#L1E"
610 Q$=Q$+"L2AAF#ABAF#P1"
620 Q$=Q$+"L4F#EDEL2F#EL1.D"
630 Q$=Q$+"P2L2.DL4DL2F#A04L1.D"
640 Q$=Q$+"03P1L2.BL4B04L2D03L2BL1.A"
650 Q$=Q$+"L4F#GL2AAL4F#F#"
660 Q$=Q$+"AAL2BAL1F#"
670 Q$=Q$+"L2EF#L4GL2F#L4EEL1D"
680 PLAY TM$+Q$
690 RETURN

700 FOR Q=1 TO 500:NEXT Q
710 CLS
720 IF Y<>B THEN PRINT @ 100,"YOU LOSE ";M
730 IF Y=B THEN M=M*5:TT=TT+M:PRINT @ 100,
    "YOU WIN ";M
740 IF TT<=0 THEN PRINT @ 100,"YOU ARE
    BROKE":FOR Z=1 TO 500:NEXT Z:GOTO 790
750 PRINT "YOU HAVE ";TT
760 INPUT "ANOTHER RACE Y/N";A$
770 A$=LEFT$(A$,1)
780 IF A$="Y" THEN GOTO 80
790 END
```

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 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 anihilate 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 upredictability of the enemy ship's course and the number of stars that appear. 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 cross-shaped disc that meanders through space. To fire your laser press the up arrow key. Your weapon will fire in a straight line from the point of your arrow. If you hit the enemy ship it will disintegrate with appropriate sound and visual effects. You have a hundred seconds in which to anihilate the enemy and once you've destroyed him the time you took will be shown.

Subroutine structure

20	Main play loop
130	Fires or rotates direction of arrow
190	Laser zap and detect hit logic
540	Moves arrow
640	Moves target
760	Title frame
880	Gets time
900	Prints stars
960	Initialises variables
1140	Draws arrow
1230	Draws target
1300	End of game

Programming details

This is a complicated program and one that illustrates a number of novel programming methods. Notice, for example, the way the revolving cross is produced by using two DRAW commands which are displayed alternately. The attacker also relies on two DRAW commands, one to draw an upward arrow, the other to draw the arrow shifted through 45 degrees. The A (angle) command is then used in the DRAW string to rotate these two shapes through 90 degrees. The way the direction of movement and velocity is set using the arrays W and V is also worth attention.

Scope for improvement

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

Program

```
 10 REM LASER ATTACK
 20 GOSUB 760
 30 GOSUB 960
 40 GOSUB 900
 50 SCREEN 1,0
 60 GOSUB 880
 70 IF T>100 THEN GOTO 1300
 80 GOSUB 130
 90 GOSUB 540
100 IF H=1 THEN GOTO 1300
110 GOSUB 640
120 GOTO 60

130 A$=INKEY$
140 IF A$="" THEN RETURN
150 IF A$=CHR$(94) THEN GOTO 190
160 K=K+1
170 IF K>8 THEN K=1
180 RETURN
```

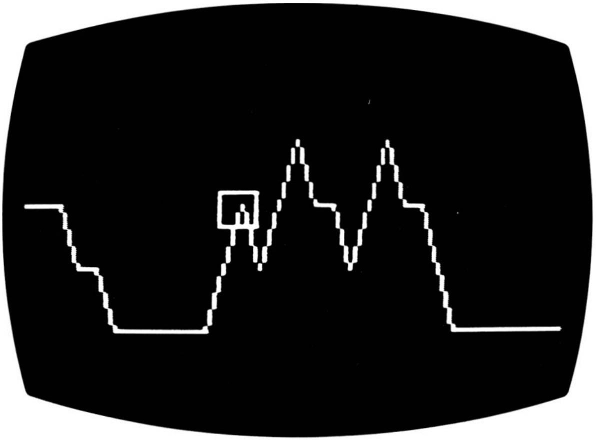
```
190 XL=X
200 YL=Y
210 GOSUB 470
220 DX=0
230 IF W(K)=1 THEN DX=255-XL
240 IF W(K)=-1 THEN DX=-XL
250 DY=0
260 IF V(K)=1 THEN DY=-YL
270 IF V(K)=-1 THEN DY=192-YL
280 IF V(K)*W(K)=0 THEN GOTO 320
290 IF ABS(DX)<ABS(DY) THEN
    DY=ABS(DX)*SGN(DY):GOTO 320
300 DX=ABS(DY)*SGN(DX)
310 COLOR 1,3
320 LINE (XL,YL)-(XL+DX,YL+DY),PSET
330 SOUND 50,4
340 LINE (XL,YL)-(XL+DX,YL+DY),PRESET
350 IF H=0 THEN RETURN
360 MX=A+4:MY=B-4
370 FOR I=1 TO RND(5)+20
380 COLOR 1,3
390 DX=10-RND(20)
400 IF MX+DX>254 OR MX+DX<0 THEN GOTO 450
410 DY=10-RND(20)
420 IF MY+DY>190 OR MY+DY<0 THEN GOTO 450
430 LINE (MX,MY)-(MX+DX,MY+DY),PSET
440 SOUND 50,5
450 NEXT I
460 RETURN
470 H=0
480 DY=B-4-Y
490 DX=A+4-X
500 IF ABS(V(K)*DX+W(K)*DY)>3 THEN RETURN
510 IF ABS(W(K))*SGN (DX)<>W(K) OR
    -ABS(V(K))*SGN (DY)<>V(K) THEN RETURN
520 H=1
530 RETURN
```

```
540 C$="C3":GOSUB 1140
550 X=X+W(K)*8
560 Y=Y-V(K)*8
570 IF X<0 THEN X=254
580 IF X>254 THEN X=0
590 IF Y<0 THEN Y=190
600 IF Y>190 THEN Y=0
610 K1=K
620 C$="C1":GOSUB 1140
630 RETURN

640 IF RND(0)>1.05-DF/20 THEN Z=Z+1
650 IF Z>8 THEN Z=1
660  C$="C3":GOSUB 1230
670 A=A+V(Z)*8
680 B=B-W(Z)*8
690 IF B<0 THEN B=189
700 IF B>189 THEN B=0
710 IF A>250 THEN A=0
720 IF A<0 THEN A=250
730 R=NOT R
740 C$="C1":GOSUB 1230
750 RETURN

760 CLS
770 PRINT @ 36," L A S E R  A T T A C K"
780 PRINT @ 96," YOU ARE IN CONTROL OF AN "
790 PRINT " ADVANCED LASER ATTACK SHIP IN"
800 PRINT " PURSUIT OF AN ENEMY CRAFT"
810 PRINT:PRINT " SHOOT IT DOWN BEFORE
    YOUR TIME"
820 PRINT " IS UP "
830 PRINT " SELECT THE DIFFICULTY LEVEL"
840 INPUT " 1 (EASY) TO 10 (DIFFICULT) ";DF
850 IF DF<1 OR DF>10 THEN GOTO 840
860 CLS
870 RETURN

880 T=TIMER/50
890 RETURN
```

```
 900 FOR Q=1 TO RND(20)+10
 910 SX=RND(254)
 920 SY=RND(190)
 930 PSET(SX,SY,2)
 940 NEXT Q
 950 RETURN

 960 DIM W(8):DIM V(8)
 970 DATA 0,1,1,1,1,0,1,-1,0,-1,-1,-1,-1,
     0,-1,1
 980 FOR I=1 TO 8
 990 READ W(I),V(I)
1000 NEXT I
1010 Z=1
1020 K=1
1030 X=150
1040 Y=100
1050 V=V(K)
1060 A=RND(253)
1070 B=RND(189)
1080 W=W(K)
1090 PMODE 3,1
1100 PCLS 3
1110 COLOR 4,3
1120 TIMER=0
1130 RETURN

1140 X$=STR$(X)
1150 Y$=STR$(Y)
1160 N$=STR$(INT(K1/2))
1170 IF K1=2 OR K1=4 OR K1=6 OR K1=8 THEN
     GOTO 1200
1180 DRAW "A"+N$+"BM"+X$+","+Y$+C$+"U8G3BE3F3"
1190 RETURN
1200 N$=STR$(INT(K1/2)-1)
1210 DRAW "A"+N$+"BM"+X$+","+Y$+C$+"E8L3BR3D3"
1220 RETURN
```

```
1230 A$=STR$(A)
1240 B$=STR$(B)
1250 IF R=0 THEN GOTO 1280
1260 DRAW "A0BM"+A$+","+B$+C$+"U8D4L4R8"
1270 RETURN
1280 DRAW "A0BM"+A$+","+B$+C$+"E6G3H3F6"
1290 RETURN

1300 IF H<>1 THEN GOTO 1350
1310 CLS
1320 PRINT @ 96," YOU DID IT !!!"
1330 PRINT "IN ";T;" SECONDS"
1340 GOTO 1360
1350 PRINT @ 96," YOUR TIME IS UP"
1360 INPUT "ANOTHER GAME Y/N";A$
1370 IF LEFT$(A$,1)="Y" THEN RUN
1380 CLS
1390 STOP
```

7
Guideline



This is a game of skill in which you have to guide a square along an irregular wavy wire. When you play this game with a real wire and a ring, it's a test of how steadily you can guide the ring along the wire. In this Dragon 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. This determines the size of the square that has to be guided along the wire. Once the square appears a tone sounds to indicate the start of the game and then it automatically starts to move 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. When you want to change direction, take care to release the key you have been pressing *before* pressing the other one.

Subroutine structure

20	Main play loop
130	End of game
200	Initialises variables and prints title frame
370	Plots wire
500	Restores screen
550	Draws and moves square

Programming details

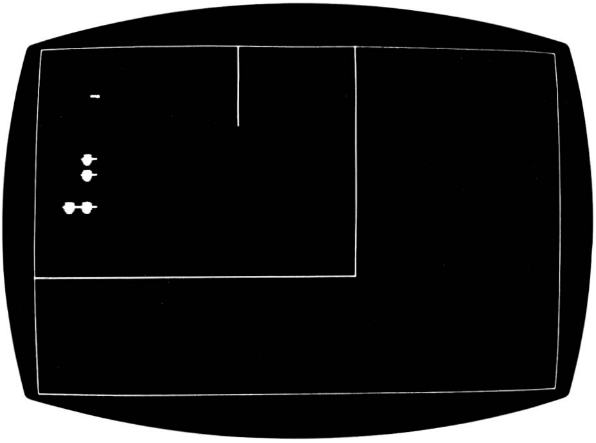
This program relies on using the PCOPY command to achieve its effect. Every time the square moves its position the wire is *redrawn* by copying from one graphics screen to the other (lines 500–510). When you play this game you'll notice that all the time you hold one of the arrows keys down it automatically *repeats*. This is controlled in lines 580 and 590 which PEEK two memory locations to discover whether or not they contain 255, the value which indicates that no key is being pressed. Line 650 is responsible for testing to see whether the current position of the square is actually over the line. It does this using the PPOINT function.

Program

```
 10 REM GUIDELINE
 20 PCLEAR 4
 30 K=341
 40 GOSUB 200
 50 PMODE 1,1:SCREEN 1,1:PCLS
 60 GOSUB 370
 70 FOR Q=1 TO 100:NEXT
 80 SOUND 50,5
 90 Y=100:X=10
100 B=Y:R=12-DF
110 R2=R/2:V=4
120 GOSUB 550

130 CLS
140 PRINT @ 100,"YOU WERE ON TARGET "
150 PRINT INT(HIT/(HIT+MISS)*10000)/100;
    "7 OF THE TIME"
160 INPUT "ANOTHER GAME ";A$
170 IF LEFT$(A$,1)="Y" THEN RUN
180 CLS
190 STOP

200 X=10
210 Y=100
220 D=30
230 P=0
240 CLS
250 PRINT @ 7, "G U I D E L I N E"
260 PRINT
270 PRINT"YOU MUST GUIDE A RING ALONG THE"
280 PRINT:PRINT "WAVY WIRE USING THE UP AND"
290 PRINT:PRINT "DOWN ARROW KEYS"
300 PRINT:PRINT"YOU WILL BE MARKED ON HOW"
310 PRINT:PRINT"ACCURATE YOU ARE"
320 PRINT:PRINT"SELECT THE DIFFICULTY LEVEL"
330 PRINT:INPUT"1(EASY) TO 5(DIFFICULT)";DF
340 IF DF<1 OR DF>5 THEN GOTO 330
350 PCLS
360 RETURN
```

```
370 PSET(X,Y,1)
380 FOR I=1 TO 248  STEP 8
390 R=D-(SGN(RND(0)-.5))*(INT(RND(0)*2)*D*2)
400 Y1=Y+R
410 IF Y1>180 THEN Y1=Y1-R:R=-R
420 IF Y1<10 THEN Y1=Y1+R:R=-R
430 X1=X+8
440 LINE (X,Y)-(X1,Y1),PSET
450 X=X1:Y=Y1
460 NEXT I
470 HIT=0
480 MISS=0
490 RETURN

500 PCOPY 1 TO 3
510 PCOPY 2 TO 4
520 PMODE 1,3
530 SCREEN 1,1
540 RETURN

550 GOSUB 500
560 LINE(X-R2,Y-R2)-(X+R,Y+R),PSET,B
570 A$=INKEY$
580 IF PEEK(K)<>255 AND Y-R2>2*V+R
    THEN B=B-2*V
590 IF PEEK(K+1)<>255 AND Y+R<188-2*V
    THEN B=B+2*V
600 LINE(X-R2,Y-R2)-(X+R,Y+R),PRESET,B
610 X=X+V
620 IF X>247 THEN RETURN
630 Y=B
640 FOR I=-R+1 TO R-1
650 IF PPOINT(X,Y+I)=8 THEN HIT=HIT+1:
    FOR Q=1 TO 10:NEXT Q:GOTO 550
660 NEXT I
670 MISS=MISS+1
680 SOUND 100,5
690 A$=INKEY$
700 GOTO 550
```

8
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 feast. 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!! Four black sheep appear in a green field surrounded by a picket fence with the faithful dog in the top righthand corner – just the right country atmosphere.

How to play

The object of the game is to herd all four sheep into the pen at the top lefthand 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 four 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.

Subroutine structure

20	Set-up routine
100	Draws field and pen
170	Plots sheep
220	Plots dog
250	Moves dog
430	Paints shapes
470	Draws sheep
540	Draws dog
620	Moves sheep
790	Scatters sheep
870	End of game

Programming details

This game uses the highest resolution graphics available on the Dragon. When you run it you may think 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 640 and 650 check to see if the dog has approached too close to the sheep. If he has (or if the random number generated is less than .01, a one-in-a-hundred chance occurence) then the sheep scatter according to the equations in 820 to 840.

Each time the dog or the sheep move they have to be undrawn from the old positions and redrawn at their new ones. Notice the way that PAINT is used at line 520 to fill in the outlines of the dog and the sheep every time this happens.

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 640 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 Dragon's timer, instead of counting the number of moves needed.

Program

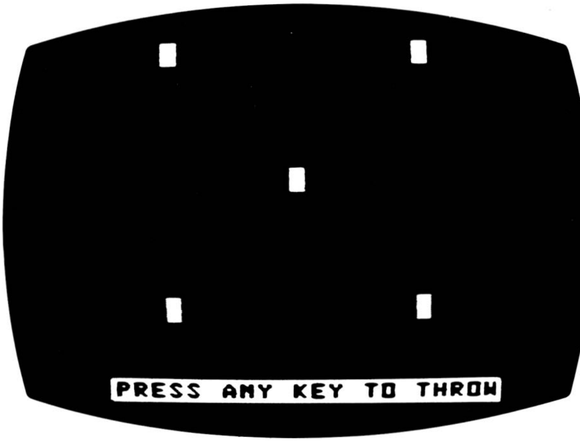
```
 10 REM SHEEPDOG
 20 DIM X(4)
 30 DIM Y(4)
 40 M=0
 50 PMODE 4,1
 60 COLOR 0,1
 70 SCREEN 1,0
 80 PCLS
 90 D=1

100 LINE(1,4)-(252,4),PSET
110 LINE(1,180)-(252,180),PSET
120 LINE(1,4)-(1,180),PSET
130 LINE(252,4)-(252,180),PSET
140 LINE(96,4)-(96,44),PSET
150 LINE(152,4)-(152,120),PSET
160 LINE(0,121)-(152,121),PSET

170 FOR S=1 TO 4
180 Y(S)=54+RND(4)*8:YS$=STR$(YS)
190 X(S)=RND(3)*8+9:XS$=STR$(XS)
200 GOSUB 470
210 NEXT S
```

```
220 YD=8+RND(3)*8:YD$=STR$(YD)
230 XD=3+RND(3)*8:XD$=STR$(XD)
240 CD$="C0":GOSUB 540

250 D$=INKEY$
260 IF D$="" THEN GOTO 250
270 CD$="C1"
280 GOSUB 540
290 IF D$=CHR$(9) AND XD=91 AND YD<45
    THEN GOTO 350
300 IF D$=CHR$(8) AND XD=99 AND YD<45
    THEN GOTO 350
310 IF D$=CHR$(8) AND XD>9 THEN XD=XD-8
320 IF D$=CHR$(9) AND XD<142 THEN XD=XD+8
330 IF D$=CHR$(10) AND YD<116 THEN YD=YD+8
340 IF D$=CHR$(94) AND YD>11 THEN YD=YD-8
350 XD$=STR$(XD):YD$=STR$(YD)
360 CD$="C0"
370 GOSUB 540
380 M=M+1
390 GOSUB 560
400 GOSUB 620
410 IF F=0 THEN GOTO 870
420 GOTO 250

430 PAINT(X,Y),1,1
440 PRESET(X-2,Y+4)
450 PRESET(X+2,Y+4)
460 RETURN

470 CIRCLE(X(S),Y(S)),3,0,1
480 COLOR 0,1
490 LINE(X(S)-1,Y(S)+3)-(X(S)-2,Y(S)+4),
    PSET
500 LINE(X(S)+1,Y(S)+3)-(X(S)+2,Y(S)+4),
    PSET
510 CIRCLE(X(S)+4,Y(S)),1,0
520 PAINT (X(S),Y(S)),0,0
530 RETURN
```

```
540 DRAW "BM"+XD$+","+YD$+CD$+"S2H2E2R5F
    4H4R4E1H1U2L10D2R1"
550 RETURN
560 FOR S=1 TO 4
570 Y=Y(S)
580 X=X(S)
590 GOSUB 430
600 NEXT S
610 RETURN

620 F=0
630 FOR S=1 TO 4
640 IF (ABS(X(S)-XD)<16 AND ABS(Y(S)-YD)<16)
    OR RND(0)<.01 THEN GOSUB 790
650 IF ABS(X(S)-XD)>16+(RND(2)*8) OR
    ABS(Y(S)-YD)>16+(RND(2)*8) THEN
    GOTO 680
660 X(S)=X(S)+SGN(X(S)-XD)*8
670 Y(S)=Y(S)+SGN(Y(S)-YD)*8
680 IF X(S)<6 THEN X(S)=6
690 IF X(S)>145 THEN X(S)=145
700 IF Y(S)<9 THEN Y(S)=9
710 IF Y(S)>115 THEN Y(S)=115
720 IF X(S)>89 AND Y(S)<49 AND X(S)<103
    THEN X(S)=89
730 IF (ABS(X(S)-XD)<6 AND ABS(Y(S)-YD)<6)
    THEN GOSUB 790
740 GOSUB 470
750 IF X(S)>96 AND Y(S)<40 THEN GOTO 770
760 F=1
770 NEXT S
780 RETURN

790 X=X(S)
800 Y=Y(S)
810 GOSUB 430
820 Y(S)=Y(S)+(SGN(RND(0)-.5))*(24+
    (RND(2)*8))
830 IF Y(S)<9 OR Y(S)>114 THEN GOTO 820
840 X(S)=X(S)+(SGN(RND(0)-.5))*(24+
    (RND(2)*8))
850 IF X(S)<7 OR X(S)>143 THEN GOTO 840
860 RETURN
```

```
870 REM END GAME
880 CLS
890 IF M<50 THEN PRINT "SUPER SHEPHERD!!!":
    GOTO 940
900 IF M<75 THEN PRINT "GOOD DOG!":GOTO 940
910 IF M<100 THEN PRINT "KEEP PRACTISING":
    GOTO 940
920 IF M<125 THEN PRINT "YOUR SHEEP WERE
    WILD!!!":GOTO 940
930 PRINT "HAND IN YOUR CROOK!!!"
940 PRINT "YOU TOOK ";M;" MOVES"
950 INPUT "ANOTHER ROUND Y/N";A$
960 IF LEFT$(A$,1)="Y" THEN RUN
970 CLS
980 STOP
```

9
Magic Dice



```
PRESS ANY KEY TO THROW
```

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

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

Of course, this program 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 as well if you run it on a colour set.

How to use the program

Using this program is simplicity itself. Type RUN and then press any key 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.

Typing tips

The words 'press any key to throw' appear in the program listing in lower case. This is because they are displayed on the screen in reverse video. To achieve this effect press SHIFT and 0 together after you've type the quotation mark and before you type the first letter concerned (in this case P). Remember to restore the normal graphics display, by pressing SHIFT and 0 together again, before you try to type the final quotation marks. Notice that you have to type 21 spaces between the quotation marks in line 350. Count these carefully as your dice display will not work with the wrong number.

Subroutine structure

20	Main play loop
170	Prints and unprints dots
340	Draws green square for dice
390	Emits sound

Programming details

The essence of a dice program is in generating random numbers. In fact, this program uses random numbers in two ways. Firstly, randomness is used in the conventional way, to determine which face of the dice will show at the next turn – this is done in line 120, 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 180). The program goes to one of the six line numbers 190, 210, 240, 260, 290 or 310 according to the value of 'R'. At 190 one dot is printed, at 210 two dots are printed and so on.

The other use of the random number generator is to give the 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. This program copies both these features by incorporating lines 70 to 90. Another random number, 'T', with a value between 6 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.

This program makes use of both colour and sound. The colour that the dots are printed in alternates between red and green (lines 100 and 130). As green is also the colour used for the background this causes them to appear and disappear. Line 390 is responsible for the beeping sound that accompanies the rolling of the dice.

Scope for improvement

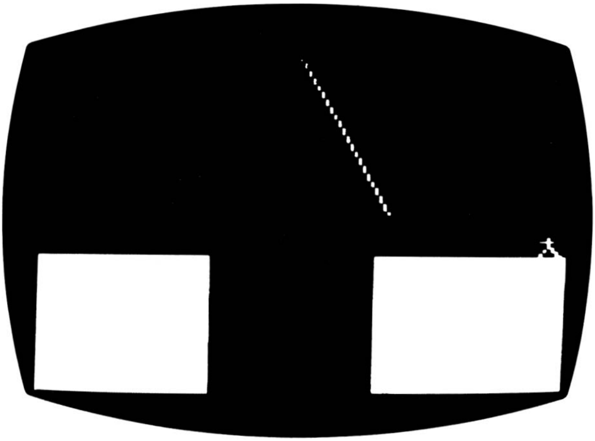
This program could be incorporated into many self-contained games. For example, you could devise a gambling game where bets were placed on which face of the dice would show.

Program

```
 10 REM DICE
 20 CLS 5
 30 GOSUB 340
 40 PRINT @ 484, "press any key to throw";
 50 PRINT @ 489,CHR$(128);:
    PRINT @ 493,CHR$(128);:
    PRINT @ 497,CHR$(128);:
    PRINT @ 500,CHR$(128);
 60 IF INKEY$="" THEN GOTO 60
 70 T=RND(10)+5
 80 FOR W=1 TO T
 90 FOR Q=1 TO W*W+50:NEXT Q
100 A$=CHR$(143)
110 GOSUB 170
120 R=RND(6)
130 A$=CHR$(191)
140 GOSUB 170
150 NEXT W
160 GOTO 60

170 GOSUB 390
180 ON R GOTO 190,210,240,260,290,310
190 PRINT @ 238,A$;
200 RETURN
210 PRINT @ 71,A$;
220 PRINT @ 405,A$;
230 RETURN
240 GOSUB 190
250 GOTO 210
260 PRINT @ 391,A$;
270 PRINT @ 85,A$;
280 GOTO 210
290 GOSUB 260
300 GOTO 190
310 PRINT @ 78,A$;
320 PRINT @ 398,A$;
330 GOTO 260
```

```
340 FOR I=1 TO 13
350 PRINT @ I*32+4,"                     ";
    :REM 21 SPACES
360 NEXT I
370 R=1
380 GOTO 170

390 SOUND 60,1
400 RETURN
```

10
Across the Ravine



The object of this game is to get your party of five intrepid explorers across a deep ravine with a fast flowing river at the bottom of steep cliffs. There is only one option, to swing across on a rope. The rope swings all the time so each man must run and leap to catch it – anyone who mistimes his jump falls into the river and is lost. Listen out for the sound effects as a man falls into the river!

How to play

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

Subroutine structure

20	Sets-up arrays
90	Main play loop
120	Swings rope
380	Calculates positions of rope
460	Plots ravine
520	Plots man on end of rope
570	Man jump routine
690	Gets next man ready
760	Man fall in water routine
930	Set-up game
1000	End of game

Programming details

One interesting feature of this program is the way the co-ordinates of the positions of the swinging rope are first calculated by subroutine 380 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. Another point to notice is the use of the LINE command in lines 470–480 to draw the river banks.

Program

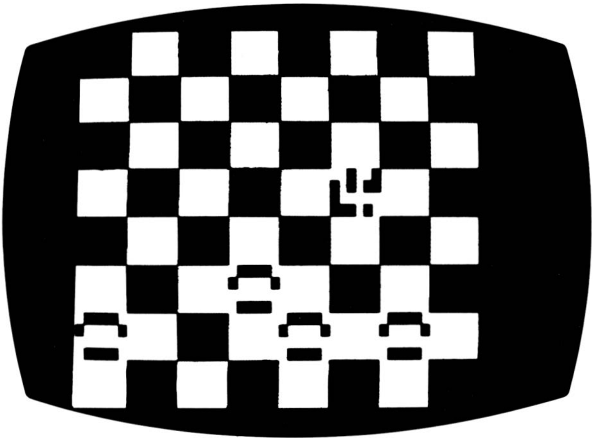
```
10 REM ACROSS THE RAVINE
20 DIM X(16)
30 DIM Y(16)

40 GOSUB 460
50 GOSUB 380
60 GOSUB 930
70 GOSUB 690
80 A$=INKEY$
```

```
 90 GOSUB 120
100 IF M=0 THEN GOTO 1000
110 GOTO 90

120 REM SWING ROPE
130 FOR T=1+R TO N-R
140 COLOR 4,2
150 LINE (124,20)-(124+X(T),20+Y(T)),PSET
160 S=1:C$="C4":GOSUB 520
170 LINE (124,20)-(124+X(T),20+Y(T)),PRESET
180 IF J=1 THEN S=2:C$="C2":GOSUB 520
190 NEXT T
200 IF C=1 THEN GOTO 290
210 FOR T=N-R TO 1+R STEP -1
220 COLOR 4,2
230 LINE (124,20)-(124+X(T),20+Y(T)),PSET
240 S=3:C$="C4":GOSUB 520
250 LINE (124,20)-(124+X(T),20+Y(T)),PRESET
260 IF J=1 THEN S=4:GOSUB 520
270 NEXT T
280 RETURN
290 AC=AC+1
300 DX$=STR$((AC-1)*10+8)
310 DY$="120"
320 C$="C4":GOSUB 550
330 C=0
340 M=M-1
350 IF M=0 THEN GOTO 740
360 GOSUB 690
370 RETURN

380 N=0
390 PI=4.0*ATN(1.0)
400 FOR T=-PI/4 TO PI/4 STEP .1
410 N=N+1
420 X(N)=-INT(90*SIN(T))
430 Y(N)=INT(90*COS(T))
440 NEXT T
450 RETURN

460 PMODE 3,1:SCREEN 1,0:PCLS 2
470 LINE (1,120)-(80,189),PSET,BF
480 LINE (160,120)-(252,189),PSET,BF
490 C=0
500 J=0
510 RETURN
```

```
520 IF C=0 THEN GOTO 570
530 DX$=STR$(120+X(T))
540 DY$=STR$(20+Y(T)+10)
550 DRAW "BM"+DX$+","+DY$+C$+"R1E4F4R1L1
    U1L1H4U2R3L6R3U2"
560 RETURN

570 IF INKEY$="" AND J=0 THEN FOR Q=1 TO
    100:NEXT Q:RETURN
580 J=1
590 DY$=STR$(INT(MY))
600 DX$=STR$(INT(MX))
610 C$="C2":GOSUB 550
620 MX=MX-10
630 DY$=STR$(MY)
640 DX$=STR$(MX)
650 C$="C4":GOSUB 550
660 IF ABS(MX-(124+X(T)))<10AND S=2
    THEN C=1:C$="C2":GOTO 550
670 IF MX<160 THEN GOTO 760
680 RETURN

690 MY=119
700 MX=240
710 DY$=STR$(MY)
720 DX$=STR$(MX)
730 J=0
740 C$="C4":GOSUB 550
750 RETURN

760 C$="C2":GOSUB 550
770 MX=MX-6
780 MY=MY+8
790 DY$=STR$(MY)
800 DX$=STR$(MX)
810 C$="C4":GOSUB 550
820 SOUND INT((200-MY)*2),5
830 C$="C2":GOSUB 550
840 IF MY<150 THEN GOTO 780
850 SOUND 50,5
860 LO=LO+1
870 C=0
880 J=0
890 M=M-1
900 IF M=0 THEN GOTO 1000
910 GOSUB 690
920 RETURN
```

```
 930 M=5
 940 LO=0
 950 AC=0
 960 J=0
 970 C=0
 980 R=RND(4)
 990 RETURN

1000 FOR Q=1 TO 500:NEXT Q
1010 CLS
1020 PRINT @68,"YOU SAVED ";5-LO
1030 PRINT @ 100,"YOU LOST  ";LO
1040 PRINT @416
1050 INPUT "ANOTHER GAME Y/N";A$
1060 IF LEFT$(A$,1)="Y" THEN RUN
1070 CLS
1080 STOP
```

11
Corner the Dragon



This game is played on an eight-by-eight checkered board and the object of the game is to trap the dragon and prevent him from reaching the bottom of the board. To do this you have two, three or four goblin pieces (determined at random) which can be moved diagonally one square at a time and only up the board. The dragon 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 dragon is at the top of the board. It is your move first. You'll notice that one of your pieces is displayed in inverse graphics. This is the piece that is ready to move. If you want to move another piece hit any key and *control* will pass to the next piece in an anti-clockwise direction. Try pressing keys to see this in operation. When you are ready to move a piece, press the left arrow key if you want to move diagonally forward left and the right arrow key if you want to move diagonally forward right. Once you have moved, the dragon will make his move automatically and it's your turn again. The Dragon will display a message when the game is won, either by you or the dragon – 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

Be very careful when typing in the PRINT @ statements in this program. Firstly, notice how many blank spaces there are after some of the short words. These spaces are required to blank out previous messages. Also pay attention to semi-colons. If you omit these, whole lines of the display can be missed out! There are three spaces between the quotation marks in lines 1050 and 1090 – these lines are part of the routine that display the chequered board. The IF statement in line 120 looks as though its wrong as there are no relational signs. It is however correct – the values stored in the array are either '1' or '0' and the Dragon treats these as equivalent to 'true' or 'false'.

Subroutine structure

20	Initialises variables
50	Main play loop
110	Dragon's move logic
490	Moves goblins and validates their moves
670	Selects which goblin to move
860	Changes colour of goblin
940	Restores colour of goblin
1020	Draws board
1140	Prints goblins and dragon
1430	Defines shapes
1490	End of game

Programming details

Notice the way that the commands POKE and PEEK are used in subroutines 860 and 940 to change the colour of the goblins. The colour changes from green to blue by adding 32 to the ASCII code of each location and is restored to blue by subracting 32 from each. You may like to use the Dragon shape – an approximation in computer graphics to the Dragon logo – in your own games. It is defined in lines 1430 and 1440.

Program

```
 10 REM CORNER THE DRAGON
 20 DIM D(10,10)
 30 DIM X(4):DIM Y(4)
 40 DIM A(4):DIM B(4)

 50 GOSUB 1430
 60 GOSUB 1020
 70 H=RND(3)+1:GOSUB 1140
 80 GOSUB 670
 90 GOSUB 110
100 GOTO 80
```

```
110 PRINT @25,"DRAGONS";
120 IF D(QI+2,QJ+2) AND D(QI+2,QJ) AND
    D(QI,QJ+2) AND D(QI,QJ) THEN GOTO 1490
130 M=1
140 GOSUB 400
150 IF QI+N<1 OR QI+N>8 THEN GOTO 180
160 IF D(QI+N+1,QJ+M+1) THEN GOSUB 330
170 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
180 IF D(QI+N+1,QJ+N+1) THEN GOSUB 330
190 IF OI=QI AND OJ=QJ THEN M=-M
200 OI=QI:OJ=QJ
210 PRINT @ QY*32+QX-3,"    ";
220 PRINT @ (QY+1)*32+QX-3,"    ";
230 QX=QX+N*3
240 QY=QY+M*2
250 PRINT @ QY*32+QX-3,W$;
260 PRINT @ (QY+1)*32+QX-3,X$;
270 D(QI+N+1,QJ+M+1)=2
280 D(QI+1,QJ+1)=0
290 QI=QI+N
300 QJ=QJ+M
310 IF QJ=8 THEN GOTO 1520
320 RETURN
330 N=-N
340 IF D(QI+N+1,QJ+M+1)<>1 THEN RETURN
350 M=-M
360 IF QJ<4 THEN N=1
370 IF D(QI+N+1,QJ+M+1)<>1 THEN RETURN
380 N=-N
390 RETURN
400 R=RND(0)
410 IF QJ>6 THEN RETURN
420 IF R>.5 AND QI>3 THEN GOTO 460
430 IF QI>7 THEN GOTO 450
440 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
450 IF QI<4 OR R>.5 THEN RETURN
460 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
470 IF R>.5 THEN GOTO 430
480 RETURN
```

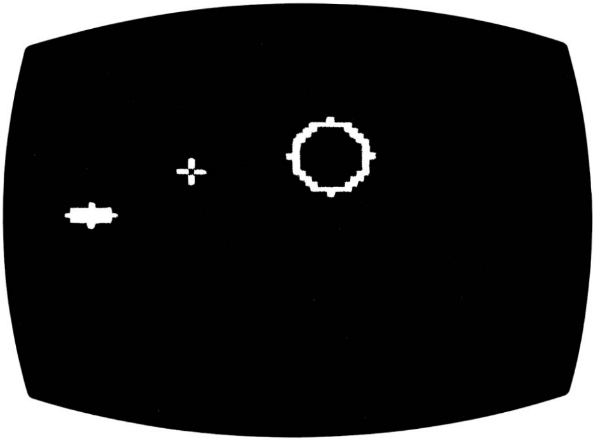
```
490 A(HM)=A(HM)+M
500 B(HM)=B(HM)-1
510 IF D(A(HM)+1,B(HM)+1)<>0 THEN
    GOTO 550
520 D(A(HM)-M+1,B(HM)+2)=0
530 D(A(HM)+1,B(HM)+1)=1
540 GOTO 590
550 A(HM)=A(HM)-M
560 B(HM)=B(HM)+1
570 SOUND 100,5
580 GOTO 670
590 PRINT @ (Y(HM)*32+X(HM)),"    ";
600 PRINT @ (Y(HM)+1)*32+X(HM),"    ";
610 Y(HM)=Y(HM)-2
620 X(HM)=X(HM)+M*3
630 PRINT @ Y(HM)*32+X(HM),Z$;
640 PRINT @ (Y(HM)+1)*32+X(HM),Y$;
650 GOSUB 860
660 RETURN

670 PRINT @ 25,"YOUR   ";
680 PRINT @ 57,"MOVE  ";
690 SOUND 100,5
700 M$=INKEY$
710 IF M$="" THEN GOTO 700
720 IF M$="R" THEN GOTO 770
730 IF M$=CHR$(8) THEN M=-1:GOTO 490
740 IF M$=CHR$(9) THEN M=+1:GOTO 490
750 GOSUB 810
760 GOTO 670
770 PRINT @25,"RESIGN ";
780 PRINT @56," Y/N";:INPUT A$
790 IF LEFT$(A$,1)="Y" THEN GOTO
800 GOTO 670
810 GOSUB 940
820 HM=HM+1
830 IF HM>H THEN HM=1
840 GOSUB 860
850 RETURN
```

```
 860 F=1025+X(HM)+Y(HM)*32
 870 POKE F-1,PEEK(F-1)+32
 880 POKE F,PEEK (F)+32
 890 POKE F+1,PEEK(F+1)+32
 900 POKE F+32,PEEK(F+32)+32
 910 POKE F+33,PEEK(F+33)+32
 920 POKE F+31,PEEK(F+31)+32
 930 RETURN

 940 F=1025+X(HM)+Y(HM)*32
 950 POKE F-1,PEEK(F-1)-32
 960 POKE F,PEEK(F)-32
 970 POKE F+1,PEEK(F+1)-32
 980 POKE F+32,PEEK(F+32)-32
 990 POKE F+33,PEEK(F+33)-32
1000 POKE F+31,PEEK(F+31)-32
1010 RETURN

1020 CLS
1030 FOR I=1 TO 4
1040 FOR J=1 TO 8
1050 PRINT "   ";CHR$(128)CHR$(128)CHR$(128);
1060 IF J/4=INT (J/4) THEN PRINT
1070 NEXT J
1080 FOR J=1 TO 8
1090 PRINT CHR$(128)CHR$(128)CHR$(128)"    ,
1100 IF J/4=INT (J/4) AND I+J<>12 THEN PRINT
1110 NEXT J
1120 NEXT I
1130 RETURN
```

```
1140 FOR I=1 TO H
1150 X=I*6
1160 PRINT @ 477+X,Y$;
1170 PRINT @ 445+X,Z$;
1180 D(I*2+1,9)=1
1190 X(I)=X-3
1200 Y(I)=14
1210 HM=1
1220 A(I)=I*2
1230 B(I)=8
1240 NEXT I
1250 GOSUB 860
1260 QI=5
1270 QJ=1
1280 QX=QI*3
1290 QY=0
1300 PRINT @ QY*32+QX-3,W$;
1310 PRINT @ (QY+1)*32+QX-3,X$;
1320 FOR I=1 TO 10
1330 D(I,1) =1
1340 D(1,I)=1
1350 D(10,I)=1
1360 D(10,I)=1
1370 D(I,10)=1
1380 NEXT I
1390 D(QI+1,QJ+1)=2
1400 QI=0
1410 QJ=0
1420 RETURN

1430 W$=CHR$(130)+CHR$(138)+CHR$(135)
1440 X$=CHR$(139)+CHR$(130)+CHR$(130)
1450 Y$=CHR$(129)+CHR$(131)+CHR$(130)
1460 Z$=CHR$(134)+CHR$(140)+CHR$(137)
1470 CLS 5
1480 RETURN
```

```
1490 PRINT @24,"YOU WIN  ";
1500 PRINT @56,"        ";
1510 GOTO 1540
1520 PRINT @ 24,"DRAGON  ";
1530 PRINT @ 56,"ESCAPES";
1540 PRINT @120,"ANOTHER";
1550 PRINT @152,"GAME ";:INPUT A$
1560 IF LEFT$(A$,1)="Y" THEN RUN
1570 CLS
1580 PRINT "BYE"
1590 STOP
```

12
Pot Shot



This game provides you with the ideal type of target practice. However many times you score a direct hit, the target continues on its way, allowing you to take aim and fire again and again. The target moves steadily across a cloudless sky where the sun always shines. The object of the game is straightforward – to line up the sight with the target and shoot it. But in reality it's not that simple. For one thing every time you fire your rifle 'kicks' to one side or the other, so you have to re-align your sight for another shot. There are a total of five targets and there is no limit to the number of hits you can store. Your total for each target and for the whole game are displayed at the end. There is a distinctive sound every time you fire your rifle and an extra sound when you hit the target.

How to play

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

Subroutine structure

20	Main play loop
130	Plots cross
170	Moves sight
310	Restores screen
360	Draws background
520	Plots target
560	Shoots, tests for hit and recoils sight
620	Title frame and calculates path of target
710	End of game

Programming details

This program uses the Dragon's facility to set up more than one graphics screen. Copying from one display to the other allows for the movement of the target across the sky and the blanking out of previous positions of both the sight and the target. The target and the sight are both drawn in the same colour (buff) but the point at which the two lines of the sight intersect is in magenta. This is to enable PPOINT to be used to detect a hit. If there wasn't this *hole* in the sight, PPOINT would detect the sight itself rather than the target and every shot would be counted as a hit! This program relies on using a repeating key feature. Lines 190 and 220 PEEK two memory locations that contain the value 255 when no key has been pressed to test their values. The effect of these lines of BASIC is that the right and left arrow keys automatically repeat all the time they are held down. The final point of interest in this program is the way in which the target's flight path is calculated and stored in the array 'B' for use later in the program.

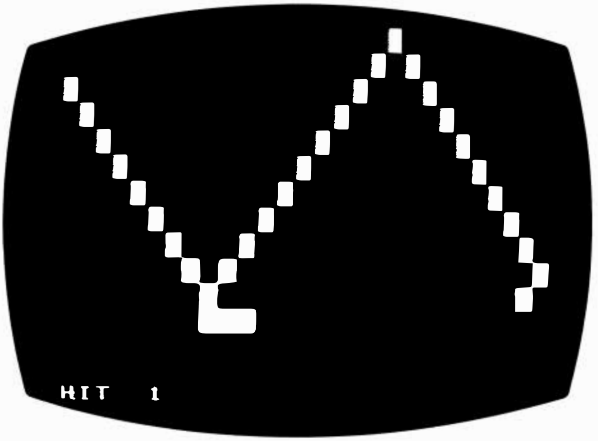
Program

```
 10 REM POT SHOT
 20 PCLEAR 4
 30 K=341
 40 GOSUB 620
 50 FOR G=1 TO 5
 60 GOSUB 360
 70 PRINT @ 100,"TARGET ";G;
 80 FOR Q=1 TO 500:NEXT Q
 90 GOSUB 170
100 NEXT G
110 GOTO 710
120 STOP

130 LINE(X-4,Y)-(X+4,Y),PSET
140 LINE(X,Y-4)-(X,Y+4),PSET
150 PRESET (X,Y)
160 RETURN

170 PMODE 1,1
180 A$=INKEY$
190 IF PEEK(K+2)<>255 AND X>11 THEN X=X-8
200 IF PEEK(K+3)<>255 AND X<248 THEN X=X+8
210 IF PEEK(K+1)<>255 AND Y<180 THEN Y=Y+8
220 IF PEEK(K)<>255 AND Y>11 THEN Y=Y-8
230 I=I+4
240 J=B(I)
250 GOSUB 310
260 COLOR 1,3:GOSUB 130
270 C=1:GOSUB 520
280 IF A$="@" THEN GOSUB 560
290 IF EN=1 THEN EN=0:RETURN
300 GOTO 180

310 PCOPY 1 TO 3
320 PCOPY 2 TO 4
330 PMODE 1,3
340 SCREEN 1,1
350 RETURN
```

```
360 CLS
370 J=RND(100)+100
380 R=RND(40)+20
390 PMODE 1,1:PCLS 3:SCREEN0,0
400 CIRCLE(J,R),15,1,1:PAINT(J-1,R-1),4,1
410 EN=0
420 X=RND(50)+50
430 Y=RND(50)+50
440 COLOR 1,3
450 I=4
460 J=B(I)
470 PCOPY 1 TO 3
480 C=1
490 PMODE 1,3
500 GOSUB 520
510 RETURN

520 CIRCLE(I,J),8,C,,5
530 PAINT (I,J),5,5
540 IF I>246 THEN EN=1
550 RETURN

560 SOUND 19,1
570 IF PPOINT(X,Y)<>5 THEN X=X+10-RND(20):
    RETURN
580 X=X+10-RND(20)
590 SOUND 100,4
600 H(G)=H(G)+1
610 RETURN

620 HT=0
630 CLS
640 PRINT @ 104,"P O T   S H O T"
650 DIM B(254)
660 FOR I=4 TO 254  STEP 4
670 B(I)=(140-I)*(140-I)/200+31
680 NEXT I
690 DIM H(5)
700 RETURN
```

```
710 CLS
720 T=0
730 FOR I=1 TO 5
740 PRINT @ 100+32*I,"TARGET  ";I;
    " HIT ";H(I)
750 T=T+H(I)
760 NEXT I
770 PRINT @ 69,"TOTAL HITS= ";T
780 PRINT @ 420
790 INPUT "ANOTHER GAME Y/N ";A$
800 A$=LEFT$(A$,1)
810 IF A$="Y" THEN RUN
820 CLS
830 END
```

13
Rainbow Squash



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

How to play

At the start of the game the bat is at the bottom of the screen and in the centre. You control the left and right movement of the bat by pressing the appropriate arrow keys. Every time you make two hits in succession the position of the bat changes – it moves nearer to the top of the screen – which makes returning the ball more difficult. If you then miss a shot the ball will move back one position, making it easier. You score a point for every hit and you will be served a total of 10 balls. Information about the number of balls played and hits scored is displayed on the screen continuously.

Subroutine structure

20	Initialises variables
80	Main loop – prints bat and score line
290	Tests keys for keypress
330	Moves balls and controls bouncing off walls
420	Controls bat movement
510	Prints top wall
550	Prints coloured court
660	End of game – prints final score and offers another game.

Programming details

This program relies on using a repeating key feature. This is looked after in line 20, in which the variable K is set to 341, and lines 300–310 which PEEK two memory locations that contain the value 255 when no key has been pressed to test their values. The effect of these lines of BASIC is that the right and left arrow keys automatically repeat all the time they are held down.

The program consists entirely of low resolution graphics. Blanking out the old positions of the bat and ball is made more difficult because a three-colour court has been drawn up. Line 240 is responsible for obliterating the previous position of the bat and line 330 removes the old position of the ball.

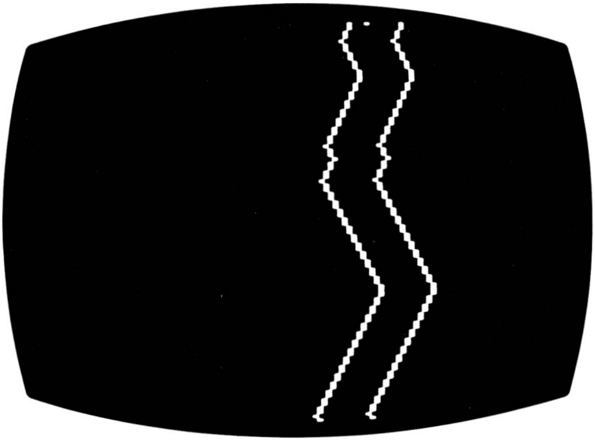
Program

```
 10 REM RAINBOW SQUASH
 20 K=341
 30 CLS
 40 FLAG=0
 50 H=0:HT=0:D=12
 60 GOSUB 550
 70 GOSUB 510

 80 BL=BL+1
 90 IF BL>10 THEN GOTO 660
100 A=2:B=2
110 V=1:W=1
120 X=10:Y=D
130 A$=INKEY$:IF A$="" THEN 130
140 PRINT @483,"BALL ";BL;
150 IF Y*32<157 THEN C1=16 ELSE IF Y*32<308
    THEN C1=64 ELSE C1=80
160 PRINT @ Y*32+X,CHR$(143+C1);CHR$(128);
    CHR$(128);CHR$(128);CHR$(143+C1);
170 PRINT @ 483,"HIT ";HT;"  ";
180 IF FLAG THEN GOSUB 330
190 FLAG=NOT FLAG
200 GOSUB 290
210 IF B+W<>Y THEN Y=D:GOTO 150
220 SOUND 100,8
230 PRINT @ Y*32+X,CHR$(143+C1);CHR$(143+C1);
    CHR$(143+C1);CHR$(143+C1);CHR$(143+C1);
240 IF B*32<157 THEN C=16 ELSE IF B*32<308
    THEN C=64 ELSE C=80
250 PRINT @ B*32+A,CHR$(143+C);
260 IF D<12 THEN D=D+1
270 H=0
280 GOTO 80

290 REM KEY PRESS ?
300 IF PEEK(K+2)<>255 AND X>0 THEN X=X-1
310 IF PEEK(K+3)<> 255 AND X<27 THEN X=X+1
320 RETURN
```

```
330 IF B*32<157 THEN C=16 ELSE IF B*32<308
    THEN C=64 ELSE C=80
340 PRINT @ B*32+A,CHR$(143+C);
350 A=A+V
360 B=B+W
370 IF A=30 OR A=1 THEN V=-V:SOUND 200,2
380 IF B=1 THEN W=-W:SOUND 200,2
390 IF B+W=Y THEN GOTO 420
400 PRINT @ 32*B+A,CHR$(175);
410 RETURN

420 R=A-X
430 IF R<1 OR R>3 THEN GOTO 400
440 W=-W
450 SOUND 200,1
460 H=H+1:HT=HT+1
470 IF H<>2 THEN GOTO 400
480 H=0:D=D-1
490 PRINT @ Y*32+X,CHR$(143+C1);CHR$(143+C1);
    CHR$(143+C1);CHR$(143+C1);CHR$(143+C1);
500 GOTO 330

510 FOR I=0 TO 31
520 PRINT @ I,CHR$(128);
530 NEXT I
540 RETURN

550 PRINT @ 128,;
560 FOR I=32 TO 159
570 PRINT @ I,CHR$(143+16)
580 NEXT I
590 FOR I=160 TO 319
600 PRINT @ I,CHR$(143+64)
610 NEXT I
620 FOR I=320 TO 447
630 PRINT @ I,CHR$(143+80)
640 NEXT I
650 RETURN
```

```
660 CLS
670 PRINT @ 100,"YOU SCORED ";HT
680 INPUT "ANOTHER GAME Y/N ";A$
690 A$=LEFT$(A$,1)
700 IF A$="Y" THEN RUN
710 CLS
720 END
```

14
Bobsleigh



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

How to play

The bobsleigh is situated at the top of the screen and the course automatically scrolls towards and past it. Do make sure that the bobsleigh is in the middle of the two lines when the course reaches you. 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 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

Do check very carefully to make sure you have typed the DATA statements in lines 570–580 correctly. They contain the values forthe machine code program and if you make a mistake it may have disastrous results – for example, an error could prevent you from being able to save the program on tape. Type this program in completely and then check it carefully before you RUN it. If, by mischance, you have made any mistakes with the machine code routine and the program *runs wild* you'll have to turn your Dragon off and on again to clear its memory.

Subroutine structure

20	Sets up screen mode and colour and main play loop
160	First part of course
240	Scrolls course off top of screen
300	Prints and moves bobsleigh
350	Adds a line to count and scrolls
390	Title frame
470	Win/lose messages and end game
570	Machine code loader

Programming details

This game relies on making the course *scroll* up the screen past the bobsleigh, which only moves to left and right and is at a fixed vertical position. The scrolling action has been achieved using a machine code routine which has then been embedded within the BASIC program. In case you are interested in using this technique here is the original machine code and the hex values obtained once it had been assembled:

```
      Hex values                                  Machine code

 0000                                      ORG       0
 0000  8E     0600                         LDX       #1536
 0003  A6     88 10          LOOP          LDA       16,X
 0006  A7     80                           STA       0,X+
 0008  8C     0BEF                         CPX       #3055
 000B  26     F6                           BNE       LOOP
 000D  86     FF                           LDA       #255
 000F  A7     80             LOOP2         STA       0,X+
 0011  8C     0C00                         CPX       #3072
 0014  26     F9                           BNE       LOOP2
 0016  39                                  RTS

0 ERROR(S) DETECTED

SYMBOL TABLE:

LOOP     0003    LOOP2    000F
```

These hex values were then converted into their decimal equivalents which are to be found in the DATA statements in lines 570-580. Line 630 indicates the address that is to be used for the start of the machine code and each of the 23 values in the DATA statements is POKEd into a separate memory location (line 610).

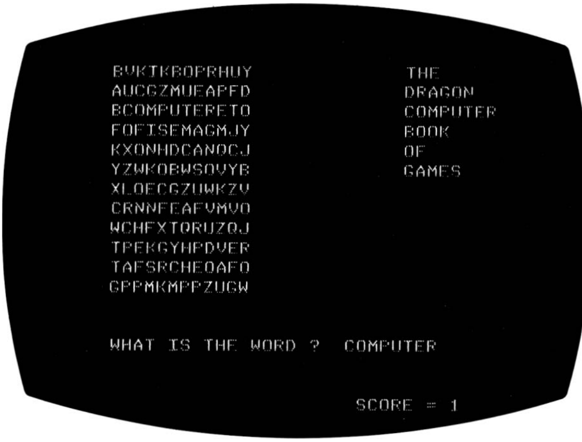
For this game the normal foreground and background colours available in PMODE 0,1 with SCREEN 1,1 have been reversed to give you a black bobsleigh on a buff (that is white!) background. In line 320 PPOINT is used to test whether or not the bobsleigh has hit the side wall. This is done simply by testing what colour is present at the next position that the bobsleigh will be printed at. If the colour is black then you've crashed into the wall.

Program

```
 10 REM BOBSLEIGH
 20 CLEAR 200,32599
 30 K=341
 40 GOSUB 570
 50 GOSUB 390
 60 YB=0
 70 S=1
 80 PMODE 0,1
 90 SCREEN 1,1
100 COLOR 0,5
110 PCLS 5
120 CLS
130 X=RND(50)+100
140 XB=X+25
150 Y=188

160 FOR I=1 TO 500
170 GOSUB 350
180 GOSUB 300
190 X=X+S
200 IF RND(0)>.9 THEN S=-S
210 IF X>220 THEN X=220:S=-S
220 IF X<0 THEN X=0:S=-S
230 NEXT I

240 FOR Z=1 TO 100
250 A=USR0(0)
260 GOSUB 300
270 FOR T=1 TO 20:NEXT T
280 NEXT Z
290 GOSUB 470

300 IF PEEK(K+2)<>255 THEN XB=XB-2
310 IF PEEK(K+3)<>255 THEN XB=XB+2
320 IF PPOINT(XB,YB)=0 THEN GOTO 500
330 PSET (XB,YB)
340 RETURN

350 PSET(X,Y)
360 PSET(X+D*4,Y)
370 A=USR0(0)
380 RETURN
```

```
390 CLS:PRINT @ 5,"B O B S L E I G H"
400 PRINT:PRINT"YOU MUST STEER YOUR
    BOBSLEIGH "
410 PRINT:PRINT"DOWN A DANGEROUS COURSE"
420 PRINT:PRINT"SELECT A DIFFICULTY
    LEVEL -"
430 INPUT"FROM 1- EASY TO 5- DIFFICULT ";D
440 IF D<1 OR D>5 THEN GOTO 430
450 D=9-D
460 RETURN

470 CLS
480 PRINT @ 66,"CONGRATULATIONS - YOU
    MADE IT "
490 GOTO 530
500 SOUND 100,10
510 CLS
520 PRINT @ 69,"YOU CRASHED"
530 INPUT "ANOTHER GAME Y/N";A$
540 IF LEFT$(A$,1)="Y" THEN RUN
550 CLS
560 STOP

570 DATA 142,6,0,166,136,16,167,128,140,11
580 DATA 239,38,246,134,255,167,128,140,12,
    0,38,249,57
590 FOR X=32600 TO 32600+22
600 READ M
610 POKE X,M
620 NEXT X
630 DEF USR0=32600
640 RETURN
```

15
Word Scramble



```
BVKIKBOPRHUY                THE
AUCGZMUEAPFD                DRAGON
BCOMPUTERETO                COMPUTER
FOFISEMAGMJY                BOOK
KXONHDCANOCJ                OF
YZWKOBWSOVYB                GAMES
XLOECGZUWKZV
CRNNFEAFVMVO
WCHFXTQRUZQJ
TPEKGYHPDVER
TAFSRCHEOAFO
GPPMKMPPZUGW


WHAT IS THE WORD ?  COMPUTER


                         SCORE = 1
```

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

The object of the game is to find all the hidden words, using an inverted character as a pointer to identify the *first* letter of each word you find. If you're correct you score a point, otherwise you hear a dismal tone.

How to play

The Dragon 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 one. Then it has to create the puzzle – which takes it a little time – longer the more long words you've included. It tells you that it's 'WORKING' on it so that you don't think it has 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 inverted. 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 Dragon 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 till go up by one), but if you are wrong you will hear a tone. Once you've completed the puzzle you'll see the message "YOU GOT THEM ALL". If you want to give up during the game type "R" and you'll be given the option of resigning. This puzzle can cope with up to eight words, each up to eight letters long.

Typing tips

Notice that you need to type fifty blank spaces between the quotation marks in line 1460. This is to blank out the message in line 1440 when necessary.

Subroutine structure

20	Main play loop
90	Asks for words to be input
250	Finds longest word left in list
310	Constructs puzzle

640	Prints puzzle
870	Controls cursor
1010	Asks for guess of word
1120	Word correct routine
1200	Checks for word in word list
1270	Checks for word in word square
1390	Blanks word square
1440	End of game

Programming details

Some interesting techniques are used in this program. The words are fitted into the square by choosing starting points at random (lines 380–390) and also by choosing directions at random (lines 400–410). 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.

Line 170 appears to be a little strange as it refers to itself. It is however perfectly correct. The program will not move on unless you input a word of eight letters maximum.

Scope for improvement

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

Program

```
 10 REM WORD SCRAMBLE
 20 CLS
 30 CLEAR 1440
 40 GOSUB 90
 50 GOSUB 310
 60 SCORE=0
 70 GOSUB 740
 80 GOSUB 870

 90 DIM L$(8,8)
100 DIM C(8)
110 LST=0
120 INPUT "HOW MANY WORDS ";W
130 IF W<2 OR W>8 THEN SOUND 100,8:GOTO 120
140 FOR I=1 TO W
150 PRINT @ 100+32*I,"WORD NUMBER ";I;"
160 INPUT W$
170 IF LEN(W$)>8 THEN INPUT "MAXIMUM OF
    EIGHT LETTERS'";W$:GOTO 170
180 IF LEN(W$)<1 THEN GOTO 160
190 FOR J=1 TO LEN(W$)
200 L$(I,J)=MID$(W$,J,1)
210 NEXT J
220 C(I)=LEN(W$)
230 NEXT I
240 RETURN

250 M=0
260 J=0
270 FOR Z=1 TO W
280 IF M<C(Z) THEN M=C(Z):J=Z
290 NEXT Z
300 RETURN
```

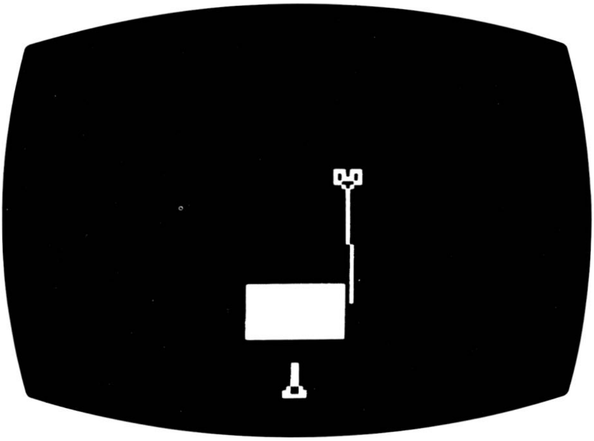
```
310 DIM D$(12,12)
320 GOSUB 1390
330 CLS
340 FOR I=1 TO W
350 GOSUB 250
360 L=C(J)
370 C(J)=0
380 X=RND(12-L)
390 Y=RND(12-L)
400 V=RND(3)-2
410 U=RND(3)-2
420 IF U=0 AND V=0 THEN GOTO 400
430 A=X
440 B=Y
450 IF V<0 THEN A=A+L
460 IF U<0 THEN B=B+L
470 FOR K=1 TO L
480 IF D$(A,B)<>" " AND D$(A,B)<>L$(J,K)
    THEN GOTO 380
490 A=A+V
500 B=B+U
510 NEXT K
520 PRINT @ 2, "WORKING"
530 A=X
540 B=Y
550 IF V<0 THEN A=A+L
560 IF U<0 THEN B=B+L
570 FOR K=1 TO L
580 D$(A,B)=L$(J,K)
590 A=A+V
600 B=B+U
610 NEXT K
620 NEXT I
630 RETURN
```

```
 640 CLS
 650 FOR M=1 TO 12
 660 FOR N=1 TO 12
 670 IF D$(M,N)=" " THEN PRINT ".";
 680 IF D$(M,N)<>" " THEN PRINT D$(M,N);
 690 NEXT N
 700 IF LST=0 OR M>8 THEN PRINT
 710 IF LST=1 AND M<=8 THEN FOR G=1 TO 8:
     PRINT @ 20+G+32*(M-1),L$(M,G):NEXT G
 720 NEXT M
 730 RETURN
 740 INPUT "DO YOU WANT TO PREVIEW THE
     GAME";A$
 750 IF LEN(A$)=0 THEN GOTO 740
 760 IF LEFT$(A$,1)="Y" THEN GOSUB 640
 770 FOR I=1 TO 12
 780 FOR J=1 TO 12
 790 IF D$(I,J)=" " THEN LET
     (I,J)=CHR$(RND(26)+64)
 800 NEXT J
 810 NEXT I
 820 INPUT "DO YOU WANT TO DISPLAY THE
     WORDS BESIDE THE PUZZLE ";A$
 830 IF LEN(A$)=0 THEN GOTO 820
 840 IF LEFT$(A$,1)="Y" THEN LST=1
 850 GOSUB 640
 860 RETURN

 870 X=0
 880 Y=0
 890 PRINT @ 32*Y+X,CHR$(ASC(D$(Y+1,X+1))
     +32);
 900 A$=INKEY$
 910 IF A$="" THEN GOTO 900
 920 IF A$="W" THEN GOTO 1010
 930 PRINT @ 32*Y+X,D$(Y+1,X+1);
 940 IF A$=CHR$(8) AND X>0 THEN X=X-1
 950 IF A$=CHR$(9) AND X<11 THEN X=X+1
 960 IF A$=CHR$(10) AND Y<11 THEN Y=Y+1
 970 IF A$=CHR$(94) AND Y>0 THEN Y=Y-1
 980 IF A$="R" THEN GOSUB 1440
 990 PRINT @Y*32+X,CHR$(ASC(D$(Y+1,X+1))
     +32);
1000 GOTO 900
```

```
1010 PRINT @ 384," ":INPUT "WHAT IS THE
     WORD ";W$
1020 IF LEN(W$)=0 OR LEN(W$)>8 THEN
     SOUND 100,8:GOTO 1010
1030 GOSUB 1200
1040 IF MATCH=0 THEN SOUND 100,8:GOTO 900
1050 FOR U=-1 TO 1
1060 FOR V=-1 TO 1
1070 IF U=0 AND V=0 THEN GOTO 1090
1080 GOSUB 1280
1090 IF MATCH=1 THEN GOTO 1140
1100 NEXT V
1110 NEXT U

1120 SOUND 100,8
1130 GOTO 900
1140 SCORE=SCORE+1
1150 PRINT @ 500,"SCORE =";SCORE;" ";
1160 SOUND 100,8
1170 L$(WORD,1)=" "
1180 IF SCORE=W THEN GOTO 1520
1190 GOTO 900

1200 MATCH=0
1210 FOR I=1 TO W
1220 G$=""
1230 FOR K=1 TO LEN(W$):G$=G$+L$(I,K):
     NEXT K
1240 IF W$=G$ THEN MATCH=1:WORD=I
1250 NEXT I
1260 RETURN

1270 REM CHECKS WORD IN SQUARE
1280 MATCH=0
1290 A=X+1
1300 B=Y+1
1310 FOR I=1 TO LEN(W$)
1320 IF MID$(W$,I,1)<>D$(B,A) THEN RETURN
1330 A=A+U:B=B+V
1340 IF A<1 OR A>12 THEN RETURN
1350 IF B<1 OR B>12 THEN RETURN
1360 NEXT I
1370 MATCH=1
1380 RETURN
```

```
1390 FOR Q=1 TO 12
1400 FOR Z=1 TO 12
1410 D$(Q,Z)=" "
1420 NEXT Z,Q
1430 RETURN

1440 PRINT @ 416," ";:INPUT "ARE YOU SURE
     THAT YOU CANNOT   FIND ANY MORE
     WORDS ";A$
1450 A$=LEFT$(A$,1)
1460 IF A$<>"Y" THEN PRINT @ 416,

                    ": RETURN
1470 CLS
1480 PRINT @ 100,"FINAL SCORE= ";SCORE
1490 INPUT "ANOTHER GAME Y/N ";A$
1500 IF LEFT$(A$,1)="Y" THEN RUN
1510 END
1520 PRINT @ 13*32, "YOU GOT THEM ALL"
1530 GOTO 1490
```

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 twenty percent of it left you will no longer have any protection from the enemies' lasers.

How to play

In this game it is important to keep an eye on the state of your shield since once there is only twenty percent of it left you will be vunerable to attack. Every time you have fired a missile the remaining 'shield strength' is displayed on the screen. Once the shield is so eroded the enemy lasers can destroy you wherever they hit you, including inside the missile base. The object of the game is to score as many hits as possible, so it is worth watching each new enemy ship's orbit at least once or twice before you try to shoot it down. To fire you have to leave your base. Do this by pressing the right or left arrow key. This will take you to a fixed position on the right or left of the screen and launch the missile. Remember to take into account the time it will take for your missile to reach the enemy ship which will continue on its path! At the end of the game your score is displayed and you are given the option of another game.

Subroutine structure

20	Set up routine
60	Main play loop
340	End of game
400	Prints attacker and fire laser
530	Checks to see if player has activated missile
640	Calculates shield strength
710	Moves and fires missile and tests for hit or miss
820	Laser zap routine
880	Sets up attack orbit
1000	Draws initial screen display
1070	Sets up graphics screen
1120	Draws attacker
1150	Draws missile
1170	Explosion graphics and sound

Programming details

One interesting point to note about this program is the way in which the path of the attacking ship is calculated by subroutine 880 and stored in a pair of arrays X and Y to be used repeatedly for the various orbits used during the game. You may also be interested in the way the shield is eroded away as the laster beam passes through it. This effect is achieved by redrawing the path of the laser beam in the background colour. The current shield strength is estimated in line 670 using the PPOINT command to count the number of red points remaining.

Program

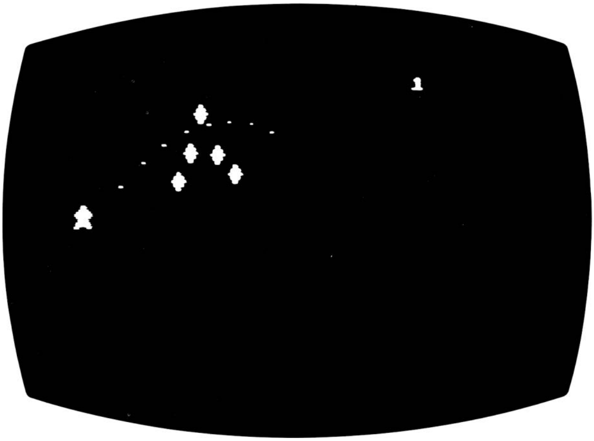
```
  10 REM MIGHTY MISSILE
  20 GOSUB 1070
  30 GOSUB 880
  40 GOSUB 1000
  50 GOSUB 640

  60 FOR A=1 TO 10
  70 DIR=SGN(RND(0)-.5)
  80 R=7-RND(14)
  90 IF R<0 THEN S=3-R:E=29
 100 IF R>=0 THEN S=3:E=29-R
 110 IF DIR=-1 THEN T=S:S=E:E=T
 120 FOR I=S TO E STEP DIR
 130 GOSUB 400
 140 GOSUB 530
 150 IF F=1 THEN GOSUB 710
 160 NEXT I
 170 C$="C3":GOSUB 1150
 180 IF F=1 THEN EN=1
 190 GOSUB 640
 200 IF EN=2 THEN A=11:GOTO 330
 210 IF F=1 THEN F=0:GOSUB 1170
 220 C$="C3":GOSUB 1120
 230 MX=120:MY=189
 240 C$="C2":GOSUB 1120
 250 IF EN=0 THEN GOTO 120
 260 EN=0
 270 IF A=10 THEN GOTO 330
 280 CLS
 290 PRINT "ATTACKER ";A+1
 300 PRINT "SHIELD STRENGTH ";C
 310 FOR Q=1 TO 2000:NEXT Q
 320 SCREEN 1,0
 330 NEXT A
```

```
340 CLS:IF EN=2 THEN PRINT @ 100,
    "THEY GOT YOU"
350 PRINT "YOU HIT ";HIT
360 INPUT "ANOTHER GAME Y/N ";A$
370 IF LEFT$(A$,1)="Y" THEN RUN
380 CLS
390 STOP

400 C$="C3"
410 W$=STR$(X(I)):Q$=STR$(Y(I+R))
420 GOSUB 1150
430 W$=STR$(X(I+DIR)):Q$=STR$(Y(I+R+DIR))
440 C$="C2"
450 GOSUB 1150
460 IF RND(0)<.5 THEN RETURN
470 IF C<=20 AND ABS(MX-X(I+DIR))<8
    THEN EN=2:F=0:GOTO 820
480 D=10-RND(20)
490 LINE (X(I+DIR)+6,Y(I+R+DIR)+4)-
    (X(I+DIR)+D,Y(I+R+DIR)+60),PSET
500 SOUND 50,5
510 LINE (X(I+DIR)+6,Y(I+R+DIR)+4)-
    (X(I+DIR)+D,Y(I+R+DIR)+60),PRESET
520 RETURN

530 IF F=1 THEN RETURN
540 A$=INKEY$
550 IF A$="" THEN RETURN
560 C$="C3":GOSUB 1120
570 IF A$=CHR$(8) THEN MX=MX-64:GOTO 600
580 IF A$=CHR$(9) THEN MX=MX+64:GOTO 600
590 RETURN
600 C$="C2"
610 GOSUB 1120
620 F=1
630 RETURN

640 C=0
650 J=135
660 FOR I=100 TO 149
670 IF PPOINT(I,J)=4 THEN C=C+1
680 NEXT I
690 C=C/50*100
700 RETURN
```

```
 710 C$="C3":GOSUB 1120
 720 MY=MY-8
 730 IF MY<16 THEN F=0:EN=1:GOTO 840
 740 C$="C2":GOSUB 1120
 750 IF ABS(MX-X(I+DIR))>5  THEN RETURN
 760 IF MY-Y(I+R+DIR)>16 OR MY-Y(I+R+DIR)<0
     THEN RETURN
 770 EN=1
 780 C$="C3":GOSUB 1120
 790 HIT=HIT+1
 800 SOUND 150,3
 810 F=0

 820 LINE (X(I+DIR)+6,Y(I+R+DIR)+4)-(MX+4,MY),
     PSET
 830 C$="C2":GOSUB 1120
 840 SOUND 50,5
 850 GOSUB 1170
 860 I=E+DIR
 870 RETURN

 880 DIM X(50):DIM Y(50)
 890 X=0:Y=0
 900 N=31
 910 FOR I=1 TO N
 920 X=X+8
 930 Y=100-INT(((150-X)*(150-X))/190)
 940 X(I)=X
 950 Y(I)=Y
 960 NEXT I
 970 I=1
 980 HIT=0
 990 RETURN

1000 LINE (100,130)-(150,160),PSET,B
1010 PAINT (105,155),4,4
1020 C$="C2"
1030 MX=120
1040 MY=189
1050 GOSUB 1120
1060 RETURN
```

```
1070 PMODE 1,1
1080 SCREEN 1,0
1090 PCLS 3
1100 COLOR 4,3
1110 RETURN

1120 MX$=STR$(MX):MY$=STR$(MY)
1130 DRAW C$+"BM"+MX$+","+MY$+"R10U2L1U2
     HHUU10L2D10G3D2L1D2"
1140 RETURN

1150 DRAW "BM"+W$+","+Q$+C$+"R2F4R1E4R2U
     5L4D3L4U3L4D3"
1160 RETURN

1170 FOR Q=1 TO 30
1180 X=MX+RND(10)
1190 Y=MY-RND(12)
1200 IF RND(0)>.5 THEN PSET(X,Y,2) ELSE
     PSET(X,Y,3)
1210 SOUND RND(255),1
1220 NEXT Q
1230 FOR X=MX TO MX+10
1240 FOR Y=MY-12 TO MY
1250 PSET(X,Y,3)
1260 NEXT Y
1270 NEXT X
1280 RETURN
```

17
Nine Hole Golf



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

How to play

At the start of each hole you are told the distance to the hole – marked on the screen by a flag – and asked to select which club you wish to use. If you've ever played golf or watched it on TV you'll know that the lower the number of the club the further it will drive the ball. 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
80	Input club strength
180	Draws course
260	Driving section
470	Putting section
860	Reports score for each hole
1000	End of game
1040	Plots balls' flight
1290	Lost ball routine
1360	Displays swinging club
1490	Draws golfer

Programming details

An interesting feature of this game is the way the player appears to swing his club. This is done in subroutine 1370 which draws a line which is the continually shifting radius of a circle. This is done using PI, the value of which is calculated in line 1380. The path that the ball appears to follow (subroutine 1050) is a distorted parabola that always carries the ball in the direction of the flag. A combination of two methods are used to blank out the old positions of the golfer. In lines 350–370 and again in lines 730–760, his legs are obliterated by using the PRESET version of the DRAW command while his torso is redrawn at the same position in the background colour.

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 T(9)
 30 B=1
 40 XH=0:XC=0
 50 YH=0:HT=0:YC=0
 60 PMODE 4,1
 70 CLS

 80 H=1
 90 X=RND(10)+10:X$=STR$(X)
100 Y=RND(20)+60:Y$=STR$(Y)
110 XT=RND(80)+120:XT$=STR$(XT)
120 YT=RND(10)+15:YT$=STR$(YT)
130 D=SGN(XT-X)*SQR((XT-X)*(XT-X)
    +(YT-Y)*(YT-Y))
140 PRINT "DISTANCE TO NEXT HOLE IS";
    INT(D)
150 INPUT "WHICH CLUB 1 TO 8";C
160 IF C<1 OR C>8 THEN GOTO 150
170 C=INT((9-C)/B)+1

180 PMODE 4,1:SCREEN 1,0:PCLS 1
190 COLOR 0,1
200 FOR B=1 TO 5
210 XB=RND(30)+50
220 YB=RND(40)+20
230 CIRCLE(XB,YB),2,0,2
240 NEXT B
250 DRAW "BM"+XT$+","+YT$+
    "C0R3L1U5G1R1"
```

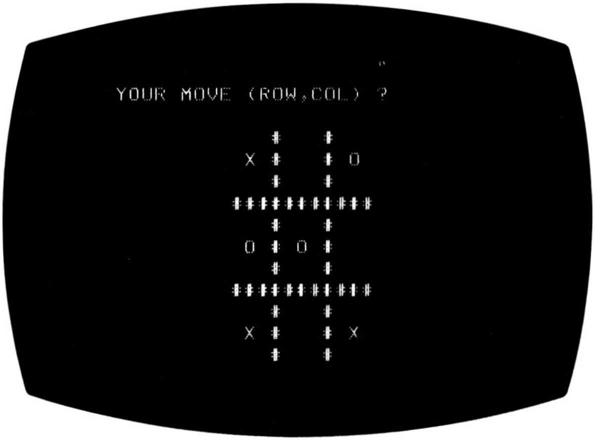
```
260 GOSUB 1490
270 GOSUB 1360
280 GOSUB 1040
290 IF T(H)=-1 THEN GOTO 890
300 B=1
310 D=SGN(XT-XHT)*SQR((XT-XHT)*(XT-XHT)+
    (YT-YHT)*(YT-YHT))
320 FOR T=1 TO 1000:NEXT T
330 IF D<-10 THEN PRINT "YOU OVERSHOT-
    TRY ANOTHER HOLE":FOR W=1 TO 300:
    NEXT W:GOTO 90
340 IF D<10 THEN GOTO 470
350 LINE(X,Y)-(X-3,Y+5),PRESET
360 LINE(X,Y)-(X+3,Y+5),PRESET
370 DRAW "BM"+X$+","+Y$+"C1U5D1G3F3E3H3"
380 X=XHT
390 Y=YHT
400 CLS
410 PRINT "DISTANCE TO HOLE IS NOW";INT(D)
420 INPUT "WHICH CLUB 1 TO 8";C
430 IF C<1 OR C>8 THEN GOTO 420
440 C=INT((9-C)/B)+1
450 SCREEN 1,0
460 GOTO 250

470 CLS
480 PRINT "ON THE GREEN - DISTANCE= ";
    ABS(INT(D))
490 INPUT "WHICH CLUB 1 TO 8";C
500 IF C<1 OR C>8 THEN GOTO 490
```

```
510 PCLS 1
520 SCREEN 1,0
530 XG=RND(20)+20
540 YG=100
550 XH=ABS(D)*15
560 D=XH-XG
570 CIRCLE(XH,YG),1,0
580 COLOR 0,1
590 XG=INT(XG)
600 LINE(XG,YG)-(XG-3,YG+5),PSET
610 LINE(XG,YG)-(XG+3,YG+5),PSET
620 XG$=STR$(XG)
630 YG$=STR$(YG)
640 DRAW "BM"+XG$+","+YG$+"C0U5D1G3F3E3H3"
650 T(H)=T(H)+1
660 H1=(9-C)*(7+RND(3))
670 D=D-H1
680 FOR Z=XG+8 TO XG+H1
690 PSET(Z,YG,0)
700 PRESET(Z,YG)
710 NEXT Z
720 PSET (Z,YG,0)
730 FOR T=1 TO 900:NEXT T
740 LINE(XG,YG)-(XG-3,YG+5),PRESET
750 LINE(XG,YG)-(XG+3,YG+5),PRESET
760 DRAW "BM"+XG$+","+YG$+"C1U5D1G3F3E3H3"
770 XG=XG+H1
780 IF ABS(XG-XH)<5 THEN GOTO 860
790 IF XH-XG<0 THEN XG=2*XH-XG
800 CLS
810 INPUT "WHICH CLUB 1 TO 8";C
820 IF C<1 OR C>8 THEN GOTO 810
830 PCLS 1
840 SCREEN 1,0
850 GOTO 560

860 CLS:PRINT @ 100,"YOU TOOK ";T(H);
    " STROKES"
870 FOR Q=1 TO 1000:NEXT Q
880 CLS
890 H=H+1
900 IF H<=9 THEN GOTO 90
910 CLS
```

```
 920 PRINT @ 33,"THIS ROUND"
 930 PRINT
 940 FOR I=1 TO 9
 950 PRINT "   HOLE ";I;
 960 IF T(I)=-1 THEN PRINT " LOST BALL":
     GOTO 980
 970 PRINT " ";T(I);" STROKES"
 980 NEXT I
 990 PRINT

1000 INPUT "ANOTHER ROUND ";A$
1010 IF LEFT$(A$,1)="Y" THEN RUN
1020 CLS 1
1030 STOP

1040 PRESET(XH+XC,YH+HT+YC)
1050 VT=(C*(1.7+RND(0)*.3))
1060 HT=0
1070 XH=0
1080 Q=(YT-Y)/(XT-X)
1090 VV=-VT*(SIN(45*PI/180))
1100 XC=X+8
1110 YC=Y
1120 VH=VT*(COS(45*PI/180))
1130 HT=HT+VV
1140 YH=Q*XH
1150 VV=VV+2.5
1160 XH=XH+VH
1170 YH=Q*XH
1180 IF XH+XC>255 THEN GOTO 1290
1190 IF (YH+HT+YC)<0 THEN GOTO 1290
1200 IF HT>=0 THEN GOTO 1250
1210 PSET(XH+XC,YH+HT+YC,0)
1220 FOR A=1 TO 50:NEXT A
1230 PRESET(XH+XC,YH+HT+YC)
1240 GOTO 1130
1250 XHT=XH+XC
1260 YHT=YH+HT+YC
1270 PSET(XHT,YHT,0)
1280 RETURN
```

```
1290 PRINT "YOU'VE LOST YOUR BALL."
1300 SOUND 30,4
1310 T(H)=-1
1320 FOR A=1 TO 100:NEXT A
1330 HT=1:YH=1:YC=1
1340 XH=1:XC=1
1350 RETURN

1360 T(H)=T(H)+1
1370 PI=4.0*ATN(1.0)
1380 FOR S=-25 TO 15 STEP 5
1390 A=S/30*PI
1400 SX=7*SIN (A)
1410 SY=7*COS (A)
1420 LINE(X,Y)-(X+SX,Y+SY),PSET
1430 FOR Q=1 TO 50:NEXT Q
1440 LINE(X,Y)-(X+SX,Y+SY),PRESET
1450 GOSUB 1490
1460 IF S=0 THEN SOUND 20,3
1470 NEXT S
1480 RETURN

1490 LINE(X,Y)-(X-3,Y+5),PSET
1500 LINE(X,Y)-(X+3,Y+5),PSET
1510 X$=STR$(INT(X))
1520 Y$=STR$(INT(Y))
1530 DRAW "BM"+X$+","+Y$+"C0U5D1G3F3E3H3"
1540 RETURN
```

18
Noughts and Crosses



```
YOUR MOVE (ROW,COL) ?

           #   #
         X #   # O
           #   #
       ###########
           #   #
         O # O #
           #   #
       ###########
           #   #
         X #   # X
           #   #
```

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

How to play

This game is played on a simple three-by-three grid in the traditional way. You have the 'X' and play first. To make your move you have to specify which square to place your mark on. Type in the row number first, then a comma, then the column number. 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, or a number that does not correspond to a position on the grid, for example 4,1, the Dragon 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 Dragon will display "I WIN" if it has been successful, "YOU WIN" if you've been successful and "DRAW" if it's stalemate. The board has to be completely filled for the game to be over.

Typing tips

Be very careful to type in the semi-colons at the end of 'PRINT @' statements. If you leave them out you run the risk of losing parts of the display.

Subroutine structure

20	Main play loop
190	Evaluates computers move
660	Tries each move
800	Gets player's move
920	Prints board
1010	Plots frame
1160	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'll make – which is why it slips up sometimes and allows you to win!

Program

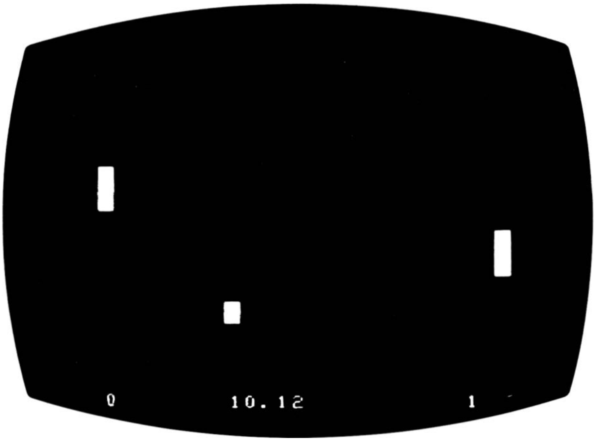
```
 10 REM NOUGHTS AND CROSSES
 20 CLS
 30 DIM X(4)
 40 DIM Y(4)
 50 DIM A(3,3)
 60 DIM B(3,3)
 70 GOSUB 1010
 80 GOSUB 800
 90 CLS
100 GOSUB 1010
110 GOSUB 920
120 GOSUB 660
130 IF ED=1 THEN GOTO 1200
140 IF ED=2 THEN GOSUB 920:GOTO 1160
150 IF DR=1 THEN GOTO 1180
160 GOSUB 920
170 GOTO 80
180 END
```

```
190 FOR L=1 TO 4
200 X(L)=0:Y(L)=0
210 NEXT L
220 FOR L=1 TO 3
230 S=0
240 T=0
250 FOR K=1 TO 3
260 IF A(L,K)=1 THEN S=S+1
270 IF B(L,K)=1 THEN T=T+1
280 NEXT K
290 IF S=0 THEN Y(T+1)=Y(T+1)+1
300 IF T=0 THEN X(S+1)=X(S+1)+1
310 NEXT L
320 FOR L=1 TO 3
330 T=0
340 S=0
350 FOR K=1 TO 3
360 IF A(K,L)=1 THEN S=S+1
370 IF B(K,L)=1 THEN T=T+1
380 NEXT K
390 IF S=0 THEN Y(T+1)=Y(T+1)+1
400 IF T=0 THEN X(S+1)=X(S+1)+1
410 NEXT L
420 GOSUB 480
430 GOSUB 570
440 IF X(4)=1 THEN ED=1:RETURN
450 IF Y(4)=1 THEN ED=2
460 E=128*Y(4)-63*X(3)+31*Y(3)-15*X(2)+
    7*Y(2)
470 RETURN
480 T=0
490 S=0
500 FOR K=1 TO 3
510 T=T+A(K,K)
520 S=S+B(K,K)
530 NEXT K
540 IF S=0 THEN X(T+1)=X(T+1)+1
```

```
550 IF T=0 THEN Y(S+1)=Y(S+1)+1
560 RETURN
570 T=0
580 S=0
590 FOR K=1 TO 3
600 T=T+A(4-K,K)
610 S=S+B(4-K,K)
620 NEXT K
630 IF S=0 THEN X(T+1)=X(T+1)+1
640 IF T=0 THEN Y(S+1)=Y(S+1)+1
650 RETURN

660 M=-256
670 DR=1
680 FOR J=1 TO 3
690 FOR I=1 TO 3
700 IF A(I,J)=1 OR B(I,J)=1 THEN GOTO 760
710 DR=0:B(I,J)=1
720 GOSUB 190
730 IF ED=1 THEN RETURN
740 IF E>M THEN M=E:A=I:B=J
750 B(I,J)=0
760 NEXT I
770 NEXT J
780 B(A,B)=1
790 RETURN

800 PRINT @ 1,;:INPUT "YOUR MOVE
    (ROW COL) ";A$,B$
810 IF LEN(A$)<>1 OR LEN(B$)<>1
    THEN SOUND 50,10:GOTO 800
820 J=VAL( A$)
830 I=VAL( B$)
840 IF I<1 OR I>3 THEN SOUND 50,10:GOTO 800
850 IF J<1 OR J>3 THEN SOUND 50,10:GOTO 800
860 IF A(I,J)=1 OR B(I,J)=1 THEN GOTO 890
870 A(I,J)=1
880 RETURN
890 PRINT @ 480,"POSITION ALREADY OCCUPIED";
900 SOUND 50,10
910 GOTO 800
```

```
 920 FOR J=1 TO 3
 930 FOR I=1 TO 3
 940 IF A(I,J)=1 THEN
     PRINT @ (J*128)+(I*4)-57,"X";
 950 IF B(I,J)=1 THEN
     PRINT @ (J*128)+(I*4)-57,"O";
 960 IF A(I,J)+B(I,J)=0 THEN
     PRINT @ (J*128)+(I*4)-57," ";
 970 NEXT I
 980 PRINT
 990 NEXT J
1000 RETURN

1010 PRINT @  44,"   #   # ";
1020 PRINT @  74,"   #   # ";
1030 PRINT @ 106,"   #   #";
1040 PRINT @ 138,"###########";
1050 PRINT @ 170,"   #   # ";
1060 PRINT @ 202,"   #   # ";
1070 PRINT @ 234,"   #   # ";
1080 PRINT @ 266,"###########";
1090 PRINT @ 298,"   #   #";
1100 PRINT @ 330,"   #   #";
1110 PRINT @ 362,"   #   #";
1120 GOSUB 920
1130 ED=0
1140 DR=0
1150 RETURN

1160 PRINT "I WIN"
1170 GOTO 1210
1180 PRINT "DRAW"
1190 GOTO 1210
1200 PRINT "YOU WIN"
1210 INPUT "ANOTHER GAME Y/N ";A$
1220 IF A$="Y" THEN RUN
1230 CLS
```

19
Anyone for Tennis



This is a two-person game of tennis and you need to have a pair of joysticks in order to play it. As it has been written in BASIC it is not a particularly fast game but it does demonstrate the way in which you can write programs involving the use of joysticks. It is a very short program to type in and makes a pleasant change from the usual one-person computer games like Invaders and Squash.

How to use this program

Firstly, find two joysticks and plug them in to the left hand and right hand joystick connectors! When you run this game, your Dragon asks you to type in first the name of the player operating the left hand joystick and then the name of the player operating the right hand joystick. The players then have two hundred seconds in which to play the game. Although the bats move up and down the screen, the players actually have to move their joystick controls to the right and left – if you find it easier, turn your control through ninety degrees and then your bat's movement will correspond to the joystick's! Each player gains a point for every time the ball bounces against his bat – even when it bounces on the rebound from the wall. The scores are displayed at the bottom of the screen. When the time is up the Dragon announces which player is the winner, or declares a draw, and offers another game.

Subroutine Structure

20	Main play loop
80	Initialises variables, title frame and name input routine
240	Reads joysticks
270	Prints bats
400	Bounces ball
520	End of game

Programming details

The routine that reads the input from the joysticks is a short and simple one – lines 240 to 260. The crucial part of this program is the ball bouncing routine which detects when the ball hits either a bat or the sides of the tennis court and causes the ball to reverse its horizontal or vertical velocity according to what was hit. The part of the program that asks for the two players' names, and then uses this information to declare the winner, is also very easy to understand. It is little additions like these that turn a very simple routine into a properly finished program. Of course, there is always scope for more embellishment of this type – for example you could add comments on the final score to this program.

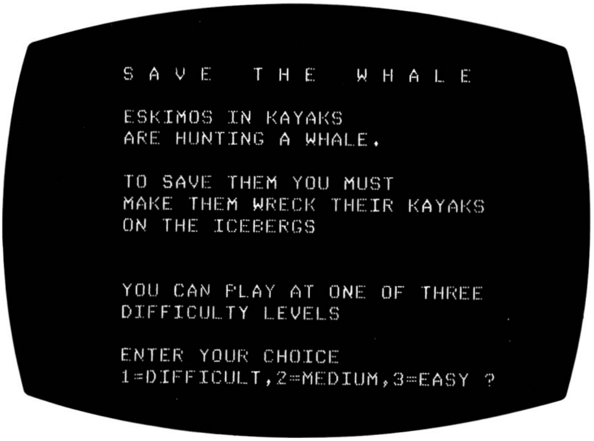
Program

```
10 REM TENNIS
20 GOSUB 80:GOSUB 240
30 GOSUB 270
40 GOSUB 400
50 IF TIMER<200*50 THEN GOTO 20
60 GOTO 520
70 STOP
```

```
 80 CLS
 90 X=28
100 M=32
110 P=32
120 W=3
130 V=1
140 U=1
150 H1=0
160 H2=0
170 PRINT @43,"ANYONE FOR"
180 PRINT @106,"T E N N I S"
190 PRINT @416,;:INPUT "LEFT HAND
    PLAYER'S NAME";L$
200 INPUT "RIGHT HAND PLAYER'S NAME";R$
210 TIMER=0
220 CLS
230 RETURN

240 Y=INT(JOYSTK(0)/4)
250 Z=INT(JOYSTK(2)/4)
260 RETURN

270 PRINT @M," ";
280 PRINT @M-32," ";
290 IF Y=0 THEN Y=1
300 M=X+32*Y
310 PRINT @M,CHR$(128);
320 PRINT @M-32,CHR$(128);
330 PRINT @P," ";
340 PRINT @P-32," ";
350 IF Z=0 THEN Z=1
360 P=W+32*Z
370 PRINT @P,CHR$(128);
380 PRINT @P-32,CHR$(128);
390 RETURN
```

```
400 PRINT @N," ";
410 A=A+V
420 B=B+U
430 IF A=0 OR A=>31 THEN V=-V:SOUND 50,3
440 IF B=0 OR B=14 THEN U=-U:SOUND 100,1
450 N=A+32*B
460 IF PEEK(1024+N)<>96 THEN V=-V:SOUND 150,1:
    IF A<10 THEN H1=H1+1 ELSE H2=H2+1
470 PRINT @N,CHR$(128);
480 PRINT @482,H1;
490 PRINT @490,;TIMER/50;
500 PRINT @505,H2;
510 RETURN

520 CLS
530 IF H1=H2 THEN PRINT "IT'S A DRAW":GOTO 560
540 IF H1>H2 THEN PRINT @128,;L$ ELSE
    PRINT @128,;R$
550 PRINT "WON THE MATCH"
560 PRINT
570 INPUT "ANOTHER GAME ";A$
580 IF A$="Y" THEN RUN
590 IF A$<>"N" THEN GOTO 570
```

20
Save the Whale



```
S A V E   T H E   W H A L E

ESKIMOS IN KAYAKS
ARE HUNTING A WHALE.

TO SAVE THEM YOU MUST
MAKE THEM WRECK THEIR KAYAKS
ON THE ICEBERGS


YOU CAN PLAY AT ONE OF THREE
DIFFICULTY LEVELS

ENTER YOUR CHOICE
1=DIFFICULT,2=MEDIUM,3=EASY ?
```

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

How to play

At the beginning of the game you can select the difficulty level for your turn. Your selection governs the starting positions of the icebergs and so makes the game easier, or harder, to play. To move the whale you press any of the arrow keys. If any kayak runs into an iceberg the kayak vanishes, if the whale runs into an iceberg, the iceberg vanishes – this of course reduces his protection, so it's 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	Set-up routine
60	Title frame
200	Sets-up icebergs
290	Sets-up kayaks
360	Sets-up whale
390	Checks for game over
480	Move whale routine
580	Move kayaks routine
760	Paints whale
880	Paints kayaks
920	Paints icebergs
950	End of game

Programming details

The initial positions of the kayaks are set at random within a band at the edges of the screen (lines 300 and 310). The initial positions of the icebergs are set in a similar fashion (lines 210 and 220) but account is also taken of the difficulty factor input at 150. The PPOINT function is used in lines 670–680 and 710–720 to detect whether a kayak has landed on an iceberg – in which case that is the end of the kayak – or landed on the whale – in which case that is the end of the game.

Program

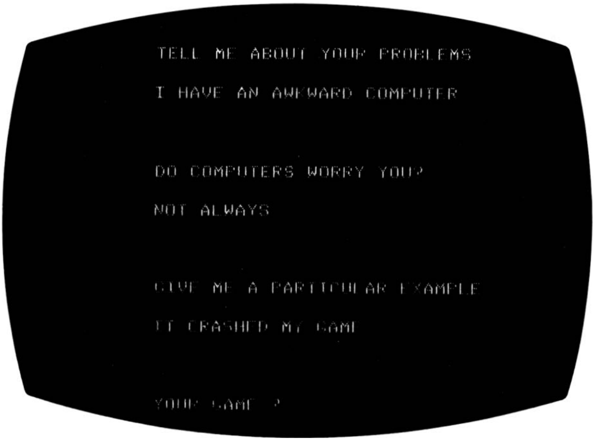
```
10 REM SAVE THE WHALE
20 PCLEAR 4
30 DIM X(15)
40 DIM Y(15)
50 CLS
```

```
 60 PRINT @ 33,"S A V E   T H E   W H A L E"
 70 PRINT @ 97,"ESKIMOS IN KAYAKS"
 80 PRINT " ARE HUNTING A WHALE."
 90 PRINT " TO SAVE THE WHALE YOU MUST"
100 PRINT " MAKE THEM WRECK THEIR KAYAKS"
110 PRINT " ON THE ICEBERGS"
120 PRINT:PRINT " YOU CAN PLAY AT ONE
    OF THREE"
130 PRINT " DIFFICULTY LEVELS"
140 PRINT " ENTER YOUR CHOICE"
150 INPUT " 1=DIFFICULT,2=MEDIUM,3=EASY ";D
160 IF D<1 OR D>3 THEN GOTO 150
170 PMODE 3,1
180 SCREEN 1,1
190 PCLS 6

200 FOR C=1 TO 16
210 X=(SGN(RND(0)-.5))*(RND(30)+40-D*10)+120
220 Y=(SGN(RND(0)-.5))*(RND(30)+30-D*10)+90
230 X=INT(X)
240 Y=INT(Y)
250 X$=STR$(X)
260 Y$=STR$(Y)
270 GOSUB 920
280 NEXT C

290 FOR C=1 TO 15
300 X=(SGN(RND(0)-.5))*(RND(30)+70)+125
310 Y=(SGN(RND(0)-.5))*(RND(30)+60)+95
320 X(C)=X
330 Y(C)=Y
340 GOSUB 880
350 NEXT C

360 X=RND(10)+110
370 Y=RND(10)+85
380 GOSUB 820
```

```
390 GOSUB 480
400 F=0
410 FOR C=1 TO 15
420 IF X(C)=0 THEN GOTO 450
430 F=1
440 GOSUB 580
450 NEXT C
460 IF F=0 THEN GOTO 1000
470 GOTO 390

480 SOUND 100,5
490 A$=INKEY$
500 IF A$="" THEN GOTO 490
510 GOSUB 760
520 IF A$=CHR$(8) THEN X=X-8
530 IF A$=CHR$(9) THEN X=X+8
540 IF A$=CHR$(10) THEN Y=Y+8
550 IF A$=CHR$(94) THEN Y=Y-8
560 GOSUB 820
570 RETURN

580 GOSUB 900
590 E=0
600 D=0
610 D=SGN(X(C)-X)*8
620 X(C)=INT(X(C)-D)
630 E=SGN(Y(C)-Y)*8
640 Y(C)=INT(Y(C)-E)
650 FOR S=X(C)-2 TO X(C)+2 STEP 4
660 FOR Z=Y(C)-1 TO Y(C)+1
670 IF PPOINT(S,Y(C))=5 THEN X(C)=0:
    GOTO 750
680 IF PPOINT(S,Y(C))=7 THEN GOTO 950
690 NEXT S
700 FOR S=Y(C)-4 TO Y(C)+4 STEP 8
710 IF PPOINT(X(C),S)=5 THEN X(C)=0:
    GOTO 750
720 IF PPOINT(X(C),S)=7 THEN GOTO 950
730 NEXT S
740 GOSUB 880
750 RETURN
```

```
 760 PAINT(X,Y),6,6
 770 COLOR 7,6
 780 CIRCLE(X,Y),4,6
 790 LINE(X+5,Y)-(X+7,Y),PRESET
 800 LINE(X+7,Y+2)-(X+7,Y-2),PRESET
 810 RETURN
 820 CIRCLE(X,Y),4,7,1
 830 PAINT(X,Y),7,7
 840 COLOR 7,6
 850 LINE(X+5,Y)-(X+7,Y),PSET
 860 LINE(X+7,Y+2)-(X+7,Y-2),PSET
 870 RETURN

 880 CIRCLE(X(C),Y(C)),2,8,2
 890 RETURN
 900 CIRCLE(X(C),Y(C)),2,6,2
 910 RETURN

 920 DRAW "BM"+X$+","+Y$+"C5R10H4G2H1G3"
 930 PAINT(X+6,Y-2),5,5
 940 RETURN

 950 CLS
 960 PRINT
 970 PRINT "THE WHALE IS DEAD"
 980 GOSUB 760
 990 GOTO 1030
1000 CLS
1010 PRINT
1020 PRINT "  YOU SAVED THE WHALE THIS TIME"
1030 PRINT
1040 INPUT "ANOTHER HUNT Y/N";A$
1050 IF LEFT$(A$,1)="Y" THEN RUN
1060 CLS
1070 PRINT " BYE"
1080 END
```

21
Dragontalk



```
TELL ME ABOUT YOUR PROBLEMS
I HAVE AN AWKWARD COMPUTER

DO COMPUTERS WORRY YOU?
NOT ALWAYS

GIVE ME A PARTICULAR EXAMPLE
IT CRASHED MY GAME

YOUR GAME ?
```

Do you ever find yourself *talking* to your Dragon? Well, if you do you may be disappointed that it never answers back. This program, however, changes all that and gives your Dragon the chance to start a conversation with you. Although it may not be able to rival the agony aunts of the glossy magazines, your Dragon 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 great deal of fun trying to conduct a dialogue with your Dragon.

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 Dragon 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 linefuleach times, 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 Dragon.

When you RUN this program you'll notice that the Dragon's replies are displayed in inverse graphics. You need to input some of the program in inverse graphics – the parts that appear in lowercase in the listings. Press SHIFT and zero together before you type these words and phrases and afterwards press SHIFT and zero again to restore the normal display.

Subroutine structure

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

Programming details

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

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

Scope for improvement

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

Program

```
10 REM DRAGONTALK
20 DIM W(20,2)
30 GOSUB 100
40 GOSUB 400
50 GOSUB 560
60 GOSUB 920
70 IF NM<>0 THEN ON NM GOSUB 1250,1270,1290,
   1310,1730,1780,1690,1710,1610,1380,2140,
   1330,1470,1900,2040,1800,1820,1840,1330,
   1880,1860
80 GOSUB 2210
90 GOTO 40
```

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

```
400 A$=""
410 B$=INKEY$
420 REM INPUT ROUTINE
430 IF B$="" THEN GOTO 410
440 IF INKEY$<>"" THEN GOTO 440
450 IF ASC(B$)=13 THEN GOTO 510
460 IF ASC(B$)=8 AND A$<>"" THEN
    A$=LEFT$(A$,LEN(A$)-1):GOTO 490
470 IF ASC(B$)<32 OR ASC(B$)>126 THEN
    GOTO 400
480 A$=A$+B$
490 PRINT @ 450, A$;" "
500 GOTO 410
510 IF A$=R$ THEN PRINT:PRINT @ 450,
    "YOU'RE REPEATING YOURSELF":GOTO 400
520 R$=A$
530 IF A$="" THEN GOTO 400
540 A$=" "+A$
550 RETURN
```

```
560 REM SPLITS SENTENCES INTO WORDS
570 N=1
580 B=0
590 FOR I=1 TO LEN(A$)
600 IF ((MID$(A$,I,1)=" " OR MID$(A$,I,1)=
    ",") AND B=0) THEN B=1
610 IF ((MID$(A$,I,1)<>" " AND MID$(A$,I,1)
    <>",") AND B<=1) THEN W(N,1)=I:B=2
620 IF ((MID$(A$,I,1)=" " OR MID$(A$,I,1)=
    ",") AND B=2) THEN W(N,2)=I-1:N=N+1:B=0
630 NEXT I
640 W(N,2)=LEN(A$)
650 FOR I=1 TO N

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

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

```
 920 RESTORE:FOR I=1 TO 34:READ Q$:NEXT I
 930 READ B$,NM
 940 IF B$="S" THEN GOTO 1010
 950 FOR I=1 TO N
 960 IF MID$(A$,W(I,1),(W(I,2)-W(I,1)+1))<>
     B$ THEN GOTO 990
 970 T$=RIGHT$(A$,LEN(A$)-(W(I,2)))
 980 RETURN
 990 NEXT I
1000 GOTO 930
1010 NM=0
1020 IF M$<>"" THEN GOTO 1050
1030 P$=N$(RND(3))
1040 RETURN
1050 P$=I$(RND(3))+M$
1060 RETURN

1070 DATA "COMPUTER",1,"MACHINE",1,
     "PROGRAM",1
1080 DATA "LIKE",2,"SAME",2,"ALIKE",2
1090 DATA "IF",3,"EVERYBODY",4
1100 DATA "CAN",5,"CERTAINLY",6
1110 DATA "HOW",7,"BECAUSE",8
1120 DATA "ALWAYS",9
1130 DATA "EVERYONE",4,"NOBODY",4
1140 DATA "WAS",10
1150 DATA "I*",11
1160 DATA "NO",12
1170 DATA "YOUR*",13
1180 DATA "YOU'RE*",14,"YOU*",15
1190 DATA "HELLO",16,"MAYBE",17
1200 DATA "MY*",18,"NO",19
1210 DATA "YES",6,"WHY",20
1220 DATA "PERHAPS",17,"SORRY",21
1230 DATA "WHAT",20
1240 DATA "S",0
```

```
1250 P$="DO COMPUTERS WORRY YOU ?"
1260 RETURN
1270 P$="in what way ?"
1280 RETURN
1290 P$="why talk of possibilities"
1300 RETURN
1310 P$="really "+B$+" ?"
1320 RETURN
1330 IF I=N THEN GOTO 1360
1340 I=I+1
1350 IF MID$(A$,W(I,1),(W(I,2)-W(I,1)+1))=
     "ONE" THEN B$=B$+" ONE":GOTO 1310
1360 P$=J$(RND(2))
1370 RETURN
1380 IF I=N THEN GOTO 2270
1390 I=I+1
1400 IF I>N THEN GOTO 1020
1410 IF MID$(A$,W(I,1),(W(I,2)-W(I,1)+1))<>
     "YOU*" THEN GOTO 1440
1420 P$="WHAT IF YOU WERE "+
     RIGHT$(A$,(LEN(A$)-W(I,2)))+" ?"
1430 RETURN
1440 IF MID$(A$,W(I,1),(W(I,2)-W(I,1)+1))<>
     "I*" THEN GOTO 1020
1450 P$="WOULD YOU LIKE TO BELIEVE I WAS "+
     RIGHT$(A$,(LEN(A$)-(W(I,1)+1)))
1460 RETURN
1470 I=I+1
1480 IF I>N THEN GOTO 1360
1490 IF MID$(A$,W(I,1),(W(I,2)-W(I,1)+1))=
     "MOTHER" THEN GOTO 1590
1500 IF MID$(A$,W(I,1),(W(I,2)-W(I,1)+1))=
     "FATHER" THEN GOTO 1590
1510 IF MID$(A$,W(I,1),(W(I,2)-W(I,1)+1))=
     "SISTER" THEN GOTO 1590
1520 IF MID$(A$,W(I,1),(W(I,2)-W(I,1)+1))=
     "BROTHER" THEN GOTO 1590
1530 IF MID$(A$,W(I,1),(W(I,2)-W(I,1)+1))=
     "WIFE" THEN GOTO 1590
1540 IF MID$(A$,W(I,1),(W(I,2)-W(I,1)+1))=
     "HUSBAND" THEN GOTO 1590
1550 IF MID$(A$,W(I,1),(W(I,2)-W(I,1)+1))=
     "CHILDREN" THEN GOTO 1590
1560 IF LEN(T$)>10 THEN M$=T$
1570 P$="your "+T$+" ?"
```

```
1580 RETURN
1590 P$="tell me more about your family"
1600 RETURN
1610 P$="give me a particular example"
1620 RETURN
1630 I=I+1
1640 IF I>N THEN P$="am i what ?":RETURN
1650 P$="why are you interested in
     whether i am "+RIGHT$(A$,W(I,1)+
     " or not ?"
1660 RETURN
1670 P$="do you think you are "+
     RIGHT$(A$,LEN(A$)-W(I,2))
1680 RETURN
1690 P$="why do you ask ?"
1700 RETURN
1710 P$="tell me about any other reasons"
1720 RETURN
1730 I=I+1
1740 IF I>N THEN P$="what ?":RETURN
1750 IF MID$(A$,W(I,1),(W(I,2)-W(I,1))+1)=
     "I*" THEN P$="do you believe i can "+
     RIGHT$(A$,(LEN(A$)-W(I,2)))+" ?":RETURN
1760 IF MID$(A$,W(I,1),(W(I,2)-W(I,1))+1)=
     "YOU*" THEN P$="do you believe you
     can "+RIGHT$(A$,LEN(A$)-W(I,2))+" ?":
     RETURN
1770 GOTO 1010
1780 P$="YOU SEEM VERY POSITIVE"
1790 RETURN
1800 P$="PLEASED TO MEET YOU - LET'S TALK
     ABOUT YOUR PROBLEMS"
1810 RETURN
1820 P$="COULD YOU TRY TO BE MORE POSITIVE"
1830 RETURN
1840 P$="WHY ARE YOU CONCERNED ABOUT MY "+T$
1850 RETURN
1860 P$="YOU DON'T HAVE TO APOLOGISE TO  ME"
1870 RETURN
1880 P$="SOME QUESTIONS ARE DIFFICULT TO
     ANSWER..."
1890 RETURN
1900 I=I+1
1910 IF I>N THEN GOTO 1020
```

```
1920 P$="I AM SORRY TO HEAR THAT YOU ARE "+
     MID$(A$,W(I,1),(W(I,2)-W(I,1)+1))
1930 IF MID$(A$,W(I,1),(W(I,2)-W(I,1)+1))=
     "SAD" THEN RETURN
1940 IF MID$(A$,W(I,1),(W(I,2)-W(I,1)+1))=
     "UNHAPPY" THEN RETURN
1950 IF MID$(A$,W(I,1),(W(I,2)-W(I,1)+1))=
     "DEPRESSED" THEN RETURN
1960 IF MID$(A$,W(I,1),(W(I,2)-W(I,1)+1))=
     "SICK" THEN RETURN
1970 P$="HOW HAVE I HELPED YOU TO BE "
     +MID$(A$,W(I,1),(W(I,2)-W(I,1)+1))
1980 IF MID$(A$,W(I,1),(W(I,2)-W(I,1)+1))=
     "HAPPY" THEN RETURN
1990 IF MID$(A$,W(I,1),(W(I,2)-W(I,1)+1))=
     "ELATED" THEN RETURN
2000 IF MID$(A$,W(I,1),(W(I,2)-W(I,1)+1))=
     "GLAD" THEN RETURN
2010 IF MID$(A$,W(I,1),(W(I,2)-W(I,1)+1))=
     "BETTER" THEN RETURN
2020 P$="IS IT BECAUSE YOU ARE "+
     MID$(A$,W(I,1),W(I,2)-W(I,1)+1)+
     " YOU WOULD LIKE TO TALK TO ME ?"
2030 RETURN
2040 IF I=1 THEN GOTO 2060
2050 IF MID$(A$,W(I-1,1),(W(I-1,2)-
     W(I-1,1)+1))="ARE*" THEN GOTO 1670
2060 I=I+1
2070 IF I>N THEN GOTO 2270
2080 IF MID$(A$,W(I,1),(W(I,2)-W(I,1)+1))=
     "ARE*" THEN GOTO 1900
2090 IF MID$(A$,W(I,1),(W(I,2)-W(I,1)+1))=
     "WANT" OR MID$(A$,W(I,1),(W(I,2)-
     W(I,1)+1))="NEED" THEN P$="WHAT WOULD
     IT MEAN IF YOU GOT "+RIGHT$(A$,LEN(A$)-
     W(I,2)):RETURN
2100 IF MID$(A$,W(I,1),(W(I,2)-W(I,1)+1))=
     "THINK" THEN P$="DO YOU REALLY THINK
     SO":RETURN
2110 IF MID$(A$,W(I,1),(W(I,2)-W(I,1)+1))=
     "CAN'T" OR MID$(A$,W(I,1),(W(I,2)-
     W(I,1)+1))="CANNOT" THEN P$="HOW DO YOU
     KNOW YOU CAN'T "+RIGHT$(A$,LEN(A$)-
     W(I,2)):RETURN
2120 IF MID$(A$,W(I,1),(W(I,2)-W(I,1)+1))=
     "FEEL" THEN P$="TELL ME MORE ABOUT HOW
     YOU FEEL":RETURN
2130 GOTO 1020
2140 IF I-1<1 THEN GOTO 2160
```

```
2150 IF MID$(A$,W(I,1),(W(I,2)-W(I,1)+1))=
     "AM" THEN GOTO 1630
2160 I=I+1
2170 IF I>=N THEN P$="WHAT AM I ?":RETURN
2180 IF MID$(A$,W(I,1),(W(I,2)-W(I,1)+1))=
     "AM" THEN P$="WHY DO YOU THINK SO ?":
     RETURN
2190 P$="IS THAT WHAT YOU THINK OF ME ?":
     RETURN
2200 RETURN

2210 FOR J=1 TO LEN(P$)
2220 IF MID$(P$,J,1)="*" THEN GOTO 2240
2230 IF ASC(MID$(P$,J,1))>64 AND
     ASC(MID$(P$,J,1))<91 THEN PRINT
     CHR$(ASC(MID$(P$,J,1))+32); ELSE PRINT
     MID$(P$,J,1);
2240 NEXT J
2250 PRINT:PRINT:PRINT:PRINT
2260 RETURN

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

21 TOP QUALITY GAMES TO FIRE YOUR IMAGINATION!

Here are twenty-one fantastic, high quality games written specially for your Dragon 32 and designed to provide hours of fun. All the games have been fully tested and crash-proofed and our selection makes full use of the exciting range of the Dragon's facilities. Each program comes with an explanation of how to play the game and how the program works. Tips on how to change them creatively are also provided, so that you can enjoy an almost infinite range of variations if you wish.

Some of the games included here are based on popular favourites like Invaders and Squash while others are completely novel like Corner the Dragon, Across the Ravine and Commando Jump. There is a word puzzle to keep all the family absorbed and a fascinating conversational program that lets your Dragon answer you back.

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

The Authors

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

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

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

Front cover photograph shows the Dragon 32 Home Computer.

GRANADA PUBLISHING
Printed in Great Britain 0 246 12102 5

£5.95 net

JAMES, GEE AND EWBANK **THE DRAGON 32 BOOK OF GAMES** GRANADA