# The Second Book of Listings

Martin Bryant

BRITISH BROADCASTING CORPORATION

# The Second Book of Listings

BRITISH BROADCASTING CORPORATION

# The Second Book of Listings

More great Model B programs

Martin Bryant

© Martin Bryant 1984
First published 1984

Phototypeset by Rowland Phototypesetting Ltd.,
Printed in Great Britain by St. Edmundsbury Press,
Bury St. Edmunds, Suffolk.

Published by the British Broadcasting Corporation
35 Marylebone High Street, London W1M 4AA.

ISBN 0 563 21071 0

# Contents

7 **Introduction**

*Reflex/Reaction Games*

12 **Ricochet Golf**
Great fun for the family and friends or for the one-man crazy golfer.

19 **Meteors**
A simple, standard 'shoot-the-hostiles' scenario.

23 **Rollers**
You and the computer play at frenziedly trying to brush each other out of the screen.

27 **Slalom**
Screen skiing: avoid the hazards and pick up speed along with your success.

31 **Rebel**
Line up the enemy craft in your sights, and fire.

*Tactical Games*

35 **Balrog**
A deadly game of cat-and-mouse.

41 **Graves**
A game to play after the midnight movie.

46 **March**
Get your army to safety.

52 **Mine**
Cross the minefield before your detector loses its potency.

*Puzzle Games*

56 **Solitaire**
The ancient puzzle of pegs-in-holes.

60 **Towers**
The famous Towers of Hanoi transfer puzzle.

64 **Rotate**
Rearrange the puzzle to its correct shape.

68 **Quiz**
A general-knowledge quiz that you can modify.

*Strategy Games*

74 **Backgammon**
A superb version of the game of kings, with four levels.

85 **Awari**
This subtle African game has five levels.

*Demonstration*

92 **Queens**
The eight dancing chess queens: a classical problem solved. The listing is invaluable for techniques.

*Text Editor*

95 **Edit**
A simple, general-purpose word processor in 6502 assembly code.

*Tutorial*

108 **Adventure**
A simple text adventure for total beginners at writing this type of game.

# Introduction

This book has been written to be different from other books of its kind. Instead of just providing a mass of fairly poor programs, we have tried to provide a good selection of quality programs, both entertaining and worth studying.

This means that the programs are in general a little longer, though fewer in number. However, many of these programs are at least as good as other commercially available programs which you might have to pay several pounds for.

Structured programming techniques have been used, to aid easy understanding and any future enhancements which you may wish to make yourself. A general section on program improvements/development is also included.

Each program is accompanied by a description of the variables, procedures, algorithms, display, operation and rules, as well as information on how to change or add your own features, and a list of suggestions. Also included for several programs are my own personal 'best-scores' for you to try to beat!

## Typing-in programs

All programs have been listed with the list option LISTO7. This indents certain statements by a predetermined number of spaces to aid readability. These leading spaces (between the line number and the first statement on that line) may be omitted to save typing. They can, of course, be automatically induced again by typing LISTO7<RETURN> before you LIST the program yourself.

If a long program-line is printed over several lines in the book, then you must not press the <RETURN> key until the last printed-line of the whole program-line has been entered. (*New* program-lines always start with a line number.)

REM statements have only been used to separate blocks of code. This aids the readability of a program, which speeds development. The detailed program information is included in the accompanying explanations. You need not type in the REM's at all, but it is advisable to type in the line number and the word REM only, so that you can ensure that you have typed in the correct number of lines by using RENUMBER.

Every effort has been made to ensure that no errors remain in these programs. If, after typing in a program, you encounter difficulties, then check the listings line-by-line to try to find all typing errors. Remember that computers are very fussy about the exact format of the languages they use. A missing semi-colon or full-stop could mean disaster to the running of the program.

Before printing, these programs were all renumbered, starting at line 10, with an interval of 10 (the default values). When you have finished typing in a program, a quick easy test is to type:

RENUMBER<RETURN> and then check that the last line number in your program matches the last line number in the book. If it does not, then you have missed out a line (or block of lines) or even added some lines of your own!

You should then check the program from the beginning until you find the first incorrectly numbered line; correct the error; and repeat the process.

Here is a list of other possible problems and suggested solutions:

| *Problem* | *Suggestion* |
|---|---|
| eeee ERROR at nnn0. | Check carefully the line in error, with the book. Also check associated lines; ie the lines where the various variables/procedures are defined. Check also for use of square brackets where round brackets are needed, or for missing brackets. Also, spaces after certain words are essential. |
| Program hangs when run. | Probably caused by an incorrect VDU statement; eg missing off the final semi-colon on the statement VDU23;8202;0;0;0; may cause the computer to hang. |
| Program runs but seems to behave strangely. | Check equations have been typed correctly; eg a missing minus sign may make a spaceship run away from you rather than attack you. |
| Layout of screen messages is messy. | Check TAB statements are correct. Also check that you have typed the correct number of spaces in PRINT statements and also that semi-colons on the end of PRINT statements haven't been omitted. |

**Note** The # symbol in the listings should be replaced by the £ symbol and vice versa.

## Improvements

*General*

Certain FX calls can be used to put finishing touches to programs. These are not essential and, because they will add to the program length, have not been included in all the listings.

They are, however, explained here so that you can add them to your favourite programs if you wish.

Simply add lines at the start of the program to include whichever of the following FX calls you desire:

| | |
|---|---|
| To disable the cursor keys | *FX4,1 |
| To disable the <ESCAPE> key | *FX229,1 or *FX200,1 |
| To disable the function keys | *FX225,0 |
| To disable all sounds | *FX210,1 |
| To enable all sounds | *FX210,0 |
| To flush memory if <BREAK> pressed | *FX200,2 |

The <BREAK> key can also be set up to allow easy re-starting of the program. It can be achieved by including a line:
*KEY10OLD|MRUN|M (see page 143 of the USER GUIDE for more information). You may find it useful to add a line to each program so that, when it ends, it goes into MODE 7.

*Joysticks*

Certain of the programs could be adapted for use with joysticks. You simply need to change the section of code which reads the keyboard entries for left, right, up, down, fire etc. to read the joystick value using ADVAL. You must of course do some sort of translation of the joystick value into, for example, a binary state value, to distinguish right from left, and so on.

Remember that some people may prefer to use keyboard controls, so it would be best to add the code to handle the joysticks with some sort of selection code to allow the user to choose *whichever* method he prefers.

*Pausing*

When playing certain video-games it can be very annoying if the phone rings when you're heading for your all-time high score! It would be very useful to have a 'pause' facility built into the program. This can be done very easily, by adding a simple test in the main loop, for a 'pause'-key being pressed. If it isn't then just carry on. If it is, then just wait in a delay loop until a 'continue'-key is pressed.

*Personalization*

If you change a program drastically you may wish to put your own mark on it. This can be done with simple PRINT statements on a header page, before the main program runs. Also including your name in REM statements is another way to 'personalize' your program.

*Protection*
When you've added your mark to a program you may wish to protect it from prying eyes. There are many methods. Here are a few such:
● Disabling all the usual 'interrupt' keys (<ESCAPE>,<BREAK>) is a good start.
● Then, as well as having your name in obvious PRINT statements, you could include some copyright message in a coded form deep within the program. This message may only be activated if a certain secret (not to you!) combination of keys is held down at a specific moment in the program.
● Adding a password system at the start of the program will also help. The secret password should not be easily visible within the program listing, or easily decoded.
● Checksumming your program is very good at catching somebody else's alterations! Do checksums of small sections of the program at frequent intervals. Then, if the program discovers that it has been tampered with, don't just 'crash' the program in one fell swoop. Be devious! Change a few important locations so that the program starts to behave slightly strangely. You may convince the pirate that the change he made introduced a bug. He may spend hours trying to find his error and eventually give up!

The rule for protecting programs is simple. The more effort you spend protecting it, the more people you'll stop from tampering with it.

Unfortunately, no amount of effort is going to stop the dedicated, intelligent, professional pirate. But at least you can make it as hard as possible for him.

*Display Equipment*
Displays vary greatly in their picture clarity and colour/grey scale distinction. These programs have been developed and tested on ordinary colour and black/white televisions and the most suitable colour options chosen. You may, however, find that your own display equipment would show up different colours better.

You can go through the programs, changing the COLOUR and GCOL statements where necessary, but this would be rather tedious. The BBC Micro provides a much easier method of instant colour switching with the VDU19 statement (see page 382 of the BBC Micro User Guide for more details).

Including a few VDU19 statements at the start of a program allows you to play around and find the best colours for the programs for your particular equipment.

*Sound*
The BBC Micro provides some very good sound facilities. Using them to their full, however, may be more difficult than it appears. Putting complicated whoops and zaps in a program may only irritate the user rather than enhance the game. The volume of general background noises might be better if it were lower than the important explosions or input-prompts.

Envelope design is also very tricky. A simple program to allow easy manipulation of envelope parameters, with instant sound feed-back, may aid development greatly.

*Layout*
The layout of a program is a very important factor to consider during development. It should be tidy, logical and consistent. Haphazard programs will be much harder to follow and debug.

The programs in this book all have:

The main program section at the start; followed by

The various procedures.

Declarations of arrays are done early in the main section.

Logically similar procedures are grouped together; eg all display handling subroutines are put together at the end of the program. Even the alphabetical ordering of procedure names can save time when searching for the first line of a procedure.

## Development/target systems

All the programs were developed on a BBC Model B microcomputer (OS 1.20) with discs (DFS 0.9H).

All the programs will work on either disc- or tape-based systems.

If, however, you are using the programs on a disc-based system you may need to reset PAGE to &1100 before CHAINing certain programs, because of their larger size.

The programs are not meant to, but should, work on operating systems previous to OS 1.20.

Some of the programs will work on a Model A micro as they stand, and even most of the others could be adapted to work on a model A, by simply using a lower-resolution graphics mode.

**Note** There is a character-count scale on page 115.

# Ricochet Golf

## Rules

The rules are as for normal golf; ie hit your ball into the hole using as few shots as possible.

The edges of this golf course, however, are elastic and so the ball bounces off anything that it hits.

Up to nine players can play at once, each taking it in turn to complete the current hole.

## Display

The display shows the current hole, its par rating and the current player's name, along with his shot number.

The ball is shown with a line near it, to show the direction of aim.

When all players have completed the hole, the par ratings for each player are shown on a scoreboard.

## Operation

To aim the ball the 'cue' near the ball can be rotated with the keys:
'Z' – rotate cue clockwise
'X' – rotate cue anti-clockwise

To hit the ball press a number key from '1' to '9'. The weakest strength hit is '1' and the strongest (longest) hit a '9'.

Because different display equipment shows different colours better, a facility has been provided to change the foreground and background colours easily! The colours may be moved one at a time through the eight possible colours on the BBC micro.
*To advance the foreground colour press 'F'.*
*To advance the background colour press 'B'.*
(Note that when the foreground and background colours selected are the same, the hole will 'disappear' until you change one of the colours.) You could, perhaps at a certain stage of a party, invite people to play blind ricochet golf!

## Program

The program reads the hole shapes from the data statements at the end of the program. The first number is the par value for the hole, followed by the X,Y coordinates of the apexes, and finally the hole and tee coordinates. A negative apex X-coordinate signifies an absolute move to the current coordinate pair. A positive apex X-coordinate signifies an absolute draw to the current coordinate pair. The final apex coordinates are specified with a negative Y-coordinate. The hole coordinates specify the centre of the drawn hole. The tee is specified by a lower-X-coordinate, an upper-X-coordinate and a Y-coordinate. The ball is teed off from a random position along the tee line.

| *Section/Variables* | *Function* |
|---|---|
| Main routine | Initialize data,setup players, main game loop, game over |
| HCX% | Store hole X-coordinates |
| HCY% | Store hole Y-coordinates |
| SC% | Player scores |
| N$ | Player names |
| BC% | Background colour |
| FC% | Foreground colour |
| NP% | Number of players |
| NH% | Number of holes |
| TPAR% | Total par |
| HN% | Current hole number |
| PN% | Current player number |
| PROC PLAYHOLE | Play current hole to completion for current player |
| SH% | Shot number |
| CH% | Cue angle |
| LWI% | Last-wall-hit index |
| K$ | Input key |
| BE | Ball energy |
| BA | Ball angle |
| PROC WHOOP | Play 'hole-in-one' fanfare |
| PROC DELAY | Delay for one second |
| FN HOLED | Check if ball in hole |
| PROC MOVEBALL | Move ball when hit |
| BX | Ball X-coordinate |
| BY | Ball Y-coordinate |
| BDDX | Saved increment in ball X-coordinate |
| BDDY | Saved increment in ball Y-coordinate |
| BDX | Increment in ball X-coordinate |
| BDY | Increment in ball Y-coordinate |
| MISS% | Missed-wall flag |
| PROC SETD | Set X,Y increments for ball movement |
| PROC PBALL | Print the ball |
| PROC MOVECLUB | Erase, move and redraw club |
| A% | Angle change |

| | |
|---|---|
| PROC PCLUB | Print club |
| PROC PSCORES | Print player's scoresheet |
| PROC READHOLE | Read hole 'shape' from data tables |
| PAR% | Par value for current hole |
| HCI% | Hole coordinate pair index |
| HX% | Hole X-coordinate |
| HY% | Hole Y-coordinate |
| TLX% | Tee lower X-coordinate |
| TUX% | Tee upper X-coordinate |
| TY% | Tee Y-coordinate |
| PROC PHOLE | Print current hole |

## Suggestions

Construct your own data statements for a collection of different golf courses!

For variety you could change the course to only play nine holes say, but select which nine randomly from the whole list of eighteen holes (or many more if you add your own).

There is a minor infelicity: The message at the top of the screen can be 'after 1 holes'. Make it grammatical!
(My best score: 9 under par)

## The Listing

```
   10 *FX4,1
   20 DIMHCX%(99),HCY%(99),SC%(9),N$(9)
   30 BC%=2:FC%=7
   40 REPEAT
   50   RESTORE
   60   MODE7:PRINTTAB(5,1)"Number of players(1-9)
?";
   70   REPEAT NP%=ASCGET$-ASC"0"
   80     UNTILNP%>=1ANDNP%<=9
   90   PRINT;NP%
  100   FORI%=1TONP%:PRINT'"Name of player ";I%;:I
NPUTN$:N$(I%)=LEFT$(N$,15)
  110     NEXT
  120   FORI%=1TONP%:SC%(I%)=0
  130     NEXT
  140   NH%=18:TPAR%=0
  150   FORHN%=1TONH%
  160     PROCREADHOLE
  170     FORPN%=1TONP%
  180       MODE4:VDU23;8202;0;0;0;
  190       PROCPHOLE
  200       PROCPLAYHOLE
  210       NEXT
  220       MODE7
  230       PROCPSCORES
  240       NEXT
  250     PRINT'"Another round?";:*FX15,1
  260     UNTILGET$="N"
  270 MODE7
  280 END
  290 REM******************************
  300 DEFPROCPLAYHOLE
  310 GCOL3,1
  320 SH%=0
  330 REPEAT
  340   CA%=270:PROCPCLUB:SH%=SH%+1:LWI%=-1
  350   PRINTTAB(14,3)"Shot ";SH%" for "N$(PN%)
  360   REPEAT
  370     VDU19,0,BC%;0;19,1,FC%;0;
  380     *FX15,1
  390     K$=GET$
  400     IFK$="Z" PROCMOVECLUB(-10) ELSEIFK$="X"
PROCMOVECLUB(10)
  410     IFK$="F" FC%=FC%+1:IFFC%>7 FC%=0
  420     IFK$="B" BC%=BC%+1:IFBC%>7 BC%=0
  430     UNTILK$>="1"ANDK$<="9"
  440   PROCPCLUB
  450   SOUND0,-15,4,1
  460   BE=(ASCK$-ASC"0")*400:BA=(CA%+180)MOD360:P
ROCSETD
  470   REPEAT
  480     PROCMOVEBALL
  490     UNTILBE<0
  500   UNTILFNHOLED
  510 IFSH%=1 PRINTTAB(14,4)"A hole in one!":PROCW
HOOP
  520 SOUND1,-15,101,10
  530 SC%(PN%)=SC%(PN%)+SH%
  540 PROCDELAY
  550 ENDPROC
  560 REM******************************
  570 DEFPROCWHOOP
  580 FORI%=1TO9
  590   SOUND1,-15,I%*10,2:SOUND2,-15,I%*40,2:SOUN
D3,-15,I%*75,2
  600   NEXT
  610 ENDPROC
  620 REM******************************
  630 DEFPROCDELAY
  640 TIME=0
  650 REPEAT
  660   UNTILTIME>99
  670 ENDPROC
  680 REM******************************
  690 DEFFNHOLED
  700 =BX>HX%-10ANDBX<HX%+10ANDBY>HY%-10ANDBY<HY%+
10
  710 REM*******************************
  720 DEFPROCMOVEBALL
  730 MISS%=TRUE:BDDX=0:BDDY=0
  740 PROCPBALL:BX=BX+BDX:BY=BY+BDY
  750 IFPOINT(BX,BY) MISS%=FALSE ELSE IFPOINT(BX-B
DX,BY) MISS%=FALSE:BX=BX-BDX:BDDX=BDX ELSE IFPOINT
(BX,BY-BDY) MISS%=FALSE:BY=BY-BDY:BDDY=BDY
  760 IFMISS% THEN900
  770 IFFNHOLED BE=-1:GOTO910
  780 PROCPBALL
  790 GCOL0,1
  800 I%=0
  810 REPEAT
  820    X%=HCX%(I%):Y%=HCY%(I%):I%=I%+1
  830    IFX%<0 MOVE-X%,Y% ELSE DRAWX%,ABS(Y%)
  840    UNTILPOINT(BX,BY)
  850 GCOL3,1
  860 BX=BX+BDDX:BY=BY+BDDY:IFBE<54 BE=BE+54
  870 IFLWI%=I% THEN900
  880 WA=DEG(ATN((ABS(Y%)-HCY%(I%-2))/(X%-ABS(HCX%
(I%-2))+.00001))):LWI%=I%
  890 BX=BX-BDX:BY=BY-BDY:BE=.95*BE:BA=(2*WA-BA+36
0)MOD360:PROCSETD:SOUND1,-15,9,1
  900 PROCPBALL:BE=BE-6
  910 ENDPROC
  920 REM*******************************
  930 DEFPROCSETD
  940 BDX=4*COS(RAD(BA)):BDY=4*SIN(RAD(BA))
  950 ENDPROC
  960 REM*******************************
  970 DEFPROCPBALL
  980 MOVEBX,BY:DRAWBX,BY
  990 ENDPROC
 1000 REM*******************************
 1010 DEFPROCMOVECLUB(A%)
 1020 PROCPCLUB:CA%=CA%+A%:PROCPCLUB
 1030 ENDPROC
 1040 REM*******************************
 1050 DEFPROCPCLUB
 1060 MOVEBX,BY
 1070 PLOT0,12*COS(RAD(CA%)),12*SIN(RAD(CA%))
 1080 PLOT1,36*COS(RAD(CA%)),36*SIN(RAD(CA%))
 1090 ENDPROC
 1100 REM*******************************
 1110 DEFPROCPSCORES
 1120 PRINT''"After ";HN%" holes"
 1130 FORI%=1TONP%
 1140    PRINT'N$(I%),ABS(SC%(I%)-TPAR%);
 1150    IFSC%(I%)<TPAR% PRINT" under"; ELSE PRINT"
 over";
 1160    PRINT" par"
 1170    NEXT
 1180 PRINT'"Press any key to continue...";:*FX15,
1
 1190 K=GET
 1200 ENDPROC
 1210 REM********************************
 1220 DEFPROCREADHOLE
 1230 READPAR%:TPAR%=TPAR%+PAR%
 1240 HCI%=-1
 1250 REPEAT
 1260    READX%,Y%:HCI%=HCI%+1:HCX%(HCI%)=X%:HCY%(H
CI%)=Y%
 1270    UNTILY%<0
 1280 READHX%,HY%,TLX%,TUX%,TY%
 1290 ENDPROC
 1300 REM********************************
 1310 DEFPROCPHOLE
 1320 GCOL0,1
 1330 PRINTTAB(14,1)"Hole ";HN%" Par ";PAR%
 1340 FORI%=0TOHCI%
 1350    X%=HCX%(I%):Y%=HCY%(I%)
 1360    IFX%<0 MOVE-X%,Y% ELSE DRAWX%,ABS(Y%)
 1370    NEXT
 1380 FORI%=-6TO6
 1390    MOVEHX%+I%,HY%-6:DRAWHX%+I%,HY%+6
 1400    NEXT
 1410 BX=TLX%+RND(TUX%-TLX%):BY=TY%:PROCPBALL
 1420 ENDPROC
 1430 REM********************************
 1440 DATA2,-600,50,600,300,500,500,300,500,200,65
0,300,800,500,800,800,500,800,50,600,-50,280,600,6
25,775,75
 1450 DATA3,-600,50,150,450,750,850,950,750,550,45
0,1000,50,600,50,-250,420,250,480,-350,420,350,480
,-450,420,450,-480,850,770,650,950,75
 1460 DATA2,-400,50,600,250,400,500,400,800,800,80
0,600,500,800,250,800,50,400,-50,700,740,475,775,7
5
 1470 DATA3,-300,450,300,50,100,50,100,450,500,850
,700,650,700,200,500,50,300,200,-300,650,500,450,5
00,-200,480,690,125,275,75
 1480 DATA2,-600,600,600,50,200,50,200,450,600,850
,1000,450,1000,50,600,-50,700,150,250,550,70
 1490 DATA3,-700,480,600,480,600,400,800,400,800,6
00,300,600,300,50,100,50,100,600,300,800,800,800,1
000,600,1000,200,300,-200,640,430,125,275,75
 1500 DATA3,-450,50,450,300,200,500,600,800,1000,5
00,750,300,750,50,450,50,-450,550,450,500,600,400,
750,500,750,-550,600,450,475,725,75
 1510 DATA3,-50,50,50,250,150,350,150,750,750,750,
850,850,1050,850,1050,650,950,550,950,150,350,150,
250,50,50,50,-350,650,350,500,450,500,450,650,-750
,650,750,500,650,500,650,650,-750,250,750,400,650,
400,650,250
 1520 DATA-350,250,350,400,450,400,450,-250,950,80
0,75,225,75
 1530 DATA4,-400,450,400,50,200,50,200,850,1000,85
0,1000,50,400,50,-200,700,700,700,700,550,-700,450
,700,300,550,300,700,-450,280,780,225,375,75
 1540 DATA4,-400,450,400,50,200,50,200,850,1000,50
,400,50,-500,300,600,200,850,-200,800,120,225,375,
75
 1550 DATA5,-450,50,300,500,600,850,900,500,750,50
,450,50,-550,650,400,500,600,200,800,500,650,650,-
575,420,550,500,600,580,650,500,625,-420,600,500,4
75,725,75
 1560 DATA4,-700,150,600,400,720,650,280,650,400,4
00,250,50,100,400,250,750,750,750,900,400,800,150,
700,150,-700,400,800,400,-200,450,250,600,300,450,
-200,350,250,200,300,-350,250,400,725,775,175
 1570 DATA4,-200,50,200,550,600,850,1000,550,1000,
50,200,50,-450,600,450,450,750,450,750,600,-450,20
0,600,350,750,200,-300,400,450,300,-900,400,750,-3
00,600,250,475,725,475
 1580 DATA4,-600,300,600,650,750,750,900,650,900,5
0,200,50,200,300,500,500,500,650,350,500,200,650,4
00,850,800,850,1000,650,1000,50,900,50
 1590 DATA-350,650,450,700,500,650,-350,250,350,10
0,-450,250,450,100,-550,250,550,-100,750,650,925,9
75,75
 1600 DATA3,-300,50,300,750,400,850,800,850,900,75
0,900,50,300,50,-400,200,550,350,-800,200,650,350,
-500,400,500,500,600,600,700,500,700,400,-380,400,
380,750,430,800,-820,400,820,750,770,-800,600,750,
500,700,70
 1610 DATA4,-450,250,800,350,800,600,1000,600,1000
,50,200,50,200,600,450,850,550,850,800,600,-300,50
0,450,500,450,350,-700,500,550,500,550,350,-300,60
0,450,600,450,750,-700,600,550,600,550,-750,500,55
0,820,980,580
 1620 DATA3,-500,50,300,200,300,650,600,850,900,65
0,900,200,700,50,500,50,-450,450,600,200,750,450,-
450,650,550,450,-750,650,650,-450,600,480,520,680,
70
 1630 DATA4,-300,50,500,450,300,850,900,850,700,45
0,900,50,300,50,-450,650,600,550,750,650,-450,250,
600,350,750,-250,600,600,500,700,70
```

# Meteors

## Rules

The object is to fly your spaceship through the meteor storm, avoiding the meteors for as long as possible. Points are given for length of survival and meteors shot. A missile will destroy the first meteor it hits or disappear off the bottom of the screen. You have a maximum fire-rate of two missiles a second.

You can fly left or right, as you wish. If you fly off either edge of the screen you reappear at the opposite edge.

The longer you survive the further you move down the screen and the denser the meteor storm becomes!

## Display

You control the white spaceship which starts at the top-middle of the screen. The red meteors move up the screen and will destroy you if they hit any part of your spaceship.

Your running score is displayed briefly at the top of the screen each time you advance one line down the screen.

When the game is over, the high score and your current score are displayed.

## Operation

To control your spaceship, use the following keys:
'Z' – left
'X' – right
<RETURN> – fire missile

## Program

The program controls the required spaceship movements and generates the random meteor storm.

| *Section/Variables* | *Function* |
|---|---|
| Main routine | Initialize data, main game loop, game over |
| HSC% | Current high score |
| D% | Difficulty factor |

| | |
|---|---|
| SC% | Current score |
| X% | Spaceship X-coordinate |
| Y% | Spaceship Y-coordinate |
| NX% | New spaceship X-coordinate |
| K$ | Input key |
| T% | Time-delay controller |
| FN GO | Check if game over (ie spaceship hit by meteor) |
| GO% | Game-over flag |
| PROC FIRE | Fire laser |
| SX% | Screen pixel X-coordinate |
| SY% | Screen pixel Y-coordinate |
| ISY% | Initial screen pixel Y-coordinate |
| FN SCRN | Check particular screen position for meteor |
| X% | Screen X-coordinate to test |
| Y% | Screen Y-coordinate to test |
| FN NO | Return 'yes' or 'no' answer |
| K$ | Input key |
| PROC PSCORE | Print current scores |

## Suggestions

Enhance the game so that the meteors come from all angles and you can fly up and down as well, and aim your missile cannon.

Also add different-colour meteors (eg guns) which score additional points.

When the spaceship gets very low on the screen, the program could restart on further meteor storms which gradually get faster.

Make the spaceship explode more dramatically when it is struck by a meteor!

(My best score: 10256)

## The Listing

```
   10 *FX4,1
   20 VDU23,224,&18,&30,&60,&FF,&FF,&60,&30,&18,23
,225,0,&18,&3C,&FF,&FF,&3C,&18,0,23,226,&18,&C,6,&
FF,&FF,6,&C,&18
   30 HSC%=0:*FX11,10
   40 REPEAT
   50    MODE1:VDU23;8202;0;0;0;
   60    D%=0:SC%=0:X%=18:Y%=4
   70    REPEAT
   80       VDU17,3,31,X%,Y%,224,225,226
   90       D%=D%+1:NX%=X%
  100       K$=INKEY$(0):*FX15,1
  110        IFK$="Z"ORINKEY(-98)NX%=NX%-1:IFNX%<0 NX
```

```
%=36
  120        IFK$="X"ORINKEY(-67)NX%=NX%+1:IFNX%>36 N
X%=0
  130        IFK$=CHR$(13)ORINKEY(-74)IFTIME>50 PROCF
IRE
  140        VDU17,1,31,0,31
  150        T%=TIME+5
  160        FORI%=0TOD%DIV50
  170          IFRND(5)<2PRINTTAB(RND(39)-1,31)"*";:S
C%=SC%+Y%
  180          NEXT
  190        SOUND&12,-6,RND(I%*9),1
  200        REPEAT
  210          UNTILTIME>T%
  220        PRINTTAB(X%,Y%)"    "TAB(0,31)
  230        X%=NX%:IFD%MOD50=0 Y%=Y%+1:PROCPSCORE:IF
Y%>30Y%=30
  240        UNTILFNGO
  250      SOUND0,-15,6,9
  260      COLOUR3
  270      PRINTTAB(X%,Y%),"£££"
  280      IFSC%>HSC% HSC%=SC%
  290      PROCPSCORE
  300      PRINTTAB(14,28)"Another Game?":*FX15,1
  310      UNTILFNNO
  320 MODE7
  330 END
  340 REM********************************
  350 DEFFNGO
  360 GO%=FALSE
  370 FORI%=X%TOX%+2
  380   IFFNSCRN(I%,Y%)GO%=TRUE
  390   NEXT
  400 =GO%
  410 REM********************************
  420 DEFPROCFIRE
  430 SOUND0,-15,4,2:SOUND1,-15,Y%*6,3
  440 SX%=32*X%+48:SY%=1008-32*Y%:ISY%=SY%
  450 REPEAT
  460   SY%=SY%-32
  470   UNTILPOINT(SX%,SY%)
  480 IFSY%<0SY%=SY%+32 ELSESC%=SC%+Y%:SOUND3,-15,
RND(255),2
  490 GCOL0,2:MOVESX%,ISY%:DRAWSX%,SY%
  500 PRINTTAB(X%+1,Y%);
  510 FORI%=1TO(ISY%-SY%)DIV32
  520   PRINTCHR$(10)" "CHR$(8);
  530   NEXT
  540 SOUND0,-15,6,1
  550 TIME=0
  560 ENDPROC
  570 REM*****************************
  580 DEFFNSCRN(X%,Y%)
  590 =POINT(32*X%+16,1008-32*Y%)
  600 REM*****************************
  610 DEFFNNO
  620 REPEAT
  630   K$=GET$
  640   UNTILK$="Y"ORK$="N"
  650 =K$="N"
  660 REM*****************************
  670 DEFPROCPSCORE
  680 VDU17,2
  690 PRINTTAB(2,3)"High Score:";HSC%TAB(26,3)"Sco
re:";SC%
  700 ENDPROC
```

# Rollers

## Rules

You control a paint roller which can move up, down, left or right within the playing arena. The computer also controls a number of paint rollers.

The object is to survive as long as possible without crashing into any of your own tracks, the computer's tracks or the edges of the arena, and to trap the computer's rollers to force them to crash.

If you succeed in crashing all the computer's rollers then you start the next screen with the computer having one more roller than last time (up to a maximum of ten rollers).

You may only turn through 90 degrees at a time. A 180-degree turn would cause an immediate crash anyway.

## Display

The top line of the display shows the current high score and your score for this game.

The arena is bordered by a thick white line.

Your roller is yellow and starts at the bottom of the screen moving up.

The computer's rollers are red and start at the top of the screen moving down.

## Operation

To change the direction of your roller, use the following keys:
'Z' – left
'X' – right
':' – up
'/' – down

## Program

The program controls the requested changes in direction of the user's roller and controls the directions of its own rollers, depending on various factors.

| *Section/Variables* | *Function* |
|---|---|
| Main routine | Initialize data, main game loop, game over |
| HSC% | High score so far |

| | |
|---|---|
| S% | Step-size for roller movement |
| CX% | Computer's rollers, X-coordinates |
| CY% | Computer's rollers, Y-coordinates |
| CDX% | Computer's rollers, X-axis direction |
| CDY% | Computer's rollers, Y-axis direction |
| SC% | Current score |
| C% | Number of computer's rollers |
| HX% | Human's roller, X-coordinate |
| HY% | Human's roller, Y-coordinate |
| HDX% | Human's roller, X-axis direction |
| HDY% | Human's roller, Y-axis direction |
| CA% | Number of computer's rollers still alive |
| GO% | Game-over flag |
| FN BLKD | Check if particular direction is blocked |
| X% | Current position X-coordinate |
| Y% | Current position Y-coordinate |
| DX% | Change in X-axis position |
| DY% | Change in Y-axis position |
| PROC SMAN | Steer player's roller |
| K$ | Input key |
| PROC AMAN | Adjust player's roller direction |
| OHDX% | Old human's roller, change in X-axis position |
| OHDY% | Old human's roller, change in Y-axis position |
| PROC MMAN | Move player's roller |
| PROC SCOM | Steer computer's rollers |
| TDX% | Temporary storage of computer's change in X-axis position |
| TDY% | Temporary storage of computer's change in Y-axis position |
| S | Score of best direction so far |
| TS | Score of current direction being examined |
| PROC MCOM | Move computer's rollers |

## Suggestions

Assembly language could be used to stop the slowing-down which takes place when the program has to handle a large number of rollers.

Also, the addition of the ability to change speed from fast to slow may enhance the strategy and tactics required in the game.
(My best score: 33416)

## The Listing

```
   10 *FX4,1
   20 VDU23,240,&FF,&FF,&FF,&FF,&FF,&FF,&FF,&FF
   30 HSC%=0:S%=12
   40 DIMCX%(9),CY%(9),CDX%(9),CDY%(9),CD%(9)
   50 REPEAT
   60   SC%=0:C%=0
   70   REPEAT
   80     MODE1:VDU23;8202;0;0;0;
   90     FORI%=0TO38:PRINTTAB(I%,3)CHR$(240)TAB(I
%,31)CHR$(240);
  100       NEXT
  110     FORI%=4TO30:PRINTTAB(0,I%)CHR$(240)TAB(3
8,I%)CHR$(240);
  120       NEXT
  130     PRINTTAB(1,1)"High Score:";HSC%TAB(28,1)
"Score:";SC%
  140     HX%=1280/2-16:HY%=108:HDX%=0:HDY%=S%
  150     CA%=C%
  160     FORI%=0TOC%:CX%(I%)=(I%+1)*(1280/(C%+2))
-16:CY%(I%)=850:CDX%(I%)=0:CDY%(I%)=-S%:CD%(I%)=FA
LSE
  170       NEXT
  180     *FX15,1
  190     REPEAT
  200       VDU5
  210       FORI%=0TOC%
  220         IFNOTCD%(I%) CX%=CX%(I%):CY%=CY%(I%)
:CDX%=CDX%(I%):CDY%=CDY%(I%):PROCSCOM:PROCMCOM:CX%
(I%)=CX%:CY%(I%)=CY%:CDX%(I%)=CDX%:CDY%(I%)=CDY%
  230         NEXT
  240       PROCSMAN
  250       PROCMMAN
  260       VDU4:PRINTTAB(34,1);SC%
  270       UNTILGO%ORCA%<0
  280     IFCA%<0 C%=C%+1:IFC%>9 C%=9
  290     IFGO% SOUND0,-15,4,9
  300     TIME=0
  310     REPEAT
  320       UNTILTIME>99
  330     UNTILGO%
  340   IFSC%>HSC% HSC%=SC%:PRINTTAB(12,1);HSC%
  350   PRINTTAB(12,18)"Another game?";:*FX15,1
  360   UNTILGET$="N"
  370 MODE7
  380 END
  390 REM*******************************
  400 DEFFNBLKD(X%,Y%,DX%,DY%)
  410 IFDY%THEN460
  420 X%=X%+DX%
  430 IFDX%<0IFPOINT(X%,Y%)ORPOINT(X%,Y%-31):=TRUE
  440 IFDX%>0IFPOINT(X%+31,Y%)ORPOINT(X%+31,Y%-31)
:=TRUE
  450 =FALSE
  460 Y%=Y%+DY%
  470 IFDY%<0IFPOINT(X%,Y%-31)ORPOINT(X%+31,Y%-31)
:=TRUE
  480 IFDY%>0IFPOINT(X%,Y%)ORPOINT(X%+31,Y%):=TRUE
  490 =FALSE
  500 REM********************************
  510 DEFPROCSMAN
  520 K$=INKEY$(0):IFK$<>""PROCAMAN
  530 SOUND1,-2,20*HDX%+12*HDY%,1
  540 ENDPROC
  550 REM********************************
  560 DEFPROCAMAN
  570 *FX15,1
  580 OHDX%=HDX%:OHDY%=HDY%
  590 IFK$="Z"IFOHDX%<1 HDX%=-S%:HDY%=0
  600 IFK$="X"IFOHDX%>-1 HDX%=S%:HDY%=0
  610 IFK$=":"IFOHDY%>-1 HDX%=0:HDY%=S%
  620 IFK$="/"IFOHDY%<1 HDX%=0:HDY%=-S%
  630 ENDPROC
  640 REM********************************
  650 DEFPROCMMAN
  660 GO%=FNBLKD(HX%,HY%,HDX%,HDY%)
  670 HX%=HX%+HDX%:HY%=HY%+HDY%:SC%=SC%+1+C%
  680 MOVEHX%,HY%:GCOL0,2:PRINTCHR$(240);
  690 ENDPROC
  700 REM********************************
  710 DEFPROCSCOM
  720 TDX%=CDX%:TDY%=CDY%:S=0
  730 IFCDX%<>S% TS=-(HX%<CX%)-(CDX%=-S%)+1.3*RND(
1):IFTS>S IFNOTFNBLKD(CX%,CY%,-S%,0) S=TS:TDX%=-S%
:TDY%=0
  740 IFCDX%<>-S% TS=-(HX%>CX%)-(CDX%=S%)+1.3*RND(
1):IFTS>S IFNOTFNBLKD(CX%,CY%,S%,0) S=TS:TDX%=S%:T
DY%=0
  750 IFCDY%<>S% TS=-(HY%<CY%)-(CDY%=-S%)+1.3*RND(
1):IFTS>S IFNOTFNBLKD(CX%,CY%,0,-S%) S=TS:TDX%=0:T
DY%=-S%
  760 IFCDY%<>-S% TS=-(HY%>CY%)-(CDY%=S%)+1.3*RND(
1):IFTS>S IFNOTFNBLKD(CX%,CY%,0,S%) S=TS:TDX%=0:TD
Y%=S%
  770 CDX%=TDX%:CDY%=TDY%
  780 ENDPROC
  790 REM********************************
  800 DEFPROCMCOM
  810 CD%(I%)=FNBLKD(CX%,CY%,CDX%,CDY%):IFCD%(I%)
CA%=CA%-1:SC%=SC%+100:SOUND0,-15,6,5
  820 CX%=CX%+CDX%:CY%=CY%+CDY%
  830 MOVECX%,CY%:GCOL0,1:PRINTCHR$(240);
  840 ENDPROC
```

# Slalom

## Rules

You must ski down a mountain through the gates of the slalom course. Points are awarded for each gate you pass through. The thinner the gate and the further down the course you are, the more points you get.

You must, however, avoid crashing into the gates, or the sticks of dynamite left on the course by your less-than-sporting opponents or the stupid spectators who stand on, or even walk across, the slope.

The longer you survive the further down the slope you will move.

When you get to the bottom of the current slope you are started again on a steeper slope (ie everything goes by quicker!).

## Display

You are shown on skis, gradually moving down the slope. The gates are shown with two flags close to each other. The sticks of dynamite are in red. The spectators are matchstick men.

The high score and current score are flashed up every time you move down the screen.

## Operation

You control your man with the keys:
'Z' – left
'X' – right

## Program

The program controls the required man-movements and generates the various obstacles randomly.

| *Section/Variables* | *Function* |
|---|---|
| Main routine | Initialize data, setup level, main game loop, game over |
| HSC% | High-score |
| SC% | Current score |
| SPEED% | Current hill speed factor |
| GLX% | Gate lower X-coordinate |
| GUX% | Gate upper X-coordinate |

| | |
|---|---|
| GY% | Gate Y-coordinate |
| BX% | Bomb X-coordinate |
| BY% | Bomb Y-coordinate |
| LX% | Lunatic spectator X-coordinate |
| LY% | Lunatic spectator Y-coordinate |
| LDX% | Lunatic spectator X-axis movement direction |
| X% | Man X-coordinate |
| Y% | Man Y-coordinate |
| DX% | Man X-axis movement direction |
| D% | Downhill-position difficulty counter |
| K$ | Input key |
| FN NO | Return 'yes' or 'no' answer |
| K$ | Input key |
| FN GO | Checks if game over (ie you've crashed into something) |
| PROC WHOOP | Play fanfare at end of current hill |
| PROC GATE | Generate next randomly positioned gate |
| PROC THROUGHGATE | Check if player skied through the gate |
| V% | Gate value |
| PROC BOMB | Generate next randomly-placed stick of dynamite |
| PROC LOONY | Generate next randomly-placed lunatic spectator |
| PROC PSCORES | Print current scores |

## Suggestions

A 'jump' facility might be added to allow the player to jump over obstacles, but lose steering capability while jumping.

Also, more hazards could be added; eg snipers trying to shoot skier from side of hill, snow mounds, potholes, avalanches.
(My best score: 3366)

## The Listing

```
   10 *FX4,1
   20 VDU23,224,&1C,&1C,&1C,&08,&3E,&5E,&9E,&1C,23
,225,&1C,&17,&1C,&34,&C7,&0C,&30,&C0
   30 VDU23,226,&1C,&1C,&1C,&08,&3E,&5D,&5D,&1C,23
,227,&55,&55,&55,&63,&63,&41,&41,&41
   40 VDU23,228,&38,&38,&38,&10,&7C,&7A,&79,&38,23
,229,&38,&E8,&38,&2C,&E3,&30,&0C,&03
   50 VDU23,230,&60,&70,&78,&78,&70,&60,&40,&40,23
,231,&20,&20,&40,&80,&BF,&7F,&3F,0,23,232,&1C,&1C,
&08,&7F,&1C,&14,&22,&22
   60 HSC%=0
   70 REPEAT
```

```
  80    SC%=0:SPEED%=6
  90    REPEAT
 100      MODE1:VDU23;8202;0;0;0;19,0,7;0;19,2,2;0
;19,3,4;0;
 110      GY%=0:BY%=0:LY%=0:LX%=0:LDX%=0
 120      X%=20:Y%=6:DX%=0:D%=0
 130      SPEED%=SPEED%-1
 140      REPEAT
 150        COLOUR3
 160        PRINTTAB(X%,Y%)CHR$(226+DX%*2)CHR$(8)C
HR$(10)CHR$(227+DX%*2);
 170        IFLY%>0COLOUR1:PRINTTAB(LX%,LY%)CHR$(2
32);:SOUND&12,-5,5*LX%,1
 180        DX%=0:D%=D%+1:SOUND&10,-2,4+D%MOD3,1
 190        K$=INKEY$(0):*FX15,1
 200        IFK$="Z"ORINKEY(-98)IFX%>0DX%=-1
 210        IFK$="X"ORINKEY(-67)IFX%<39DX%=1
 220        TIME=0
 230        IFRND(10)=1 PROCGATE
 240        IFRND(12)=1 PROCBOMB
 250        IFRND(18)=1 PROCLOONY
 260        REPEAT
 270          UNTILTIME>SPEED%
 280        IFLY%>0PRINTTAB(LX%,LY%)" ";
 290        PRINTTAB(X%,Y%)" "CHR$(8)CHR$(10)" "TA
B(0,31)
 300        X%=X%+DX%
 310        IFD%MOD50=0Y%=Y%+1:PROCTHROUGHGATE:PRO
CPSCORES
 320        GY%=GY%-1:BY%=BY%-1:LY%=LY%-1
 330        LX%=LX%+LDX%
 340        IFLDX%<=0IFLX%<1OR(RND(30)=1ANDLX%<38)
 LDX%=1
 350        IFLDX%>=0IFLX%>38OR(RND(30)=1ANDLX%>1)
 LDX%=-1
 360        PROCTHROUGHGATE
 370        UNTILY%>23ORFNGO
 380      IFY%>23 PROCWHOOP
 390      UNTILFNGO
 400    SOUND0,-15,6,9
 410    PRINTTAB(X%,Y%)"£"TAB(X%,Y%+1)"£"
 420    IFSC%>HSC% HSC%=SC%
 430    PROCPSCORES
 440    PRINTTAB(12,30)"Another game?":*FX15,1
 450    UNTILFNNO
 460  MODE7
 470  END
 480  REM********************************
 490  DEFFNNO
 500  REPEAT
 510    K$=GET$
```

```
  520    UNTILK$="Y"ORK$="N"
  530 =K$="N"
  540 REM********************************
  550 DEFFNGO
  560 IFY%=GY%ORY%=GY%-1THENIFX%=GLX%ORX%=GUX% :=T
RUE
  570 IFY%=BY%ORY%=BY%-1THENIFX%=BX% :=TRUE
  580 IFY%=LY%ORY%=LY%-1THENIFX%=LX% :=TRUE
  590 =FALSE
  600 REM********************************
  610 DEFPROCWHOOP
  620 FORI%=1TO9
  630    SOUND1,-15,I%*9,1:SOUND2,-15,255-I%*9,1
  640    NEXT
  650 TIME=0
  660 REPEAT
  670    UNTILTIME>99
  680 ENDPROC
  690 REM********************************
  700 DEFPROCGATE
  710 IFGY%>=Y% THEN750
  720 GLX%=RND(31):GUX%=GLX%+8-RND(Y%DIV6+1):GY%=3
1
  730 COLOUR2
  740 PRINTTAB(GLX%,GY%)CHR$(230)TAB(GUX%,GY%)CHR$
(230);
  750 ENDPROC
  760 REM********************************
  770 DEFPROCTHROUGHGATE
  780 IFGY%=Y%THENIFGLX%<X%ANDGUX%>X% V%=Y%*(9-GUX
%+GLX%):SC%=SC%+V%:PRINTTAB(GLX%,GY%-2);V%:SOUND3,
-12,V%,1
  790 ENDPROC
  800 REM********************************
  810 DEFPROCBOMB
  820 IFBY%>=Y%ORGY%=31THEN860
  830 BX%=RND(38):BY%=31
  840 COLOUR1
  850 PRINTTAB(BX%,BY%)CHR$(231);
  860 ENDPROC
  870 REM********************************
  880 DEFPROCLOONY
  890 IFLY%>=Y%ORBY%=31ORGY%=31THEN910
  900 LX%=RND(38):LY%=31:LDX%=RND(3)-2
  910 ENDPROC
  920 REM********************************
  930 DEFPROCPSCORES
  940 COLOUR3
  950 PRINTTAB(2,5)"High Score:";HSC%TAB(26,5)"Sco
re:";SC%
  960 ENDPROC
```

# Rebel

## Rules

The idea is to shoot down as many 'Empire' fighters as fast as possible by positioning them at the centre of your sights and shooting them with your laser cannon. The fighters gradually get faster and if you let them escape from your sights they will shoot you down from around your frontal shields.

## Display

The display shows the view through your cross-wire aiming sights.

The Empire fighters are shown in red, flying around in front of you.

Your lasers fire from the bottom corners of the screen to the centre of the crosswires.

## Operation

To move the sights, use the keys:
'Z' – left
'X' – right
':' – up
'/' – down
To fire the laser, press the <RETURN> key.

## Program

The program controls the fleeing fighter and the required sight movements.

| *Section/Variables* | *Function* |
|---|---|
| Main routine | Initialize data, main game loop, game over |
| HSC% | High score |
| S% | Speed of fleeing fighter |
| SC% | Current score |
| EX% | Enemy-fighter X-coordinate |
| EY% | Enemy-fighter Y-coordinate |
| DX% | Enemy-fighter X-axis direction |
| DY% | Enemy-fighter Y-axis direction |
| ED% | Enemy-dead flag |

| | |
|---|---|
| MD% | Man-dead flag |
| NEX% | New-enemy X-coordinate |
| NEY% | New-enemy Y-coordinate |
| FN NO | Return 'yes' or 'no' answer |
| K$ | Input key |
| PROC MOVEENEMY | Move enemy-fighter position |
| X% | Change in enemy X-axis position |
| Y% | Change in enemy Y-axis position |
| PROC PENEMY | Print enemy-fighter |
| PROC FIRE | Fire laser |
| PROC PSHOT | Print laser line of fire |
| PROC EXPLODE | Draw exploded fighter |
| PROC PSCORE | Print current scores |
| PROC PSIGHTS | Print the gun-sights |
| X% | Apex X-coordinate |
| Y% | Apex Y-coordinate |

## Suggestions

The game could be enhanced by having more than one enemy fighter at a time, with different types of fighter having different points values.
(My highest score: 1220)

## The Listing

```
   10 *FX4,1
   20 HSC%=0
   30 REPEAT
   40    S%=0:SC%=0:TIME=0
   50    REPEAT S%=S%+1
   60       MODE5:VDU23;8202;0;0;0;
   70       PROCPSIGHTS
   80       EX%=400+RND(400):EY%=300+RND(400):DX%=1:
DY%=1
   90       ED%=FALSE:MD%=FALSE
  100       PROCPSCORE
  110       REPEAT
  120          PROCPENEMY:NEX%=EX%:NEY%=EY%
  130          IFRND(1)>.90-.04*((DX%>0)=(EX%>600)) D
X%=-DX%
  140          IFRND(1)>.90-.04*((DY%>0)=(EY%>500)) D
Y%=-DY%
  150          PROCMOVEENEMY(S%*DX%,S%*DY%)
  160          IFINKEY(-98)PROCMOVEENEMY(-24,0)
  170          IFINKEY(-67)PROCMOVEENEMY(24,0)
  180          IFINKEY(-73)PROCMOVEENEMY(0,24)
  190          IFINKEY(-105)PROCMOVEENEMY(0,-24)
  200          IFINKEY(-74)IFTIME>99PROCFIRE
  210          IFTIME>1500 S%=S%+1:TIME=100
  220          SOUND&13,-RND(5),RND(9),1
  230          PROCPENEMY:EX%=NEX%:EY%=NEY%
  240          UNTILED%ORMD%
  250        IFED% SC%=SC%+10*S%
  260        UNTILMD%
  270      IFSC%>HSC% HSC%=SC%:PROCPSCORE
  280      FORI%=1TO7
  290        VDU19,0,I%;0;
  300        SOUND0,-15,RND(3)+3,1
  310        FORJ%=0TO99
  320          NEXT
  330        NEXT
  340      VDU20
  350      PRINTTAB(3,15)"Another game?":*FX15,1
  360      UNTILFNNO
  370 MODE7
  380 END
  390 REM********************************
  400 DEFFNNO
  410 REPEAT
  420     K$=GET$
  430     UNTILK$="Y"ORK$="N"
  440 =K$="N"
  450 REM********************************
  460 DEFPROCMOVEENEMY(X%,Y%)
  470 NEX%=NEX%+X%:NEY%=NEY%+Y%:MD%=NEX%<36ORNEX%>
1100ORNEY%<100ORNEY%>932
  480 ENDPROC
  490 REM********************************
  500 DEFPROCPENEMY
  510 VDU18,3,1,25,4,EX%;EY%;25,1,0;-32;25,0,64;0;
25,1,0;32;25,0,-8;-16;25,1,-48;0;25,0,16;4;25,1,16
;0;25,0,0;-8;25,1,-16;0;
  520 ENDPROC
  530 REM********************************
  540 DEFPROCFIRE
  550 GCOL3,2
  560 SOUND0,-15,4,2
  570 PROCPSHOT
  580 IFEX%>535ANDEX%<601ANDEY%>499ANDEY%<533 ED%=
TRUE:SOUND0,-15,6,9:PROCEXPLODE
  590 PROCPSHOT
  600 TIME=0:S%=S%+1
  610 ENDPROC
  620 REM********************************
  630 DEFPROCPSHOT
  640 Y%=0
  650 FORX%=100TO599STEP50
  660     MOVEX%,Y%:DRAWX%+50,Y%+50:MOVE1200-X%,Y%:D
```

```
RAW1200-X%-50,Y%+50
  670    Y%=Y%+50
  680    NEXT
  690 ENDPROC
  700 REM******************************
  710 DEFPROCEXPLODE
  720 FOREY%=NEY%-12TONEY%+12STEP4
  730    PROCPENEMY
  740    NEXT
  750 ENDPROC
  760 REM******************************
  770 DEFPROCPSCORE
  780 PRINTTAB(5,1)"Score=";SC%TAB(3,31)"High scor
e=";HSC%;
  790 ENDPROC
  800 REM******************************
  810 DEFPROCPSIGHTS
  820 GCOL0,3
  830 RESTORE
  840 REPEAT
  850    READX%,Y%
  860    IFX%<0 MOVE-X%,Y% ELSE DRAWX%,ABS(Y%)
  870    UNTILY%<0
  880 ENDPROC
  890 REM******************************
  900 DATA-100,200,100,800,-100,500,400,500,-1100,
200,1100,800,-1100,500,800,500,-400,100,800,100,-6
00,100,600,300,-400,900,800,900,-600,900,600,-700
```

# Balrog

## Rules

The BALROG is a monstrous beastie who must be trapped before he eats all of your men!

The catch is that he is only visible on the first move and thereafter only when he eats one of your men!

The BALROG can normally move only up, down, left or right. It can however 'eat' diagonally if you leave one of your men within reach.

The BALROG is trapped when it cannot move: ie it has no men, it can eat only with a diagonal move, and all its lateral moves are blocked by your men or the sides of the arena.

Your men can move diagonally as well as laterally. You get two moves to the BALROG's one. You and the BALROG can only move one square at a time.

If any of your men have been eaten, then every eighth move (provided you've survived) you will be given another man to join in the search.

## Display

The arena is shown at the top middle of the display. Your men are indicated by yellow matchstick men. The BALROG is indicated by a red skull-and-crossbones. Empty squares are indicated by white dots.

The arena has a white border and is numbered along two edges.

Your moves are printed below the board.

## Operation

First you are asked for the height and width of the arena. Press a number key from '4' to '9'. The larger the arena you select the more men you will be given to start the game with.

To move a man, position the cursor (the '<' sign) next to the square from which you wish to move your man and press <RETURN>. Then move the cursor to one of the empty squares around the man and press <RETURN> again.

If you accidentally enter the wrong from-square, you can clear it by moving the cursor more than one square from the man and pressing <RETURN>. The illegal move will then be cleared.

To move the cursor, use the keys:

'Z' – left

'X' – right
':' – up
'/' – down

## Program

The program simply controls the required men movements and moves the BALROG according to its desire to eat/centralize etc.

| *Section/Variables* | *Function* |
|---|---|
| Main routine | Initialize data, setup arena size, main game loop, game over |
| B% | Current board |
| MX% | Men X-coordinates |
| MY% | Men Y-coordinates |
| BMDX% | BALROG's moves – change in X axis |
| BMDY% | BALROG's moves – change in Y axis |
| BCDX% | BALROG's captures – change in X axis |
| BCDY% | BALROG's captures – change in Y axis |
| BX% | BALROG X-coordinate |
| BY% | BALROG Y-coordinate |
| H% | Arena height |
| W% | Arena width |
| NM% | Number of men |
| BT% | BALROG-trapped flag |
| MA% | Men-alive counter |
| M% | Move number |
| DB% | Display BALROG flag |
| CX% | Cursor X value |
| CY% | Cursor Y value |
| FN DIG | Get single-digit arena dimension |
| K$ | Input key |
| PROC RNDSQ | Generate random unoccupied-square X,Y values |
| X% | Square X-coordinate |
| Y% | Square Y-coordinate |
| PROC GETSQ | Handle cursor movement till square selected |
| PROC MMAN | Move player's man |
| OQY% | Old query Y-coordinate |
| QX% | Query X-coordinate |
| QY% | Query Y-coordinate |
| FX% | From-square X-coordinate |
| FY% | From-square Y-coordinate |
| MI% | Index of man to be moved |
| PROC ADDMAN | Handle additional men |
| PROC PUTMAN | Insert new man into data table |

| | |
|---|---|
| FN EVAL | Evaluate possible BALROG move |
| PROC MBALROG | Move the BALROG |
| S% | Score of best move so far for the BALROG |
| TS% | Score of the current move being examined |
| PROC PBOARD | Print complete arena |

## Suggestions

It may be exciting to also make a ‘real-time’ version of the game, where the BALROG doesn’t wait for you to move, but moves every 3 seconds, say, even if you haven’t yet moved!

Also you could gradually increase the difficulty of the game by having more than one BALROG wandering around, with some sort of scoring system.

## The Listing

```
   10 *FX4,1
   20 DIMB%(10,10),MX%(9),MY%(9),BMDX%(3),BMDY%(3)
,BCDX%(3),BCDY%(3)
   30 VDU23,224,&38,&38,&10,&7E,&10,&38,&28,&28,23
,225,&38,&7C,&54,&7C,&6C,&BA,&7C,&82
   40 FORI%=0TO3
   50   READBMDX%(I%),BMDY%(I%),BCDX%(I%),BCDY%(I%
)
   60   NEXT
   70 DATA0,1,-1,1,1,0,1,1,0,-1,1,-1,-1,0,-1,-1
   80 REPEAT
   90   MODE1
  100   PRINT'"Enter arena height(4-9)?";:H%=FNDIG
:PRINT;H%
  110   PRINT'"Enter arena width(4-9)?";:W%=FNDIG:
PRINT;W%
  120   VDU23;8202;0;0;0;
  130   FORY%=0TO10
  140     FORX%=0TO10
  150       IFY%<1ORY%>H%ORX%<1ORX%>W% B%(Y%,X%)=-
1 ELSE B%(Y%,X%)=0
  160       NEXT
  170     NEXT
  180   PROCRNDSQ:BX%=X%:BY%=Y%:B%(Y%,X%)=2
  190   NM%=W%*H%/15+1
  200   FORI%=0TONM%
  210     PROCRNDSQ:MX%(I%)=X%:MY%(I%)=Y%:B%(Y%,X%
)=1
  220     NEXT
  230   BT%=FALSE:MA%=NM%:M%=1:DB%=TRUE:CX%=1:CY%=
1
  240   CLS
```

```
 250    REPEAT
 260      PROCPBOARD
 270      PROCMMAN
 280      IFM%MOD8=0 PROCADDMAN
 290      IFM%MOD2=0 PROCMBALROG
 300      M%=M%+1
 310      UNTILBT%ORMA%<0
 320    IFBT% PRINTTAB(9,29)"The BALROG is trapped
!!!" ELSE PRINTTAB(5,29)"All your men have been ea
ten!!!"
 330    DB%=TRUE
 340    PROCPBOARD
 350    PRINTTAB(14,31)"Another game?";:*FX15,1
 360    UNTILGET$="N"
 370 MODE7
 380 END
 390 REM*******************************
 400 DEFFNDIG
 410 REPEAT
 420    K$=GET$
 430    UNTILK$>="4"ANDK$<="9"
 440 =ASCK$-ASC"0"
 450 REM*******************************
 460 DEFPROCRNDSQ
 470 REPEAT
 480    X%=RND(W%):Y%=RND(H%)
 490    UNTILB%(Y%,X%)=0
 500 ENDPROC
 510 REM*******************************
 520 DEFPROCGETSQ
 530 REPEAT
 540    PRINTTAB(19-W%+2*CX%,2*CY%-1)"<"
 550    *FX15,1
 560    K$=GET$
 570    PRINTTAB(19-W%+2*CX%,2*CY%-1)" "
 580    IFK$="Z" CX%=CX%-1:IFCX%<1 CX%=W%
 590    IFK$="X" CX%=CX%+1:IFCX%>W% CX%=1
 600    IFK$=":" CY%=CY%-1:IFCY%<1 CY%=H%
 610    IFK$="/" CY%=CY%+1:IFCY%>H% CY%=1
 620    UNTILK$=CHR$(13)
 630 ENDPROC
 640 REM*******************************
 650 DEFPROCMMAN
 660 OQY%=VPOS+2+2*((M%-1)MOD2)
 670 REPEAT
 680    PRINTTAB(0,OQY%)CHR$(7)"Move ";M%") From?"
;:QX%=POS:QY%=VPOS:PRINT"                    "
 690    REPEAT
 700      PROCGETSQ
 710      UNTILB%(CY%,CX%)=1
 720    FX%=CX%:FY%=CY%
```

```
  730    PRINTTAB(QX%,QY%);CY%",";CX%;" To?";:QX%=P
OS:QY%=VPOS
  740    REPEAT
  750      PROCGETSQ
  760      UNTILB%(CY%,CX%)<>1
  770    PRINTTAB(QX%,QY%);CY%",";CX%
  780    UNTILABS(FX%-CX%)<2ANDABS(FY%-CY%)<2
  790 FORI%=0TONM%
  800    IFFX%=MX%(I%)ANDFY%=MY%(I%) MI%=I%
  810    NEXT
  820 MX%(MI%)=CX%:MY%(MI%)=CY%:B%(FY%,FX%)=0
  830 IFCX%=BX%ANDCY%=BY% PRINT"Straight into the
BALROGS mouth":MX%(MI%)=0:MA%=MA%-1:DB%=TRUE:SOUND
0,-15,6,9 ELSE B%(CY%,CX%)=1
  840 PROCPBOARD
  850 ENDPROC
  860 REM*********************************
  870 DEFPROCADDMAN
  880 FORI%=0TONM%
  890    IFMX%(I%)=0 I%=99
  900    NEXT
  910 IFI%<99 THEN970
  920 PRINTTAB(0,VPOS+8)"New man to join chase at?
":QY%=VPOS
  930 REPEAT
  940    PROCGETSQ
  950    UNTILB%(CY%,CX%)<>1
  960 IFB%(CY%,CX%)=2 PRINTTAB(0,QY%)"Straight int
o the BALROGS mouth":DB%=TRUE:SOUND0,-15,6,9 ELSE
PROCPUTMAN
  970 ENDPROC
  980 REM*********************************
  990 DEFPROCPUTMAN
 1000 B%(CY%,CX%)=1:MA%=MA%+1
 1010 FORI%=0TONM%
 1020    IFMX%(I%)=0 MX%(I%)=CX%:MY%(I%)=CY%:I%=NM%
 1030    NEXT
 1040 ENDPROC
 1050 REM*********************************
 1060 DEFFNEVAL
 1070 =-ABS(W%/2+.5-X%)-ABS(H%/2+.5-Y%)+RND(7)
 1080 REM*********************************
 1090 DEFPROCMBALROG
 1100 S%=-99
 1110 PRINTTAB(0,30)"The BALROG is moving!"
 1120 FORI%=0TO3
 1130    SOUND1,-15,RND(250),3
 1140    X%=BX%+BMDX%(I%):Y%=BY%+BMDY%(I%)
 1150    IFB%(Y%,X%)=0 TS%=FNEVAL:IFTS%>S% S%=TS%:N
X%=X%:NY%=Y%
 1160    X%=BX%+BCDX%(I%):Y%=BY%+BCDY%(I%)
 1170    IFB%(Y%,X%)=1 TS%=FNEVAL+4:IFTS%>S% S%=TS%
:NX%=X%:NY%=Y%
 1180    NEXT
 1190 IFS%=-99 BT%=TRUE:GOTO1290
 1200 FORI%=0TONM%
 1210    IFMX%(I%)=NX%ANDMY%(I%)=NY% MX%(I%)=0:MA%=
MA%-1:SOUND0,-15,6,7:PRINTTAB(0,30)"One of your me
n has been eaten!!"
 1220    NEXT
 1230 DB%=(B%(NY%,NX%)=1)
 1240 B%(NY%,NX%)=2:B%(BY%,BX%)=0:BX%=NX%:BY%=NY%
 1250 PROCPBOARD
 1260 TIME=0
 1270 REPEAT
 1280    UNTILTIME>70
 1290 CLS
 1300 ENDPROC
 1310 REM*******************************
 1320 DEFPROCPBOARD
 1330 MOVE32*(20-W%)-40,1024-16:PLOT1,64*W%+48,0:P
LOT1,0,-64*H%:PLOT1,-64*W%-48,0:PLOT1,0,64*H%
 1340 PRINTTAB(0,0)
 1350 FORY%=1TOH%
 1360    PRINTTAB(20-W%,VPOS);
 1370    FORX%=1TOW%
 1380      IFB%(Y%,X%)=0 PRINT"."; ELSEIFB%(Y%,X%)=
1 VDU17,2,224 ELSEIFDB% VDU17,1,225 ELSE PRINT".";
 1390      COLOUR3
 1400      VDU9
 1410      NEXT
 1420    PRINTCHR$(9);Y%'
 1430    NEXT
 1440 PRINTTAB(20-W%);
 1450 FORX%=1TOW%
 1460    PRINT;X%" ";
 1470    NEXT
 1480 ENDPROC
```

# Graves

## Rules

You are stuck in a graveyard being chased by some very angry skeletons. You must try to lure them into the open graves to destroy them. You and the skeletons cannot walk over the existing graves or off the edges of the graveyard.

## Display

The graveyard is shown with crosses on the existing graves and rectangles for open graves. You are represented by a matchstick man in white and each skeleton by a drawing of a skull in red.

## Operation

First you are asked for the height and width of the graveyard. Enter the required values (from 9 to 28).

You control your man with the keys:
'Z' – left
'X' – right
':' – up
'/' – down
Holding two keys down at once gives diagonal movement.

## Program

The program simply controls the required man movements and moves the skeletons according to their desire to catch the fleeing man.

| *Section/Variables* | *Function* |
|---|---|
| Main routine | Initialize data, setup size, main game loop, game over |
| B% | Current board position |
| SX% | Skeleton X-coordinates |
| SY% | Skeleton Y-coordinates |
| H% | Board height |
| W% | Board width |
| NG% | Number of graves |
| NS% | Number of skeletons |

| | |
|---|---|
| NAS% | Number of 'alive' skeletons |
| MX% | Man X-coordinate |
| MY% | Man Y-coordinate |
| GO% | Game-over flag |
| FN NO | Return 'yes' or 'no' answer |
| K$ | Input key |
| FN NUM | Get number for graveyard size |
| CX% | Cursor X-coordinate |
| CY% | Cursor Y-coordinate |
| N | Input number |
| PROC RNDSQ | Generate random X, Y values of squares |
| X% | Random square X-coordinate |
| Y% | Random square Y-coordinate |
| PROC MMAN | Move player's man |
| NMX% | Man's new X-coordinate |
| NMY% | Man's new Y-coordinate |
| PROC MSKELETONS | Move the remaining skeletons |
| DX% | Change in skeleton X-coordinate |
| DY% | Change in skeleton Y-coordinate |
| NX% | Skeleton new X-coordinate |
| NY% | Skeleton new Y-coordinate |
| PROC PBOARD | Print the graveyard |
| PROC PMES | Print centralized message and erase old message |
| Y% | Message Y-coordinate |
| M$ | Actual message |

## Suggestions

An option to dig or fill-in graves instead of moving, may improve the game further.

Also, having more than one type of monster and more than one way to destroy them would enhance it.

## The Listing

```
   10 *FX4,1
   20 DIMB%(29,29),SX%(27),SY%(27)
   30 VDU23,224,&38,&38,&10,&7E,&10,&38,&28,&28,23
,225,&18,&18,&7E,&18,&18,&18,&18,&18,23,226,&7E,&4
2,&42,&42,&42,&42,&42,&7E,23,227,&38,&7C,&54,&7C,&
6C,&38,&38,0
   40 ENVELOPE1,&81,-2,-2,-2,100,0,0,0,0,0,-127,12
7,127
   50 REPEAT
   60   MODE1
   70    PRINT'"Enter graveyard height(9-28)";:H%=F
NNUM
   80    PRINT'"Enter graveyard width(9-28)";:W%=FN
NUM
   90    PRINT''"Please wait while I wake the dead!
!!"
  100    VDU23;8202;0;0;0;19,2,6;0;
  110    FORY%=0TO29
  120      FORX%=0TO29
  130        IFY%<1ORY%>H%ORX%<1ORX%>W% B%(Y%,X%)=-
1 ELSE B%(Y%,X%)=0
  140        NEXT
  150      NEXT
  160    NG%=W%*H%/30:NS%=-1
  170    FORI%=0TONG%
  180      PROCRNDSQ:B%(Y%,X%)=-1
  190      PROCRNDSQ:B%(Y%,X%)=3
  200      IFRND(5)<3ORNS%<NG%/3 NS%=NS%+1:PROCRNDS
Q:B%(Y%,X%)=4:SX%(NS%)=X%:SY%(NS%)=Y%
  210      NEXT
  220    NAS%=NS%
  230    PROCRNDSQ:MX%=X%:MY%=Y%
  240    CLS
  250    PROCPBOARD
  260    COLOUR3
  270    PROCPMES(30,"Press any key to start..."):*
FX15,1
  280    K=GET
  290    PROCPMES(30,"")
  300    GO%=FALSE
  310    REPEAT
  320      PROCMMAN
  330      IFGO%=FALSE PROCMSKELETONS
  340      UNTILGO%ORNAS%<0
  350    SOUND0,-15,6,9
  360    COLOUR3
  370    IFNAS%<0 PROCPMES(30,"All the skeletons ar
e dead!")
  380    IFGO%=3 SOUND1,1,250,20:PROCPMES(30,"You w
alked into an open grave!")
  390    IFGO%=4 PROCPMES(30,"A skeleton ripped you
 to pieces!")
  400    PROCPMES(31,"Another game?"):*FX15,1
  410    UNTILFNNO
  420 MODE7
  430 END
  440 REM*********************************
  450 DEFFNNO
  460 REPEAT
  470    K$=GET$
  480    UNTILK$="Y"ORK$="N"
  490 =K$="N"
  500 REM*******************************
  510 DEFFNNUM
  520 CX%=POS:CY%=VPOS
  530 REPEAT
  540   PRINTTAB(CX%,CY%)SPC(255)TAB(CX%,CY%);:INP
UTN
  550   UNTILN>=9ANDN<=28
  560 =INT(N)
  570 REM*******************************
  580 DEFPROCRNDSQ
  590 REPEAT
  600   X%=RND(W%):Y%=RND(H%)
  610   UNTILB%(Y%,X%)=0
  620 ENDPROC
  630 REM*******************************
  640 DEFPROCMMAN
  650 NMX%=MX%:NMY%=MY%
  660 IFINKEY(-98) NMX%=NMX%-1
  670 IFINKEY(-67) NMX%=NMX%+1
  680 IFINKEY(-73) NMY%=NMY%-1
  690 IFINKEY(-105) NMY%=NMY%+1
  700 COLOUR3
  710 IFB%(NMY%,NMX%)=-1THEN760
  720 PRINTTAB(19-W%DIV2+MX%,MY%)" ";
  730 MX%=NMX%:MY%=NMY%
  740 PRINTTAB(19-W%DIV2+MX%,MY%)CHR$(224);
  750 GO%=B%(MY%,MX%)
  760 ENDPROC
  770 REM*******************************
  780 DEFPROCMSKELETONS
  790 TIME=0
  800 FORI%=0TONS%
  810   SOUND&10,-5,RND(3)+3,1
  820   X%=SX%(I%):Y%=SY%(I%)
  830   IFX%=-1THEN970
  840   IFRND(9)<2THEN970
  850   IFX%<MX% DX%=1 ELSEIFX%>MX% DX%=-1 ELSE DX
%=0
  860   IFRND(8)<2 DX%=RND(3)-2
  870   IFY%<MY% DY%=1 ELSEIFY%>MY% DY%=-1 ELSE DY
%=0
  880   IFRND(8)<2 DY%=RND(3)-2
  890   NX%=X%+DX%:NY%=Y%+DY%
  900   IFB%(NY%,NX%)=4ORB%(NY%,NX%)=-1THENNX%=X%+
RND(3)-2:NY%=Y%+RND(3)-2:IFB%(NY%,NX%)=4ORB%(NY%,N
X%)=-1THEN970
  910   B%(Y%,X%)=0:SX%(I%)=-1
  920   PRINTTAB(19-W%DIV2+X%,Y%)" ";
  930   IFB%(NY%,NX%)=3 SOUNDRND(3),1,200,20:COLOU
R1:PROCPMES(30,"A skeleton fell into a grave!"):NA
```

```
S%=NAS%-1:GOTO970
  940     B%(NY%,NX%)=4:SX%(I%)=NX%:SY%(I%)=NY%
  950     COLOUR1
  960     PRINTTAB(19-W%DIV2+NX%,NY%)CHR$(227);
  970     NEXT
  980 IFB%(MY%,MX%)=4 GO%=4:TIME=999
  990 REPEAT
 1000     UNTILTIME>99
 1010 PROCPMES(30,"")
 1020 ENDPROC
 1030 REM*****************************
 1040 DEFPROCPBOARD
 1050 MOVE32*(20-W%DIV2)-8,1024-24:PLOT1,32*W%+16,
0:PLOT1,0,-32*H%-16:PLOT1,-32*W%-16,0:PLOT1,0,32*H
%+16
 1060 PRINTTAB(0,0)
 1070 FORY%=1TOH%
 1080     PRINTTAB(20-W%DIV2,VPOS);
 1090     FORX%=1TOW%
 1100       IFMX%=X%ANDMY%=Y% VDU17,3,224 ELSEIFB%(Y
%,X%)=0 PRINT" "; ELSEIFB%(Y%,X%)=-1 VDU17,2,225 E
LSEIFB%(Y%,X%)=3 VDU17,2,226 ELSEVDU17,1,227
 1110       NEXT
 1120     PRINT
 1130     NEXT
 1140 ENDPROC
 1150 REM*****************************
 1160 DEFPROCPMES(Y%,M$)
 1170 PRINTTAB(0,Y%)SPC(39)TAB(20-LEN(M$)/2,Y%)M$;
 1180 ENDPROC
```

# March

## Rules

The board is split into three ranks for each side: the 'home', 'limbo' and 'safety' ranks.

The object of the game is to move all your men off the home-rank before the opponent does so with his army.

You move your men with the throw of two dice. You may move the man from the home-point number shown on either die or the sum of both dice, forward one rank. You may also move your men from the safety-point back one rank using the dice in a similar fashion. You may also move your opponent's men back from the limbo rank to their home-rank similarly.

If you use both dice, you may roll again for another move. Your move is over when you cannot use one of the dice, or choose not to use one of the dice for tactical purposes.

After one side finishes its move, all its opponent's men still on the limbo-rank are automatically moved to the safety-rank.

To start the game, each player rolls one die and the player with the highest die moves first.

## Display

The board shows the computer's men at the top, moving down; and your men at the bottom, moving up.

The home-ranks are coloured black, the limbo ranks red, and the safety-ranks yellow. The ranks are also labelled to avoid confusion.

The men are numbered from 1 to 10 from left to right.

The moves are displayed below the board printout.

## Operation

First the two dice are rolled to see who moves first. The computer's die is on the left and your die is on the right. The player with the highest die moves first.

You input your move by typing the letter 'H' or 'C' to signify the 'human' or 'computer' army, then a comma, then the number of the man you wish to move.

Any illegal moves are rejected.

To finish your move you may, at any time, enter a negative man-number.

## Program

The program controls and validates the human moves and selects the 'best' move for the computer.

| *Section/Variables* | *Function* |
|---|---|
| Main routine | Initialize data, main game loop, game over |
| H% | Human army |
| C% | Computer army |
| GO% | Game-over flag |
| PROC CMOVE | Handle computer's move |
| NLM% | No-legal-moves flag |
| M$ | Man character |
| M% | Man file |
| PROC HMOVE | Handle human's move |
| FN NOLEGALMOVES | Check for any legal human moves |
| PROC PBOARD | Print board display |
| PROC SETCOL | Set required colour and label each rank |
| I% | Rank number |
| PROC ROLL | Roll dice |
| D0 | 1st die value |
| D1 | 2nd die value |
| D2 | Sum of first two die |
| PROC PDICE | Print both dice |
| PROC PDIE | Print single die |
| PROC PMES | Print message at required line and clear previous message |
| Y% | Message Y-coordinate |
| M$ | Actual message |
| PROC DELAY | Delay for required time |
| L% | Limit value for delay |

## Suggestions

Additional rules or hazards may be added to enhance the skill level required in the game.

## The Listing

```
   10 *FX4,1
   20 DIMH%(2,11),C%(2,11)
   30 VDU23,224,0,0,0,&18,&18,0,0,0,23,225,&60,&60
,0,0,0,0,6,6,23,226,&60,&60,0,&18,&18,0,6,6,23,227
,&66,&66,0,0,0,0,&66,&66,23,228,&66,&66,0,&18,&18,
0,&66,&66,23,229,&66,&66,0,&66,&66,0,&66,&66
```

```
   40 VDU23,230,&38,&38,&10,&7E,&10,&38,&28,&28
   50 REPEAT
   60   FORI%=0TO2
   70     FORJ%=1TO10
   80       H%(I%,J%)=-(I%=0):C%(I%,J%)=-(I%=0)
   90       NEXT
  100     NEXT
  110   MODE1:VDU23;8202;0;0;0;
  120   GO%=FALSE
  130   PROCPBOARD
  140   PROCPMES(24,"Rolling for 1st move...")
  150   REPEAT
  160     PROCROLL
  170     UNTILD0<>D1
  180   IFD0<D1 PROCPMES(24,"You move first!") ELS
E PROCPMES(24,"I move first")
  190   PROCDELAY(200)
  200   IFD0<D1 PROCHMOVE
  210   REPEAT
  220     PROCCMOVE
  230     IFGO%=FALSE PROCHMOVE
  240     UNTILGO%
  250   FORI%=1TO9
  260     SOUND2,-15,I%*24,1
  270     NEXT
  280   IFGO%=1 PROCPMES(24,"I win!") ELSE PROCPME
S(24,"You win!")
  290   PROCPMES(25,"Another game?"):*FX15,1
  300   UNTILGET$="N"
  310 MODE7
  320 END
  330 REM********************************
  340 DEFPROCCMOVE
  350 REPEAT
  360   PROCROLL
  370   REPEAT
  380     PRINTTAB(0,24)SPC(80):PROCPMES(24,CHR$(7
)+"My move:")
  390     NLM%=FALSE
  400     M$="H"
  410     IFH%(1,D0)+H%(1,D1)+(D0=D1)<2 IFH%(1,D2)
 H%(1,D2)=0:H%(0,D2)=1:M%=D2:D2=0:GOTO500
  420     IFH%(1,D1) H%(1,D1)=0:H%(0,D1)=1:M%=D1:D
1=0:D2=11:GOTO500
  430     IFH%(1,D0) H%(1,D0)=0:H%(0,D0)=1:M%=D0:D
0=0:D2=11:GOTO500
  440     M$="C"
  450     IFC%(0,D0)+C%(0,D1)+(D0=D1)<2 IFC%(0,D2)
 C%(0,D2)=0:C%(1,D2)=1:M%=D2:D2=0:GOTO500
  460     IFC%(0,D0) C%(0,D0)=0:C%(1,D0)=1:M%=D0:D
0=0:D2=11:GOTO500
```

```
  470        IFC%(0,D1) C%(0,D1)=0:C%(1,D1)=1:M%=D1:D
1=0:D2=11:GOTO500
  480        NLM%=TRUE
  490        GOTO570
  500        PRINTM$",";M%
  510        PROCPBOARD
  520        PROCDELAY(70)
  530        GO%=1
  540        FORI%=1TO10
  550          IFC%(0,I%)=1 GO%=FALSE
  560          NEXT
  570        UNTILGO%ORNLM%OR(D0=0ANDD1=0)ORD2=0
  580      UNTILGO%ORNLM%
  590 IFNLM% PROCPMES(25,"End of my move!"):SOUND1
,-15,120,9:PROCDELAY(50)
  600 FORI%=1TO10
  610     IFH%(1,I%)=1 H%(1,I%)=0:H%(2,I%)=1
  620     NEXT
  630 PROCPBOARD
  640 ENDPROC
  650 REM*******************************
  660 DEFPROCHMOVE
  670 REPEAT
  680     *FX15,1
  690     PROCROLL
  700     REPEAT
  710       NLM%=FNNOLEGALMOVES
  720       IFNLM% THEN910
  730       PRINTTAB(0,24)SPC(80):PROCPMES(24,CHR$(7
)+"Your move")
  740       INPUTM$,M%
  750       IFM%<0THEN910
  760       M$=LEFT$(M$,1)
  770       IF(M$<>"H"ANDM$<>"C")ORM%<1ORM%>10OR(M%<
>D0 ANDM%<>D1 ANDM%<>D2) THEN990
  780       IFM$="C"THEN830
  790       IFH%(1,M%)=1 THEN990
  800       IFH%(0,M%)=1 H%(0,M%)=0 ELSE H%(2,M%)=0
  810       H%(1,M%)=1
  820       GOTO850
  830       IFC%(1,M%)=0 THEN990
  840       C%(1,M%)=0:C%(0,M%)=1
  850       IFD0=M%D0=0:D2=11 ELSEIFD1=M%D1=0:D2=11
ELSE D2=0
  860       PROCPBOARD
  870       GO%=2
  880       FORI%=1TO10
  890         IFH%(0,I%)=1 GO%=FALSE
  900          NEXT
  910       UNTILGO%ORM%<0OR(D0=0ANDD1=0)ORD2=0ORNLM
%
```

```
  920    UNTILGO%ORM%<0ORNLM%
  930  IFNLM%ORM%<0 PROCPMES(25,"End of your move!"
):SOUND1,-15,120,9:PROCDELAY(50)
  940  FORI%=1TO10
  950    IFC%(1,I%)=1 C%(1,I%)=0:C%(2,I%)=1
  960    NEXT
  970  PROCPBOARD
  980  ENDPROC
  990  PROCPMES(25,"*ILLEGAL*")
 1000  SOUND1,-15,5,9
 1010  PROCDELAY(50)
 1020  GOTO730
 1030  REM********************************
 1040  DEFFNNOLEGALMOVES
 1050  IFH%(0,D0)ORH%(0,D1)ORH%(0,D2)ORH%(2,D0)ORH%
(2,D1)ORH%(2,D2):=FALSE
 1060  IFC%(1,D0)ORC%(1,D1)ORC%(1,D2):=FALSE
 1070  =TRUE
 1080  REM*********************************
 1090  DEFPROCPBOARD
 1100  MOVE9*32-4,1024-8*32:PLOT1,21*32+4,0:PLOT1,0
,-11*32-4:PLOT1,-21*32-4,0:PLOT1,0,11*32+4
 1110  PRINTTAB(0,7)
 1120  FORI%=0TO2
 1130    PROCSETCOL
 1140    FORJ%=1TO10
 1150      PRINT" ";
 1160      IFC%(I%,J%) VDU230 ELSE VDU32
 1170      NEXT
 1180    PRINT" "
 1190    NEXT
 1200  PRINT
 1210  PRINTTAB(0,15)
 1220  FORI%=2TO0STEP-1
 1230    PROCSETCOL
 1240    FORJ%=1TO10
 1250      PRINT" ";
 1260      IFH%(I%,J%) VDU230 ELSE VDU32
 1270      NEXT
 1280    PRINT" "
 1290    NEXT
 1300  COLOUR128
 1310  FORI%=1TO10
 1320    PRINTTAB(8+2*I%,6);I%TAB(8+2*I%,20)I%
 1330    NEXT
 1340  ENDPROC
 1350  REM********************************
 1360  DEFPROCSETCOL
 1370  COLOUR128
 1380  IFI%=0 PRINT"HOME"TAB(9,VPOS);:COLOUR128
 1390  IFI%=1 PRINT"LIMBO"TAB(9,VPOS);:COLOUR129
```

```
1400 IFI%=2 PRINT"SAFETY"TAB(9,VPOS);:COLOUR130
1410 ENDPROC
1420 REM*******************************
1430 DEFPROCROLL
1440 FORI%=0TO9+RND(9)
1450   D0=RND(6):D1=RND(6):SOUND3,-2,20*D0,1:PROC
PDICE
1460   NEXT
1470 D2=D0+D1:IFD2>10 D2=11
1480 ENDPROC
1490 REM******************************
1500 DEFPROCPDICE
1510 PROCPDIE(16,13,D0):PROCPDIE(22,13,D1)
1520 ENDPROC
1530 REM******************************
1540 DEFPROCPDIE(DX%,DY%,V%)
1550 MOVEDX%*32-4,(32-DY%)*32+8:PLOT1,36,0:PLOT1,
0,-52:PLOT1,-36,0:PLOT1,0,52
1560 PRINTTAB(DX%,DY%)CHR$(223+V%)
1570 ENDPROC
1580 REM*******************************
1590 DEFPROCPMES(Y%,M$)
1600 PRINTTAB(9,Y%)SPC(30)TAB(9,Y%)M$;
1610 ENDPROC
1620 REM*******************************
1630 DEFPROCDELAY(L%)
1640 TIME=0
1650 REPEAT
1660   UNTILTIME>L%
1670 ENDPROC
```

# Mine

## Rules

The object is to move your man from the top-left corner of the minefield to the bottom-right corner (without treading on a mine!).

You may move up, down, left or right, but not diagonally.

You also cannot move off the edges of the minefield or onto the rocky areas scattered around the minefield.

Your mine detector will tell you, before each move, how many mines there are in the eight squares around you. You can then use this information to steer clear of densely mined areas and make your way to safety!

You have a limited time, however, before the batteries in your mine detector run out, so don't dawdle. If the batteries do run out, then you're on your own!

## Display

The display shows a matchstick man at the spot where you are currently positioned. The red boulders are the rocky areas. Unexplored squares are shown with a white dot.

As you move, the square you have just come from is changed to a number, from 0 to 8, to indicate the number of mines around that square (unless the batteries have run out, in which case the square is changed to a question mark).

The amount of battery life remaining is shown to the right of the minefield in seconds.

Below the minefield is displayed the current move number and the number of mines around the current square.

## Operation

First the program asks for height and width of the minefield. Enter numbers between 4 and 14 press <RETURN>.

When the minefield is displayed you use the following keys to control your man:
'Z' – left
'X' – right
':' – up
'/' – down

## Program

| *Section/Variables* | *Function* |
|---|---|
| Main routine | Initialize data, setup size, main game loop, game over |
| B% | Current board position |
| H% | Minefield height |
| W% | Minefield width |
| NM% | Number of mines in minefield |
| T% | Battery time left |
| M% | Move number |
| MX% | Man X-coordinate |
| MY% | Man Y-coordinate |
| FN NUM | Get minefield dimension |
| CX% | Cursor X-coordinate |
| CY% | Cursor Y-coordinate |
| N | Input number |
| PROC RNDSQ | Generate random unoccupied squares' X, Y coordinates |
| X% | Random X-coordinate |
| Y% | Random Y-coordinate |
| PROC MMAN | Move player's man |
| C% | Surrounding-mines counter |
| NMX% | Man's new X-coordinate |
| NMY% | Man's new Y-coordinate |
| K$ | Input key |
| PROC PBOARD | Print the whole minefield |
| MF% | Display mines flag |

## Suggestions

Sometimes no path across the minefield exists, so a feature where you can dynamite selected surrounding squares to clear mines and rocks before you step on them, might be useful. Obviously only a limited supply of dynamite would be available.

Alternatively, you could make the program only generate minefields where a path exists.

Also, diagonal movement may help get round this problem.

## The Listing

```
   10 *FX4,1
   20 DIMB%(15,15)
   30 VDU23,224,&38,&38,&10,&7E,&10,&38,&28,&28,23
,225,0,&20,&34,&3E,&7E,&7F,&FF,&FF
   40 REPEAT
   50   MODE1
   60    PRINT'"Enter minefield height(4-14)";:H%=F
NNUM
   70    PRINT'"Enter minefield width(4-14)";:W%=FN
NUM
   80    VDU23;8202;0;0;0;
   90    FORY%=0TO15
  100      FORX%=0TO15
  110        B%(Y%,X%)=-1
  120        NEXT
  130      NEXT
  140    FORY%=1TOH%
  150      FORX%=1TOW%
  160        B%(Y%,X%)=0
  170        NEXT
  180      NEXT
  190    NM%=W%*H%/9+1
  200    T%=W%*H%:TIME=0
  210    FORI%=0TONM%
  220      PROCRNDSQ:B%(Y%,X%)=-1:PROCRNDSQ:B%(Y%,X
%)=9
  230      NEXT
  240    B%(1,1)=0:B%(1,2)=0:B%(2,1)=0:B%(H%,W%)=0:
B%(H%-1,W%)=0:B%(H%,W%-1)=0
  250    M%=0:MX%=1:MY%=1
  260    CLS
  270    REPEAT
  280      M%=M%+1
  290      PROCPBOARD(FALSE)
  300      PROCMMAN
  310      UNTILB%(MY%,MX%)=9OR(MX%=W%ANDMY%=H%)
  320    PROCPBOARD(TRUE)
  330    IFB%(MY%,MX%)=9 SOUND0,-15,6,9:PRINTTAB(10
,30)"You trod on a mine!!!" ELSE PRINTTAB(11,30)"Y
ou made it across!"
  340    PRINTTAB(14,31)"Another game?";:*FX15,1
  350    UNTILGET$="N"
  360 MODE7
  370 END
  380 REM*******************************
  390 DEFFNNUM
  400 CX%=POS:CY%=VPOS
  410 REPEAT
  420    PRINTTAB(CX%,CY%)SPC(255)TAB(CX%,CY%);:INP
UTN
  430    UNTILN>=4ANDN<=14
  440 =INT(N)
  450 REM*******************************
  460 DEFPROCRNDSQ
  470 REPEAT
  480    X%=RND(W%):Y%=RND(H%)
  490    UNTILB%(Y%,X%)=0
  500 ENDPROC
  510 REM******************************
  520 DEFPROCMMAN
  530 C%=0
  540 FORX%=MX%-1TOMX%+1
  550   FORY%=MY%-1TOMY%+1
  560     IFB%(Y%,X%)=9 C%=C%+1
  570     NEXT
  580   NEXT
  590 PRINT"Move ";M%") ";
  600 IFT%=0 C%=9:PRINT"Your detector has failed!!
!  " ELSEPRINT"There are ";C%" mines around you."
  610 SOUND1,-15,25*(C%+1),1
  620 REPEAT
  630   NMX%=MX%:NMY%=MY%:IFB%(MY%,MX%)=0 B%(MY%,M
X%)=100+C%
  640   *FX15,1
  650   REPEAT
  660     IFTIME>99IFT%>0THENTIME=0:T%=T%-1
  670     PRINTTAB(21+W%,H%)"Time"TAB(22+W%,H%+1);
T%"  "
  680     K$=INKEY$(0)
  690     UNTILK$<>""
  700   IFK$="Z" NMX%=NMX%-1:IFNMX%<1 NMX%=1
  710   IFK$="X" NMX%=NMX%+1:IFNMX%>W% NMX%=W%
  720   IFK$=":" NMY%=NMY%-1:IFNMY%<1 NMY%=1
  730   IFK$="/" NMY%=NMY%+1:IFNMY%>H% NMY%=H%
  740   UNTIL(NMX%<>MX%ORNMY%<>MY%)ANDB%(NMY%,NMX%
)>=0
  750 MX%=NMX%:MY%=NMY%
  760 ENDPROC
  770 REM******************************
  780 DEFPROCPBOARD(MF%)
  790 MOVE32*(20-W%)-40,1024-16:PLOT1,64*W%+48,0:P
LOT1,0,-64*H%:PLOT1,-64*W%-48,0:PLOT1,0,64*H%
  800 PRINTTAB(0,0)
  810 FORY%=1TOH%
  820   PRINTTAB(20-W%,VPOS);
  830   FORX%=1TOW%
  840     IFMX%=X%ANDMY%=Y% VDU17,2,224 ELSEIFB%(Y
%,X%)=0 PRINT"."; ELSEIFB%(Y%,X%)=-1 VDU17,1,225 E
LSEIFB%(Y%,X%)=109 PRINT"?"; ELSEIFB%(Y%,X%)>99 PR
INT;B%(Y%,X%)-100; ELSEIFMF% PRINT"*";:SOUND1,-15,
RND(255),1 ELSE PRINT".";
  850     COLOUR3
  860     VDU9
  870     NEXT
  880   PRINT'
  890   NEXT
  900 ENDPROC
```

# Solitaire

## Rules

The board consists of 45 holes arranged in a large cross-shape. Initially all but the centre hole is occupied by a peg.

The idea is to remove as many pegs as possible before being unable to move.

You move by taking one peg and jumping over an adjacent peg into an empty hole. You can jump sideways but not diagonally.

The game is over when you cannot move any more pegs.

## Display

The best-score so far and your current score are shown at the top of the display.

The board is printed with yellow pegs and red holes, with numbers along two edges.

Below the board is displayed your current move.

## Operation

Move the cursor (the '<' sign) to the square of the peg you want to move and press <RETURN>. Then move the cursor to the empty square you wish to jump into and again press <RETURN>.

If the move you enter is illegal it is rejected and cleared from the display.

You control the cursor with the keys:
'Z' – left
'X' – right
':' – up
'/' – down

## Program

The program simply updates your selected moves and checks whether there are any legal moves left.

| *Section/Variables* | *Function* |
|---|---|
| Main routine | Initialize data, main game loop, game over |
| B% | Current board position |

| | |
|---|---|
| BSC% | Best score so far |
| CX% | Cursor X-coordinate |
| CY% | Cursor Y-coordinate |
| SC% | Current score |
| FN NOLEGALMOVES | Check if player has any legal moves left |
| NLM% | No-legal-moves flag |
| PROC GETSQ | Handle cursor till square selected |
| K$ | Input key |
| PROC MOVE | Move peg |
| FX% | Peg from-hole X-coordinate |
| FY% | Peg from-hole Y-coordinate |
| FN LEGALMOVE | Check move for validity |
| PROC PBOARD | Print current board position |
| PROC PSCORES | Print current scores |

## Suggestions

Add to the program the intelligence required to solve the puzzle itself and demonstrate the method.

Also, allowing different positions to be setup would be nice. You could then try several different lines from a position to find the best one.
(My best score: 4)

## The Listing

```
   10 *FX4,1
   20 VDU23,224,0,0,0,&18,&18,0,0,0,23,225,0,&3C,&
7E,&7E,&7E,&7E,&3C,0
   30 DIMB%(10,10)
   40 BSC%=44
   50 REPEAT
   60   FORI%=0TO10
   70     FORJ%=0TO10
   80       B%(I%,J%)=-1
   90       IF(I%>3ANDI%<7ANDJ%>0ANDJ%<10)OR(J%>3A
NDJ%<7ANDI%>0ANDI%<10) B%(I%,J%)=1
  100       NEXT
  110     NEXT
  120   B%(5,5)=0
  130   CX%=5:CY%=5
  140   SC%=44
  150   MODE1:VDU23;8202;0;0;0;
  160   PROCPBOARD
  170   REPEAT
  180     PROCMOVE
  190     PROCPBOARD
  200       UNTILFNNOLEGALMOVES
  210     IFSC%<BSC% BSC%=SC%:PROCPSCORES
  220     FORI%=0TO9
  230       SOUND2,-15,RND(255),1
  240       NEXT
  250     PRINTTAB(10,30)"Another game?":*FX15,1
  260     UNTILGET$="N"
  270 MODE7
  280 END
  290 REM*******************************
  300 DEFFNNOLEGALMOVES
  310 NLM%=TRUE
  320 FORI%=1TO9
  330   FORJ%=1TO9
  340     IFB%(I%,J%)<>1THEN400
  350     IFB%(I%-1,J%)=1 IFB%(I%-2,J%)=0 NLM%=FAL
SE
  360     IFB%(I%+1,J%)=1 IFB%(I%+2,J%)=0 NLM%=FAL
SE
  370     IFB%(I%,J%-1)=1 IFB%(I%,J%-2)=0 NLM%=FAL
SE
  380     IFB%(I%,J%+1)=1 IFB%(I%,J%+2)=0 NLM%=FAL
SE
  390     IFNOTNLM% J%=9:I%=9
  400     NEXT
  410   NEXT
  420 =NLM%
  430 REM*******************************
  440 DEFPROCGETSQ
  450 REPEAT
  460   PRINTTAB(CX%*2+8,CY%*2+2)"<";
  470   K$=GET$:*FX15,1
  480   PRINTTAB(CX%*2+8,CY%*2+2)" ";
  490   IFK$="Z" CX%=CX%-1:IFB%(CY%,CX%)=-1 CX%=CX
%+1
  500   IFK$="X" CX%=CX%+1:IFB%(CY%,CX%)=-1 CX%=CX
%-1
  510   IFK$="/" CY%=CY%+1:IFB%(CY%,CX%)=-1 CY%=CY
%-1
  520   IFK$=":" CY%=CY%-1:IFB%(CY%,CX%)=-1 CY%=CY
%+1
  530   UNTILK$=CHR$(13)
  540 ENDPROC
  550 REM*******************************
  560 DEFPROCMOVE
  570 REPEAT
  580   PRINTTAB(9,26)SPC(20)TAB(9,26)CHR$(7)"From
?"
  590   REPEAT
  600     PROCGETSQ
  610     UNTILB%(CY%,CX%)=1
  620    FX%=CX%:FY%=CY%
  630    PRINTTAB(14,26);FX%",";FY%"  To?"
  640    PROCGETSQ
  650    PRINTTAB(22,26);CX%",";CY%
  660    UNTILFNLEGALMOVE
  670 B%(FY%,FX%)=0:B%(CY%,CX%)=1:B%(FY%+(CY%-FY%)
DIV2,FX%+(CX%-FX%)DIV2)=0
  680 SOUND1,-15,220,3
  690 SC%=SC%-1
  700 ENDPROC
  710 REM*******************************
  720 DEFFNLEGALMOVE
  730 IFB%(CY%,CX%)<>0 THEN760
  740 IFABS(FX%-CX%)=2ANDFY%=CY%ANDB%(FY%,FX%+(CX%
-FX%)DIV2)=1 :=TRUE
  750 IFABS(FY%-CY%)=2ANDFX%=CX%ANDB%(FY%+(CY%-FY%
)DIV2,FX%)=1 :=TRUE
  760 SOUND3,-15,5,9
  770 TIME=0
  780 REPEAT
  790    UNTILTIME>99
  800 =FALSE
  810 REM*******************************
  820 DEFPROCPBOARD
  830 MOVE14*32,1028-3*32:PLOT1,7*32,0:PLOT1,0,-6*
32:PLOT1,6*32,0:PLOT1,0,-7*32:PLOT1,-6*32,0:PLOT1,
0,-6*32:PLOT1,-7*32,0:PLOT1,0,6*32:PLOT1,-6*32,0:P
LOT1,0,7*32:PLOT1,6*32,0:PLOT1,0,6*32
  840 PROCPSCORES
  850 PRINTTAB(0,1)
  860 FORI%=0TO10
  870    PRINTSPC(7);
  880    FORJ%=0TO10
  890       IFB%(I%,J%)=-1 VDU9 ELSEIFB%(I%,J%)=0 VD
U17,1,224 ELSE VDU17,2,225
  900       VDU17,3,9
  910       NEXT
  920    IFI%>0ANDI%<10 PRINT;I%;
  930    PRINT'
  940    NEXT
  950 PRINTSPC(9);
  960 FORJ%=1TO9
  970    PRINT;J%" ";
  980    NEXT
  990 ENDPROC
 1000 REM*******************************
 1010 DEFPROCPSCORES
 1020 PRINTTAB(2,1)"Best score:";BSC%" "TAB(22,1)"
Score:";SC%" "
 1030 ENDPROC
```

# Towers

## Rules

This is an ancient puzzle which involves moving coloured discs from one of three pegs onto one of the other two pegs.

The discs are of different sizes and you may never move a larger disc onto a smaller one.

This program uses six discs.

The minimum number of moves required is 2 to the power of (the number of discs) minus 1. Hence for six discs you require at least $2^6 - 1 = 63$ moves.

## Display

The display shows the three white pegs, and the red and yellow coloured discs, on their current pegs.

Each peg is labelled with its number.

## Operation

The pegs are numbered 1, 2 and 3.

When asked for your move you must first select a peg from which you wish to move the topmost disc. Then press the number key corresponding to this peg. Then select which peg you wish to move the disc to and again press the number key required.

Any illegal moves will be rejected. If you enter the wrong peg to move from, you can cancel it by entering the same peg number again.

## Program

The program controls the required moves and redraws the pegs after each move.

| *Section/Variables* | *Function* |
|---|---|
| Main routine | Initialize data, main game loop, game over |
| B% | Current peg positions |
| BSC% | Best score so far |
| M% | Current move number |
| PROC MOVE | Move disc |
| F% | Peg to move disc from |

|  |  |
|---|---|
| TPF% | Topmost disc on 'from' peg |
| T% | Peg to move disc to |
| TPT% | Topmost disc on 'to' peg |
| PROC GETTOWER | Get selected tower number |
| K$ | Input key |
| T% | Tower index |
| FN TP | Find topmost disc on required peg |
| T% | Selected tower (peg) |
| TP% | Topmost disc found so far |
| PROC PTOWERS | Print all three towers |
| PROC PTOWER | Print required tower |
| O% | Current disc size |
| PROC PSCORES | Print current scores |

## Suggestions

Add to the program the intelligence required to both solve the puzzle itself and demonstrate the method.

Also make the program give the option of the number of discs required (up to a reasonable maximum).
(My best score: 63)

## The Listing

```
   10 *FX4,1
   20 DIMB%(2,6)
   30 BSC%=999
   40 REPEAT
   50   MODE1:VDU23;8202;0;0;0;
   60   FORI%=0TO6
   70     B%(0,I%)=I%:B%(1,I%)=0:B%(2,I%)=0
   80     NEXT
   90   M%=0
  100   PROCPTOWERS
  110   REPEAT
  120     PROCMOVE
  130     PROCPTOWERS
  140     UNTILB%(1,1)=1ORB%(2,1)=1
  150   IFM%<BSC% BSC%=M%:PROCPSCORES
  160   FORI%=1TO9
  170     SOUND1,-15,I%*20,1
  180     NEXT
  190   PRINTTAB(9,30)"Another game?";:*FX15,1
  200   UNTILGET$="N"
  210 MODE7
  220 END
  230 REM********************************
  240 DEFPROCMOVE
  250 M%=M%+1
  260 COLOUR128:COLOUR3
  270 PRINTTAB(0,28)SPC(39)TAB(9,28)"Move ";M%") F
rom?";:PROCGETTOWER:F%=T%
  280 TPF%=FNTP(F%)
  290 IFTPF%=7 THEN360
  300 PRINT"  To?";:PROCGETTOWER
  310 TPT%=FNTP(T%)
  320 IFF%=T% THEN360
  330 IFTPT%<7 IFB%(F%,TPF%)>B%(T%,TPT%) THEN360
  340 B%(T%,TPT%-1)=B%(F%,TPF%):B%(F%,TPF%)=0
  350 ENDPROC
  360 COLOUR1
  370 PRINTTAB(9,29)"*ILLEGAL*"
  380 SOUND1,-15,5,20
  390 TIME=0
  400 REPEAT
  410   UNTILTIME>99
  420 PRINTTAB(9,29)SPC(9)
  430 GOTO260
  440 REM*******************************
  450 DEFPROCGETTOWER
  460 REPEAT
  470   K$=GET$:*FX15,1
  480   UNTILK$>="1"ANDK$<="3"
  490 T%=ASCK$-ASC"1"
  500 PRINT;T%+1;
  510 SOUND1,-15,(T%+1)*50,1
  520 ENDPROC
  530 REM*******************************
  540 DEFFNTP(T%)
  550 TP%=7
  560 FORI%=6TO1STEP-1
  570   IFB%(T%,I%)>0 TP%=I%
  580   NEXT
  590 =TP%
  600 REM*******************************
  610 DEFPROCPTOWERS
  620 PROCPTOWER(0,9,5)
  630 PROCPTOWER(1,20,14)
  640 PROCPTOWER(2,31,5)
  650 PROCPSCORES
  660 ENDPROC
  670 REM*******************************
  680 DEFPROCPTOWER(T%,X%,Y%)
  690 FORI%=0TO6
  700   O%=B%(T%,I%)
  710   IFO%=0 VDU17,128,31,X%-6,Y%+I%:PRINTSPC(13
);:VDU17,131,31,X%,Y%+I%,32 ELSE COLOUR129-(O%/2=O
%DIV2):PRINTTAB(X%-O%,Y%+I%)SPC(2*O%+1);
  720   NEXT
  730 COLOUR128
  740 PRINTTAB(X%,Y%+8);T%+1
  750 ENDPROC
  760 REM*******************************
  770 DEFPROCPSCORES
  780 COLOUR128:COLOUR2
  790 PRINTTAB(3,1)"Best score:";BSC%"    "TAB(26,1)
"Score:";M%"   "
  800 ENDPROC
```

# Rotate

## Rules

This puzzle involves arranging the letters 'A' to 'P' in their correct order within a four-by-four square. The letters are initially placed in a random order by the computer.

You can move the letters by rotating any block of four letters in a clockwise direction. Thoughtful rotations can gradually move the letters into their correct places.

Also, one special 'swap' move is allowed per game, where you can swap over any two letters. This may be vital to your completing the puzzle so don't use it up too early!

## Display

At the centre of the display is the current board with the current letters as they actually are.

To the right of the display is a smaller drawing of what the puzzle should end up looking like.

To the left is a small table to indicate the key you must press to make the computer rotate the required block of four letters.

## Operation

The program simply repeatedly asks for your next move. This is specified by a number key from '1' to '9'.

The key specifies the top left corner of the block of four letters you wish to rotate.

After you select a key, the four letters will rotate by ninety degrees and the new board displayed.

The swap move can be selected by pressing key '0', followed by the two letters you wish to swap. You can abort the swap move by entering the same letter twice. You can still then use the swap move later.

## Program

The program generates the initial random setup, controls the selected moves and checks for completion of the puzzle.

| *Section/Variables* | *Function* |
|---|---|
| Main routine | Initialize data, main game loop, game over |
| B% | Current board position |
| V% | Random letter value |
| USED% | Letter-used-already flag |
| M% | Move number |
| SWAP% | Swap-move-used flag |
| FN CORRECT | Check puzzle correctness |
| C% | Correct flag |
| PROC MOVE | Handle move |
| K$ | Input key |
| PROC SWAP | Handle special swap move |
| L1% | First letter to swap |
| L2% | Second letter to swap |
| FN GETLET | Get letter key from 'A' to 'P' |
| K$ | Input key |
| PROC ROTATE | Handle rotate move |
| I% | Board index |
| T% | Temporary storage |
| PROC PBOARD | Print board information |

## Suggestions

The program could be changed to play a selection of similar letter puzzles using the same data structures.

## The Listing

```
   10 *FX4,1
   20 DIMB%(15)
   30 REPEAT
   40   FORI%=0TO15
   50     B%(I%)=-1
   60     NEXT
   70   FORI%=0TO15
   80     REPEAT
   90       V%=RND(16)-1
  100       USED%=FALSE
  110       FORJ%=0TOI%
  120         IFV%=B%(J%) USED%=TRUE
  130         NEXT
  140       UNTILNOTUSED%
  150     B%(I%)=V%
  160     NEXT
  170   MODE5:VDU23;8202;0;0;0;
  180   PROCPBOARD
  190   M%=0:SWAP%=FALSE
```

```
 200   REPEAT
 210     PROCMOVE
 220     PROCPBOARD
 230     UNTILFNCORRECT
 240   FORI%=1TO9
 250     SOUND1,-15,I%*25,1
 260     NEXT
 270   PRINTTAB(3,27)"Another game?";:*FX15,1
 280   UNTILGET$="N"
 290 MODE7
 300 END
 310 REM*****************************
 320 DEFFNCORRECT
 330 C%=TRUE
 340 FORI%=0TO15
 350   IFB%(I%)<>I% C%=FALSE
 360   NEXT
 370 =C%
 380 REM*****************************
 390 DEFPROCMOVE
 400 M%=M%+1
 410 COLOUR128:COLOUR3
 420 PRINTTAB(5,17)CHR$(7)"Move ";M%"? "CHR$(8);
 430 REPEAT
 440   K$=GET$:*FX15,1
 450   UNTIL(K$>="1"ANDK$<="9")OR(K$="0"ANDNOTSWA
P%)
 460 PRINTK$
 470 IFK$="0" PROCSWAP ELSE PROCROTATE
 480 ENDPROC
 490 REM*****************************
 500 DEFPROCSWAP
 510 PRINTTAB(5,18)"Swap?";:L1%=FNGETLET
 520 PRINT" With?";:L2%=FNGETLET
 530 IFL1%=L2% M%=M%-1:GOTO580
 540 FORI%=0TO15
 550   IFB%(I%)=L1% B%(I%)=L2% ELSEIFB%(I%)=L2% B
%(I%)=L1%
 560   NEXT
 570 SWAP%=TRUE
 580 PRINTTAB(5,18)SPC(14)
 590 ENDPROC
 600 REM*****************************
 610 DEFFNGETLET
 620 REPEAT
 630   K$=GET$
 640   UNTILK$>="A"ANDK$<="P"
 650 PRINTK$;
 660 =ASCK$-ASC"A"
 670 REM*****************************
 680 DEFPROCROTATE
```

```
  690 I%=ASCK$-ASC"1":I%=I%+I%DIV3
  700 T%=B%(I%):B%(I%)=B%(I%+4):B%(I%+4)=B%(I%+5):
B%(I%+5)=B%(I%+1):B%(I%+1)=T%
  710 ENDPROC
  720 REM******************************
  730 DEFPROCPBOARD
  740 COLOUR131:COLOUR1
  750 PRINTTAB(0,5)
  760 FORI%=0TO15
  770    IFI%MOD4=0 PRINTTAB(6,VPOS);
  780    PRINTCHR$(ASC"A"+B%(I%))CHR$(9);
  790    IFI%MOD4=3 PRINT'
  800    NEXT
  810 FORI%=0TO4
  820    COLOUR130
  830    PRINTTAB(5,5+I%*2)SPC(9);
  840    PRINTTAB(5+I%*2,5);
  850    FORJ%=0TO8
  860      VDU32,10,8
  870      NEXT
  880    NEXT
  890 PRINTTAB(0,7)"123X"'"456X"'"789X"'"XXXX"
  900 PRINTTAB(15,7)"ABCD"TAB(15,8)"EFGH"TAB(15,9)
"IJKL"TAB(15,10)"MNOP"
  910 ENDPROC
```

# Quiz

## Rules

This is a quick-fire multiple-choice six-round general knowledge quiz. The faster you answer the questions, the more points you get. If you guess wrongly you lose 200 points for the current question but are allowed to guess again. If you do not answer before the time for the current question is up, you will be told the correct answer and moved onto the next question.

At the end of each set of six questions you will be told your rating.

## Display

The high-score and your current score are shown in a banner at the top of the screen.

Below the banner is printed the points-counter for the current question. This counter decreases rapidly with time.

The questions are printed in white below the points-counter.

The possible answers are shown in magenta below the question and are numbered from 1 to 5.

The bottom part of the display is used for various informative messages.

## Operation

You will be given five possible answers to each question. You must select one by pressing a key from '1' to '5'. If you give a wrong answer you may then press another key.

## Program

The program selects a random set of six questions from the list. It will not repeat a question until all the questions have been used at least once.

| *Section/Variables* | *Function* |
|---|---|
| Main routine | Initialize data, main game loop, game over |
| Q$ | Questions text |
| A$ | Answers text |
| CA% | Correct answer |
| U% | Used-once flag |

| | |
|---|---|
| QI% | Question index |
| HSC% | High-score |
| CQI% | Current-question index |
| R | Input key |
| SC% | Current score |
| Q% | Questions number |
| QL% | Questions left (not yet asked) in list |
| C% | Correct-answer flag |
| K$ | Input answer |
| FN NO | Return 'yes' or 'no' answer |
| K$ | Input key |
| PROC VALIDATE | Check answer for correctness |
| PROC CORRECT | Handle correct answer |
| PROC WRONG | Handle wrong answer |
| PROC PDM | Print double-height message |
| M$ | Actual message |
| PROC CLEARAQ | Clear all-questions-used flags |
| PROC DELAY | Delay for required time |
| L | Limit value of delay |
| PROC PSCORES | Print current scores |

## Suggestions

Add loads of your own questions and also get your friends to add their own until you cannot recall all the answers directly from their numbers. The more people who add questions about their own interests, the more educational and enjoyable the program becomes.
(My best score: 5520)

## The Listing

```
   10 *FX4,1
   20 DIMQ$(99),A$(4,99),CA%(99),U%(99)
   30 QI%=-1
   40 REPEAT
   50   QI%=QI%+1
   60   READQ$(QI%),CA%(QI%)
   70   FORI%=0TO4
   80     READA$(I%,QI%)
   90     NEXT
  100   UNTILCA%(QI%)<0
  110 HSC%=0
  120 PROCCLEARAQ:CQI%=RND(QI%)-1
  130 REPEAT
  140   MODE7:VDU23;8202;0;0;0;
  150   PRINTTAB(8,12)"Hit any key when ready":R=G
```

```
ET
  160    SC%=0
  170    FORQ%=1TO6
  180      CLS
  190      PROCPSCORES
  200      CQI%=(CQI%+RND(QI%))MODQI%
  210      REPEAT
  220        CQI%=(CQI%+1)MODQI%
  230        UNTILNOTU%(CQI%)
  240      QL%=QL%-1:IFQL%=0 PROCCLEARAQ
  250      U%(CQI%)=TRUE
  260      PRINTTAB(0,7)"Question ";Q%'Q$(CQI%)"?"'
  270      FORI%=0TO4
  280        PRINTCHR$(133);I%+1;")"A$(I%,CQI%)
  290        NEXT
  300      *FX15,1
  310      C%=FALSE
  320      TIME=0
  330      REPEAT
  340        K$=INKEY$(0)
  350        IFK$>="1"ANDK$<="5" PROCVALIDATE
  360        PRINTTAB(14,4)CHR$(130)"Points:";1000-
TIME;"   "
  370        UNTILTIME>999ORC%
  380      PRINTTAB(14,4)"                    "
  390      IFNOTC% PROCPDM(CHR$(136)+CHR$(129)+"TOO
 LATE"):PRINT'"The correct answer was ";CA%(CQI%);
")"A$(CA%(CQI%)-1,CQI%)
  400      PROCDELAY(200)
  410      NEXT
  420    IFSC%>HSC% HSC%=SC%:PROCPSCORES
  430    PRINTTAB(11,4)CHR$(136)CHR$(130)"Rating:";
  440    IFSC%>4999 PRINT"Genius" ELSEIFSC%>3999 PR
INT"Brainy" ELSEIFSC%>2999 PRINT"Average" ELSEIFSC
%>1999 PRINT"Pretty bad" ELSEIFSC%>999 PRINT"Turke
y" ELSE PRINT"Vegetable"
  450    PROCPDM("Another game?"):*FX15,1
  460    UNTILFNNO
  470 MODE7
  480 END
  490 REM*******************************
  500 DEFFNNO
  510 REPEAT
  520    K$=GET$
  530    UNTILK$="Y"ORK$="N"
  540 =K$="N"
  550 REM*******************************
  560 DEFPROCVALIDATE
  570 IFASC(K$)=CA%(CQI%)+ASC"0" PROCCORRECT ELSE
 PROCWRONG
  580 *FX15,1
  590 ENDPROC
  600 REM********************************
  610 DEFPROCCORRECT
  620 PROCPDM(CHR$(134)+"CORRECT")
  630 SOUND&11,-15,213,2:SC%=SC%+1000-TIME:PROCPSC
ORES:C%=TRUE
  640 ENDPROC
  650 REM********************************
  660 DEFPROCWRONG
  670 PROCPDM(CHR$(132)+"WRONG")
  680 SOUND&11,-15,5,2:TIME=TIME+142
  690 PROCDELAY(50)
  700 PROCPDM("         ")
  710 ENDPROC
  720 REM********************************
  730 DEFPROCPDM(M$)
  740 FORY%=19TO20
  750   PRINTTAB(20-LEN(M$)/2,Y%)CHR$(141)M$
  760   NEXT
  770 ENDPROC
  780 REM********************************
  790 DEFPROCCLEARAQ
  800 FORI%=0TOQI%
  810   U%(I%)=FALSE
  820   NEXT
  830 QL%=QI%
  840 ENDPROC
  850 REM********************************
  860 DEFPROCDELAY(L)
  870 T=TIME
  880 REPEAT
  890   UNTILTIME-T>L
  900 ENDPROC
  910 REM********************************
  920 DEFPROCPSCORES
  930 PRINTTAB(0,0)CHR$(7)CHR$(157)CHR$(134)'CHR$(
129)"  High score:";HSC%TAB(24,1)CHR$(131)"Score:"
SC%'CHR$(157)CHR$(134)
  940 ENDPROC
  950 REM********************************
  960 DATAWhich home computer does not use the 650
2 micro-processor,3,Apple II,Vic 20,Spectrum,BBC B
,Oric
  970 DATAWhich is nearest the sun,4,Earth,Jupiter
,Asteroid belt,Venus,Pluto
  980 DATA7 TIMES 12 MINUS 22 EQUALS,2,44,62,72,86
,68
  990 DATAIn 'STAR WARS' the light sabre was the w
eapon of,3,Han Solo,Imperial guards,Jedi Knights,L
ando Calrisian,Chewbacca
 1000 DATAShakespeare did not write,5,Hamlet,Macbe
```

```
th,Julius Caesar,Henry the Fifth,The Hobbit
 1010 DATANot a 'STAR WARS' character,1,Gythan,Luk
e Skywalker,Jabba the Hut,The Emperor,C3PO
 1020 DATAThe inventor of the telescope was,3,Erat
osthenes,Euclid,Galileo,Newton,Kepler
 1030 DATAR.A.M. stands for,2,Read all magazines,R
andom access memory,Route a monde,Right after meal
s,Ready and manoeuvrable
 1040 DATAWon the 1981 F.A.Cup,5,Q.P.R.,Man.Utd.,M
an.City,Liverpool,Spurs
 1050 DATASteve Ovett is associated with which spo
rt,4,Swimming,Snooker,Motor Racing,Athletics,Darts
 1060 DATAWho won the Jules Rimet World Cup outrig
ht,1,Brazil,West Germany,Argentina,England,Holland
 1070 DATAWorld professional snooker champion 1980
,2,Ray Reardon,Cliff Thorburn,Alex Higgins,Terry G
riffiths,Steve Davies
 1080 DATAWorld professional darts champion 1983,4
,John Lowe,Eric Bristow,Tony Brown,Keith Deller,Jo
cky Wilson
 1090 DATADavid Bryant is associated with which sp
ort,3,Football,Golf,Bowls,Show jumping,Squash
 1100 DATANot a micro-processor,2,6502,6522,6809,Z
80,8086
 1110 DATANot a chess term,4,Pawn,Castling,Promoti
on,Huffing,Pin
 1120 DATANot a chess grandmaster,1,Sharif,Karpov,
Miles,Spassky,Donner
 1130 DATAIn 'STAR WARS' who put up the bounty on
Han Solo,5,The Empire,Darth Vader,Lando Calrisian,
Chewbacca,Jabba the Hut
 1140 DATAWho plays the 'Fonz',4,Marlon Brando,Tom
 Baker,John Williams,Henry Winkler,Christopher Ree
ve
 1150 DATAWhich is on the east coast of America,1,
Maine,Oregon,California,Washington,Nevada
 1160 DATAWhich is not about space,5,Star Wars,Bla
kes Seven,Doctor Who,Buck Rogers,Logans Run
 1170 DATANot a 'Police' hit,2,Roxanne,Don't you w
ant me,Message in a bottle,Every breath you take,W
alking on the moon
 1180 DATANot a magician,3,Paul Daniels,Ali Bongo,
John Parsons,David Nixon,Tommy Cooper
 1190 DATANot a feature of the BBC micro-computer,
4,Multi-processor capability,Multi-channel sounds,
Analogue inputs,Pre-programmed sound effects,Full
size keyboard
 1200 DATAA 'Star Trek' slogan,1,Live long and pro
sper,The force be with you,Open all hours,Not a lo
t,Just like that
 1210 DATAA computer language,5,Humps,Heaps,Styx,D
```

```
rubs,Mumps
 1220 DATAWhere was the 'Industrial Revolution',3,
Holland,Italy,England,China,Germany
 1230 DATANot a computer manufacturer,1,NBC,IBM,IC
L,CDC,DEC
 1240 DATANot a county cricket club,2,Middlesex,Av
on,Somerset,Kent,Essex
 1250 DATANot a card game,4,Bridge,Baccara,Canasta
,Gothic,Whist
 1260 DATANot a character in 'The Hobbit',3,Bilbo,
Gandalf,Aragorn,Thorin,Smaug
 1270 DATAFrank Bruno is associated with which spo
rt,1,Boxing,Badminton,Squash,Judo,Rugby
 1280 DATA18 divided by 6 times 9 equals,3,54,24,2
7,39,15
 1290 DATAR.O.M. stands for,4,Right on man,Rancid
old meat,Rock on momma,Read only memory,Roll on Mo
nday
 1300 DATANot a T.V. news reader,2,Jan Leeming,Mic
hael Fish,Sandy Gaul,Moira Stewart,John Humphries
 1310 DATAOdd one out,4,Shot putt,Marathon,Archery
,Golf,Shooting
 1320 DATANot a fish,5,Herring,Mackerel,Salmon,Chu
b,Dolphin
 1330 DATAOdd one out,3,7,11,9,13,5
 1340 DATANot a car manufacturer,4,Ford,General Mo
tors,Toyota,Raleigh,Vauxhall
 1350 DATAOdd one out,1,Mauve,Blue,Yellow,Green,Or
ange
 1360 DATANot British,3,Michael Parkinson,Eric Mor
ecambe,Pamela Stephenson,Margaret Thatcher,Michael
 Caine
 1370 DATAQ,-1,A1,A2,A3,A4,A5
```

# Backgammon

## Rules

The 'game of kings' is played between two 15-piece armies on a 24 'point' board.

To decide who moves first, each player throws a die and the player with the highest die moves first. These dice are used by the first player on his first move.

The object is to move all your pieces around the board and off the end before your opponent does so with his army.

The dark pieces move anti-clockwise while the light pieces move clockwise.

The roll of two dice are used to move your pieces by the number of points on the dice. Each die is considered as a separate move and different pieces can be moved with either die; or the same piece can be moved twice. If a double is thrown, this counts as four moves of the value on either die.

If any 'point' has more than one piece of the same side on it, then the point is said to be 'made' or 'blocked' and the opponent is not allowed to land on this point with any of his men.

If a point has only one piece on it, and the opponent lands on that square, the lone piece is said to be 'hit' and is moved onto the 'bar' (the thick line dividing the two halves of the board) and must then restart its journey around the board from the beginning.

If a side has any pieces on the bar they must be moved back onto the board before any other pieces of that side can be moved.

When all pieces of one side have been moved into its 'inner' quadrant (the last quadrant on the army's journey) the process of 'bearing off' (taking the pieces off the board) begins.

You must move off the board with an exact throw if possible. If you cannot use the die exactly to bear off, you must use it up exactly with an ordinary move on the board. If this is not possible, you must bear off on the highest available point.

When one army has moved all its pieces off the board, it is awarded from one to three points depending on the nature of its win.

If the losing side has borne off any of its pieces, the winner gets one point for a 'standard' win. If the losing side has borne off no pieces but at least has all its pieces out of the winning side's inner quadrant, the winner gets two points for a 'gammon' win. If the losing side hasn't even got all of its pieces off the bar and out of the winning side's inner quadrant, the winner gets three points for a 'backgammon' win.

A gambling facility is provided in backgammon by means of a 'doubling-cube' but this has not been implemented in the program.

## Display

The board is drawn with green and red 'points' and blue and white 'pieces'. The points are lettered from 'B' to 'Y'. The computer's bar is labelled 'A' while your bar is labelled 'Z'.

The number of pieces on a point is shown by the corresponding number of circular counters. If more than eight pieces are on a point, the last counter has a flashing number super-imposed on top of it, to indicate the number of counters it actually represents.

Two dice faces are shown on the left hand side of the board. These are 'rolled' when required.

The moves by each player are printed below the board. They are in the format 'point-letter' followed by 'die value'.

## Operation

First the program asks which level it should play at. The higher the level the better the computer will play, but the more time it will take on its move. (On level four the program may take several minutes to decide how to best use a double!)

When asked for your move you must first press a letter from 'B' to 'Z' to indicate where you wish to move from. Then press a number from '1' to '6' to indicate the die value you wish to use. Then press <RETURN> to enter your move or <DELETE> if you have made a mistake in either the point or die.

Any illegal moves will be rejected and the point-letter and die-value cleared. The computer will detect any occurrence when either side cannot move and then proceed with the next player's roll.

## Program

The program selects its moves by examining the tree of possible moves to a certain fixed depth. Each final position is evaluated and the path to the best position chosen.

| *Section/Variables* | *Function* |
|---|---|
| Main routine | Initialize data, setup level, main game loop, game over |
| FC% | Foreground colour |
| DBC% | Dark background colour |
| LBC% | Light background colour |
| DPC% | Dark-piece colour |
| LPC% | Light-piece colour |
| B% | Current board position |
| DIE% | Dice values |

| | |
|---|---|
| BS% | Best-scores found in tree |
| CSC% | Computer's total score |
| HSC% | Human's total score |
| GO% | Game-over flag |
| CM% | Number of computer's men left |
| HM% | Number of human's men left |
| LEVEL% | Current level (maximum search depth in tree) |
| FM% | First-move flag |
| PROC CWIN | Handle computer's win |
| PROC HWIN | Handle human's win |
| PROC CMOVE | Handle computer's move |
| CG% | Can't-go flag |
| D% | Current depth in tree |
| BF% | Best-move 'from' point (found in tree search) |
| BT% | Best-move 'to' point (found in tree search) |
| BDN% | Best-move die index (found in tree search) |
| PROC PLY | Handle each ply of game tree |
| DN% | Die number |
| F% | Current 'from' point |
| T% | Current 'to' point |
| RF% | Restorable value of current 'from' point |
| RT% | Restorable value of current 'to' point |
| RB% | Restorable value of opponent's bar |
| PROC EVAL | Evaluate terminal position of tree |
| ADV% | Army advancement value |
| BLOT% | Blot penalty value |
| MADE% | Made-point bonus value |
| O% | Current point occupiers |
| PROC SETLM | Find rearmost man of each army |
| LCM% | Last (most backward) computer's man |
| LHM% | Last (most backward) human's man |
| TB% | Temporary board point storage |
| PROC HMOVE | Handle human's move |
| NLM% | No-legal-moves flag |
| FN GETSQ | Return code letter for required square |
| K$ | Input Key |
| FN GETDIE | Return value for required die |
| FN LEGALDIE | Check die value for legality |
| LD% | Legal-die flag |
| FD% | Found-die index |
| FN ALLDICEUSED | Check if all dice have been used |
| DI% | Die index |
| ADU% | All-dice-used flag |
| FN Empty | Check range of squares for emptiness |
| LL% | Lower limit of range |

| | |
|---|---|
| UL% | Upper limit of range |
| SIGN% | Side-to-move adjuster |
| BI% | Board index |
| EMP% | Empty flag |
| FN LARGERHMOVES | Check for any larger human moves |
| LL% | Lower limit for scan |
| D% | Die value |
| BI% | Board index |
| LM% | Larger-moves flag |
| FN LARGERCMOVES | Check for any larger computer moves |
| UL% | Upper limit for scan |
| PROC PBOARD | Print complete board |
| PROC PSQ | Print particular square |
| S% | Current point selected |
| UD$ | Up/down adjustment string |
| PROC PSQB | Print particular square's background triangle |
| C$ | Background character required |
| PROC PSQP | Print particular square's pieces |
| AV% | Actual number of pieces on current point |
| PROC ROLL | Roll dice |
| PROC PDICE | Print both dice |
| PROC PDIE | Print required die |
| DX% | X-coordinate specifier |
| DY% | Y-coordinate specifier |
| V% | Die value |
| PROC PMES | Print message at selected line and erase old message |
| M$ | Actual message |
| PROC DELAY | Delay for required time |
| L% | Limit value of delay |

## Suggestions

If you are a very strong backgammon player you could perhaps enhance the evaluation function to strengthen the program's play, eg by including knowledge of hit probabilities of blots, bearing-off efficiency, book openings.

The search is rather slow in BASIC and would benefit greatly from assembly language coding. It would then be possible to include much more detailed knowledge in the evaluation routine.

Knowledge of the doubling cube could be added for those serious players who require this feature.

```
   10 *FX4,1
   20 *FX229,1
   30 *FX225,0
   40 FC%=6:DBC%=1:LBC%=2:DPC%=4:LPC%=7
   50 VDU23,224,0,0,0,&18,&18,0,0,0,23,225,&60,&60
,0,0,0,0,6,6,23,226,&60,&60,0,&18,&18,0,6,6,23,227
,&66,&66,0,0,0,0,&66,&66,23,228,&66,&66,0,&18,&18,
0,&66,&66,23,229,&66,&66,0,&66,&66,0,&66,&66
   60 VDU23,230,0,&3C,&7E,&7E,&7E,&7E,&3C,0,23,231
,&FF,&FF,&FF,&FF,&FF,&FF,&FF,&FF,23,232,&7E,&7E,&7
E,&7E,&7E,&7E,&7E,&7E,23,233,&3C,&3C,&3C,&3C,&3C,&
3C,&3C,&3C,23,234,&18,&18,&18,&18,&18,&18,&18,&18
   70 DIMB%(25),DIE(3),BS%(5)
   80 CSC%=0:HSC%=0
   90 REPEAT
  100    RESTORE
  110    DATA0,2,0,0,0,0,-5,0,-3,0,0,0,5
  120    FORI%=0TO12
  130       READB%(I%):B%(25-I%)=-B%(I%)
  140       NEXT
  150    GO%=FALSE:CM%=15:HM%=15
  160    MODE2:VDU23;8202;0;0;0;
  170    PROCPBOARD
  180    PROCPMES("Level (1-4)?")
  190    REPEAT
  200       LEVEL%=GET-ASC"0"
  210       UNTILLEVEL%>0ANDLEVEL%<5
  220    PROCPMES("Rolling for 1st move")
  230    REPEAT
  240       PROCROLL
  250       UNTILDIE(2)=-99
  260    IFDIE(0)<DIE(1) PROCPMES("You move first!"
) ELSE PROCPMES("I move first")
  270    PROCDELAY(200)
  280    FM%=TRUE
  290    IFDIE(0)<DIE(1) PROCHMOVE
  300    REPEAT
  310       PROCCMOVE
  320       IFCM%>0PROCHMOVE
  330       UNTILCM%=0ORHM%=0
  340    COLOUR1
  350    PRINTTAB(0,27);
  360    IFCM%=0 PROCCWIN ELSE PROCHWIN
  370    COLOUR2
  380    IFS%=1PRINT"Standard stake" ELSEIFS%=2PRIN
T"Gammon stake" ELSEPRINT"Backgammon stake"
  390    COLOUR3
  400    PRINT"Me: ";CSC%"  You: ";HSC%
  410    PROCPMES("Another game?"):*FX15,1
  420    UNTILGET$="N"
  430 MODE7
  440 END
  450 REM*********************************
  460 DEFPROCCWIN
  470 PRINT"I win!"
  480 IFHM%<15 S%=1 ELSE IFLHM%<19 S%=2 ELSE S%=3
  490 CSC%=CSC%+S%
  500 ENDPROC
  510 REM*********************************
  520 DEFPROCHWIN
  530 PRINT"You win!"
  540 IFCM%<15 S%=1 ELSE IFLCM%>6 S%=2 ELSE S%=3
  550 HSC%=HSC%+S%
  560 ENDPROC
  570 REM*********************************
  580 DEFPROCCMOVE
  590 PROCPMES("My move: ")
  600 IFNOTFM% PROCROLL
  610 CG%=FALSE
  620 REPEAT
  630    D%=0
  640    PROCPLY
  650    IFBS%(1)=-9999 CG%=TRUE:SOUND1,-15,5,9:PRO
CPMES("I can't move!!"):PROCDELAY(99):GOTO740
  660    COLOUR7
  670    PRINTTAB(8,30)CHR$(BF%+ASC"A");DIE(BDN%)
  680    B%(BF%)=B%(BF%)-1:SOUND1,-15,117,3:PROCPSQ
(BF%)
  690    IFBT%>24 CM%=CM%-1:GOTO0720
  700    IFB%(BT%)=-1 B%(25)=B%(25)-1:B%(BT%)=0:SOU
ND1,-15,197,4:PROCPSQ(25)
  710    B%(BT%)=B%(BT%)+1:SOUND1,-15,89,5:PROCPSQ(
BT%)
  720    DIE(BDN%)=-DIE(BDN%)
  730    PROCDELAY(99)
  740    UNTILCM%=0ORCG%ORFNALLDICEUSED
  750 FM%=FALSE
  760 ENDPROC
  770 REM*********************************
  780 DEFPROCPLY
  790 LOCALDN%,F%,T%,RF%,RT%,RB%
  800 D%=D%+1:BS%(D%)=-9999
  810 FORDN%=0TO3
  820    IFDIE(DN%)<0THEN1020
  830    DIE(DN%)=-DIE(DN%)
  840    FORF%=0TO24
  850       IFB%(F%)<1THEN990
  860       T%=F%-DIE(DN%):IFT%>24IFNOTFNEMPTY(1,18,
1)THEN980
  870       IFT%>25IFFNLARGERCMOVES(F%-1,-DIE(DN%))T
```

```
HEN980
  880       IFT%<25IFB%(T%)<-1THEN980
  890       RF%=B%(F%):B%(F%)=B%(F%)-1
  900       IFT%>24THEN940
  910       RT%=B%(T%):RB%=B%(25)
  920       IFRT%=-1 B%(25)=B%(25)-1:B%(T%)=0
  930       B%(T%)=B%(T%)+1
  940       IFD%=LEVEL%ORFNALLDICEUSED PROCEVAL ELSE
 PROCPLY
  950       B%(F%)=RF%
  960       IFT%<25B%(T%)=RT%:B%(25)=RB%
  970       IFBS%(D%+1)>BS%(D%) BS%(D%)=BS%(D%+1):IF
D%=1 BF%=F%:BT%=T%:BDN%=DN%
  980       IFB%(0)>0 F%=25
  990       NEXT
 1000     DIE(DN%)=-DIE(DN%)
 1010     IFDIE(2)>-99 DN%=4
 1020     NEXT
 1030 IFBS%(D%)=-9999 IFD%>1 PROCEVAL:BS%(D%)=BS%(
D%+1)
 1040 D%=D%-1
 1050 ENDPROC
 1060 REM*******************************
 1070 DEFPROCEVAL
 1080 ADV%=0:BLOT%=0:MADE%=0
 1090 PROCSETLM
 1100 FORI%=1TO24
 1110    O%=B%(I%):IFO%=0THEN1190
 1120    IFO%<0THEN1170
 1130    ADV%=ADV%+O%*(I%-30+(I%-19)*(I%>19))
 1140    IFO%>1MADE%=MADE%+I% ELSEIFI%<LHM%BLOT%=BL
OT%-I%
 1150    IFB%(25)<0IFI%>18IFO%>1MADE%=MADE%+1
 1160    GOTO1190
 1170    ADV%=ADV%+O%*(25-I%+(6-I%)*(I%<6))
 1180    IFO%<-1MADE%=MADE%+I%-25 ELSEIFI%>LCM%BLOT
%=BLOT%+(25-I%)DIV8
 1190    NEXT
 1200 IFLHM%<LCM% BLOT%=0:MADE%=0
 1210 BS%(D%+1)=ADV%*5+3*(LCM%+LHM%-25+(LCM%<19)-(
LHM%>6))-B%(LCM%)-B%(LHM%)+BLOT%*6+MADE%*7-30*B%(2
5)-60*(B%(25)<-1)+RND(2)
 1220 ENDPROC
 1230 REM*******************************
 1240 DEFPROCSETLM
 1250 LCM%=-1:TB%=B%(24):B%(24)=1
 1260 REPEAT
 1270    LCM%=LCM%+1
 1280    UNTILB%(LCM%)>0
 1290 B%(24)=TB%
 1300 LHM%=26:TB%=B%(1):B%(1)=-1
 1310 REPEAT
 1320    LHM%=LHM%-1
 1330    UNTILB%(LHM%)<0
 1340 B%(1)=TB%
 1350 ENDPROC
 1360 REM*******************************
 1370 DEFPROCHMOVE
 1380 PROCPMES("Your  move?")
 1390 IFNOTFM% PROCROLL
 1400 REPEAT
 1410    COLOUR7
 1420    NLM%=TRUE
 1430    FORDN%=0TO3
 1440       IFDIE(DN%)<0THEN1520
 1450       FORF%=25TO1STEP-1
 1460          IFB%(F%)>-1OR(B%(25)<0ANDF%<>25)THEN15
10
 1470          T%=F%-DIE(DN%):IFT%<1IFNOTFNEMPTY(7,24
,-1)THEN1510
 1480          IFT%<0IFFNLARGERHMOVES(F%+1,DIE(DN%))T
HEN1510
 1490          IFT%>0IFB%(T%)>1THEN1510
 1500          NLM%=FALSE:F%=0:DN%=4
 1510          NEXT
 1520       NEXT
 1530    IFNLM% SOUND1,-15,5,9:PROCPMES("You can't
move!!"):PROCDELAY(99):GOTO1730
 1540    PROCSETLM
 1550    REPEAT
 1560       COLOUR7
 1570       PRINTTAB(10,30)"   ";
 1580       *FX15,1
 1590       F%=FNGETSQ:PRINTTAB(10,30)K$;:IFB%(F%)>=
0OR((B%(25)<0)AND(F%<>25))THEN1760
 1600       D%=FNGETDIE:PRINTK$;:IFNOTFNLEGALDIE THE
N1760
 1610       T%=F%-D%:IFT%<1IFNOTFNEMPTY(7,24,-1) THE
N1760
 1620       IFT%<0IFFNLARGERHMOVES(F%+1,D%) THEN1760
 1630       IFT%>0IFB%(T%)>1THEN1760
 1640       REPEAT
 1650          K$=GET$
 1660          UNTILK$=CHR$(127)ORK$=CHR$(13)
 1670       UNTILK$=CHR$(13)
 1680    B%(F%)=B%(F%)+1:SOUND1,-15,117,3:PROCPSQ(F
%)
 1690    IFT%<1 HM%=HM%-1:GOTO1720
 1700    IFB%(T%)=1 B%(0)=B%(0)+1:B%(T%)=0:SOUND1,-
15,197,4:PROCPSQ(0)
 1710    B%(T%)=B%(T%)-1:SOUND1,-15,89,5:PROCPSQ(T%
)
```

```
1720    DIE(FD%)=-99
1730    UNTILHM%=0ORNLM%ORFNALLDICEUSED
1740 FM%=FALSE
1750 ENDPROC
1760 COLOUR1
1770 PRINTTAB(0,31)"*ILLEGAL*";
1780 SOUND1,-15,5,9:PROCDELAY(60)
1790 PRINTTAB(0,31)SPC(9);
1800 GOTO1560
1810 REM********************************
1820 DEFFNGETSQ
1830 REPEAT
1840    K$=GET$
1850    UNTILK$>="A"ANDK$<="Z"
1860 =ASCK$-ASC"A"
1870 REM********************************
1880 DEFFNGETDIE
1890 REPEAT
1900    K$=GET$
1910    UNTILK$>="1"ANDK$<="6"
1920 =ASCK$-ASC"0"
1930 REM********************************
1940 DEFFNLEGALDIE
1950 LOCALLD%
1960 LD%=FALSE
1970 FORI%=0TO3
1980    IFDIE(I%)=D% FD%=I%:LD%=TRUE
1990    NEXT
2000 =LD%
2010 REM********************************
2020 DEFFNALLDICEUSED
2030 LOCALDI%,ADU%
2040 ADU%=TRUE
2050 FORDI%=0TO3
2060    IFDIE(DI%)>0 ADU%=FALSE:DI%=4
2070    NEXT
2080 =ADU%
2090 REM********************************
2100 DEFFNEMPTY(LL%,UL%,SIGN%)
2110 LOCALBI%,EMP%
2120 EMP%=TRUE
2130 FORBI%=LL%TOUL%
2140    IFB%(BI%)*SIGN%>0 EMP%=FALSE:BI%=25
2150    NEXT
2160 =EMP%
2170 REM********************************
2180 DEFFNLARGERHMOVES(LL%,D%)
2190 LOCALBI%,LM%
2200 LM%=FALSE
2210 FORBI%=6TOLL%STEP-1
2220    IFB%(BI%)>-1THEN2240
```

```
 2230    IFD%>=BI% LM%=TRUE ELSEIFB%(BI%-D%)<2 LM%=
TRUE
 2240    NEXT
 2250 =LM%
 2260 REM******************************
 2270 DEFFNLARGERCMOVES(UL%,D%)
 2280 LOCALBI%,LM%
 2290 LM%=FALSE
 2300 FORBI%=19TOUL%
 2310    IFB%(BI%)<1THEN2330
 2320    IFD%>=25-BI% LM%=TRUE ELSEIFB%(BI%+D%)>-2
LM%=TRUE
 2330    NEXT
 2340 =LM%
 2350 REM******************************
 2360 DEFPROCPBOARD
 2370 COLOURFC%
 2380 FORI%=0TO16
 2390    PRINTTAB(I%,2)CHR$(231)TAB(I%,24)CHR$(231)
;
 2400    NEXT
 2410 FORI%=3TO23
 2420    PRINTTAB(0,I%)CHR$(231)TAB(7,I%)CHR$(231)C
HR$(231)CHR$(231)TAB(16,I%)CHR$(231);
 2430    NEXT
 2440 FORI%=0TO25
 2450    PROCPSQ(I%)
 2460    NEXT
 2470 ENDPROC
 2480 REM******************************
 2490 DEFPROCPSQ(S%)
 2500 IFS%<=12UD$=CHR$(10)ELSEUD$=CHR$(11)
 2510 PROCPSQB(S%):IFB%(S%)<>0PROCPSQP(S%)
 2520 ENDPROC
 2530 REM******************************
 2540 DEFPROCPSQB(S%)
 2550 COLOUR7
 2560 PRINTTAB(S%-(25-S%-S%)*(S%>12)-3*(S%>6ANDS%<
19)-8*(S%=0ORS%=25),1-24*(S%>12))CHR$(ASC"A"+S%)UD
$UD$;
 2570 IFS%=0ORS%=25COLOURFC%ELSEIFS%/2=S%DIV2COLOU
RDBC%ELSECOLOURLBC%
 2580 FORJ%=0TO7
 2590    IFS%=0ORS%=25C$=CHR$(231)ELSEC$=CHR$(231+J
%DIV2)
 2600    PRINTCHR$(127)C$UD$;
 2610    NEXT
 2620 ENDPROC
 2630 REM******************************
 2640 DEFPROCPSQP(S%)
 2650 PRINTTAB(S%-(25-S%-S%)*(S%>12)-3*(S%>6ANDS%<
19)-8*(S%=0ORS%=25),2-22*(S%>12));
 2660 VDU5
 2670 MOVE64*POS,32*(32-VPOS)-4
 2680 IFB%(S%)>0GCOL0,LPC% ELSEGCOL0,DPC%
 2690 J%=1:AV%=ABS(B%(S%))
 2700 REPEAT
 2710   PRINTUD$CHR$(230)CHR$(8);
 2720   J%=J%+1
 2730   UNTILJ%>AV%ORJ%>8
 2740 IFJ%<=AV% GCOL0,10:PRINT;AV%-7
 2750 VDU4
 2760 ENDPROC
 2770 REM******************************
 2780 DEFPROCROLL
 2790 COLOUR7:GCOL0,7
 2800 FORI%=0TO9+RND(9)
 2810   DIE(0)=RND(6):DIE(1)=RND(6):SOUND3,-2,20*D
IE(0),1:PROCPDICE
 2820   NEXT
 2830 DIE(2)=-99:DIE(3)=-99
 2840 IFDIE(0)=DIE(1) DIE(2)=DIE(0):DIE(3)=DIE(1)
 2850 ENDPROC
 2860 REM******************************
 2870 DEFPROCPDICE
 2880 PROCPDIE(2,13,DIE(0)):PROCPDIE(5,13,DIE(1))
 2890 ENDPROC
 2900 REM******************************
 2910 DEFPROCPDIE(DX%,DY%,V%)
 2920 MOVEDX%*64-8,(32-DY%)*32+8:PLOT1,72,0:PLOT1,
0,-52:PLOT1,-72,0:PLOT1,0,52
 2930 PRINTTAB(DX%,DY%)CHR$(223+V%)
 2940 ENDPROC
 2950 REM******************************
 2960 DEFPROCPMES(M$)
 2970 COLOUR7
 2980 PRINTTAB(0,30)M$SPC(19);
 2990 ENDPROC
 3000 REM******************************
 3010 DEFPROCDELAY(L%)
 3020 TIME=0
 3030 REPEAT
 3040   UNTILTIME>L%
 3050 ENDPROC
```

# Awari

## Rules

This is an ancient African board game with 36 pebbles and 14 holes. The object is to get as many pebbles into your own 'home' hole as possible.

Each player has six holes on his side of the board and his 'home' hole at his right-hand end of the board.

The game starts with every hole on both sides of the board having three pebbles in it, thus:

```
    3   3   3   3   3   3
0                           0←Your home hole
    3   3   3   3   3   3
```

To move you take all the pebbles from one of the non-empty holes on your side of the board and sow the pebbles, one at a time, in an anti-clockwise direction, into the adjacent holes until all pebbles have been sown.
For example, if you move first from the leftmost hole on your side of the board the position would end up thus:

```
    3   3   3   3   3   3
0                           0
    0   4   4   4   3   3
```

If the last pebble is sown into in your own home, then you may take one (but only one) extra move.

Also, if the last pebble is sown into an empty hole, and the *opposite* hole on the other side of the board is not empty, then you capture all the pieces in the opposite hole and the last seed you sowed, and move them into your own 'home'.

The game is over when all the holes on either side of the board are empty. The winner is then the player with the most pebbles in his 'home' hole.

## Display

The display is divided into three sections.

The top of the display shows the running score between you and the computer in games and pebbles scored.

The middle of the display shows the board.

The computer's holes are at the top of the board with its 'home' at the left end of the board.

Your holes are at the bottom of the board with your 'home' at the right end of the board.

The holes are lettered 'A'–'F' on your side of the board and 'G'–'L' on the computer's side.

Your holes are printed in yellow while the computer's holes are printed in red.

The bottom of the display shows the moves by either side and various other queries, as needed.

## Operation

First the program asks which level you require it to play at. The higher the level chosen, the better it will play, but the longer it will take to make its moves.

The computer will move first in the first game and then alternate who moves first in all successive games.

When it is your turn to move you must type a letter from 'A' to 'F' to choose one of the non-empty holes, on your side of the board, to move from. All illegal moves are rejected with an error message and a low-pitched beep.

## Program

The program selects its moves by using a full-width minimax search algorithm to fixed depth, with alpha-beta pruning. This is the same algorithm as used in most modern chess programs. It can be adapted to work with any strategy game eg draughts, othello, gomoku.

| *Section/Variables* | *Function* |
|---|---|
| Main routine | Initialize data, setup level, main game loop, game over |
| MB% | Main board position |
| BS% | Best-score found at each ply in the tree |
| B% | Temporary boards used in the tree |
| NN% | Not-null-move flags in tree |
| MP% | Move-pointers in tree |
| PG% | Number of program's won games |
| OG% | Number of opponent's won games |
| PP% | Number of program's won points |
| OP% | Number of opponent's won points |
| OF% | Opponent-moves-first flag |
| D% | Current search depth in tree |
| MD% | Maximum search depth in tree (related to current level) |
| GO% | Game-over flag |
| GOM$ | Game-over message |

| | |
|---|---|
| PROC MANMOVE | Handle human player's move |
| N% | Number of current move (extra move flag) |
| M% | Current move hole number |
| PROC ILLEG | Indicate illegal move attempted |
| X% | Current cursor position X-coordinate |
| Y% | Current cursor position Y-coordinate |
| PROC COMPMOVE | Handle computer's move |
| N% | Number of current move (extra move flag) |
| PROC ODD | Search odd plies of the game tree |
| PROC EVEN | Search even plies of the game tree |
| PROC UPDATE | Update complete move |
| H% | Home hole of current side to move |
| M% | Current hole to move from |
| PROC MOVE | Move pebbles around main board |
| M% | Current hole to move from |
| N% | Number of pebbles in current hole |
| S% | Current hole to sow next pebble into |
| FN EVAL | Evaluate terminal position in same tree |
| E% | Evaluation of current position |
| FN EMPTY | Test range of squares for emptiness |
| L% | Lower limit of range |
| U% | Upper limit of range |
| PROC PRB | Print whole board |
| PROC PRH | Print one hole |
| I% | Current hole index |
| PROC PRCENTRAL | Print centralised message |
| Y% | Message Y-coordinate |
| S$ | Actual message |
| PROC PRSCORE | Print running scores |

## Suggestions

The board display could be enhanced to a full-blown graphics board with holes showing individual pebbles within them. Then each time a move is made the pebbles could be seen to be sown one at a time into the following holes.

Level 5 is already quite slow because of the limited speed of BASIC. Using assembly language for the search procedures would allow much higher levels of skill to be achieved by the program.

As stated earlier, the algorithm in this program can be adapted for any strategy game. Perhaps if you study it you may then like to try to implement it within a suite of different strategy games of your own.

## The Listing

```
   10 *FX4,1
   20 DIMMB%(13),BS%(9),B%(9,13),NN%(9),MP%(9)
   30 PG%=0:OG%=0:PP%=0:OP%=0:OF%=TRUE
   40 D%=2:BS%(1)=-999:BS%(2)=999
   50 REPEAT
   60    MODE1
   70    FORI%=0TO13:MB%(I%)=3
   80       NEXT
   90    MB%(0)=0:MB%(7)=0
  100    PROCPRSCORE:PROCPRB
  110    PROCPRCENTRAL(19,"Level (1-5)?")
  120    REPEAT
  130       MD%=GET-ASC"0"+2
  140       UNTILMD%>2ANDMD%<8
  150    OF%=NOTOF%
  160    IFOF% PROCPRCENTRAL(19,"Now you can move f
irst!"):PROCMANMOVE
  170    REPEAT
  180       PROCCOMPMOVE
  190       IFNOTGO% PROCMANMOVE
  200       UNTILGO%
  210    IFMB%(7)>MB%(0) GOM$="I win!":PG%=PG%+1 EL
SEIFMB%(7)<MB%(0) GOM$="You win!":OG%=OG%+1 ELSE G
OM$="A draw"
  220    PP%=PP%+MB%(7):OP%=OP%+MB%(0)
  230    PROCPRCENTRAL(29,GOM$)
  240    PROCPRSCORE
  250    PROCPRCENTRAL(31,"Another game?"):*FX15,1
  260    UNTILGET$="N"
  270 MODE7
  280 END
  290 REM*******************************
  300 DEFPROCMANMOVE
  310 N%=1
  320 REPEAT
  330    IFN%=1 PRINTTAB(14,22); ELSE PRINTTAB(14,2
3)"and again"TAB(14,24);
  340    PRINT"Your move?";:*FX15,1
  350    REPEAT
  360       REPEAT
  370          M%=GET-ASC"A"+8
  380          IFM%<8ORM%>13 PROCILLEG
  390          UNTILM%>7ANDM%<14
  400       IFMB%(M%)<1 PROCILLEG
  410       UNTILMB%(M%)>0
  420    PRINTCHR$(ASC"A"+M%-8)CHR$(7)
  430    PROCUPDATE(0,M%)
  440    PROCPRB
  450    N%=N%+1
  460    GO%=FNEMPTY(1,6)ORFNEMPTY(8,13)
  470    UNTILS%<>0ORN%>20RGO%
  480 ENDPROC
  490 REM*******************************
  500 DEFPROCILLEG
  510 X%=POS:Y%=VPOS
  520 PRINTTAB(14,26)"*ILLEGAL*";
  530 SOUND1,-15,5,9
  540 TIME=0
  550 REPEAT
  560    UNTILTIME>50
  570 PRINTTAB(14,26)SPC(9)TAB(X%,Y%);
  580 ENDPROC
  590 REM*******************************
  600 DEFPROCCOMPMOVE
  610 VDU28,0,31,39,16,12,26:PRINTTAB(14,19)"Let m
e think";
  620 NN%(1)=1:NN%(2)=1:S%=1
  630 N%=1
  640 REPEAT
  650    IFN%=2 NN%(2)=0:PRINTTAB(14,20)"and again"
;
  660    PROCODD
  670    PRINTTAB(14,21);:IFN%=1 PRINTTAB(14,19);
  680    PRINT"My move is "CHR$(ASC"F"+BM%)CHR$(7)
  690    PROCUPDATE(7,BM%)
  700    PROCPRB
  710    N%=N%+1
  720    GO%=FNEMPTY(1,6)ORFNEMPTY(8,13)
  730    UNTILS%<>7ORN%>20RGO%
  740 ENDPROC
  750 REM*******************************
  760 DEFPROCODD
  770 D%=D%+1:MP%(D%)=M%:BS%(D%)=BS%(D%-2)
  780 FORI%=0TO13:B%(D%,I%)=MB%(I%)
  790    NEXT
  800 IFS%=0 IFNN%(D%-2) M%=0:NN%(D%)=0:GOTO860
  810 IFFNEMPTY(1,6)ORFNEMPTY(8,13)ORD%>MD% BS%(D%
)=FNEVAL:GOTO930
  820 NN%(D%)=1:M%=6
  830 IFMB%(M%)=0 THEN920
  840 PROCMOVE(M%)
  850 IFS%<>7 IFS%<>0 IFMB%(S%)=1 IFMB%(14-S%) MB%
(7)=MB%(7)+1+MB%(14-S%):MB%(S%)=0:MB%(14-S%)=0
  860 PROCEVEN
  870 FORI%=0TO13:MB%(I%)=B%(D%,I%)
  880    NEXT
  890 IFBS%(D%+1)<=BS%(D%) THEN920
  900 BS%(D%)=BS%(D%+1):IFBS%(D%)>=BS%(D%-1) M%=0
  910 IFD%=3 BM%=M%
  920 M%=M%-1:IFM%>0THEN830
```

```
 930 M%=MP%(D%):D%=D%-1
 940 ENDPROC
 950 REM*******************************
 960 DEFPROCEVEN
 970 D%=D%+1:MP%(D%)=M%:BS%(D%)=BS%(D%-2)
 980 FORI%=0TO13:B%(D%,I%)=MB%(I%)
 990   NEXT
1000 IFS%=7 IFNN%(D%-2) M%=7:NN%(D%)=0:GOTO1060
1010 IFFNEMPTY(8,13)ORFNEMPTY(1,6)ORD%>MD% BS%(D%
)=FNEVAL:GOTO1110
1020 NN%(D%)=1:M%=13
1030 IFMB%(M%)=0THEN1100
1040 PROCMOVE(M%)
1050 IFS%<>0 IFS%<>7 IFMB%(S%)=1 IFMB%(14-S%) MB%
(0)=MB%(0)+1+MB%(14-S%):MB%(S%)=0:MB%(14-S%)=0
1060 PROCODD
1070 FORI%=0TO13:MB%(I%)=B%(D%,I%)
1080   NEXT
1090 IFBS%(D%+1)<BS%(D%) BS%(D%)=BS%(D%+1):IFBS%(
D%)<=BS%(D%-1) M%=7
1100 M%=M%-1:IFM%>7THEN1030
1110 M%=MP%(D%):D%=D%-1
1120 ENDPROC
1130 REM*******************************
1140 DEFPROCUPDATE(H%,M%)
1150 PROCMOVE(M%)
1160 IFS%<>0 IFS%<>7 IFMB%(S%)=1 IFMB%(14-S%) MB%
(H%)=MB%(H%)+1+MB%(14-S%):MB%(S%)=0:MB%(14-S%)=0
1170 ENDPROC
1180 REM*******************************
1190 DEFPROCMOVE(M%)
1200 LOCALN%,I%
1210 N%=MB%(M%):MB%(M%)=0:S%=M%
1220 FORI%=1TON%:S%=S%+1:IFS%=14 S%=0
1230   MB%(S%)=MB%(S%)+1
1240   NEXT
1250 ENDPROC
1260 REM*******************************
1270 DEFFNEVAL
1280 LOCALE%
1290 E%=(MB%(7)-MB%(0))*4+RND(3)
1300 IFMB%(7)>18 :=E%+99 ELSEIFMB%(0)>18 :=E%-99
1310 IFFNEMPTY(1,6)ORFNEMPTY(8,13) :=E%*16
1320 =E%
1330 REM*******************************
1340 DEFFNEMPTY(L%,U%)
1350 FORI%=L%TOU%:IFMB%(I%) I%=98
1360   NEXT
1370 =I%<90
1380 REM*******************************
1390 DEFPROCPRB
```

```
 1400 MOVE6*32,22.5*32:PLOT1,26*32,0:PLOT1,0,-4*32
:PLOT1,-26*32,0:PLOT1,0,4*32
 1410 COLOUR3
 1420 PRINTTAB(9,8);
 1430 FORI%=0TO5
 1440   PRINT"  "CHR$(ASC"L"-I%);
 1450   NEXT
 1460 COLOUR1
 1470 PRINT'
 1480 PRINTTAB(9,VPOS);
 1490 FORI%=6TO1STEP-1:PROCPRH(I%)
 1500   NEXT
 1510 PRINT
 1520 PRINTTAB(6,VPOS);:PROCPRH(7):PRINTTAB(27-(MB
%(0)>9),VPOS);:COLOUR2:PROCPRH(0):PRINT" "
 1530 PRINTTAB(9,VPOS);
 1540 FORI%=8TO13:PROCPRH(I%)
 1550   NEXT
 1560 COLOUR3
 1570 PRINT'
 1580 PRINTTAB(9);
 1590 FORI%=0TO5
 1600   PRINT"  "CHR$(ASC"A"+I%);
 1610   NEXT
 1620 ENDPROC
 1630 REM********************************
 1640 DEFPROCPRH(I%)
 1650 VDU9:IFMB%(I%)<10 PRINT" ";
 1660 PRINT;MB%(I%);
 1670 ENDPROC
 1680 REM********************************
 1690 DEFPROCPRCENTRAL(Y%,S$)
 1700 PRINTTAB(20-(LEN(S$)/2),Y%)S$;
 1710 ENDPROC
 1720 REM********************************
 1730 DEFPROCPRSCORE
 1740 @%=&01000408
 1750 PROCPRCENTRAL(1,"SCORE"):PROCPRCENTRAL(2,"Pr
ogram Opponent")
 1760 PRINT'"Games",PG%,OG%'"Pebbles",PP%,OP%
 1770 ENDPROC
```

# Queens

## Rules

The 'eight-queens' puzzle, a classic, involves placing eight queens on a chess board such that no queen is attacking any other.

## Display

The display shows a red and yellow chess board and the eight queens 'dancing' – being shuffled around by the program until it finds the next solution.

When the program finds a solution it displays the number of positions it has examined so far.

## Operation

The program searches until it finds a solution, then pauses so you can see the solution. You can then press any key to make it continue its search for the next solution.

Holding any key down, while the program is searching, will cause it to slow down its search to a speed where you can see it checking each step.

The program produces beeps at each level in the tree to indicate how deep it is looking. The higher-pitched the note, the deeper into the tree it is.

## Program

The program searches the tree of possible positions, checking at each stage if the solution is still possible. If it is, then it proceeds to place the next queen. If it is not, it backtracks to the previous queen to find its next valid placing. This continues until all eight queens have been placed.

This program demonstrates clearly the method of tree searching used in most modern chess and other strategy games programs.

The program also demonstrates how easy it is to create graphics drawings using the user-definable characters available on the BBC micro.

| *Section/Variables* | *Function* |
|---|---|
| Main routine | Initialize data, start search |
| Q% | Queen squares |
| D% | Depth in tree |

| | |
|---|---|
| P% | Positions examined |
| PROC SEARCH | Handle the tree of possible positions |
| J% | Queen index counter |
| MATCH% | Number of clashes |
| PROC FOUND | Handle solution |
| K | Input key |
| PROC DELAY | Fixed time delay |
| PROC PBOARD | Print chess board |
| PROC PSMES | Print 'searching' message |
| PROC PMES | Print centralised message and erase old message |
| Y% | Message Y-coordinate |
| M$ | Actual message |

## Suggestions

Modify the program to handle the 'Knights Tour' chess board problem.

Also, if you feel ambitious enough to write your own chess program you may like to expand the user-definable graphics characters to include all the chess pieces.

## The Listing

```
   10 *FX4,1
   20 DIMQ%(8)
   30 VDU23,224,0,0,4,&E,&24,&74,&24,&3F,23,225,0,
0,&20,&70,&24,&2E,&24,&FC,23,226,&3F,&3F,&10,&1F,8
,&F,0,0,23,227,&FC,&FC,8,&F8,&10,&F0,0,0
   40 MODE1:VDU23;8202;0;0;0;
   50 PROCPBOARD
   60 D%=0:P%=0
   70 PROCPSMES
   80 PROCSEARCH
   90 MODE7
  100 END
  110 REM**********************************
  120 DEFPROCSEARCH
  130 LOCALJ%
  140 D%=D%+1
  150 SOUND&11,-9,9*D%,1
  160 FORJ%=1TO8
  170   Q%(D%)=J%
  180   COLOUR129-((D%+J%)/2=(D%+J%)DIV2)
  190   VDU31,9+J%*2,3+D%*2,224,225,10,8,8,226,227
  200   MATCH%=0
  210   FORI%=D%TO1STEP-1
  220     IFQ%(I%)=Q%(D%) MATCH%=MATCH%+1
  230     IFABS(Q%(I%)-J%)=ABS(I%-D%) MATCH%=MATCH
```

```
%+1
  240       IFMATCH%>2 I%=0
  250       NEXT
  260    P%=P%+1
  270    IFINKEY$(0)<>"" PROCDELAY
  280    IFMATCH%>2THEN300
  290    IFD%<8 PROCSEARCH ELSE PROCFOUND
  300    COLOUR129-((D%+J%)/2=(D%+J%)DIV2)
  310    VDU31,9+J%*2,3+D%*2,32,32,10,8,8,32,32
  320    NEXT
  330 D%=D%-1
  340 ENDPROC
  350 REM*******************************
  360 DEFPROCFOUND
  370 PROCPMES(27,"Positions examined=   "):PRINTT
AB(POS-3,VPOS);P%;
  380 PROCPMES(29,CHR$(7)+"Here is a solution.")
  390 PROCPMES(30,"Hit any key to continue...")
  400 K=GET
  410 PROCPSMES
  420 ENDPROC
  430 REM*******************************
  440 DEFPROCDELAY
  450 *FX15,1
  460 TIME=0
  470 REPEAT
  480   UNTILTIME>50
  490 ENDPROC
  500 REM*******************************
  510 DEFPROCPBOARD
  520 FORI%=1TO8
  530   PRINTTAB(11,I%*2+3);
  540   FORJ%=1TO8
  550     COLOUR129-((I%+J%)/2=(I%+J%)DIV2)
  560     PRINT"  "CHR$(10)CHR$(8)CHR$(8)"  "CHR$(
11);
  570     NEXT
  580   PRINT'
  590   NEXT
  600 ENDPROC
  610 REM*******************************
  620 DEFPROCPSMES
  630 PROCPMES(27,"")
  640 PROCPMES(29,"Searching")
  650 PROCPMES(30,"")
  660 ENDPROC
  670 REM*******************************
  680 DEFPROCPMES(Y%,M$)
  690 COLOUR128
  700 PRINTTAB(0,Y%)SPC(39)TAB(20-LEN(M$)/2,Y%)M$;
  710 ENDPROC
```

# Edit

## General

This program allows you to create or change text files. The files must contain ASCII text not tokenised BASIC programs. (If you do wish to edit BASIC programs you can use *SPOOL to convert from BASIC to text and *EXEC to convert from text back to BASIC after the edit. See the User Guide for more details.)

The program ignores all linefeed characters within the file, thus allowing it to handle all different formats for specifying end-of-line; eg CR only, CR–LF or LF–CR.

This editor uses a screen/cursor format rather than the more old-fashioned line-editors.

## Display

The file being edited is displayed in all but the bottom two lines of the screen.

The bottom line is used for various queries within the program. The line above this is merely a separator line for clarity.

The cursor is shown at the current position in the file.

## Operation

First the program asks you for the name of the file to be edited. If you wish to edit an existing file, enter its name and start your recorder (a maximum of ten characters may be entered for the filename). If you wish to start editing a new file from scratch then just press <RETURN>.

When the file has been loaded, its first page will be displayed and the cursor will be at the top left hand corner of the display (the first character of the file).

The program will then accept the following commands:

| | |
|---|---|
| <CTRL>S | *Save file*. The program asks for the filename to save the new data to. Enter the required filename and start your recorder. If you press this accidentally then just entering <RETURN>, when asked for the filename, will return you to the editor.<br>After the file has been saved the program asks 'Continue edit?'. If you press 'N' you will be |

| | |
|---|---|
| | returned to BASIC. Any other keypress returns you to the editor to continue the edit. |
| <CTRL>Q | *Quit edit.* The program asks 'Quit?'. If you press 'Y' the edit is aborted and you are returned to BASIC. Pressing any other key returns you to the editor. |
| <CTRL>T | *Move pointer to top of file.* The first page of the file is displayed with the cursor at the top left corner. |
| <CTRL>B | *Move pointer to bottom of file.* The last line of the file is displayed with the cursor placed one position after the last character. |
| <CTRL>U | *Move pointer up one page (20 lines).* The previous page (if any) is displayed. If you try to page before the start of the file the cursor will be moved to the top of the file and the first page displayed. |
| <CTRL>N | *Move pointer down one page.* The next page (if any) is displayed. If you try to page past the end of the file the cursor will be moved one position after the last character. |
| up-arrow | *Move pointer to end of previous line.* If you try to move before the start of the file, the pointer will move to the first character of the file. |
| down-arrow | *Move pointer to start of next line.* If you try to move past the end of the file, the pointer will move to the position after the last character. |
| left-arrow | *Move pointer to previous character.* If you move before the start of the current line, the pointer will move to the last character of the previous line. |
| right-arrow | *Move pointer to next character.* If you move past the end of the current line, the pointer will move to the first character of the next line. |
| <CTRL>F | *Find string.* The program asks 'ARG?' whereupon you must enter the string you wish it to search for, followed by <RETURN> (a maximum of thirty-five characters may be entered). The program then searches forwards through the file until it finds the specified string. If the string does not exist the pointer is left at the end-of-file. If you press this accidentally, then just pressing <RETURN> will return you to the editor with the pointer unchanged. |
| <CTRL>R | *Replace string.* As <CTRL>F but when the first string has been found it asks for a second |

| | |
|---|---|
| | string to replace the first. If you just press <RETURN> the first string will just be deleted. |
| <CTRL>D | *Delete character at pointer.* |
| <CTRL>E | *Erase whole line at pointer.* The current line is deleted wherever the cursor is on that line. |
| <DELETE> | *Delete character before pointer.* |
| Any ASCII character | *Insert character at pointer.* The character typed is inserted into the file at the current pointer position. |
| <COPY> | *Repeat last command.* Pressing COPY causes the last command entered to be repeated. This is most useful for the 'FIND' and 'REPLACE' commands to work on a section or the whole of the file. |

## Program

The program displays the current page of the file and handles the required user commands.

The whole file is stored in memory at one time.

| *Section/Variables* | *Function* |
|---|---|
| Main routine | Sets up various addresses, declares variable space, inputs file data, and main loop |
| PT% | General-purpose pointer |
| CA% | Current cursor address within file |
| TCA% | Temporary storage for CA% |
| CX% | Cursor screen X-coordinate |
| CY% | Cursor screen Y-coordinate |
| SA% | File start address |
| FA% | File finish address |
| MC% | Machine code space |
| FCB% | File-control-block |
| KBUF% | Keyboard buffer |
| SSA% | Address of start of screen text |
| FSA% | Address of finish of screen text |
| LIN% | Line number on screen |
| LL% | Line lengths |
| ENDOF% | At end-of-file flag |
| LIM% | Input line length limit |
| ARG1% | First argument |
| ARG2% | Second argument |
| ARGLEN1% | First argument length |
| ARGLEN2% | Second argument length |
| CTR% | General-purpose counter |
| PADCHAR% | Character saved while padding out space |

| | |
|---|---|
| LCOM% | Last command character |
| CCOM% | Copy command active flag |
| SAV | Handles 'save' command |
| QUIT | Handles 'quit' command |
| KLA,KRA,KDA,KUA | Handle arrow key commands |
| TTOP | Handle 'top' command |
| TBOT | Handle 'bottom' command |
| FIND | Handle 'find' command |
| REPLACE | Handle 'replace' command |
| LD | Load named file |
| SV | Save named file |
| GTARG1 | Get first argument |
| GTARG2 | Get second argument |
| GTARG | Get string argument for 'find' or 'replace' commands |
| GTCH | Get input character |
| GTFN | Get filename |
| GTLN | Get line of input |
| GTPARAM | Get parameter for message subroutine |
| CLRMES | Clear current bottom line message |
| PRCH | Print character |
| PRMES | Print message on bottom line |
| PRPAGE | Print current page of data |
| MESCURSOR | Move cursor to start of bottom line |
| CURSOR | Move cursor to specified coordinates |
| FINDCURSOR | Generate cursor coordinates from pointer address |
| DECCA | Decrement file pointer |
| INCCA | Increment file pointer |
| DECPT | Decrement general printer |
| INCPT | Increment general pointer |
| SAVCA | Save file pointer temporarily |
| RESCA | Restore file pointer |
| NCHAR | Move pointer to next character |
| PCHAR | Move pointer to previous character |
| NLINE | Move pointer to start of next line |
| PLINE | Move pointer to end of previous line |
| SPLINE | Move pointer to start of previous line |
| NPAGE | Move pointer to next page |
| PPAGE | Move pointer to previous page |
| FARG1 | Find specified string argument |
| CPARG | Compare two strings for equality |
| DELCHAR | Delete character at pointer |
| DELLINE | Delete line at pointer |
| INSCH | Insert character into file |
| PACK | Close up gap in file when something deleted |
| PAD | Open up gap in file when something to be inserted |

## Suggestions

Word-processors/text-editors range from the very simple to the very complicated. Finding one which does all the things you require within your budget may be very difficult. This program includes all the most often used commands. If you find in use that you wish to do something that the program currently cannot, then add this new facility and so gradually build it into a tailor made editor for your own personal requirements.

Also, the program currently uses only MODE 7. You may find that using the eighty-column MODE 0 or MODE 3 is better for letter writing. But remember that adding new options and using higher-resolution modes cuts down the space available for actually storing text.

## The Listing

```
   10 *TV255
   20 MODE7
   30 HIMEM=&5000
   40 OSBYTE=&FFF4:OSFILE=&FFDD:OSRDCH=&FFE0:OSWRC
H=&FFEE
   50 PT%=&70:CA%=&72:TCA%=&74:CX%=&76:CY%=&77:SA%
=&78:FA%=&7A
   60 DIMMC%8*256,FCB%17,KBUF%99,SSA%1,FSA%1,LIN%0
,LL%22,ENDOF%0,LIM%0,ARG1%38,ARG2%38,ARGLEN1%0,ARG
LEN2%0,CTR%0,PADCHAR%0,LCOM%0,CCOM%0
   70 REM******************************
   80 FORAO%=0TO2STEP2
   90   P%=MC%
  100   [OPTAO%
  110    LDA£HIMEM MOD256:STASA%:STAFA%:LDA£HIMEM
DIV256:STASA%+1:STAFA%+1
  120    JSRGTFN:BEQCFNM:JSRLD
  130   .CFNM JSRCLRMES
  140    JSRTTOP
  150   .NXTCCLR LDA£12:JSROSWRCH
  160    LDX£0:LDY£23:JSRCURSOR
  170    LDX£40:LDA£ASC"="
  180   .NSC JSROSWRCH:DEX
  190    BNENSC
  200   .NXTCREPRINT JSRPRPAGE
  210   .NXTCOMMAND
  220    JSRPRMES
  230    ]$P%=CHR$(23)+CHR$(1)+CHR$(1)+CHR$(0)+CHR
$(0)+CHR$(0)+CHR$(0)+CHR$(0)+CHR$(0)+CHR$(0)+"Comm
and"+CHR$(ASC"?"+128):P%=P%+18:[OPTAO%
  240    JSRFINDCURSOR:LDXCX%:LDYCY%:JSRCURSOR
  250    LDA£0:STACCOM%
  260    JSRGTCH
  270    CMP£&87:BNESTLC
  280     INCCCOM%:LDALCOM%
  290    .STLC STALCOM%
  300     CMP£ASC"S"-&40:BNEQU:JMPSAV
  310    .QU CMP£ASC"Q"-&40:BNELA:JMPQUIT
  320    .LA CMP£&88:BNERA:JMPKLA
  330    .RA CMP£&89:BNEDA:JMPKRA
  340    .DA CMP£&8A:BNEUA:JMPKDA
  350    .UA CMP£&8B:BNETTO:JMPKUA
  360    .TTO CMP£ASC"T"-&40:BNETBO:JSRTTOP:JMPNXTC
CLR
  370    .TBO CMP£ASC"B"-&40:BNEFPAG:JSRTBOT:JMPNXT
CCLR
  380    .FPAG CMP£ASC"N"-&40:BNERPAG:JMPNPAGE
  390    .RPAG CMP£ASC"U"-&40:BNEFSTR:JMPPPAGE
  400    .FSTR CMP£ASC"F"-&40:BNERSTR:JMPFIND
  410    .RSTR CMP£ASC"R"-&40:BNEDELCC:JMPREPLACE
  420    .DELCC CMP£ASC"D"-&40:BNEDELPC:JMPDELCCHAR
  430    .DELPC CMP£127:BNEDELL:JMPDELFCHAR
  440    .DELL CMP£ASC"E"-&40:BNEINS:JMPDELLINE
  450    .INS CMP£&D:BEQJINS:CMP£&20:BCCNXTCA:CMP£1
27:BCSNXTCA:.JINS JMPINSCH
  460    .NXTCA JMPNXTCOMMAND
  470    \********************************
  480    .SAV JSRGTFN:BEQNXTCA:JSRSV
  490     JSRPRMES
  500     ]$P%="Continue edit"+CHR$(ASC"?"+128):P%=
P%+14:[OPTAO%
  510     JSRGTCH:CMP£ASC"N":BNENXTCA
  520     RTS
  530    .QUIT
  540     JSRPRMES
  550     ]$P%="Quit"+CHR$(ASC"?"+128):P%=P%+5:[OPT
AO%
  560     JSRGTCH:CMP£ASC"Y":BNENXTCA
  570     RTS
  580    \********************************
  590    .KLA JSRPCHAR:JMPKLUE
  600    .KRA JSRNCHAR:JMPKRDE
  610    .KDA JSRNLINE
  620    .KRDE LDAENDOF%:BNEKLRDUE
  630     LDACA%+1:CMPFSA%+1:BCCKLRDUE:BNESCRD:LDAC
A%:CMPFSA%:BCCKLRDUE
  640    .SCRD JSRSAVCA
  650     LDASSA%:STACA%:LDASSA%+1:STACA%+1:JSRNLIN
E:LDACA%:STASSA%:LDACA%+1:STASSA%+1
  660     JSRRESCA
  670     JSRPRPAGE
  680     JMPKRDE
  690    .KUA JSRPLINE
  700    .KLUE LDASSA%+1:CMPCA%+1:BCCKLRDUE:BNESCRU
:LDACA%:CMPSSA%:BCSKLRDUE
```

```
  710   .SCRU JSRSAVCA
  720    LDASSA%:STACA%:LDASSA%+1:STACA%+1:JSRSPLI
NE:LDACA%:STASSA%:LDACA%+1:STASSA%+1
  730    JSRRESCA
  740    JSRPRPAGE
  750   .KLRDUE JMPNXTCA
  760   \********************************
  770   .TTOP
  780    LDASA%:STASSA%:STACA%:LDASA%+1:STASSA%+1:
STACA%+1
  790    RTS
  800   .TBOT
  810    JSRCTBOT
  820    JSRSPLINE:LDACA%:STASSA%:LDACA%+1:STASSA%
+1
  830   .CTBOT LDAFA%:STACA%:LDAFA%+1:STACA%+1
  840    RTS
  850   \********************************
  860   .FIND
  870    JSRGTARG1:BNEFFARG:JMPNXTCA
  880   .FFARG JSRFARG1
  890   .STPL JSRSAVCA
  900    JSRSPLINE
  910    LDACA%:STASSA%:LDACA%+1:STASSA%+1
  920    JSRRESCA
  930    JMPNXTCCLR
  940   \********************************
  950   .REPLACE
  960    JSRGTARG1:BNERFARG:JMPNXTCA
  970   .RFARG JSRFARG1:LDAENDOF%:BNESTPL
  980    JSRSAVCA
  990    LDACA%:STAPT%:CLC:ADCARGLEN1%:STACA%:LDAC
A%+1:STAPT%+1:ADC£0:STACA%+1
 1000    JSRPACK
 1010    JSRRESCA
 1020    JSRGTARG2:BNERFARG2:JMPRARGE
 1030   .RFARG2 JSRSAVCA
 1040    LDAFA%:LDYCA%:STACA%:STAPT%:STYFA%:LDAFA%
+1:LDYCA%+1:STACA%+1:STAPT%+1:STYFA%+1
 1050    LDACA%:CLC:ADCARGLEN2%:STACA%:LDACA%+1:AD
C£0:STACA%+1
 1060    JSRPAD
 1070    JSRRESCA
 1080    LDY£0
 1090   .RNC LDAARG2%,Y:STA(CA%),Y:INY
 1100    LDAARG2%,Y:CMP£&D
 1110    BNERNC
 1120   .RARGE JMPSTPL
 1130   \********************************
 1140   .LD
 1150    LDA£KBUF%MOD256:STAFCB%:LDA£KBUF%DIV256:S
```

```
TAFCB%+1
 1160     LDASA%:STAFCB%+2:LDASA%+1:STAFCB%+3
 1170     LDA£0:STAFCB%+6
 1180     LDA£&FF:LDX£FCB%MOD256:LDY£FCB%DIV256:JSR
OSFILE
 1190     LDASA%:CLC:ADCFCB%+10:STAFA%:LDASA%+1:ADC
FCB%+11:STAFA%+1
 1200     RTS
 1210    .SV
 1220     LDA£KBUF%MOD256:STAFCB%:LDA£KBUF%DIV256:S
TAFCB%+1
 1230     LDASA%:STAFCB%+10:LDASA%+1:STAFCB%+11
 1240     LDAFA%:STAFCB%+14:LDAFA%+1:STAFCB%+15
 1250     LDA£0:LDX£FCB%MOD256:LDY£FCB%DIV256:JSROS
FILE
 1260     RTS
 1270    \**********************************
 1280    .GTARG1
 1290     LDACCOM%:BNEGA1E
 1300     JSRGTARG
 1310     STYARGLEN1%
 1320     LDY£0
 1330    .SNARG1 LDAKBUF%,Y:STAARG1%,Y:INY
 1340     CMP£&D
 1350     BNESNARG1
 1360    .GA1E LDYARGLEN1%
 1370     RTS
 1380    .GTARG2
 1390     LDACCOM%:BNEGA2E
 1400     JSRGTARG
 1410     STYARGLEN2%
 1420     LDY£0
 1430    .SNARG2 LDAKBUF%,Y:STAARG2%,Y:INY
 1440     CMP£&D
 1450     BNESNARG2
 1460    .GA2E LDYARGLEN2%
 1470     RTS
 1480    .GTARG
 1490     JSRPRMES
 1500     ]$P%="Arg"+CHR$(ASC"?"+128):P%=P%+4:[OPTA
 0%
 1510     LDY£35:JSRGTLN:DEY
 1520     RTS
 1530    .GTCH
 1540     JSROSRDCH
 1550     BCCGTCHEX
 1560     LDA£&7E:JSROSBYTE
 1570     JMPGTCH
 1580    .GTCHEX RTS
 1590    .GTFN
 1600     JSRPRMES
 1610     ]$P%="Filename"+CHR$(ASC"?"+128):P%=P%+9:
[OPTAO%
 1620      LDY£10:JSRGTLN
 1630      RTS
 1640     .GTLN
 1650      STYLIM%
 1660      LDY£0
 1670     .GTLNN JSRGTCH:CMP£127:BNEGTLNAC
 1680      CPY£0:BEQGTLNA
 1690      DEY
 1700      JMPGTLNPC
 1710     .GTLNAC CPYLIM%:BEQGTLNA
 1720      CMP£&D:BEQGTLSC
 1730      CMP£&20:BCCGTLNN
 1740     .GTLSC STAKBUF%,Y:INY
 1750     .GTLNPC JSROSWRCH
 1760     .GTLNA CMP£&D
 1770      BNEGTLNN
 1780      LDAKBUF%:CMP£&D
 1790      RTS
 1800     .GTPARAM
 1810      LDY£0:JSRINCPT:LDA(PT%),Y
 1820      RTS
 1830     \********************************
 1840     .CLRMES
 1850      JSRMESCURSOR
 1860      LDX£39:LDA£ASC" "
 1870     .CLRMN JSROSWRCH:DEX
 1880      BNECLRMN
 1890      RTS
 1900     .PRCH
 1910      CMP£&D:BEQPRCHCLREOL
 1920      INCCX%
 1930      JMPOSWRCH
 1940     .PRCHCLREOL LDA£134:JSROSBYTE:LDA£ASC" "
 1950     .PRCHCN JSROSWRCH:INX:CPX£40
 1960      BNEPRCHCN
 1970      INCLIN%
 1980      LDX£0:STXCX%
 1990      RTS
 2000     .PRMES
 2010      JSRCLRMES
 2020      JSRMESCURSOR
 2030      PLA:STAPT%:PLA:STAPT%+1
 2040     .PRMESN JSRGTPARAM:PHA:AND£&7F:JSROSWRCH:P
LA
 2050      BPLPRMESN
 2060      JSRINCPT
 2070      JMP(PT%)
 2080     .PRPAGE
 2090      JSRPRMES
 2100     ]$P%=CHR$(23)+CHR$(1)+CHR$(0)+CHR$(0)+CHR
$(0)+CHR$(0)+CHR$(0)+CHR$(0)+CHR$(0)+CHR$(0+128):P
%=P%+10:[OPTAO%
 2110     LDA£30:JSROSWRCH
 2120     LDY£22:LDA£&FF
 2130    .CNLL STALL%,Y:DEY
 2140     BPLCNLL
 2150     LDA£0:STAENDOF%
 2160     LDASSA%:STAPT%:LDASSA%+1:STAPT%+1:LDY£0:S
TYLIN%:STYCX%:TYA:STA(FA%),Y
 2170    .PRPAGEN LDY£0:LDA(PT%),Y:BEQPRPAGEOF
 2180     LDXLIN%:INCLL%,X:JSRINCPT:CMP£&A:BEQPRPAG
EA:JSRPRCH
 2190    .PRPAGEA LDXCX%:CPX£40:BNEPRPAGEC:INCLIN%:
LDX£0:STXCX%
 2200    .PRPAGEC LDYLIN%:CPY£23
 2210     BNEPRPAGEN
 2220     JMPPRPAGER
 2230    .PRPAGEOF JSRINCPT
 2240     DECENDOF%:LDXLIN%:INCLL%,X
 2250    .CLREOSN LDA£134:JSROSBYTE:LDA£ASC" "
 2260    .CEOSNL CPY£23:BEQPRPAGER
 2270    .CEOSNC JSROSWRCH:INX:CPX£40
 2280     BNECEOSNC
 2290     LDX£0
 2300     INY
 2310     JMPCEOSNL
 2320    .PRPAGER LDYPT%:STYFSA%:LDYPT%+1:STYFSA%+1
 2330     RTS
 2340     \*********************************
 2350    .MESCURSOR
 2360     LDX£0:LDY£24
 2370    .CURSOR
 2380     LDA£31:JSROSWRCH:TXA:JSROSWRCH:TYA:JSROSW
RCH
 2390     RTS
 2400    .FINDCURSOR
 2410     LDASSA%:STAPT%:LDASSA%+1:STAPT%+1
 2420     LDY£&FF
 2430    .FCNL INY:LDAPT%:CLC:ADCLL%,Y:STAPT%:LDAPT
%+1:ADC£0:STAPT%+1:JSRINCPT
 2440     LDAPT%+1:CMPCA%+1:BCCFCNL:BNEFCSL:LDACA%:
CMPPT%:BCSFCNL
 2450    .FCSL STYCY%:LDAPT%:SEC:SBCLL%,Y:STAPT%:LD
APT%+1:SBC£0:STAPT%+1:JSRDECPT
 2460    .FCFC LDX£0:LDY£0
 2470    .FCFCN LDAPT%:CMPCA%:BNEFCFCLF:LDAPT%+1:CM
PCA%+1:BEQFCSC
 2480    .FCFCLF LDA(PT%),Y:CMP£&A:BEQFCFCA:INX
 2490    .FCFCA JSRINCPT:JMPFCFCN
 2500    .FCSC STXCX%
 2510     RTS
 2520    \********************************
 2530    .DECCA
 2540     DECCA%:LDXCA%:INX:BNEDECCE:DECCA%+1
 2550    .DECCE RTS
 2560    .INCCA
 2570     INCCA%:BNEINCCE:INCCA%+1
 2580    .INCCE RTS
 2590    .DECPT
 2600     DECPT%:LDXPT%:INX:BNEDECPE:DECPT%+1
 2610    .DECPE RTS
 2620    .INCPT
 2630     INCPT%:BNEINCPE:INCPT%+1
 2640    .INCPE RTS
 2650    \********************************
 2660    .SAVCA
 2670     LDACA%:STATCA%:LDACA%+1:STATCA%+1
 2680     RTS
 2690    .RESCA
 2700     LDATCA%:STACA%:LDATCA%+1:STACA%+1
 2710     RTS
 2720    \********************************
 2730    .NCHAR
 2740     LDACA%:CMPFA%:BNENCA:LDACA%+1:CMPFA%+1:BE
QNCE
 2750    .NCA JSRINCCA
 2760     LDY£0:LDA(CA%),Y:CMP£&A:BEQNCHAR
 2770    .NCE RTS
 2780    .PCHAR
 2790     LDACA%:CMPSA%:BNEPCA:LDACA%+1:CMPSA%+1:BE
QPCE
 2800    .PCA JSRDECCA
 2810     LDY£0:LDA(CA%),Y:CMP£&A:BEQPCHAR
 2820    .PCE RTS
 2830    .NLINE
 2840    .NLNOC LDY£0:LDA(CA%),Y:CMP£&D:BEQNLNC
 2850     JSRNCHAR:BEQNLE
 2860     JMPNLNOC
 2870    .NLNC JSRNCHAR
 2880    .NLE RTS
 2890    .PLINE
 2900    .PLPOC JSRPCHAR:BEQPLE
 2910     LDY£0:LDA(CA%),Y:CMP£&D:BNEPLPOC
 2920     LDA£1
 2930    .PLE RTS
 2940    .SPLINE JSRPLINE:JSRPLINE:BEQSPLE:JSRNLINE
 2950    .SPLE RTS
 2960    .NPAGE
 2970     LDASSA%:STACA%:LDASSA%+1:STACA%+1
 2980     LDX£20:STXCTR%
 2990    .NPNL JSRNLINE
```

```
3000     DECCTR%
3010     BNENPNL
3020     LDACA%:STASSA%:LDACA%+1:STASSA%+1
3030     JMPNXTCCLR
3040    .PPAGE
3050     LDASSA%:STACA%:LDASSA%+1:STACA%+1
3060     LDX£19:STXCTR%
3070    .PPPL JSRPLINE
3080     DECCTR%
3090     BNEPPPL
3100     JSRSPLINE
3110     LDACA%:STASSA%:LDACA%+1:STASSA%+1
3120     JMPNXTCCLR
3130    \********************************
3140    .FARG1
3150     LDA£0:STAENDOF%
3160     JSRNCHAR:BEQFARGEOF
3170    .FARGN JSRCPARG:BEQFARGE
3180     JSRNCHAR:BEQFARGEOF
3190     JMPFARGN
3200    .FARGEOF INCENDOF%
3210    .FARGE RTS
3220    .CPARG
3230     LDY£0
3240    .CPAN LDA(CA%),Y:CMPARG1%,Y:BNECPAE
3250     INY
3260     LDAARG1%,Y:CMP£&D
3270     BNECPAN
3280    .CPAE RTS
3290    \********************************
3300    .DELPCHAR
3310     JSRPCHAR:BNEDPCD:JMPNXTCA
3320    .DPCD JSRDELCHAR
3330     LDASSA%+1:CMPCA%+1:BCCDPCE:BNEDPCSU:LDACA
%:CMPSSA%:BCSDPCE
3340    .DPCSU JMPSCRU
3350    .DPCE JMPNXTCREPRINT
3360    .DELCCHAR
3370     JSRDELCHAR
3380     JMPNXTCREPRINT
3390    .DELCHAR
3400     JSRSAVCA
3410     LDACA%:STAPT%:LDACA%+1:STAPT%+1
3420     JSRNCHAR
3430     JSRPACK
3440     JSRRESCA
3450     RTS
3460    .DELLINE
3470     JSRPLINE:BEQDLSCA:JSRNLINE
3480    .DLSCA JSRSAVCA
3490     LDACA%:STAPT%:LDACA%+1:STAPT%+1
```

```
 3500     JSRNLINE
 3510     JSRPACK
 3520     JSRRESCA
 3530     JMPNXTCREPRINT
 3540    \********************************
 3550    .INSCH
 3560     STAKBUF%
 3570     JSRSAVCA
 3580     LDAFA%:LDYCA%:STACA%:STAPT%:STYFA%:LDAFA%
+1:LDYCA%+1:STACA%+1:STAPT%+1:STYFA%+1:JSRINCCA
 3590     JSRPAD
 3600     JSRRESCA
 3610     LDY£0:LDAKBUF%:STA(CA%),Y:JSRINCCA
 3620     JSRPRPAGE
 3630     JMPKRDE
 3640    \********************************
 3650    .PACK LDY£0:TYA:STA(FA%),Y
 3660    .PACKN LDA(CA%),Y:STA(PT%),Y:BEQPACKE
 3670     INY
 3680     BNEPACKN
 3690     INCPT%+1:INCCA%+1
 3700     JMPPACKN
 3710    .PACKE TYA:CLC:ADCPT%:STAFA%:LDA£0:ADCPT%+
1:STAFA%+1
 3720     RTS
 3730    .PAD LDY£0:LDA(FA%),Y:STAPADCHAR%:LDA£1:ST
A(PT%),Y:TYA:STA(FA%),Y
 3740     LDACA%:STAFA%:LDACA%+1:STAFA%+1
 3750    .PADN LDA(PT%),Y:STA(CA%),Y:BEQPADE
 3760     DEY:CPY£&FF
 3770     BNEPADN
 3780     DECPT%+1:DECCA%+1
 3790     JMPPADN
 3800    .PADE LDAPADCHAR%:STA(CA%),Y
 3810     RTS
 3820    \********************************
 3830    ]
 3840    IFP%>MC%+8*256 STOP
 3850    NEXT
 3860 REM*******************************
 3870 *FX4,1
 3880 CALLMC%
 3890 MODE7
```

# Adventure

When you come to think of it, in spite of everything we human beings are quite something. We are immensely curious, inventive, always wanting to see what's over the horizon or what's beyond the stars. If one frontier is closed, we find another – or even invent territories to explore. Adventure games are territories we invent. We fill them with locations, objects, events, and even (sometimes) creatures of our imagination. We can play someone else's adventure and other people can play our adventures.

Adventure games can be immensely complicated. Some have moving graphics, though very many people prefer purely text adventures because, they say, text descriptions trigger their 'graphical' imagination.

What are the ingredients of an adventure? Take the very simple coffee adventure here.

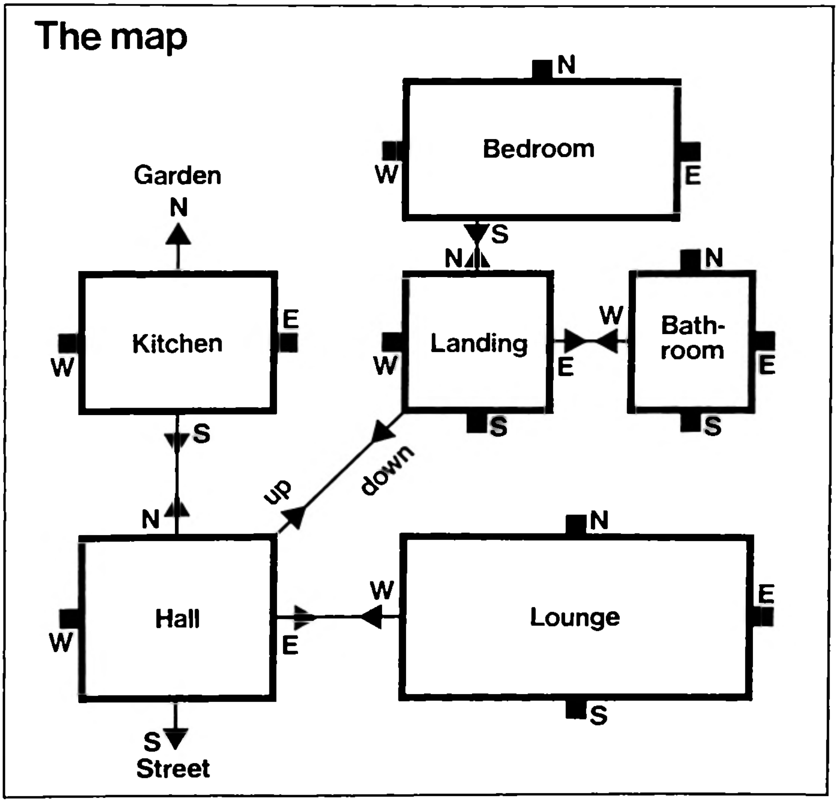
| | |
|---|---|
| *Its purpose* | We want to make a cup of coffee. |
| *The 'catch'* | We have to find the ingredients. |
| *Hazards* | None in this program, but necessary in big adventures. If you elaborate the coffee adventure, you could increase the number of rooms in the house, make one or two rooms dark, allow for dropping and shattering a cup or saucer, add a demented dog or a couple of deadly spiders; and so on. |
| *Locations* | These are the rooms in the house. We must know: ● how many locations there are ● what they are (descriptions) ● what object or objects (if any) each contains ● in which location you are at the moment ● whether it's possible to go from one location to the adjacent one (the directions). |
| *Objects* | We find certain things in certain locations. Some of these things could be red herrings; others could be essential for our successful adventuring. Here the objects are things like the kettle. We must know: ● how many objects there are ● what the object is (its description) ● where it is (its location). |

(By the way, what's a 'data table'? This simply is the way we *represent* the locations; the directions in which we can or cannot go from one location to another; and what objects, if any, we can find in each location.)

| | |
|---|---|
| *Verbs* | You have an object. You must do something with it or to it: *pick* it, *drop* it, and so on. You are in a location. You will want to *go* north, south, east, west, up or down, etc. |

When you are asked 'what now?', the response you give is a *command* to the computer. The line you type in is the *command line*. Suppose you say: *Pick cup*. This adventure is ultra-simple, so, as its store of verbs (eg *pick*) and nouns (eg *cup*) is not large, there are bound to be words it can't handle. Now try: *get cup*. That's satisfactory.

Before you do anything, draw a 'physical' *map* of your adventure. Here is the map of the coffee adventure:



What's happening inside the program?

Your command line has to be *analysed* by the program. This is called 'parsing'. The command is split up into a verb and a noun (object). The program has to look up its store, or list, of verbs and nouns and decide whether you've given it a command that's legal (in its terms). Remember, also, that it is not enough that both verb and noun are in its vocabulary. It must also check to see that, for example, the object is something that is available to you at that point in the game. If there were an unlit lamp in an adventure, and you said 'light lamp' without having matches, it would have to tell you that.

The program can give you a list of things that you may be carrying. This is called an *inventory*.

You may want to go in a direction that's impossible. To internally signal or *flag* this to itself, the program keeps track of this.

You may or may not be carrying something. Again, the program keeps a *flag* to check on this. If, for instance, you try to use an object that you're not carrying, the program can check you.

If the water in the kettle isn't boiled, you can't make a cup of coffee. Again, the program keeps a 'boiled' *flag* to keep track of this.

To signal to itself whether you have or haven't found all the ingredients (the objects) the program keeps yet another *flag*.

Notice that the program has to perform all kinds of tests: Have all the objects been found? Are you carrying an object? Is the command legal? Is the kettle boiled? Can you go in a particular direction? Is the game over?

If you are interested in writing your own adventures, you should start here and study this program carefully. The next step is to try and modify it and add to it, but cautiously and gradually. Later, if you feel that you have a fair idea of how to write an adventure, restrain yourself and first buy a couple of books on adventuring; also keep an eye out for magazine articles on the subject. You will find many techniques, such as compressing the text so as to fit more in, that we could not go into here. There are also 'adventure generators' on the market. These are supposed to save you programming effort and allow you to concentrate on the creative aspects of your invention. We are bringing these to your attention but you will have to decide for yourself whether you would like to invest in one.

## Rules

Adventure games allow you to become a different person in a different land full of magic, monsters and myths. This book hasn't the space to list a full-blown adventure, but this program should serve as a good example of how to start writing an adventure. It merely involves the exploration of a house to find the ingredients to make a cup of coffee. The program will describe for you the location and ask you what to do next.

## Display

The display consists of scrolling text, describing your location and the various things you see. Special responses may be printed in reply to certain of your commands.

## Operation

The program understands six verbs and contains five objects. Type in your commands in a verb-noun format; eg GET CUP, GO WEST.

## Program

The program analyses the user-input into a verb and noun string. These are then processed to try to make some sense of them. If they are understood then the required action is taken; otherwise the program tells you that it doesn't understand.

| *Section/Variables* | | *Function* |
|---|---|---|
| Main routine | | Initialize data, main game loop, game over |
| D$ | | Location descriptions |
| N% | | Northward routes |
| S% | | Southward routes |
| E% | | Eastward routes |
| W% | arrays | Westward routes |
| U% | | Upward routes |
| D% | | Downward routes |
| OH$ | | Object 'handle' string |
| OD$ | | Object descriptions |
| OLOC% | | Object locations |
| NLOC% | | Number of locations |
| NOBJ% | | Number of objects |
| LOC% | | Current location number |
| BOILED% | | Kettle-boiled flag |
| COM$ | | Command line |
| VERB$ | | Verb string |
| NOUN$ | | Noun string |
| MA% | | Movement-allowed flag |
| FN FOUNDALL | | Test if all objects found by adventurer |
| FA% | | Found-all-objects flag |
| PROC BOIL | | Handle 'boil' verb |
| PROC HELP | | Handle 'help' verb |
| PROC GET | | Handle 'get' verb |
| PROC DROP | | Handle 'drop' verb |
| PROC INVENTORY | | Handle 'inventory' verb |
| NC% | | Nothing-carried flag |
| PROC DIRECTION | | Handle 'go' or direction verb |
| C$ | | First character of direction |
| PROC MOVE | | Handle required movement |
| NL% | | New location number |
| PROC VERBNOUN | | Parse command line |
| PROC FINDOBJ | | Find object in list |

## Suggestions

Commercially available adventure programs are vastly more complicated than this small example. It does, however, contain some of the most useful routines in an adventure program. Try to expand on the descriptions and the map. Also add 'special' puzzles to solve; eg how to raise a magic portcullis.

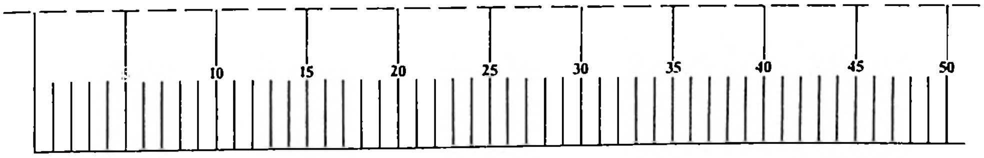
Notice that, because the 'parsing' is minimal, the program will not accept a command like BOIL WATER. Try to modify the program to make it more 'intelligent'.

## The Listing

```
   10 MODE7
   20 DIMD$(9),N%(9),S%(9),E%(9),W%(9),U%(9),D%(9)
,OH$(9),OD$(9),OLOC%(9)
   30 NLOC%=7
   40 FORI%=0TONLOC%
   50    READD$(I%),N%(I%),S%(I%),E%(I%),W%(I%),U%(
I%),D%(I%)
   60    NEXT
   70 NOBJ%=4
   80 FORI%=0TONOBJ%
   90    READOH$(I%),OD$(I%),OLOC%(I%)
  100    NEXT
  110 LOC%=5:BOILED%=FALSE
  120 REPEAT
  130    PRINT''"You are in the ";D$(LOC%)
  140    FORI%=0TONOBJ%
  150      IFOLOC%(I%)=LOC% PRINT"You see ";OD$(I%)
  160      NEXT
  170    INPUT'"What now",COM$
  180    PROCVERBNOUN
  190    IFVERB$="BOIL" PROCBOIL:GOTO270
  200    IFVERB$="HELP" PROCHELP:GOTO270
  210    IFVERB$="GET" PROCGET:GOTO270
  220    IFVERB$="DROP" PROCDROP:GOTO270
  230    IFLEFT$(VERB$,1)="I" PROCINVENTORY:GOTO270
  240    IFVERB$="GO" VERB$=NOUN$
  250    PROCDIRECTION
  260    IFNOTMA% PRINT"I don't understand!?"
  270    UNTILBOILED%ANDFNFOUNDALL
  280 PRINT'''"Congratulations!!! You can now make
"'"yourself a real cup of coffee as a"'"reward!(an
d make me one while you're at"'"it!)"
  290 END
  300 REM********************************
  310 DEFFNFOUNDALL
  320 FA%=TRUE
  330 FORI%=0TONOBJ%
  340    IFOLOC%(I%)<>LOC%ANDOLOC%(I%)<>99 FA%=FALS
E
  350    NEXT
  360 =FA%
  370 REM********************************
  380 DEFPROCBOIL
  390 IFNOUN$="KETTLE"AND(LOC%=OLOC%(0)OROLOC%(0)=
99) PRINT"The kettle is now boiled!":BOILED%=TRUE
ELSE PRINT"You can't boil "NOUN$
  400 ENDPROC
  410 REM********************************
  420 DEFPROCHELP
  430 PRINT"This adventure's too easy for you to n
eed help!!!"
  440 ENDPROC
  450 REM********************************
  460 DEFPROCGET
  470 PROCFINDOBJ
  480 IFOBJI%<0OROLOC%(ABS(OBJI%))<>LOC% PRINT"I s
ee no "NOUN$ ELSEOLOC%(OBJI%)=99:PRINT"O.K."
  490 ENDPROC
  500 REM********************************
  510 DEFPROCDROP
  520 PROCFINDOBJ
  530 IFOBJI%<0OROLOC%(ABS(OBJI%))<>99 PRINT"You a
ren't carrying "NOUN$ ELSEOLOC%(OBJI%)=LOC%:PRINT"
O.K."
  540 ENDPROC
  550 REM********************************
  560 DEFPROCINVENTORY
  570 PRINT"You are carrying :-"
  580 NC%=TRUE
  590 FORI%=0TONOBJ%
  600    IFOLOC%(I%)=99 PRINTOD$(I%):NC%=FALSE
  610    NEXT
  620 IFNC% PRINT"Nothing!"
  630 ENDPROC
  640 REM********************************
  650 DEFPROCDIRECTION
  660 MA%=FALSE
  670 C$=LEFT$(VERB$,1)
  680 IFC$="N"PROCMOVE(N%(LOC%))
  690 IFC$="S"PROCMOVE(S%(LOC%))
  700 IFC$="E"PROCMOVE(E%(LOC%))
  710 IFC$="W"PROCMOVE(W%(LOC%))
  720 IFC$="U"PROCMOVE(U%(LOC%))
  730 IFC$="D"PROCMOVE(D%(LOC%))
  740 ENDPROC
  750 REM********************************
  760 DEFPROCMOVE(NL%)
  770 MA%=TRUE
  780 IFNL%<0 PRINT"You can't go that way!" ELSE L
OC%=NL%
  790 ENDPROC
  800 REM********************************
  810 DEFPROCVERBNOUN
  820 I%=0:VERB$="":NOUN$=""
  830 REPEAT
  840   I%=I%+1
  850   C$=MID$(COM$,I%,1)
  860   VERB$=VERB$+C$
  870   UNTILC$=" "ORC$=""
  880 IFC$=" " VERB$=LEFT$(VERB$,LEN(VERB$)-1)
  890 REPEAT
  900   I%=I%+1
  910   UNTILMID$(COM$,I%,1)<>" "
  920 I%=I%-1
  930 REPEAT
  940   I%=I%+1
  950   C$=MID$(COM$,I%,1)
  960   NOUN$=NOUN$+C$
  970   UNTILC$=" "ORC$=""
  980 IFC$=" " NOUN$=LEFT$(NOUN$,LEN(NOUN$)-1)
  990 ENDPROC
 1000 REM********************************
 1010 DEFPROCFINDOBJ
 1020 OBJI%=-1:I%=0
 1030 REPEAT
 1040   IFNOUN$=OH$(I%) OBJI%=I%
 1050   I%=I%+1
 1060   UNTILI%>NOBJ%
 1070 ENDPROC
 1080 REM********************************
 1090 DATA"bedroom.A door leads south.",-1,1,-1,-1
,-1,-1
 1100 DATA"landing.Doors lead north and east.A fli
ght of stairs leads down.",0,-1,2,-1,-1,3
 1110 DATA"bathroom.The door is to the west.",-1,-
1,-1,1,-1,-1
 1120 DATA"hall.The front door is to the south.Doo
rs also lead north and east.The stairs lead upward
s.",4,7,5,-1,1,-1
 1130 DATA"kitchen.The back door is to the north.A
nother door leads south.",6,3,-1,-1,-1,-1
 1140 DATA"lounge.A door leads west.",-1,-1,-1,3,-
1,-1
 1150 DATA"garden.The back door of the house lies
to the south.",-1,4,-1,-1,-1,-1
 1160 DATA"street.The front door of the house lies
 to the north.",3,-1,-1,-1,-1,-1
 1170 DATA"KETTLE","an electric kettle(full of wat
er)",4
```

```
1180 DATA"COFFEE","a large jar of instant coffee"
,4
1190 DATA"CUP","a china cup",5
1200 DATA"MILK","a pint bottle of milk",7
1210 DATA"SUGAR","a kilo bag of sugar",2
```

**Character Count Scale** (cut out to use)



## Notes

## Notes

## Notes

## Notes

## Notes

This software feast is ready to be served up on the British Broadcasting Corporation Model B Microcomputer. It has been prepared by Martin Bryant, author of *White Knight*, the chess program that won first prize in the home computer section of the 1983 PCW European Microchess Championship. Spread the book out before you and relish the reflex/reaction games (*Ricochet Golf*, *Meteors*, *Rollers*, *Slalom*, *Rebel*); the tactical games (*Balrog*, *Graves*, *March*, *Mine*); the puzzle games (*Solitaire*, *Towers*, *Rotate*, *Quiz*); the strategy games (*Backgammon*, *Awari*); and the demonstration 'dance' of the eight chess queens (*Queens*). The backgammon program alone is the equal of similar commercial games.

The book also includes an extremely simple adventure that allows you to grasp completely the principles of constructing such games of your own (*Adventure*); and a simple text editor (*Edit*) written in 6502 assembly language, that can be used on its own or as the basis for developing a text editor of your own.

The programs in this book are suitable for running on a Model B with Operating System 1.0 onward and with either cassette or disk storage.

£3.95

BRITISH BROADCASTING CORPORATION