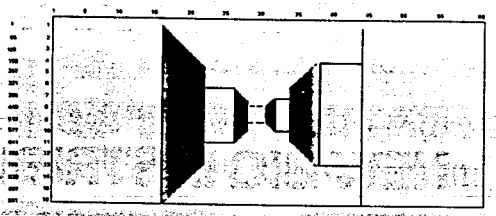
Don Scales

LABYRINTH

Lose yourself in this amazing game

Labyrinth is a fairly large program written in Tiny BASIC. Each time the program is run it will construct a different two dimensional maze and then allow the player to explore a three dimensional projection of this maze.

The program is divided roughly into two halves. The first half randomly builds a maze with a single route through it. A 2D plot of the maze is available at the end of this stage for those who suffer from claustrophobia. The second half of the program produces 3D projections as the player wanders along the corridors of the maze.



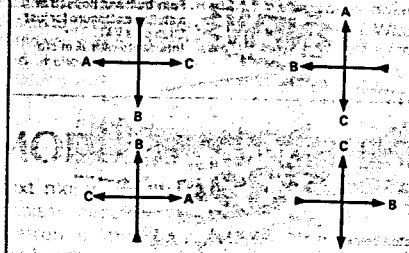
Building The Maze

The basic maze is a 'simple connected' maze (one which has no closed circuits). It is constructed using two, two dimensional arrays. The first array holds an indication of which cells of the maze have been used and the order in which they have been allocated. The second array holds the description of the topology of the maze.

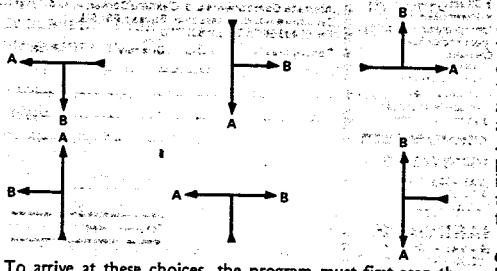
The maze construction starts by randomly selecting an entrance along the width of the maze. This location is saved in a spare element of the array.

From this start location the maze is constructed. At each cell, the program scans the adjacent cells to see which are available to use. Having decided which are available, the program then selects one cell randomly.

Consider the following examples. In each of these four there are three possible choices, A, B and C



hence the route can be chosen from the three possibilities. Next there are six combinations of two choices.



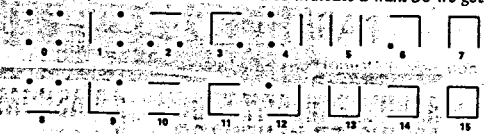
To arrive at these choices, the program must first scan the adjacent cells. As the program knows the direction it has just come from, it only needs to check the other three directions.

The program continues its random route through the maze until it hits a dead end. A branch is then made from the first route at this point and continued until the next dead end. This procedure is continued until the maze is complete.

At this point, the player can obtain a two dimensional display of the maze. Each element of the second array contains information about one cell of the maze. This information is incomplete as it is only for the top and right hand wall.

The Third Dimension

To produce a three dimensional picture, it is necessary to complete the cell information and organize it in such a manner that it can be rotated. The binary system fulfils both these requirements. A bit is used to indicate a wall. So we get



To turn left, the cell information is cyclically shifted right one bit. 2 becomes 4, 3 becomes 6, 8 becomes 1. To turn right, the cell information is cyclically shifted left one bit. 2 becomes 1, 1 becomes 8, 10 becomes 5.

The information for the 2d maze is therefore translated and the information completed by inspecting the neighbouring cells. The 3D pictures are produced using memory mapping and the graphics available on the TRITON. As most systems have both memory mapped displays and graphic symbols, it should be very easy to convert this part of the program to run on most machines.

The display is constructed simply with horizontal, vertical and diagonal lines. A reasonable display would be possible with | — and /\. To move in the maze, the player can turn left or right or move forward. The players current position can also be obtained.

Giving The Picture

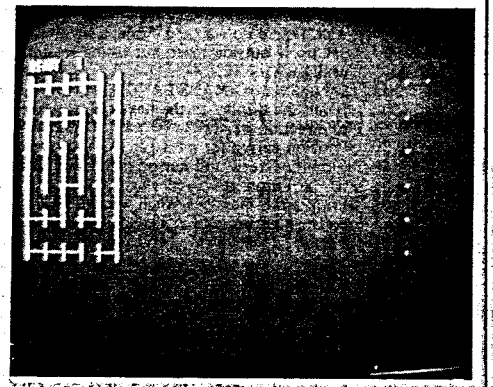
To produce the 3D picture, the program starts with the cell corresponding to the players current position. This cell is then rotated, as described earlier, until facing the same way as the player. The program then decodes the cell information and checks for the walls left, right and in front of the player. At the first depth, either a blank wall or two columns are produced. If a blank wall is produced, no further information is available. If looking out of the maze, no further information is produced and if outside the maze and looking away from it, a blank screen is all you get.

If, on the other hand, a passage exists to the next cell, the program obtains the information about the next cell by making the appropriate index and rotates and decodes this cell. At the second depth, it is possible to have walls or passages to the left, right and straight ahead.

Each depth has its own display routine which checks for and plots the three walls or passages. Each depth produces a display continuing from the previous and maintains the perspective. The display stops either with a blank wall or when depth 5 has been reached.

The program listing following contains the full Tiny BASIC commands and is commented to make it easier to follow and to translate. If using a floating point BASIC, take great care in the rotate and decode routines as they rely on integer rounding effects. A large number of INT commands will be required.

The program will fit on a TRITON with mother board and an extra 8K of static RAM but the Tiny BASIC commands should be abbreviated for size and speed reasons. A tape of the program in abbreviated Tiny BASIC is available from TRANSAM of Chapel Street, London.



Program Notes

LINE NUMBERS	
5–40	Clear Screen and print heading.
45–70	Ask for size of maze.
95–120	Clear arrays used to construct the maze and initialize variables. Obtain random entry point.
125–150	Save entry point and start the maze.
155–1295	Maze build routine.
155–200	Finds the next starting point when a route comes to a dead end.
210–270	Does an initial check on the number of allowable routes from the current position in the maze.
275–310	Randomly select Left, Down or Right as the next route.
320–350	More route checking. Z = 1 when an exit exists.
355–390	Randomly select Left, Down or Up.
395–410	Use when exit already exists or no way up. Randomly select Left or Down.
420–470	Move route checking.
475–510	Randomly select Left, Right or Up.
515–530	No way up. Randomly select Left or Right.
540–570	Move route checking.
575–600	Randomly select Left or Up.
610–680	Move route checking.
685–720	Randomly select Down, Right or Up.
725–740	No way up, select Down or Right.
750–780	Yet more route checking.
785–810	Just Down or Up.
820–870	Not much more route to check.
875–900	Right or Up.
910–950	Last bit of route checking.
955–990	Set up maze for route to go Left. Check if maze finished, if not, see where it goes next.
995–1030	Route goes Down.
1035–1100	Route goes Right.
1105–1160	Route goes Up. Checks if exit made.
1165–1200	Make exit at top, loop back if maze not complete.
1205–1210	Make sure maze has an exit.
1300–1320	Keyboard scan to see if 2D print required. READ 0, I scans a byte from the keyboard on the Triton. Substitute INPUT if necessary.
1330–1570	2D print routine.
1330–1335	Clear screen and print 'CHEAT'.
1340	Loop for height of maze.
1350–1420	Print the top of a line of cells checking to see if wall or gap required. To use Triton graphics change + to w and – to s.
1430–1500	Print the sides of a line of cells, checking to see if wall or gap required. To use Triton graphics change I to t.
1510	End of height loop.
1520–1570	Print bottom of last row of cells, leaving an entrance.
1595–1620	Reset cursor to top of screen and loop on the keyboard until a key is pressed. Again, INPUT can be substituted.
1625–1630	Call the instruction print routine.
1635–1870	Translate the maze into binary cell information and then give each cell the information about all its walls.
1635–1670	Translate maze to convenient notation and move into other buffer.
1710–1870	Take each cell in turn and check with adjacent cells to obtain information about all the walls.
1875–1890	Set up start parameters and go display entrance to maze in 3D.
1895–1950	Print instruction for wandering in maze.
1995–2100	Print helpful information when lost. Note the ^J and ^J which perform a carriage return without clearing the screen and a line feed.
2195–2270	Another keyboard scan routine. Routine loops scanning the keyboard until L, R, F or H are pressed. When pressed it jumps to the appropriate routine. No real problem to substitute INPUT.
2295–2320	Turn Left, then go display new view.
2345–2370	Turn Right, and go display new view.
2395–2440	Clear screen and wait while it is cleared. VDU 0, 12 is the clear screen command for a Triton.
2445–2460	Reset cursor to top of screen and wait. VDU 0, 28 is the reset cursor command.
2495–2540	Routine to space cursor and erase messages.
2595–2790	Rotate routine.
2595–2630	Check current position (A,B) and extract cell information if inside maze.
2635–2660	Rotate the cell information if not facing north until facing right direction.
2670–2700	Decode the cell information into C, D and E.

LABYRINTH

C is Left wall, D is Right wall and E is front wall. If zero no wall and if one, a wall.

2705–2750 Set up if outside maze but facing retaining wall.

2755–2790 Set up if in NO MANS LAND.

2795–2850 Index the display to the next cell according to direction faced.

2855–2920 Position cursor for messages. ΛJ and ΛI perform line feed and cursor right commands on the Triton.

2930–2980 Print error messages when you hit a dead end or no mans land.

2995–3040 Routine to move the player forward to the next cell.

3045–4980 3D display routines.

3045–3060 Set up start position, rotate and look from 1st cell.

3065–3080 Set up loop for up to 5 depths and call display routine.

3085–3140 Check if possible to see into next cell. If so, index to and rotate next cell. Loop to a depth of 5 unless wall in way. Return to keyboard routine.

3195–3200 Jump to appropriate depth routine.

3205–3300 Clear screen and check if facing no mans land, if yes, nothing to display. Otherwise display first depth.

3240–3270 Map vertical lines of walls. Triton screen is 64 wide by 16 high. The screen is numbered left to right, top to bottom from 1 to 1024. VDU 1,116 maps graphic 116 at the location in I.

3280–3330 Check for a wall ahead and if so map top and bottom. Graphic 107 is and 108 is .

3600–3940 Display second depth.

3600–3720 Check for left wall or passage and map projection. Graphic 114 is \, 113 is /.

3730–3840 Check for right wall or passage and map.

3850–3880 Map end walls.

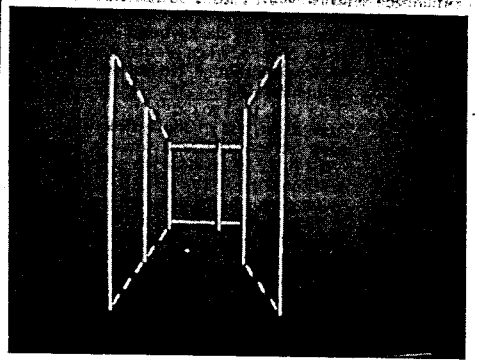
3890–3940 Check for end wall, return if no wall otherwise map top and bottom.

4000–4300 Display third depth.

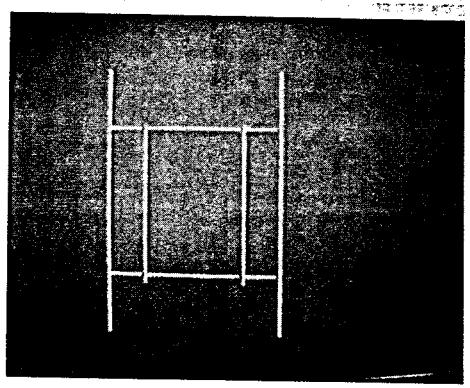
4400–4620 Display fourth depth.

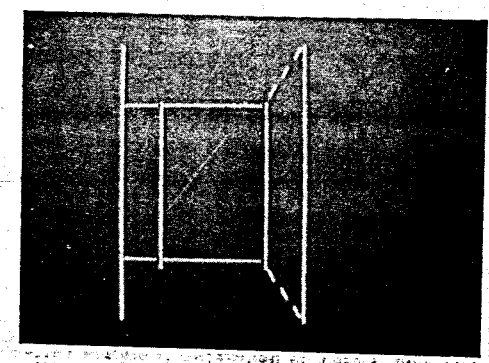
4800–4980 Display fifth depth. Graphic 106 is | and 105 is |.

4995–5030 Clear screen and display WAY OUT. End of game.



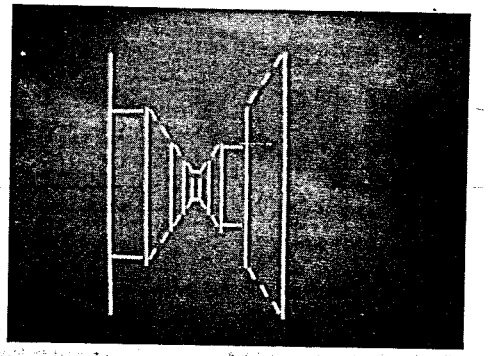
```
  5 REM-CLEAR SCREEN AND PRINT HEADING
 10 GOSUB 2400
 20 PRINT '   **********'
 30 PRINT '   *LABYRINTH*'
 40 PRINT '   **********'
 45 REM-GET MAZE DIMENSIONS
 50 PRINT 'ENTER SIZE OF MAZE'
 60 INPUT 'WIDTH 'H, 'HEIGHT 'V
 70 PRINT 'THINKING'
 95 REM-CLEAR MAZE ARRAY
100 A=H*V+1
110 FOR I=1 TO A+A:@(I)=0:NEXT I
120 Q=0,Z=0,X=RND(H)
125 REM-SAVE MAZE ENTRY POINT
130 @(A)=X
140 @(X)=1,C=2
150 R=X,S=1:GOTO 220
155 REM-START OF MAZE BUILD ROUTINE
160 IF P=H GOTO 200
170 IF S=V GOTO 190
180 R=1,S=1:GOTO 210
190 R=1,S=S+1:GOTO 210
200 R=R+1
210 IF @(P+(S-1)*H)=0 GOTO 160
220 IF P-1=0 GOTO 610
230 IF @(R-1+(S-1)*H)=0 GOTO 610
240 IF S-1=0 GOTO 420
250 IF @(R+(S-2)*H)=0 GOTO 420
260 IF R=H GOTO 320
270 IF @(R+1+(S-1)*H)=0 GOTO 320
275 REM-LEFT/DOWN/RIGHT
280 X=RND(3)
290 IF X=1 GOTO 960
300 IF X=2 GOTO 1000
310 GOTO 1040
320 IF S=V GOTO 350
330 IF Z=1 GOTO 400
340 Q=1:GOTO 360
350 IF @(R+S*H)=0 GOTO 400
355 REM-LEFT/DOWN/UP
360 X=RND(3)
370 IF X=1 GOTO 960
380 IF X=2 GOTO 1000
390 GOTO 1110
395 REM-LEFT/DOWN
400 X=RND(2)
410 GOTO 370
420 IF R=H GOTO 540
430 IF @(R+1+(S-1)*H)=0 GOTO 540
440 IF S=V GOTO 470
450 IF Z=1 GOTO 520
460 Q=1:GOTO 480
470 IF @(R+S*H)=0 GOTO 520
475 REM-LEFT/RIGHT/UP
480 X=RND(3)
490 IF X=1 GOTO 960
500 IF X=2 GOTO 1040
510 GOTO 1110
515 REM-LEFT/RIGHT
520 X=RND(2)
530 GOTO 490
540 IF S=V GOTO 570
550 IF Z=1 GOTO 960
560 Q=1:GOTO 580
570 IF @(R+S*H)=0 GOTO 960
575 REM-LEFT/UP
580 X=RND(2)
590 IF X=1 GOTO 960
600 GOTO 1110
610 IF S-1=0 GOTO 820
620 IF @(P+(S-2)*H)=0 GOTO 820
630 IF P=H GOTO 750
640 IF @(P+1+(S-1)*H)=0 GOTO 750
650 IF S=V GOTO 680
660 IF Z=1 GOTO 730
670 Q=1:GOTO 690
680 IF @(P+S*H)=0 GOTO 730
685 REM-DOWN/RIGHT/UP
690 X=RND(3)
700 IF X=1 GOTO 1000
710 IF X=2 GOTO 1040
720 GOTO 1110
725 REM-DOWN/RIGHT
730 X=RND(2)
740 GOTO 700
750 IF S=V GOTO 780
760 IF Z=1 GOTO 1000
770 Q=1:GOTO 790
780 IF @(R+S*H)=0 GOTO 1000
785 REM-DOWN/UP
790 X=RND(2)
800 IF X=1 GOTO 1000
810 GOTO 1110
820 IF R=H GOTO 910
830 IF @(R+1+(S-1)*H)=0 GOTO 910
840 IF S=V GOTO 870
850 IF Z=1 GOTO 1040
860 Q=1:GOTO 880
870 IF @(P+S*H)=0 GOTO 1040
875 REM-RIGHT/UP
880 X=RND(2)
890 IF X=1 GOTO 1040
900 GOTO 1110
910 IF S=V GOTO 940
920 IF Z=1 GOTO 160
930 Q=1:GOTO 950
940 IF @(R+S*H)=0 GOTO 160
950 GOTO 1110
955 REM-LEFT
960 @(R-1+(S-1)*H)=C
970 C=C+1,@(A+R-1+(S-1)*H)=2,R=R-1
980 IF C=A GOTO 1210
990 Q=0:GOTO 220
995 REM-DOWN
1000 @(R+(S-2)*H)=C
1010 C=C+1
1020 @(A+R+(S-2)*H)=1,S=S-1:IF C=A GOTO 1210
1030 Q=0:GOTO 220
1035 REM-RIGHT
1040 @(R+1+(S-1)*H)=C
1050 C=C+1:IF @(A+P+(S-1)*H)=0 GOTO 1070
1060 @(A+R+(S-1)*H)=3:GOTO 1080
1070 @(A+R+(S-1)*H)=2
1080 R=R+1
1090 IF C=A GOTO 1210
1100 GOTO 610
1105 REM-UP
1110 IF Q=1 GOTO 1170
1120 @(R+S*H)=C,C=C+1:IF @(A+R+(S-1)*H)=0 GOTO 1140
1130 @(A+R+(S-1)*H)=3:GOTO 1150
1140 @(A+R+(S-1)*H)=1
```

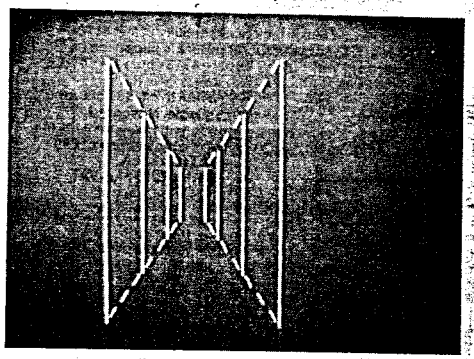




```
1150 S=S+1:IF C=A GOTO 1210
1160 GOTO 220
1165 REM-EXIT AT TOP OF SCREEN
1170 Z=1
1180 IF @(A+R+(S-1)*H)=0 GOTO 1200
1190 @(A+R+(S-1)*H)=3,Q=0:GOTO 160
1200 @(A+R+(S-1)*H)=1,Q=0,R=1,S=1:GOTO 210
1205 REM-MAKE EXIT IF NOT THERE
1210 IF Z=1 X=A+RND(H)+(V-1)*H,@(X)=@(X)+1
1295 REM-END OF MAZE BUILD
1300 PRINT 'DO YOU WANT TO SEE THE MAZE?';
1310 READ 0,I:IF I<128 GOTO 1310
1320 IF I=249 GOTO 1630
1330 GOSUB 2400:PRINT 'CHEAT!!!!'
1335 REM-2D DISPLAY ROUTINE
1340 FOR J=V TO 1 STEP -1
1350 FOR I=1 TO H
1360 IF @(A+I+(J-1)*H)=0 GOTO 1400
1370 IF @(A+I+(J-1)*H)=2 GOTO 1400
1375 REM-PRINT TOP OF CELLS
1380 PRINT '+ ';
1390 GOTO 1410
1400 PRINT '+--';
1410 NEXT I
1420 PRINT '+'
1430 PRINT 'I',
1440 FOR I=1 TO H
1450 IF @(A+I+(J-1)*H)<2 GOTO 1480
1455 REM-PRINT SIDES OF CELLS
1460 PRINT '  ';
1470 GOTO 1490
1480 PRINT '  I';
1490 NEXT I
1500 PRINT
1510 NEXT J
1520 FOR I=1 TO H
1530 IF-I=@(A) GOTO 1550
1535 REM-PRINT BOTTOM OF MAZE
1540 PRINT '+--';:GOTO 1560
1550 PRINT '+ ';
1560 NEXT I
1570 PRINT '+'
1595 REM-PAUSE FOR VIEWING
1600 GOSUB 2450
1610 PRINT 'READY ';
1620 READ 0,I:IF I<128 GOTO 1620
1625 REM-PRINT INSTRUCTION
1630 GOSUB 1900
1635 REM-TRANSLATE ROUTINE
1640 FOR I=1 TO A-1
1650 J=I+A
1660 @(I)=(3-@(J))*2
1670 NEXT I
1710 W=@(A)
1715 REM-COMPLETE CELL INFORMATION
```

LABYRINTH





```
1720 FOR J=1 TO V
1730 K=(J-1)*H
1740 FOR I=1 TO H
1750 L=I+K
1760 IF J*1 GOTO 1790
1770 IF I=W GOTO 1820
1780 M=1:GOTO 1310
1790 M=@(L-H)/2
1800 M=M-(M/2)*2
1810 @(L)=@(L)+M*3
1820 IF I=1 M=1:GOTO 1850
1830 M=@(L-1)/4
1840 M=M-(M/2)*2
1850 @(L)=@(L)+M
1860 NEXT I
1870 NEXT J
1875 REM-SET UP START PARMS
1880 X=W,Y=0,Z=1
1890 GOTO 3050
1895 REM-INSTRUCTION PRINTOUT
1900 GOSUB 2400
1910 PRINT "ENTER L TO TURN LEFT"
1920 PRINT "    R TO TURN RIGHT"
1930 PRINT "    F TO GO FORWARD"
1940 PRINT "    H FOR HELP"
1950 RETURN
1995 REM-HELP ROUTINE
2000 PRINT "YOU ARE AT";X;",";Y;
2010 PRINT #1,X;" EAST ";X;",";Y;
2020 PRINT #1,Y;" NORTH ";X;",";Y;
2030 PRINT "YOU ARE FACING";Z;",";Y;
2040 IF Z=1 PRINT "NORTH"
2050 IF Z=2 PRINT "EAST"
2060 IF Z=3 PRINT "SOUTH"
2070 IF Z=4 PRINT "WEST"
2080 PRINT ",";Y;
2090 GOSUB 2450
2100 GOTO 2200
2195 REM-KEYBOARD ROUTINE
2200 IF Y>V GOTO 5000
2210 READ @,A
2220 IF A<128 GOTO 2210
2230 IF A=236 GOTO 2500
2240 IF A=242 GOTO 2550
2250 IF A=230 GOTO 3000
2260 IF A=232 GOTO 2000
2270 GOTO 2210
2295 REM-LEFT TURN
2300 Z=Z-1
2310 IF Z<1 Z=Z+4
2320 GOTO 3050
2345 REM-RIGHT TURN
2350 Z=Z+1
2360 IF Z>4 Z=Z-4
2370 GOTO 3050
2395 REM-CLEAR SCREEN AND WAIT
2400 I=12
2410 VDU 0,I
2420 FOR I=1 TO 600
2430 NEXT I
2440 RETURN
2445 REM-RESET CURSOR AND WAIT
2450 I=28
2460 GOTO 2410
2495 REM-ERAZE MESSAGE ROUTINE
2500 GOSUB 2860
2510 PRINT
2520 GOSUB 2450
2530 S=0
2540 RETURN
2595 REM-ROTATE AND LOOK ROUTINE
2600 IF B=0 GOTO 2710
2610 IF B>V E=2:RETURN
2620 F=@(A+(B-1)*H)
2630 IF Z=1 GOTO 2670
2635 REM-ROTATE
2640 FOR I=2 TO Z
2650 F=F/2+(F-(F/2)*2)*3
2660 NEXT I
2670 C=F-(F/2)*2
2680 D=F/4-(F/8)*2
2690 E=F/2-(F/4)*2
2700 RETURN
2705 REM-OUTSIDE MAZE
2710 C=0,D=0,E=-1
2720 IF Z*1 GOTO 2760
2730 E=1
2740 IF A=W E=0
2750 RETURN
2755 REM-NO MANS LAND
2760 IF Z=3 E=2
2770 IF Z=2 IF A=H E=2
2780 IF Z=4 IF A=1 E=2
2790 RETURN
2795 REM-INDEX TO NEXT CELL
2800 IF E>0 GOTO 2930
2810 IF Z=1 B=B+1
2820 IF Z=2 A=A+1
2830 IF Z=3 B=B-1
2840 IF Z=4 A=A-1
2850 RETURN
2855 REM-MESSAGE ROUTINE
2860 FOR I=1 TO 8
2870 PRINT ^J,
2880 NEXT I
2890 FOR I=1 TO 23
2900 PRINT ^I,
2910 NEXT I
2920 RETURN
2930 GOSUB 2860
2940 IF E=1 PRINT "  DEAD END",
2950 IF E=2 PRINT "NO MANS LAND",
2960 GOSUB 2450
2970 S=1
2980 RETURN
2995 REM-FORWARD ROUTINE
3000 A=X,B=Y
3010 GOSUB 2600
3020 GOSUB 2800
3030 X=A,Y=B
3040 IF E>0 GOTO 2200
3045 REM-3D DISPLAY ROUTINE
3050 A=X,B=Y
3060 GOSUB 2600
3065 REM-5 DEPTHS
3070 FOR I=1 TO 5
3080 GOSUB 3200
3085 REM-CHECK FOR NEXT DEPTH
3090 IF E*0 GOTO 2200
3100 GOSUB 2800
3110 GOSUB 2600
3120 IF E=2 GOTO 2200
3130 NEXT I
3140 GOTO 2200
3195 REM-JUMP TO DISPLAY DEPTH
3200 GOTO I*400+2810
3205 REM-DISPLAY DEPTH 1
3210 GOSUB 2400
3220 IF E<0 RETURN
3230 IF E>1 RETURN
3240 FOR I=30 TO 976 STEP 64
3250 VDU I,116
3260 VDU I+28,116
3270 NEXT I
3280 IF E=0 RETURN
3290 FOR I=81 TO 107
3300 VDU I,107
3310 VDU I+896,108
3320 NEXT I
3330 RETURN
3600 REM-DISPLAY DEPTH 2
3610 IF C=0 GOTO 3690
3620 VDU 81,114
3630 VDU 147,114
3640 VDU 213,114
3650 VDU 977,113
3660 VDU 915,113
3670 VDU 853,113
3680 GOTO 3750
3690 FOR I=273 TO 277
3700 VDU I,107
3710 VDU I+512,108
3720 NEXT I
3730 IF D=0 GOTO 3810
3740 VDU 107,113
3750 VDU 169,113
3760 VDU 231,113
3770 VDU 1003,114
3780 VDU 957,114
3790 VDU 871,114
3800 GOTO 3850
3810 FOR I=295 TO 299
3820 VDU I,107
3830 VDU I+512,108
3840 NEXT I
3850 FOR I=273 TO 730 STEP 64
3860 VDU I,116
3870 VDU I+16,116
3880 NEXT I
3890 IF E=0 RETURN
3900 FOR I=279 TO 293
3910 VDU I,107
3920 VDU I+512,108
3930 NEXT I
3940 RETURN
4000 REM-DISPLAY DEPTH 3
4010 IF C=0 GOTO 4070
4020 VDU 273,114
4030 VDU 345,114
4040 VDU 791,113
4050 VDU 723,113
4060 GOTO 4110
4070 FOR I=407 TO 409
4080 VDU I,107
4090 VDU I+256,108
4100 NEXT I
4110 IF D=0 GOTO 4170
4120 VDU 293,113
4130 VDU 355,113
4140 VDU 805,114
4150 VDU 739,114
4160 GOTO 4210
4170 FOR I=419 TO 421
4180 VDU I,107
4190 VDU I+256,108
4200 NEXT I
4210 FOR I=410 TO 666 STEP 64
4220 VDU I,116
4230 VDU I+8,116
4240 NEXT I
4250 IF E=0 RETURN
4260 FOR I=411 TO 417
4270 VDU I,107
4280 VDU I+256,108
4290 NEXT I
4300 RETURN
4400 REM-DISPLAY DEPTH 4
4410 IF C=0 GOTO 4450
4420 VDU 411,114
4430 VDU 667,113
4440 GOTO 4470
4450 VDU 475,107
4460 VDU 603,108
4470 IF D=0 GOTO 4510
4480 VDU 417,113
4490 VDU 673,114
4500 GOTO 4530
4510 VDU 481,107
4520 VDU 609,108
4530 FOR I=476 TO 604 STEP 64
4540 VDU I,116
4550 VDU I+4,116
4560 NEXT I
4570 IF E=0 RETURN
4580 FOR I=477 TO 479
4590 VDU I,107
4600 VDU I+123,108
4610 NEXT I
4620 RETURN
4800 REM-DISPLAY DEPTH 5
4810 IF C=0 GOTO 4850
4820 VDU 477,114
4830 VDU 605,113
4840 GOTO 4870
4850 VDU 477,108
4860 VDU 605,107
4870 IF D=0 GOTO 4910
4880 VDU 479,113
4890 VDU 607,114
4900 GOTO 4930
4910 VDU 479,108
4920 VDU 607,107
4930 VDU 541,106
4940 VDU 543,105
4950 IF E=0 RETURN
4960 VDU 473,108
4970 VDU 606,107
4980 RETURN
4995 REM-WAY OUT FOUND
5000 GOSUB 2400
5010 GOSUB 2860
5020 PRINT "  WAY OUT
5030 STOP
```