often appear clumsy and unwieldy. Languages such as LOGO and ALGOL are much better equipped to carry out this sort of task. In BASIC we have two main tasks to carry out. Firstly we must derive our tree from the maze data, as presented to the program. And for each square of the maze we must have four pointers showing which square lies in each of the four directions. The best way to store this pointer system is in a two-dimensional array, TR(N,D), where N is the square number and D is the direction 1 to 4. Thus in our simple maze, TR(9,1) would be 5— the square lying to the north of square 9. When the square in a particular direction is not free, or there is a boundary to the maze then this can be marked by a special value, for example -1.

As the tree is negotiated the route taken is stored in a pseudo-stack, implemented using a one-dimensional array and a variable, D, to point to the next available space on the stack. The shortest route encountered at any time is also stored in a one-dimensional array, with the number of steps for the route stored in the first element of the array.

When the program has worked its way through the tree, a record of the best route will be held as a series of square numbers. On the assumption that the vehicle originally faces north in the start square, it can be directed using the simple mathematical relationships between the

IAN McKINNELL

direction to be travelled and the difference between two consecutive square numbers in the route array. For example, in our simple maze, a difference of +4 would indicate north, -4 indicate south, and so on. We must then calculate the angle to be turned through to change direction, before proceeding one square forwards. As the vehicle uses simple DC electric motors, turning angles and distances travelled are governed by the length of time that a particular combination of motors is on for. To make practical use of the program some initial experiments need to be done to determine the time intervals required to turn through 90° and to advance one square. This information should be entered in the variables AF and FF, respectively. The BBC version requires units of 1/100th of a second, the Commodore 64 version requires 1/60th of a second units.

Solving The Maze

```
900 REM ********************
910 REM ********************
920 REM **                  **
930 REM ** CBM 64 MAZE     **
950 REM **                  **
960 REM ********************
970 REM ********************
980 :
990 REM **** MAIN CALLING PROGRAM ****
1000 GOSUB 1600:REM READ MAZE DATA
1100 GOSUB3700:REM PRINT MAZE
1200 GOSUB5400:REM CONSTRUCT TREE
1210 GOSUB7700:REM TRAVERSE TREE
1215 IF S(0)=9999 THEN PRINT "NO SOLUTION":END
1220 GOSUB8200:REM DIRECT VEHICLE
1400 END
1500 :
1600 REM **** READ MAZE DATA/INITIALISE ****
1700 READ SX,SY
1800 DIM MZ(SX,SY),TR(SX*SY,4),DR(4),RT(SX*SY),
     LN(SX*SY),CN(SX*SY)
1900 FOR Y=0 TO SX-1
2000 FOR X=0 TO SX-1
2100 READ A:MZ(X,Y)=A
2200 NEXT X,Y
2300 :
2400 READ XS,YS,XF,YF
2500 DATA 4,4
2600 DATA 1,0,0,0,0,0,1,0
2700 DATA 0,0,1,0,0,0,0,0
2800 DATA 1,1:REM START COORDS
2900 DATA 2,3:REM FINISH COORDS
3000 :
3100 DR(1)=-SX:DR(2)=1:DR(3)=SX:DR(4)=-1
3110 ID(1)=3:ID(2)=4:ID(3)=1:ID(4)=2
3120 SR(0)=9999:REM INIT SHORTEST ROUTE
3130 DDR=56579:DATREG=56577:POKE DDR,255
3140 AF=30:FF=45:REM ANGLE AND FWRD TIME FACTORS
3200 FOR I=1 TO 25:CD$=CD$+CHR$(17):NEXT I
3500 RETURN
3600 :
3700 REM **** PRINT MAZE ****
3800 PRINTCHR$(147):REM CLEAR SCREEN
3900 FOR X=0 TO SX-1
4000 FOR Y=0 TO SY-1
4100 GOSUB 4900:REM POSITION CURSOR
4200 PRINT CHR$(32+MZ(X,Y)*134)
4300 NEXT Y,X
4400 :
4500 X=XS:Y=YS:GOSUB4900:PRINT"S"
4600 X=XF:Y=YF:GOSUB4900:PRINT"F"
4700 RETURN
4800 :
4900 REM **** POSITION CURSOR AT X,Y ****
5000 PRINT CHR$(19);:PRINTTAB(X)LEFT$(CD$,Y);:RETURN
5300 :
5400 REM **** CONSTRUCT TREE ****
5500 :
5600 REM ** INITIALISE WITH TERMINATORS ****
5700 FOR P=0 TO SX*SY-1:FORI=1TO4:TR(P,I)=-1:NEXT I,P
6100 :
6110 REM ** CALCULATE START & FINISH ****
6120 X=XS:Y=YS:GOSUB9200:S=N
6130 X=XF:Y=YF:GOSUB9200:F=N
6140 :
6200 REM ** CONSTRUCT **
6300 LC=1:CC=0:REM INIT STACK PTRS
6400 LN(LC)=S:REM START POINT
6450 CN=LN(LC):REM GET CURR NODE OFF STACK
6500 DF=0
6600 FOR D=1 TO 4
6700 N=CN+DR(D):GOSUB 8900:REM CONVERT TO X,Y
6800 IF(X<0 OR X>SX-1 OR Y<0 OR Y>SY-1)THEN 7300
6900 IF MZ(X,Y)=1 THEN 7300
7000 IF(D=2 AND N/SX=INT(N/SX))THEN7300
7100 IF(D=4 AND CN/SX=INT(CN/SX))THEN7300
7200 IF TR(N,ID(D))=CN THEN 7300
7210 TR(CN,D)=N:DF=1
7220 CC=CC+1:CN(CC)=N:REM PUSH ONTO CURRENT STACK
7300 NEXT D
7310 IF(DF=0 AND LC=1) THEN RETURN:REM TERMINAL NODE
7320 :
7330 LC=LC-1:REM DEC LAST STACK PTR
7340 IF LC>0 THEN 6450:REM NEXT NODE
7345 :
7350 REM ** COPY CURR. STACK TO LAST STACK **
7360 FOR I=1 TO CC:LN(I)=CN(I):NEXT I
7390 LC=CC:CC=0:GOTO6450:REM NEXT NODE
7600 :
7700 REM **** TRAVERSE TREE ****
7720 C=0:RN=S:CN=RN:EF=0
7730 C=C+1:RT(C)=CN
7740 IF CN=F THEN GOSUB8100:GOSUB8000:IF EF=0 THEN 7730
7745 IF EF=1 THEN RETURN
7750 DF=0
7760 FOR D=1 TO 4
7770 IF TR(CN,D)<>-1 THEN CN=TR(CN,D):DF=1:DR=D:D=4
7780 NEXT D
7790 IF DF=0 THEN GOSUB8000
7800 IF EF=0THEN 7730
7810 IF EF=1 THEN RETURN
7820 :
8000 REM **** RESTART AT ROOT ****
8010 TR(RT(C-1),DR)=-1
8020 CN=RN:C=0
8030 IF(TR(CN,1)ANDTR(CN,2)ANDTR(CN,3)ANDTR(CN,4))=-1 THEN EF=1
8040 RETURN
8050 :
8100 REM **** SAVE ARRAY ****
8110 IF C>SR(0) THEN RETURN:REM IS NEW ROUTE SHORTER?
8120 SR(0)=C
8130 FOR I=1 TO C:SR(I)=RT(I):NEXT I:RETURN
8140 :
8200 REM **** DIRECT VEHICLE ****
8205 PD=1:REM ASSUME INITIAL
     DIRECTION NORTH
8210 FOR C=1 TO SR(0)-1
8220 DF=SR(C+1)-SR(C)
8222 :
8225 REM ** FIND REQUIRED DIRECTION **
8230 FOR I=1 TO 4
8240 IF DF=DR(I) THEN D=I:I=4
8250 NEXT I
8260 :
8265 DR=D-PD:PD=D
8270 H=INT(4+DR/4):R=(4+DR)-4*H
8275 :
8277 REM ** DO TURN **
8280 FOR I=1 TO R
8290 POKE DATREG=9:REM CLOCKWISE TURN
8300 T=TI
8310 IF (TI-T)<AF THEN 8310:REM WAIT
8320 POKE DATREG=0:REM OFF
8330 NEXT I
8340 :
8350 REM ** FORWARD **
8360 POKE DATREG=5
8370 T=TI
8380 IF (TI-T)<FF THEN 8380
8390 POKE DATREG=0
8400 NEXT C
8410 :
8420 RETURN
8430 :
8900 REM **** CONVERT N TO X,Y ****
9000 Y=INT(N/SX):X=N-SX*Y:RETURN
9200 :
9210 REM **** CONVERT X,Y TO N ****
9220 N=Y*SX+X:RETURN
```

For The BBC

Make the following changes:

```
3130 DDR=&FE62:DATREG=&FE60

8290 ?DATREG=9
8300 TIME=0
8310 REPEAT UNTIL TIME>=AF
8320 ?DATREG=0

8360 ?DATREG=5
8370 TIME=0
8380 REPEAT UNTIL TIME>=FF
8390 ?DATREG=0
```