Cross the minefield at your own risk

PROGRAMMING

Pete Gerrard steers Vic 20 owners through the minefield of programming

ONE OF THE more commonly recommended ways of learning to program is by studying someone else's programs, and adopting any useful techniques that they might have used. We can follow that procedure by looking at a games program for the unexpanded Vic 20.

The game is called Minefield, and the object of the game is to guide a little blob across a 9 by 10 square grid. Some of the squares on the grid have mines in them, and walking into such a square will result in instant destruction!

Rusty radar

You are equipped with a rusty radar that allows you to scan the immediate area around you. If there is a mine within one square of your present position, this information will be displayed at the top of the screen. However, it's up to you to determine which square the radar is indicating.

Getting successfully across the grid will send you onto the next level where there are even more hidden mines. And so it goes on, with more mines appearing all the time, until the inevitable happens and you meet your doom.

The program can be broken down into several main chunks, and the first 10 lines just clear the screen, tell you who to blame for the program, set the screen and border colours, and then go into displaying the instructions on the screen before dimensioning an array M(10,11) in line 10. This array will hold the mines information.

Line 20 contains the level counter L, and the variable J, which determines your position on the screen relative to the start of the screen memory, which on the unexpanded Vic, starts at location 7608.

Lines 30 to 50 need some explanation as the printer used to produce the listing can't reproduce Vic graphic symbols. Line 30 consists of PRINTing within quotes a space, then shifted C, then a space, then shifted C, and so on, until you have 9 spaces and 9 shifted Cs. Line 53 is exactly the same as line 30.

Line 40 is PRINTing within quotes a shifted B, a space, a shifted B, a space, and so on, until you have 10 shifted Bs and 10 spaces. This draws the grid for the ▶

```
1 PRINT"[CLR]"
2 REM ********************
3 REM * BY PETE GERRARD *
4 REM ********************
5 POKE 36879,8:REM SET SCREEN AND BORDER
8 GOSUB 2000:REM RULES
10 DIMM(10,11):REM ARRAY FOR GRID
20 L=L+1:J=441
27 PRINT"[CLR,WHT]"
28 FORI=1TO10:REM PLOT GRID
30 REM SEE ACCOMPANYING ARTICLE FOR DRAWING OF GRID
40 REM MORE GRID DRAWING
50 NEXT
53 REM AND THE BOTTOM LINE OF THE GRID
55 IFEOG=1THEN RETURN:REM END OF GAME
56 POKE 7680+J,102:POKE 7680+61,42:X=2:Y=2:REM PUT YOU ON GRID
57 FORI=1TO4+L*3:A=INT(RND(.5)*9):B=INT(RND(.5)*10):M(A,B)=1:REM GENERATE MINES
58 IF(A=0ANDB=0)OR(A=9ANDB=10)OR(A=1ANDB=0)THEN M(A,B)=0
59 NEXT
60 IFX=10ANDY=11THEN5000
61 GOSUB 1500
62 Q=J
64 GETP$:IFP$=""THEN 60:REM MOVEMENT
70 IFP$="I"THEN1000:REM UP
75 IFP$="M"THEN1200:REM DOWN
```

Continued on page 20

◀ game onto the screen.

Line 56 positions you on the grid, and places an asterisk in the top right hand corner of the grid: this is the spot that you're aiming for.

Lines 57 through 59 are generating the positions of the mines on the grid by producing random numbers in the range 0 through 9 for the X-direction, 0 through 10 for the Y-direction, and then storing those X,Y random numbers into our M(10,11) array by putting a value of 1 into the array where the co-ordinates meet. A 0 in the array means that there is no mine at that location.

Line 60 is a universal check to see if you've reached the top right hand corner of the grid. Lines 64 through 90 check for pressing of one of the four movement keys, and going to the REMmed subroutine to process moving up, down, left or right.

All of these moving routines are the same in that they start by checking for the validity of a move to ensure that you don't go over the borders of the grid. They then update your position on the grid and POKE your new position onto the screen using the J variable offset from the start of screen memory at 7680. Your old position is indicated by a dot on the screen, positioned using the variable Q.

Bomb search

The next step is to update your X or Y co-ordinates on the grid, before going to the bomb checking subroutine starting at line 1500. This checks the X and Y co-ordinates of all the surrounding squares for the presence of a bomb by using the original M(10,11) array and seeing if any of the surrounding X,Y co-ordinate squares contain a 1, or in other words a bomb.

Line 1502 checks to see if you're on a square with a bomb in it. If you are it's off to line 3000 and a message that tells you you're dead, before showing the offending bomb's position on the grid.

If you managed to reach the corner of the grid, the routine starting at line 5000 informs you of your success, updates the level counter L (so that we can have more bombs), and then re-sets all the bomb locations to 0, before going back to line 20 and coming up with some new bombs.

The main things to look at in the listing are the way that the grid is handled and the way that the surrounding squares are checked for the presence of a bomb. The use of the array M(10,11) to hold the bomb information and the use of the X and Y co-ordinates provides most of the lessons in this short program.

Other things to see are generating random numbers (line 57), checking that the bomb is not placed on either the starting square, one of the squares next to it (in which case you could never start the game), and the end square, all of which is done in line 58, and the handling of key pressing in lines 64 to 90. You should also have a fun game at the end of it!

This is a special Vic 20 adaption of the original Minefield game for the Commodore 64, as published in *Using the 64*, author Pete Gerrard, publishers Gerald Duckworth. ■

```
80 IFP$="A"THEN1400:REM LEFT
85 IFP$="D"THEN1600:REM RIGHT
90 GOTJ 64
1000 IFJ<66THEN60
1020 J=J-44:POKE7680+Q,46:POKE7680+J,102
1022 Y=Y+1
1025 GOSUB 1500
1040 GOTO60
1200 IFJ>412THEN60
1220 J=J+44:POKE7680+Q,46:POKE7680+J,102
1222 Y=Y-1
1225 GOSUB 1500
1240 GOTO60
1400 IF INT((J-1)/22)=(J-1)/22THEN60
1420 J=J-2:POKE 7680+Q,46:POKE 7680+J,102
1422 X=X-1
1425 GOSUB 1500
1440 GOTO 60
1500 REM CHECK FOR BOMBS
1502 IFM(X-2,Y-2)=1 THEN3000
1510 X1=X+1:
1511 IFM(X1-2,Y-1)=1THENB1=B1+1
1512 IFM(X1-2,Y-2)=1THENB1=B1+1
1513 IFY<3THEN1515
1514 IFM(X1-2,Y-3)=1THENB1=B1+1
1515 X1=X
1516 IFM(X1-2,Y-1)=1THENB1=B1+1
1517 IFM(X1-2,Y-2)=1THENB1=B1+1
1518 IFY<3THEN1520
1519 IFM(X1-2,Y-3)=1THENB1=B1+1
1520 X1=X-1:IFX<3THEN1570
1521 IFM(X1-2,Y-1)=1THENB1=B1+1
1522 IFM(X1-2,Y-2)=1THENB1=B1+1
1523 IFY<3THEN1570
1524 IFM(X1-2,Y-3)=1THENB1=B1+1
1570 PRINT"[HOME]"B1"BOMB(S) 1 SQ. AWAY"
1575 B1=0
1599 RETURN
1600 IF INT((J+5)/22)=(J+5)/22THEN60
1620 J=J+2:POKE 7680+Q,46:POKE 7680+J,102
1622 X=X+1
1625 GOSUB 1500
1640 GOTO 60
1998 GETZ$:IFZ$=""THEN1998
1999 END
2000 PRINT"[CLR,WHT]WELCOME TO MINEFIELD!"
2001 PRINT"[CD]YOU HAVE TO FIND YOUR WAY ACROSS A MINE-"
2002 PRINT"FIELD, TO REACH SAFETYIN THE TOP RIGHT"
2003 PRINT"CORNER OF THE SCREEN"
2004 PRINT"[CD]YOUR RUSTY RADAR ONLY SHOWS MINES THAT"
2005 PRINT"ARE ONE SQUARE AWAY : TREAD CAREFULLY!"
2006 PRINT"[CD]USE 'A' TO MOVE LEFT, 'D' RIGHT, 'I' UP"
2007 PRINT"AND 'M' DOWN'"
2008 PRINT"[CD]PRESS 'SPACE' WHEN    READY TO START"
2009 GETSW$:IFSW$=" "THENPRINT"":RETURN
2010 GOTO 2009
3000 PRINT"[CLR]DESTROYED!"
3002 PRINT"[CD]PRESS 'SPACE' TO SEE  THE BOMBS"
3004 GETSP$:IFSP$=" "THEN3008
3006 GOTO 3004
3008 EOG=1:GOSUB 27
3009 POKE 7680+J,102
3010 FORA=0TO10:FORB=0TO11
3015 IFM(A,B)=1THENSS=A*2+1+(9-B)*44+44:POKE 7680+SS,81
3020 NEXT B,A
3040 PRINT"[HOME]ANOTHER GAME (Y OR N)"
3045 GETG$:IFG$="Y"THENRUN
3050 IFG$="N"THENPRINT"[CLR]BYE!":END
3055 GOTO 3045
5000 REM SURVIVED A LEVEL! ONTO NEXT ONE
5002 PRINT"[CLR]YOU WERE LUCKY ON     LEVEL ";L
5005 FORI=0TO10:FORK=0TO11:M(I,K)=0:NEXTK,I
5010 PRINT"[CD]BUT JUST PRESS 'SPACE'AND I'LL PUT YOU"
5020 PRINT"ON LEVEL ";L+1
5030 GETSP$:IFSP$=" "THENPRINT"[CLR]":GOTO20
5040 GOTO5030
```