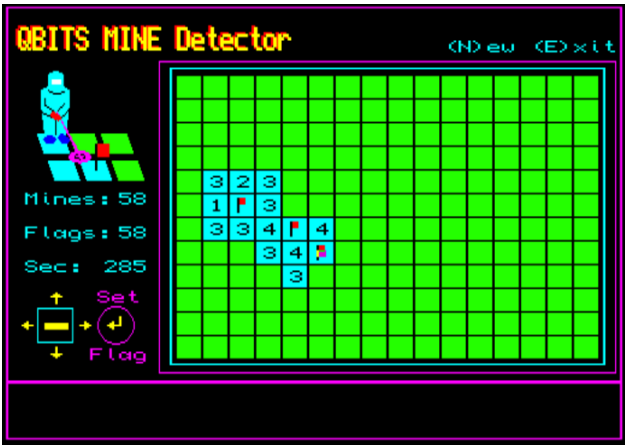
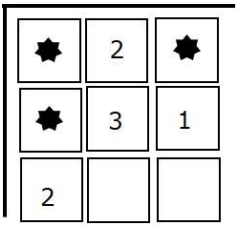


The Game of Minesweeper

Minesweeper is a Single-player Puzzle Game where success is largely contingent on being able to eliminate all possible positions of the distributed Mines within the shortest amount of time. Minesweeper was first introduced as a Mainframe Game of the 1960 and 1970's it then became popular as part of the Puzzle Video Game Genre during the 1980's. Each Game starts out with a grid of unmarked squares. You **Click** on a square to reveal its **Status** and some of the surrounding squares or **Mark** it with a **Flag** as one holding a **Mine**. The object is to Locate and **Mark** all of the "**Mines**" in the shortest possible time. If a player **Clicks** an unmarked square that is **mined**, the game ends.

The difficulty levels were Beginner, Intermediate, and Expert. **Beginner** usually came with a total of **10 Mines** and a board size either **8×8, 9×9**, or **10×10**. **Intermediate** with **40 Mines** and larger board sizes up to **16×16**. **Expert** had **99 Mines** and a grid of **16×30** (or 30×16). The eighty's introduction was partly due to encourage use of the mouse, the left button (**Spacebar**) for **Status** and the right button (**Enter**) to **Mark** with a Flag.

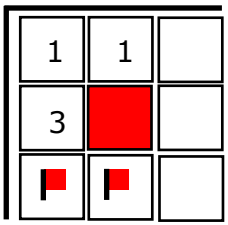
The player starts on a safe square and **Click's** to reveal a number on the square occupied and then some of the surrounding squares. The **numbers represent** how many mines are adjacent to the current square. For example, if a square has a "**3**" on it, then there are 3 mines placed in the surrounding squares. The mines could be positioned above, below, left or right, or in one of the four corners diagonally positioned squares.



Apart from squares adjacent to the boundaries and comers of the board, a square has the possibility of each of the surrounding squares holding a mine. Counting the possible mines this would be between zero and eight.

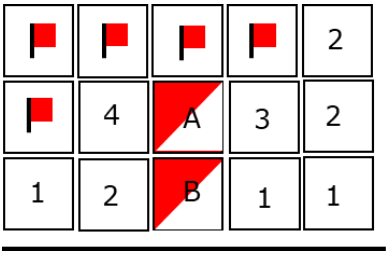
Minesweeper Logic or Probability
This game can be considered to be played as one of **Logic** or of **Probability**. Although technically Probability will include some level of Logic. If Logic suggests a mine is occupying a position, then in all Probability it has considerable certainty of being correct.

Local Probabilities
The squares shown has two with **Flags** where mines have been identified. For **Square** with the number **3** only one other **Mine** needs to be identified and this can only be in the Square shown in RED.



This is shown to be correct as the Squares with a 1 are also satisfied by the Mined Square shown in RED.

Local Probability Conflicts
The squares with a **Flag** are those with a **Mine**. Squares in **RED/ WHITE** are where the other **Mine** maybe located. This is implied by the numbers 4,3,2,1 shown in the adjacent squares. **Note** that in the surrounding grid of the Square containing a 4 it already has four Flags, so its true value is 5. It can't be 6 as this would break with the numbering of the other Squares.



Clearly either square **A** or **B** containing a **Mine** will satisfy the squares with numbers 4,3,2,1. If the choice is simply between **A** or **B,** which is it? Surprisingly by taking the lowest number square counts adjacent to a potential **Mine** square, in this case **B**, more often than not turns out to be the right choice.

QBITS APM
The 1980's QBITS first version of a Minesweeper game was called **Anti-Personnel Mine Detection (APM)**. However today with many Charities and Government Organisation providing schemes for clearing minefields of old and recent conflicts, **QBITS Mine Detector** is a Title perhaps more succinct. Changes to the display layout follow similarities with the other QBITS Games with this collection.

QBITS Mine Detector Game
Press (**N**) to start a New Game. The Aim of the Game is to **Mark** with a **Flag** all of the **Mines** in the grid and complete this within the Time Limit of **300sec** (**GTime**) which can be altered (see code line 1010).

Use the Cursor keys to move the highlight to a selected square on the grid. **Click** with **Spacebar** to reveal Squares **Status** or use **Enter** to **Mark** with a **Flag**. To add more difficulty **QBITS Mine Detector c**ounts surrounding mines to a max of four. Therefore, it is necessary to review several adjacent squares to determine if a mine is present. On a few occasions this will simply come down to a best guess.

QBITS Mine Detector Code

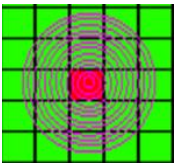
```
1000 REMark QBITS_MDETR_bas_[QBITS Mine Detector 2024 Review - QPC2] vM20

1002 dev$='win1_':MODE 8:gx=0:gy=0:      :REMark Basic Settings

1004 WHEN ERRor :CONTINUE:END WHEN

1006 REMark Import QBITSCongif settings - QPC2
1007 OPEN_IN#9,dev$&'QBITSConfig':INPUT#9,gx\gy\dn$:close#9

1010 Init_win:GTime=300: MF_Game

1012 DEFine PROCedure Init_win
1013 WINDOW#2,512,222,gx,gy             :PAPER#2,0:BORDER#2,1,3:CLS#2
1014 WINDOW#1,362,176,gx+136,gy+36:PAPER#1,0:BORDER#1,1,5
1015 WINDOW#0,512, 34,gx,gy+222      :PAPER#0,0:BORDER#0,1,3:CLS#0
1016 CSIZE#2,2,1:OVER#2,1:SCALE#1,120,0,0:CSIZE#0,2,0
1017 INK#2,2:FOR i=0 TO 1:CURSOR#2,4+i,8:PRINT#2,'QBITS MINE Detector'
1018 INK#2,6:FOR i=0 TO 1:CURSOR#2,6+i,9:PRINT#2,'QBITS MINE Detector'
1019 CSIZE#2,2,0:OVER#2,0
1020 INK#2,3:LINE#2,42,2 TO 42,86 TO 170,86 TO 170,2 TO 42,2:RESTORE
1021 FOR i=1 TO 10:READ col,x,y,str$:INK#2,col:CURSOR#2,x,y:PRINT#2,str$
1022 DATA 6,33,166,'⬆',6,33,198,'⬇',6,9,182,'⬅  ➡',6,80,182,'¼'
1023 DATA 5,360,18,'(N)ew (E)xit',3,72,166,'Set',3,66,200,'Flag'
1024 DATA 5,12,108,'Mines:',5,12,128,'Flags:',5,12,148,'Sec:'
1025 BLOCK#2,20,4,30,186,6:BLOCK#2, 2,6,92,182,6
1026 INK#2,5:LINE#2,8,10 TO 8,19 TO 18,19 TO 18,10 TO 8,10    :INK#2,3
1027 CIRCLE#2,30,14.5,5:MF_Detector 2,8,82:MF_PPE 2,5,13,82:INK#2,5
1028 END DEFine

1030 DEFine PROCedure MF_Game
1031 DIM mes$(4,40):MF_Seed:chk=0
1032 REPeat main
1033  IF chk=1
1034    CURSOR#2,78,148:PRINT#2,FILL$(' ',3-LEN(gsec))&gsec
1035    gsec=GTime-(DATE-OldTime):IF gsec=0:End_Game
1036  END IF
1037  x1=x:y1=y:OVER#1,-1:BLOCK#1,8,4,(a+(x1-1)*w),(b+(y1-1)*h),6
1038  k=CODE(INKEY$(20)) :BLOCK#1,8,4,(a+(x1-1)*w),(b+(y1-1)*h),6:OVER#1,0
1039  SELect ON k
1040   =192:x=x -1:BEEP 400:IF x< 1:x=1
1041   =200:x=x+1:BEEP 400:IF x>16:x=16
1042   =208:y=y -1:BEEP 400:IF y< 1:y=1
1043   =216:y=y+1:BEEP 400:IF y>12:y=12
1044   = 81,113:LRUN dn$ :STOP                          :REMark (Q)uit
1045   = 77,109:MF_Locate:mines=255:End_Game            :REMark Show Mines
1046   = 78,110:chk=1:CLS#0:MF_Seed:OldTime=DATE        :REMark (N)ew Game
1047   = 10:IF chk=1:MF_Mark:IF mines=0:End_Game        :REMark Mark mines
1048   = 32:IF chk=1:IF Board(x,y)=255:MF_Explode:ELSE :MF_Show
1049   = 27:MODE 4:CSIZE#2,0,0:INK#2,7:CSIZE#0,0,0:INK#0,7:PRINT#0,'Bye…':STOP
1050  END SELect
1051 END REPeat main
1052 END DEFine
```

```
1054 DEFine PROCedure MF_Show
1055 IF y-1>=1 AND x-1>=1    :x1=x-1:y1=y-1    :MF_Status:END IF
1056 IF y-1>=1              :x1=x  :y1=y-1    :MF_Status:END IF
1057 IF y-1>=1 AND x+1<=16 :x1=x+1:y1=y-1    :MF_Status:END IF
1058 IF x-1>0               :x1=x-1:y1=y      :MF_Status:END IF
1059 x1=x:y1=y                                :MF_Status
1060 IF x+1<=16             :x1=x+1:y1=y      :MF_Status:END IF
1061 IF y+1<=12 AND x-1>=1 :x1=x-1:y1=y+1    :MF_Status:END IF
1062 IF y+1<=12             :x1=x  :y1=y+1    :MF_Status:END IF
1063 IF y+1<=12 AND x+1<=16:x1=x+1:y1=y+1 :MF_Status:END IF
1064 END DEFine
```


```
1066 DEFine PROCedure MF_Status
1067 IF Board(x1,y1)>0 AND Board(x1,y1)<200
1068   BLOCK#1,w-2,h-1,4+(x1-1)*w,4+(y1-1)*h,5
1069   CURSOR#1,7+w DIV 21+(x1-1)*w,-1+h DIV 2+(y1-1)*h
1070   STRIP#1,5:INK#1,0:PRINT#1,Board(x1,y1)
1071 END IF
1072 END DEFine
```

```
1074 DEFine PROCedure MF_Mark
1075 IF Board(x,y)=255
1076   Board(x,y)=200:mines=mines-1:flags=flags-1:x1=10+(x-1)*w:y1=6+(y-1)*h
1077   BLOCK#1,w-2,h-1,x1-6,y1-2,5:BLOCK#1,7,4,x1,y1,2:BLOCK#1,2,9,x1,y1,0
1078 END IF
1079  CURSOR#2,90,108:PRINT#2,FILL$(' ',2-LEN(mines))&mines
1080  CURSOR#2,90,128:PRINT#2,FILL$(' ',2-LEN(flags))&flags
1081 END DEFine
```

```
1083 DEFine PROCedure MF_Explode
1084 BLOCK#1,w-2,h-1,4+(x-1)*w,4+(y-1)*h,2
1085 BEEP 0,100,50,1,100,6,15,15:hscale=(16/12)*116
1086 INK#1,3:FOR i=1 TO 12:CIRCLE#1,x*11.2-4,(12-y+1)*9.8-4,1.5*i
1087 PAUSE 80:BEEP:End_Game
1088 END DEFine
```



```
1090 DEFine PROCedure End_Game
1091 CLS#0:gsec=GTime-gsec
1092 mes$(1)='  You left '&mines&' Mines to Find    '
1093 mes$(2)='    after '&gsec&' Seconds of Play    '
1094 mes$(3)='    Leaving '&flags&' marker Flags    '
1095 mes$(4)='     Game Over - Press any Key...     '
1096 IF mines=0:mes$(1)='Well done all Mines Cleared...  '
1097 IF mines<255
1098  FOR m=1 TO 4
1099      IF m>1:Pause 60:cls#0
1100      FOR i=1 TO LEN(mes$(m)):CURSOR#0,490,5:PRINT#0;mes$(m,i):PAUSE 5:PAN#0,-12
1101  END FOR m
1102 END IF
1103 k$=INKEY$(-1):CLS#0:MF_Seed:OldTime=DATE
1104 END DEFine
```

```
1106 DEFine PROCedure MF_Seed
1107 DIM Board(17,13):w=22:h=14:mct=0:mines=0:flags=0:CLS#1
1108 FOR m=1 TO 72:x=RND(1 TO 16):y=RND(1 TO 12):Board(x,y)=255      Random Mine Distribution
1109 FOR y=1 TO 12                                                  Gather Info to Identify Mine Locations
1110  FOR x=1 TO 16                                                  and Number of distributed Mines
1111    IF Board(x-1,y-1)   =255:mct=mct+1
1112    IF Board(x, y-1)    =255:mct=mct+1
1113    IF Board(x+1,y-1)   =255:mct=mct+1
1114    IF Board(x-1,y)     =255:mct=mct+1
1115    IF Board(x+1,y)     =255:mct=mct+1
1116    IF Board(x-1,y+1)   =255:mct=mct+1
1117    IF Board(x,y+1)     =255:mct=mct+1
1118    IF Board(x+1,y+1)   =255:mct=mct+1
1119    IF mct>4:mct=4
1120    IF Board(x,y)=255:mines=mines+1:flags=flags+1:ELSE Board(x,y)=mct
1121    BLOCK#1,w-2,h-1,4+(x-1)*w,4+(y-1)*h,4:mct=0
1122  END FOR x
1123 END FOR y
1124 y=RND(4 TO 8):x=RND(2 TO 4):a=-1+w DIV 2:b=2+h DIV 2:MF_Mark:k=0
1125 END DEFine

1127 DEFine PROCedure MF_Locate                                     End Game Show Mine distribution
1128 FOR y=1 TO 12
1129    FOR x=1 TO 16:x1=x:y1=y:IF Board(x,y)=255 OR Board(x,y)=200:MF_Mark
1130 END FOR y
1131 END DEFine

1133 DEFine PROCedure MF_Detector(ch,x,y)
1134 RESTORE 1127:FOR i=1 TO 6:READ col,tx,ty:MF_Tile 2,col,7,5,tx,ty
1135 DATA 5,x,y-16,4,x+9,y-16,4,x+18,y-16,5,x+3,y-23,5,x+12,y-23,4,x+21,y-23
1136 END DEFine

1138 DEFine PROCedure MF_Tile(ch,col,tw,td,tx,ty)
1139 INK#ch,col:x1=tx:x2=tx+tw:x3=tx+tw+tw/4:x4=tx+tw/4:y1=ty:y2=ty-td
1140 FILL#ch,1:LINE#ch,x1,y1 TO x2,y1 TO x3,y2 TO x4,y2 TO x1,y1:FILL#ch,0
1141 END DEFine

1143 DEFine PROCedure MF_PPE(ch,col,x,y)
1144 INK#ch,col:FILL#ch,1:ARC#ch,x+2,y TO x-2,y,PI/2
1145 LINE#ch TO x-2,y-3 TO x-4,y-4 TO x-4,y-8 TO x-3,y-16
1146 LINE#ch TO x+3,y-16 TO x+4,y-8 TO x+4,y-4 TO x+2,y-3 TO x+2,y
1147 FILL#ch,0:INK#ch,7:FILL#ch,1
1148 LINE#ch,x+1.5,y-1 TO x+1.5,y-2 TO x-1,y-2 TO x-1,y-1 TO x+1.5,y-1
1149 FILL#ch,0:INK#ch,0
1150 LINE#ch,x,y-16 TO x,y-11:LINE#ch,x-4,y-10 TO x-1,y-11
1151 LINE#ch,x-3,y-6 TO x-3,y-8 TO x,y-10:LINE#ch,x+4,y-9 TO x+1,y-12
1152 LINE#ch,x+.5,y-10 TO x+3,y-7:ARC#ch,x-2,y-4 TO x+3,y-4,PI/2
1153 INK#ch,1:FILL#ch,1:CIRCLE#ch,x-1.5,y-17,1.6,.6,PI/2:FILL#ch,0
1154 INK#ch,1:FILL#ch,1:CIRCLE#ch,x+2.5,y-17,1.6,.6,PI/3:FILL#ch,0
1155 INK#ch,3:FILL#ch,1:CIRCLE#ch,x+7,y-22,3,.5,PI/2:FILL#ch,0
1156 INK#ch,0:FILL#ch,0:CIRCLE#ch,x+7,y-22,1.5,.4,PI/2:FILL#ch,0
1157 INK#ch,3:FILL#ch,1:LINE#ch,x+.4,y-10 TO x+7,y-22 TO x+7.4,y-22 TO x+.4,y-10
1158 FILL#ch,0:INK#ch,2:FILL#ch,1:CIRCLE#ch,x,y-11,1.5,.5,PI/3:FILL#ch,0
1159 BLOCK#ch,12,8,72,80,2:BLOCK#ch,2,16,72,80,0
1160 END DEFine
```