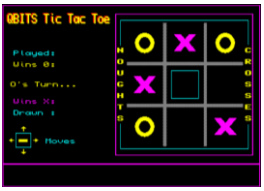
QBITS SuperBASIC Progs

GAMES One

Tic Tac Toe



Mine Detector

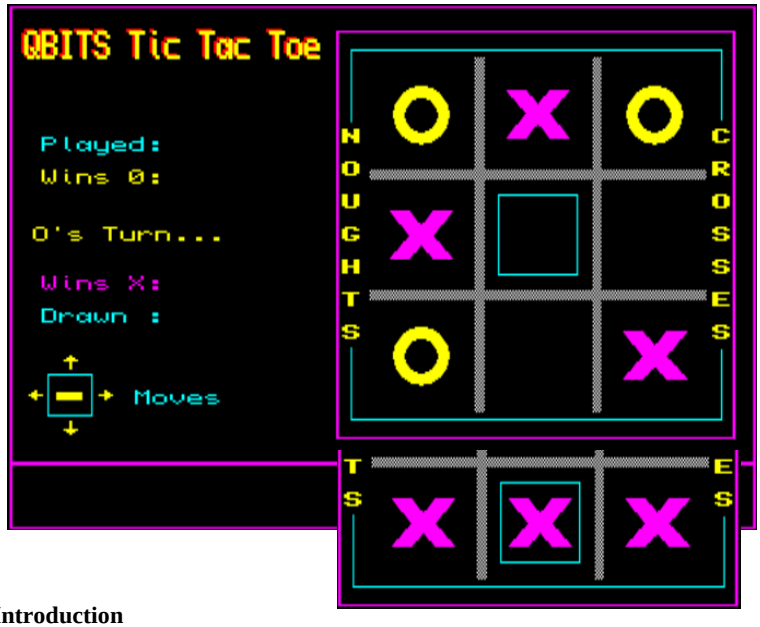
Tile Slider

Conundrum

Darts

Golf

.



TIC TAC TOE Introduction

Described as a deceptively easy game the origins of Nought and Crosses is unknown, but some evidence suggests that something very similar was played by the Ancient Egyptians. Each game has three possible outcomes, a win by either side, by Noughts or Crosses or a draw, in which neither side wins.

QBITS Tic Tac Toe

This QBITS SuperBASIC Game presents a three-by-three Grid. Who goes first the Nought or Crosses player is randomly selected? Both players on average will get an even chance at going first, the choice of Player indicated with **O's** or **X's Turn…** Use the Cursor keys to select a grid position, then press the spacebar and the relevant Nought or Cross is drawn.

Trying to decide if the outcome will be a draw would require analysing multiple combinations. There are 255,168 possible ways of playing Tic Tac Toe. The coding for resultant moves was therefore limited to deciding the state of play for three of a kind either in a horizontal, vertical or diagonal row of cells across the grid.

QBITS Tic Tac Toe Strategy

If you are the first to go a simple **Strategy** is placing your **Nought** or **Cross** in any corner. The aim is to create two possible ways to complete a three in a row. This move will give your opponent the most opportunities to make a mistake and thereby give you the best chance of a win. However, if your opponent places their Nought or Cross in the centre, this will make it all the harder.

Games with two equally matched players invariably ends in a draw (i.e. undecided). The game is too short for any initiative by the second player to force a win, they must rely on the first player making a mistake.

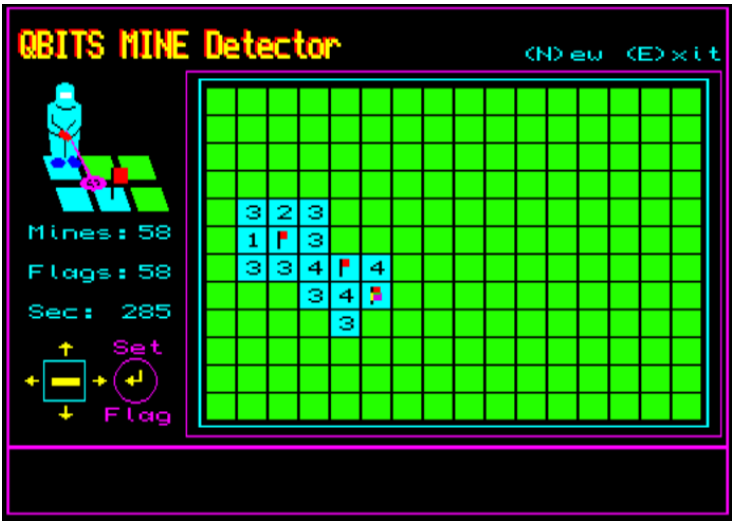
QBITS TIC TAC TOE Code

```
1000 REMark QBITS_TTT_bas [QBITS Tic Tac Toe 2023 Review – QPC2]

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

1004 WHERN ERRor :CONTINUE:END WHEN

1006 REMark Import QBITSConfig Settings – QPC2
1007 OPEN_IN#9,'ram2_QBITSConfig':INPUT#9,gx\gy\dn$:CLOSE#9

1010 Init_win:chk=0:QBITS_TTT

1012 DEFine PROCedure Init_win                                  Create screen layout
1013 WINDOW#2,512,224,gx,gy          :PAPER#2,0:BORDER#2,1,3:CLS#2
1014 WINDOW#1,274,200,gx+224,gy+12:PAPER#1,0:BORDER#1,1,3:CLS#1
1015 WINDOW#0,512, 32,gx,gy+224   :PAPER#0,0:BORDER#0,1,3:CLS#0
1016 CSIZE#2,2,1:OVER#2,1:SCALE#1,100,0,0
1017 INK#2,2:FOR i=0 TO 1:CURSOR#2,4+i,8:PRINT#2,'QBITS Tic Tac Toe'
1018 INK#2,6:FOR i=0 TO 1:CURSOR#2,6+i,9:PRINT#2,'QBITS Tic Tac Toe'
1019 CSIZE#2,2,0:Str1$='NOUGHTS':Str2$='CROSSES' :c=1:r=2
1020 FOR a=1 TO 7
1021   FOR b=0 TO 2:CURSOR#2,224+b,42+a*16:PRINT#2,Str1$(a)
1022   FOR b=0 TO 2:CURSOR#2,478+b,42+a*16:PRINT#2,Str2$(a)
1023 END FOR a
1024 OVER#2,0:m=0:OG=0:XG=0:INK#1,5:LINE#1,3,22 TO 3,4 TO 98, 4 TO 98,22
1025 RESTORE 1021:PG=0:FG=0:LINE#1,3,78 TO 3,96 TO 98,96 TO 98,78:INK#1,248
1026 FOR i=1 TO 8:READ col,x,y,str$:INK#2,col:CURSOR#2,x,y:PRINT#2,str$
1027 DATA 6,33,168,'⬆',6,33,200,'⬇',6,9,184,'⬅  ➡'
1028 DATA 5,82,186,'Moves',5,18,62,'Played:',6,18,78,'Wins 0:'
1029 DATA 3,18,130,'Wins X:',5,18,146,'Draw  :':BLOCK#2,18,4,30,188,6
1030 FOR i=1 TO 4
1031   READ x1,y1,x2,y2,x3,y3,x4,y4
1032   FILL 1:LINE x1,y1 TO x2,y2 TO x3,y3 TO x4,y4 TO x1,y1:FILL 0
1033 END FOR i
1034 DATA 8,36,94,36,94,34,8,34, 8,66,94,66,94,64,8,64
1035 DATA 35,6,35,94,37,94,37,6, 65,6,65,94,67,94,67,6
1036 INK#2,5: LINE#2,8,10 TO 8,19 TO 18,19 TO 18,10 TO 8,10
1037 END DEFine

1039 DEFine PROCedure QBITS_TTT                                 Game Control
1040 REPeat loop
1041   IF chk=0:INK#2,5:CURSOR#2,12,106:PRINT#2,"(N)EW / (Q)uit"
1042   IF chk=1:INK#2,6:CURSOR#2,12,106:PRINT#2,"O's Turn...  "
1043   IF chk=2:INK#2,3:CURSOR#2,12,106:PRINT#2,"X's Turn...  "
1044   x=11+c*30:y=r*30:Tile_Hgl x,y:k=CODE(INKEY$(-1)):Tile_Hgl x,y   calculate x-y from c-r
1045   SELect ON k
1046     =27:MODE 4:CSIZE#2,0,0:CSIZE#0,0,0:CLS#0:PRINT#0,'Bye...':STOP
1047     =69,101:IF chk=0:LRUN dn$:ELSE chk=0  :REMark (E)xit Game
1048     =77,110:Tile_CLS:chk=RND(1 TO 2)      :REMark (N)ew Game
1049     =192:c=c -1:IF c<0:c=0                :REMark Left
1050     =200:c=c+1:IF c>2:c=2                 :REMark Right
1051     =208:r =r+1:IF r >3:r=3               :REMark Up
1052     =216:r =r -1:IF r <1:r=1              :REMark Down
1053     =  32:RESTORE 1076:IF chk=1:Nought:ELSE IF chk=2:Cross
1054   END SELect
1055 END REPeat loop
1056 END DEFine
```

```
1058 REMark QBITS TicTackToe Graphics

1060 DEFine PROCedure Tile_Hgl(x,y)
1061 INK 5:OVER -1:LINE x,y TO x+20,y TO x+20,y-20 TO x,y-20 TO x,y:OVER 0
1062 END DEFine

1064 DEFine PROCedure Nought
1065 IF Grid(c+1,r)=0:Grid(c+1,r)=79:chk=2:x=x+10:y=y-10:ELSE RETurn
1066 INK 6:FILL 1:CIRCLE x,y,7:FILL 0:INK 0:FILL 1:CIRCLE x,y,4:FILL 0
1067 BEEP 2000,5,10,0,0,0,0,0:Tile_Chk 79
1068 END DEFine

1070 DEFine PROCedure Cross
1071 IF Grid(c+1,r)=0:Grid(c+1,r)=88:chk=1:ELSE RETurn
1072 INK 3:x=x+10:y=y-10:FILL 1:LINE x-8,y+6 TO x-3,y+6 TO x+2,y TO x-3,y-6
1073 LINE TO x-8,y-6 TO x-3,y TO x-8,y+6:FILL 0:FILL 1:LINE x+8,y+6 TO x+3,y+6
1074 LINE TO x-2,y TO x+3,y-6 TO x+8,y-6 TO x+3,y TO x+8,y+6:FILL 0
1075 BEEP 2000,5,10,0,0,0,0,0:Tile_Chk 88
1076 END DEFine

1078 REMark QBITS TicTackToe Game Checks
```

For three in a row, there are eight possible combinations. A **FOR** loop **READ**'s the combinations as **DATA** sets and compares the three Grid entries. If they all test the same ie. all Noughts or all Crosses then a win is declared and the score is updated.

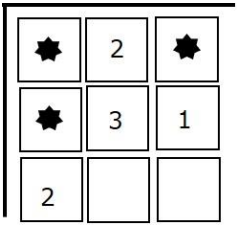
```
1080 DEFine PROCedure Tile_Chk(n)
1081 FOR i=1 TO 8
1082   READ a,b,c,r,d,e
1083   IF Grid(a,b)=n AND Grid(c,r)=n AND Grid(d,e)=n:Tile_Win:RETurn
1084 END FOR i
1085 m=m+1:IF m=9:m=0:n=0:Tile_Win
1086 DATA 1,1,1,2,1,3, 1,1,2,2,3,3, 1,1,2,1,3,1, 1,2,2,2,3,2
1087 DATA 3,3,3,2,3,1, 3,3,2,3,1,3, 1,3,2,2,3,1, 2,1,2,2,2,3
1088 END DEFine

1090 DEFine PROCedure Tile_Win                                  Displays Updated Score
1091 chk=0  :PG=PG+1:INK#2,5:CURSOR#2,124, 64:PRINT#2,FILL$(' ',2-LEN(PG))&PG
1091 IF n=79:OG=OG+1:INK#2,6:CURSOR#2,124, 80:PRINT#2,FILL$(' ',2-LEN(OG))&OG
1093 IF n=88:XG=XG+1:INK#2,3:CURSOR#2,124,132:PRINT#2,FILL$(' ',2-LEN(XG))&XG
1094 IF n=0 :FG=FG+1:INK#2,5:CURSOR#2,124,148:PRINT#2,FILL$(' ',2-LEN(FG))&FG
1095 BEEP 5000,20,12,40,8,8,0,0
1096 END DEFine

1098 DEFine PROCedure Tile_CLS
1099 DIM Grid(3,3):m=0:n=0:c=1:r=2:RESTORE 1093:INK 0
1100 FOR i=1 TO 9
1101   READ x,y,x1,y1:FILL 1:LINE x,y TO x1,y TO x1,y1 TO x,y1 TO x,y:FILL 0
1102 END FOR i
1113 DATA 11,30,31,10, 41,30,61,10, 71,30,91,10, 11,60,31,40, 41,60,61,40
1104 DATA 71,60,91,40,11,90,31,70, 41,90,61,70,71,90,91,70
1105 END DEFine
```



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 **Mark's** 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.

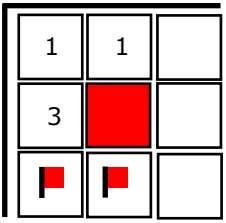


The player starts on a safe square and **Click's** to reveal a number on the square occupied and then some of the surrounding squares. The **numbers represent** how many mines are adjacent to the current square. For example, if a square has a "**3**" on it, then there are 3 mines placed in the surrounding 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.

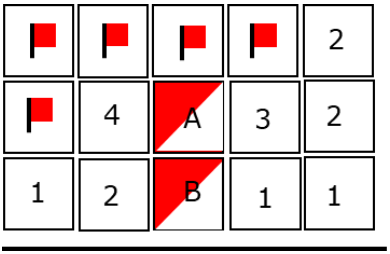
Minesweeper Logic or Probability
This game can be considered to be played as one of **Logic** or of **Probability**. Although technically Probability will include some level of Logic. If Logic suggests a mine to occupy a position, then in all Probability it has considerable certainty of being correct.

Local Probabilities
The squares shown have 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.

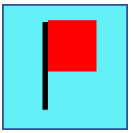
QBITS Mine Detector Code

```
1000 REMark QBITS_MDETR_bas_[QBITS Mine Detector 2023 Review - QPC2]

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$(20)) :BLOCK#1,8,4,(a+(x1-1)*w),(b+(y1-1)*h),6:OVER#1,0
1039  SELect ON k
1040    =192:x=x -1:BEEP 400:IF x< 1:x=1
1041    =200:x=x+1:BEEP 400:IF x>16:x=16
1042    =208:y=y -1:BEEP 400:IF y< 1:y=1
1043    =216:y=y+1:BEEP 400:IF y>12:y=12
1044    = 69,101:LRUN dn$                                    :REMark (E)xit
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
```

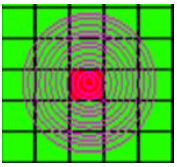
```
1054 DEFine PROCedure MF_Show
1055 IF y-1>=1 AND x-1>=1   :x1=x-1:y1=y-1    :MF_Status:END IF
1056 IF y-1>=1              :x1=x  :y1=y-1    :MF_Status:END IF
1057 IF y-1>=1 AND x+1<=16 :x1=x+1:y1=y-1    :MF_Status:END IF
1058 IF x-1>0               :x1=x-1:y1=y      :MF_Status:END IF
1059 x1=x:y1=y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

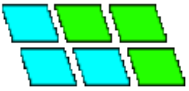


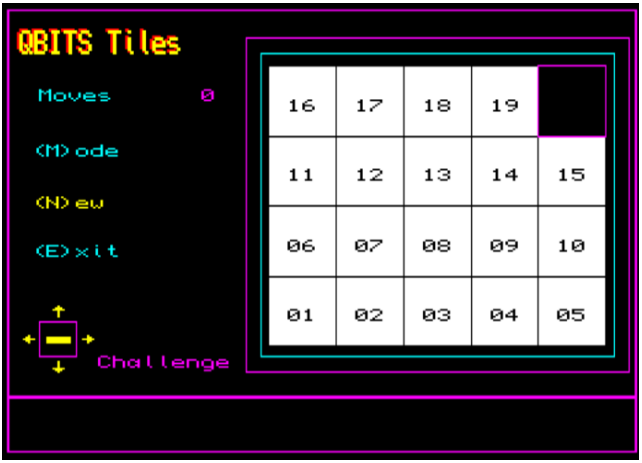
```
1083 DEFine PROCedure MF_Explode
1084 BLOCK#1,w-2,h-1,4+(x-1)*w,4+(y-1)*h,2
1085 BEEP 0,100,50,1,100,6,15,15:hscale=(16/12)*116
1086 INK#1,3:FOR i=1 TO 12:CIRCLE#1,x*11.2-4,(12-y+1)*9.8-4,1.5*i
1087 PAUSE 80:BEEP:End_Game
1088 END DEFine
```



```
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
```





Sliding Tile Puzzle

The origins of the Sliding Puzzle are mostly credited to Noyes Chapman whose invention of the 15-Puzzle started a craze in the early 1880's. Sliding Puzzles continued to be popular in the 1950s and through the 1980s when letters were employed to spell out popular phrases. The 1980's Popularity was further enhanced by the Rubik Cube a Rotational three-dimensional version of the Sliding Tile Puzzle. Each cube had different colours on each side and had to be matched in groups to one of six sides.

Early puzzles were tokens on a flat board; the Rules prohibited lifting any piece from the board, and the challenge was to find moves within the two-dimensional plane to solve the Puzzle. Later Manufactured versions used togue and grove designs to interlink the sliding tiles mechanically and so facilitated these restrictions.

Sam Loyd introduced his version of the 15-tile Puzzle in 1886. A square grid with 15 Tile squares and one blank space. The game starts with the Tiles in some Random order then following the rules the aim is to slide the Tiles around until they are arranged in their Home or Goal sequence as shown on the right. Interest was further fuelled by Loyd offering a $1,000 prize for anyone who could achieve a solution to a particular combination namely with 14 and 15 inverted. No one claimed the prize as it proved to be impossible.

The game can be played with any size grid, not just a 4 by 4 as in the original puzzle and pieces may be imprinted with colours, patterns, sections of a larger picture (like a jigsaw puzzle), numbers, or letters. Although the original had evenly sized tiles or squares, some can have two or more different sized tiles.

QBITS Sliding Tile Challenge

The QBITS Sliding Tile Grid is a two dimensional Five by Four. The Tiles are set one to five on the lowest row and ending on row four with the Blank Tile number twenty at the top most right.

16	17	18	19	
11	12	13	14	15
06	07	08	09	10
01	02	03	04	05

Only half of the 2,432,902,000,000,000,000 - two quintillion, four hundred and thirty-two quadrillion, nine hundred and two trillion possible combinations are solvable when abiding by the rules.

QBITS Solvable Tile Configurations

Choosing a 5x4 Grid makes it simpler to determine if a Sliding Puzzle is Solvable. As a General Rule a Grid with an odd number of columns and even number of Inversions is solvable. If the Tile numbers of the displaced order are set out as a single row an Inversion occurs whenever **a>b** in any **a-b** combination of the Home Tile Column Row sequence. The Blank position is treated as 0.

For each New Game. The Tiles are Randomly Shuffled to set a new order of displacement.

16 07	17 08	18 15	19 19	03
11 16	12 17	13 12	14 02	15 11
06 10	07 14		09 04	10 18
01 06	02 09	03 13	04 01	05 05

In this example QBITS Tile 01 holds the number 6
a>b Inversions 6>1 6>5 6>4 6>2 6>3 =5

Looking at QBITS Tile 02 this holds the number 9
a>b Inversions 9>1 9>5 9>4 9>2 9>7 9>8 9>3=7

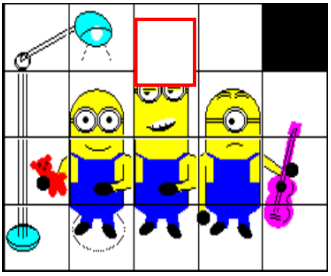
Blank

Tile String: 06 09 13 01 05 10 14 00 04 10 16 17 12 02 11 07 08 15 19 03
I**nversions**: +5 +7 +11 +0 +4 +4 +7 +2 +4 +6 +6 +5 +0 +3 +1 +1 +1 +1 0 = 68

The Inversions result of 68 is an Even number so this 5x4 configuration is Solvable.

QBITS Sliding Tile Picture

For the challenge of replacing Numbered Tile with an Image I used Vector Graphics. Each Tile is drawn with a different Graphic combination that when correctly sequenced creates a picture. This will in part explain use of the S/SuperBASIC Graphic coordinates and why QBITS Tile numbers begin bottom left, progressing to top right.



QBITS Tile Moves

Press (**N**) for New Game and **Cursor keys** to move highlight over an adjacent Tile to the Blank (black) Tile space. Pressing **Spacebar** then Swaps the Tile positions. Use (**M**) key to switch between Numbers and Picture Mode use (E) to Exit Game.

QBITS Sliding Tile Permutations

The act of changing the arrangement of a given number of elements is a permutation. The number of moves to position a Tile in its correct row and column is **n** the number of tiles in the set. A maximum number of moves for each tile is **n**-1 and each switch involves two Tile positions, therefore this equates to $\mathbf{n}^2(\mathbf{n}\text{-}1)$. Therefor the maximum number of moves for a QBITS 5x4 Grid = 16*2*(16-1) = 480. Taking the minimum number of moves as equal to the number of inversions then the required moves by an average player should fall somewhere between.

After reviewing a series of games, the average number of Inversion was around 200, it suggested an inexperienced player might take twice this number of moves to solve the Puzzle. Therefore, the smallest number of moves taken will be dependent on the number of Inversions and Skill of the player.

QBITS Tile Strategy

Sliding Puzzles can be incredibly difficult to solve and there is no universal rule, it is more about developing an intuition on how you move the pieces around the grid. The main mathematical idea in solving the Sliding Puzzle Challenge is recursion, performing a task by repeatedly carrying out a basic procedure.

QBITS Solving the Puzzle

One simplifying method is to reduce the Puzzle grid size into smaller ones.

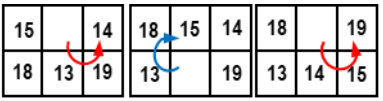
16	17	18	19	
11	12	13	14	15
06	07	08	09	10
01	02	03	04	05

Steps 1: Position Tiles correctly for Column One: 01,06,11,16. Reduces Grid to 4x4
Steps 2: Complete Row One: 01 02 03 04 05 and then Column Two: 02 07 12 17
Steps 3: Complete Row Two: 06 07 08 09 10 and then Column Three: 03 08 13 18
Steps 4: Solve the remaining 2x2 Grid at top right of Puzzle board.

Placing a Tile in its correct position is achieved by cycling the Blank and numbered Tiles around a group of 2x2 : 2x3 : 2x4 : 3x2 : 4x2 in a clockwise or anticlockwise motion.

15	13	
18	19	14

The last moves of a 2x2 may not always be possible and more often than not you are left with a 2x3 Puzzle to solve.



18	19	
13	14	15

QBITS Sliding Tile code

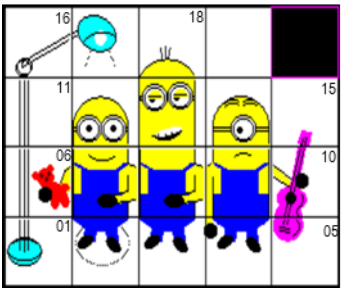
```
1000 REMark QBITS_Tiles_bas [QBITS Tiles 2023 Review - QPC2]

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

1004 WHEN ERRor :CONTINUE:END WHEN

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

1010 DIM Tile(20,3):cp=3:Init_win:QBITS_Tiles

1012 DEFine PROCedure Init_win
1013 WINDOW#2,512,224,gx,gy             :PAPER#2,0:BORDER#2,1,3:CLS#2:INK#2,3
1014 WINDOW#1,280,167,gx+212,gy+28:SCALE#1,164,0,0:CSIZE#1,2,0
1015 WINDOW#0,512, 32,gx,gy+224       :BORDER#0,1,3:PAPER#0,0:CLS#0
1016 LINE#2,64,6 TO 168,6 TO 168,93 TO 64,93 TO 64,6:INK#2,5
1017 LINE#2,68,10 TO 164,10 TO 164,89 TO 68,89 TO 68,10
1018 CSIZE#2,2,1:OVER#2,1
1019 INK#2,2:FOR i=0 TO 1:CURSOR#2,4+i,8:PRINT#2,'QBITS Tiles'
1020 INK#2,6:FOR i=0 TO 1:CURSOR#2,6+i,9:PRINT#2,'QBITS Tiles'
1021 CSIZE#2,2,0:OVER#2,0:RESTORE 1016
1022 FOR i=1 TO 8:READ col,x,y,str$:INK#2,col:CURSOR#2,x,y:PRINT#2,str$
1023 DATA 5,22,44,'Moves',5,16,76,'(M)ode',6,16,106,'(N)ew',5,16,134,'(E)xit'
1024 DATA 3,70,198,'Challenge',6,33,168,'¾',6,33,200,'¿',6,9,184,'¼  ½'
1025 INK#2,3:LINE#2,8,10 TO 8,19 TO 18,19 TO 18,10 TO 8,10:BLOCK#2,20,4,30,188,6
1026 END DEFine

1028 DEFine PROCedure QBTiles
1029 t=0:Tsel=0:Init_Tiles:Set_Tiles:tc=4:tr=3:m=0
1030 REPeat G_lp
1031   INK#1,3:Tile_Hgl 1,40,40,tc*40,tr*40
1032   IF m<999:CURSOR#2,130,44:PRINT#2,FILL$(' ',3-LEN(m))&m
1033   k=CODE(INKEY$(-1)):INK#1,0:Tile_Hgl 1,40,40,tc*40,tr*40
1034   SELect ON k
1035     = 27:MODE 4:CSIZE#2,0,0:INK#2,7:CSIZE#0,0,0:CLS#0:PRINT#0,'Bye...':STOP
1036     = 32:Tile_Chg:IF chk=1:m=m+1:Tile_Draw 1,ts:Tile_Draw 1,tn:Game_Chk
1037     = 69,101:QExit:BLOCK#2,40,10,96,134,0
1038     = 77,109:IF Tsel=0:Tsel=1:Set_Tiles:ELSE Tsel=0:Set_Tiles
1039     = 78,110 :t=0:Init_Tiles:Set_Tiles:PAUSE 50:Sort_Tiles:Set_Tiles
1040     =192:tc=tc -1:IF tc<0:tc=0
1041     =200:tc=tc+1:IF tc>4:tc=4
1042     =208:tr =tr+1:IF tr >3:tr=3
1043     =216:tr =tr -1:IF tr <0:tr=0
1044   END SELect
1045 END REPeat G_lp
1046 END DEFine8

1048 DEFine PROCedure Game_Chk
1049 cnt=0:FOR i=1 TO 20:IF Tile(i,1)=i:cnt=cnt+1
1050 IF m>999 OR cnt=20
1051    CURSOR#2,90,106:PRINT#2,'Game End':PAUSE
1052    BLOCK#2,100,10,90,106,0:BLOCK#2,40,10,132,184,0:m=0:t=0
1053 END IF
1054 END DEFine
```

```
1056 DEFine PROCedure QExit
1057 CURSOR#2,96,134:PRINT#2,'Y/N':PAUSE:IF KEYROW(5)=64:LRUN dn$
1058 END DEFine

1060 DEFine PROCedure Tile_Hgl(ch,w,d,x,y)
1061 LINE#ch,x,y TO x+w,y TO x+w,y+d TO x,y+d TO x,y
1062 END DEFine

1064 DEFine PROCedure Tile_Chg
1065 tn=1+tc+tr*5:chk=0       :IF Tile(tn,1)=20       :RETurn
1066 IF tc>0:ts=tn-1           :Tile_Chk ts:IF chk=1:RETurn
1067 IF tc<4:ts=tn+1           :Tile_Chk ts:IF chk=1:RETurn
1068 IF tr>0:ts=1+tc+(tr-1)*5  :Tile_Chk ts:IF chk=1:RETurn
1069 IF tr<3:ts=1+tc+(tr+1)*5 :Tile_Chk ts:IF chk=1:RETurn
1070 END DEFine

1072 DEFine PROCedure Tile_Chk(ts)
1073 IF Tile(ts,1)=20:Tile(ts,1)=Tile(tn,1):Tile(tn,1)=20:chk=1
1074 END DEFine

1076 DEFine PROCedure Tile_Draw(ch,ts)
1077 IF Tile(ts,1)<20:STRIP#ch,7:INK#ch,7:ELSE STRIP#ch,0:INK#ch,0
1078 BEEP 2000,5,10,0,0,0,0,0:t$=Tile(ts,1):x=Tile(ts,2):y=Tile(ts,3)
1079 FILL#ch,1:LINE#ch,x,y TO x+40,y+1 TO x+40,y+40 TO x,y+40 TO x,y:FILL#ch,0
1080 INK#ch,0:Tile_Hgl 1,40,40,x,y:IF Tsel=1:Tile_Pic ts,x,y
1081 IF Tsel=0:CURSOR#ch,x,y,14,-22:PRINT#ch,FILL$('0',2-LEN(t$))&t$
1082 END DEFine

1084 DEFine PROCedure Init_Tiles
1085 FOR r=0 TO 3
1086   FOR c=0 TO 4:t=t+1:Tile(t,1)=t:Tile(t,2)=c*40:Tile(t,3)=r*40
1087 END FOR r
1088 END DEFine

1090 DEFine PROCedure Set_Tiles
1091 BLOCK#2,64,10,120,164,0:ch=1:PAPER#ch,0:CLS#ch:FOR t=1 TO 20:Tile_Draw 1,t
1092 END DEFine

1094 DEFine PROCedure Sort_Tiles
1095 FOR t=20 TO 3 STEP -1
1096   ran=RND(1 TO t-1):temp=Tile(t,1):Tile(t,1)=Tile(ran,1):Tile(ran,1)=temp
1097 END FOR t
1098 m=0:ct=0:Solvable
1099 FOR t=1 TO 20:IF Tile(t,1)=20:tc=Tile(t,2)/40:tr=Tile(t,3)/40
110 END DEFine

1102 DEFine PROCedure Solvable
1103 FOR t=1 TO 19
1104   IF Tile(t,1)=20:NEXT t:ELSE FOR c=t+1 TO 20:IF Tile(t,1)>Tile(c1,1):ct=ct+1
1105 END FOR t
1106 CURSOR#2,130,184:PRINT#2,INT(ct+ct/cp):IF ct MOD 2>0:Sort_Tiles
1107 END DEFine
```

QBITS Minions Graphics

Creating a Tiled picture gave the opportunity to try out Vector Graphics instead of using Pixel Bitmaps. The result will have variations of picture quality across the range of QL Platforms and will perform better on those that run at speeds of 10 or more times that of the original QL hardware.

```
1109 DEFine PROCedure Tile_Pic(ts,x,y)
1110 ch=1:tp=Tile(ts,1)
1111 SELect ON tp
1112  = 1:ML5 x,y
1113  = 2:ML1 x,y:MB2 x,y
1114  = 3:MB2 x,y
1115  = 4:MB2 x+5,y:MA3 x,y
1116  = 5:MG3 x,y
1117  = 6:MT x+11,y-6:ML4 x,y
1118  = 7:MB1 x,y:MS1 x,y:MA2 x,y
1119  = 8:MB1 x,y:MA2 x,y
1120  = 9:MB1 x+5,y:MS2 x,y:MA1 x,y
1121  =10:MG2 x,y
1122  =11:ML4 x,y
1123  =12:MH3 x,y:ME1 10,10,x,y:ME2 10,10,x,y:ME1 25,10,x,y:ME2 25,10,x,y
1124  =13:MH2 x,y:ME1 10,28,x,y:ME3 10,28,x,y:ME1 25,29,x,y:ME3 25,29,x,y:MS3 x,y
1125  =14:MH3 x+5,y:ME1 19,10,x+5,y:ME2 17,10,x+5,y-1:MH4 x+5,y+1
1126  =15:MG1 x,y
1127  =16:ML3 x,y
1128  =17:ML2 x,y
1129  =18:MH1 x,y
1130 END SELect
1131 END DEFine

1150 REMark Vector Graphics
```



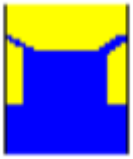
Note: Sliding Tiles presented as a Picture. The Vector Graphics are draw as a combination of individual parts.

.

```
1152 DEFine PROCedure MH1(x,y)          :REMark Head Top
1152 INK#ch,6:FILL#ch,1:LINE#ch,x+1,y+1 TO x+34,y+1
1153 ARC#ch TO x+1,y+1,PI/1.5:FILL#ch,0:INK#ch,0
1154 ARC#ch,x+1,y+1 TO x+34,y+1,-PI/1.5:LINE#ch,x+16,y+10 TO x+16,y+18
1156 LINE#ch,x+14,y+10 TO x+13,y+17:LINE#ch,x+18,y+10 TO x+19,y+17
1157 END DEFine
```

```
1159 DEFine PROCedure MH2(x,y)          :REMark Head Long
1160 INK#ch,6:FILL#ch,1:LINE#ch,x+1,y+1 TO x+34,y+1 TO x+34,y+39
1161 LINE#ch TO x+34,y+39 TO x+1,y+39 TO x+1,y+1:FILL#ch,0
1162 INK#ch,0:LINE#ch,x+34,y+1 TO x+34,y+39
1163 LINE#ch,x+1,y+27 TO x+34,y+27 TO x+34,y+29 TO x+1,y+29 TO x+1,y+27
1164 FILL#ch,0:REMark MS3 x,y
1165 END DEFine
```

```
1167 DEFine PROCedure MH3(x,y)              :REMark Head Average
1168 INK#ch,6:FILL#ch,1:ARC#ch,x+1,y+12 TO x+34,y+12,-PI
1169 LINE#ch TO x+34,y+1 TO x+1,y+1 TO x+1,y+12:FILL#ch,0:INK#ch,0
1170 LINE#ch,x+1,y+1 TO x+1,y+12:ARC#ch TO x+34,y+12,-PI:LINE#ch TO x+34,y+1
1171 LINE#ch,x+1,y+11 TO x+34,y+11 TO x+34,y+9 TO x+1,y+9 TO x+1,y+11
1172 END DEFine

1174 DEFine PROCedure MH4(x,y)              :REMark Hair
1175 INK#ch,0
1176 ARC#ch,x+8,y+24 TO x+17,y+25,-PI/2:ARC#ch,x+19,y+25 TO x+28,y+24,-PI/2
1177 ARC#ch,x+8,y+22 TO x+17,y+23,-PI/2:ARC#ch,x+19,y+23 TO x+28,y+22,-PI/2
1178 END DEFine

1180 DEFine PROCedure ME1(w,d,x,y)          :REMark Eye Cover
1181 INK#ch,7:FILL#ch,1:CIRCLE#ch,x+w,y+d,7:FILL#ch,0
1182 INK#ch,0:CIRCLE#ch,x+w,y+d,7.4:CIRCLE#ch,x+w,y+d,6
1183 END DEFine

1185 DEFine PROCedure ME2(w,d,x,y)          :REMark Eye
1186 FILL#ch,1:CIRCLE#ch,x+w,y+d,1.6:FILL#ch,0
1187 END DEFine
:
1189 DEFine PROCedure ME3(w,d,x,y)          :REMark Eye Lid
1190 INK#ch,0:ME2 w,d-1,x-2,y:INK#ch,6:FILL#ch,1
1191 ARC#ch,x+w-4,y+d TO x+w+4,y+d,-PI:LINE#ch TO x+w-4,y+d:FILL#ch,0
1192 INK#ch,0:LINE#ch,x+w-4,y+d+1 TO x+w+4,y+d+1
1193 END DEFine

1195 DEFine PROCedure MS1(x,y)              :REMark Smile
1196 INK#ch,0:ARC#ch,x+10,y+35 TO x+26,y+35,PI/2
1197 END DEFine

1199 DEFine PROCedure MS2(x,y)              :REMark Frown
1200 INK#ch,0:ARC#ch,x+18,y+34 TO x+28,y+34,-PI/2
1201 END DEFine

1203 DEFine PROCedure MS3(x,y)              :REMark Teeth
1204 FILL#ch,1:LINE#ch,x+8,y+6 TO x+26,y+9:ARC#ch TO x+9,y+6,-PI/2:FILL#ch,0
1205 INK#ch,7:FILL#ch,1:CIRCLE#ch,x+18,y+8,8,.2,-PI/2.4:FILL#ch,0
1206 INK#ch,0:LINE#ch,x+16,y+7 TO x+16,y+4:LINE#ch,x+20,y+8 TO x+19,y+4
1207 LINE#ch,x+12,y+7 TO x+12,y+4
1208 END DEFine

1210 DEFine PROCedure MB1(x,y)              :REMark Body Trunk
1211 FILL#ch,1:INK#ch,6
1212 LINE#ch,x+1,y+8 TO x+34,y+8 TO x+34,y+38 TO x+1,y+38 TO x+1,y+8
1213 FILL#ch,0:FILL#ch,1:INK#ch,1
1214 LINE#ch,x+1,y+1 TO x+34,y+1 TO x+34,y+12 TO x+27,y+12 TO x+27,y+28
1215 LINE#ch TO x+7,y+28 TO x+7,y+12 TO x+1,y+12 TO x+1,y+1:FILL#ch,0:FILL#ch,1
12161 LINE#ch,x+1,y+30 TO x+4,y+30 TO x+10,y+26 TO x+8,y+26 TO x+1,y+30
1217 FILL#ch,0:FILL#ch,1
1218 LINE#ch,x+31,y+30 TO x+34,y+30 TO x+28,y+26 TO x+26,y+26 TO x+31,y+30
1219 FILL#ch,0:INK#ch,0:LINE#ch,x+1,y+1 TO x+1,y+39:LINE#ch,x+34,y+1 TO x+34,y+39
1220 END DEFine
```



```
1222 DEFine PROCedure MB2(x,y)              :REMark Body Feet
1223 INK#ch,1:FILL#ch,1:LINE#ch,x+1,y+39 TO x+34,y+39
1224 ARC#ch TO x+1,y+39,-PI/2:FILL#ch,0:FILL#ch,1:LINE#ch,x+8,y+34 TO x+14,y+34
1225 LINE#ch TO x+14,y+26 TO x+8,y+26 TO x+8,y+34:FILL#ch,0:FILL#ch,1
1226 LINE#ch,x+26,y+34 TO x+20,y+34 TO x+20,y+26 TO x+26,y+26 TO x+26,y+34
1227 FILL#ch,0:FILL#ch,1:INK#ch,0:CIRCLE#ch,x+9,y+25,5,.6,-PI/3
1228 FILL#ch,0:FILL#ch,1:CIRCLE#ch,x+24,y+24,5,.6,PI/4:FILL#ch,0
1229 END DEFine
```

```
1231 DEFine PROCedure MA1(x,y)              :REMark Arm Straight
1232 INK#ch,6:FILL#ch,1:LINE#ch,x+1,y+28 TO x+1,y+1 TO x+5,y+1 TO x+5,y+24
1233 LINE#ch TO x+5,y+24 TO x+5,y+30 TO x+1,y+28:FILL#ch,0
1234 LINE#ch,x+6,y+28 TO x+6,y+20:INK#ch,0:LINE#ch,x+1,y+28 TO x+6,y+30
1235 END DEFine
```

```
1237 DEFine PROCedure MA2(x,y)              :REMark Arm Left
1238 INK#ch,6:FILL#ch,1
1239 LINE#ch,x+34,y+30 TO x+39,y+28 TO x+39,y+9 TO x+20,y+6 TO x+20,y+10
1240 LINE#ch TO x+34,y+12 TO x+34,y+30:FILL#ch,0:FILL#ch,1:INK#ch,0
1241CIRCLE#ch,x+21,y+9,6,.6,PI/2:FILL#ch,0:LINE#ch,x+34,y+30 TO x+39,y+28
1242 LINE#ch,x+20,y+6 TO x+39,y+9:LINE#ch,x+20,y+10 TO x+34,y+14 TO x+34,y+24
1243 END DEFine
```

```
1245 DEFine PROCedure MA3(x,y)              :REMark Arm Right
1246 INK#ch,6:FILL#ch,1:LINE#ch,x+1,y+39 TO x+4,y+39 TO x+4,y+36
1247 LINE#ch TO x+1,y+36 TO x+1,y+39:FILL#ch,0
1248 INK#ch,0:FILL#ch,1:CIRCLE#ch,x+3,y+32,4:FILL#ch,0
1249 END DEFine
```

```
1251 DEFine PROCedure MT(x,y)               :REMark Teddy Bear
1252 INK#ch,6:FILL#ch,1
1253 LINE#ch,x+29,y+34 TO x+29,y+20 TO x+22,y+20 TO x+26,y+32 TO x+29,y+34
1254 FILL#ch,0:INK#ch,0:LINE#ch,x+29,y+34 TO x+26,y+32 TO x+22,y+21:INK#ch,2
1255 FILL#ch,1:CIRCLE#ch,x+14,y+28,5:FILL#ch,0          :REMark head
1256 FILL#ch,1:CIRCLE#ch,x+ 8,y+29,2:FILL#ch,0          :REMark ear 1
1257 FILL#ch,1:CIRCLE#ch,x+17,y+33,2:FILL#ch,0          :REMark ear 2
1258 FILL#ch,1:CIRCLE#ch,x+20,y+20,8,.6,PI/3:FILL#ch,0  :REMark body
1259 FILL#ch,1:CIRCLE#ch,x+26,y+22,3,.6,PI/2:FILL#ch,0  :REMark arm 1
1260 FILL#ch,1:CIRCLE#ch,x+20,y+14,4,.6,PI:FILL#ch,0    :REMark leg 1
1261 FILL#ch,1:CIRCLE#ch,x+26,y+14,3,.8,PI:FILL#ch,0:INK#ch,0 :REMark hand
1262 FILL#ch,1:CIRCLE#ch,x+14,y+18,4:FILL#ch,0          :REMark leg 2
1263 CIRCLE#ch,x+12,y+29,1:CIRCLE#ch,x+16,y+30,1:CIRCLE#ch,x+15,y+27,1
1264 END DEFine
```

```
1266 DEFine PROCedure ML1(x,y)              :REMark Lamp Highlight
1267 INK#ch,248:CIRCLE#ch,x+18,y+25,16,.8,PI/2
1268 END DEFine
```

```
1270 DEFine PROCedure ML2(x,y)              :REMark Lamp Head
1271 INK#ch,0:LINE#ch,x+1,y+25 TO x+10,y+30 TO x+10,y+27 TO x,y+22
1272 INK#ch,5:FILL#ch,1:CIRCLE#ch,x+15,y+25,12,.8,PI/3:FILL#ch,0
1273 INK#ch,7:FILL#ch,1:CIRCLE#ch,x+17,y+22,4,.8,PI:FILL#ch,0:INK#ch,0
1274 CIRCLE#ch,x+15,y+25,12,.8,PI/3:ARC#ch,x+7,y+19 TO x+26,y+23,-PI/2
1275 INK#ch,248:LINE#ch,x+12,y+15 TO x+8,y+5:LINE#ch,x+22,y+15 TO x+26,y+5
1276 END DEFine
```

```
1278 DEFine PROCedure ML3(x,y)              :REMark Lamp Arm
1279 INK#ch,0:LINE#ch,x+39,y+24 TO x+10,y+8 TO x+10,y+5 TO x+39,y+21
1280 CIRCLE#ch,x+11,y+5,4.5:CIRCLE#ch,x+12,y+6,4.5
1281 END DEFine
```

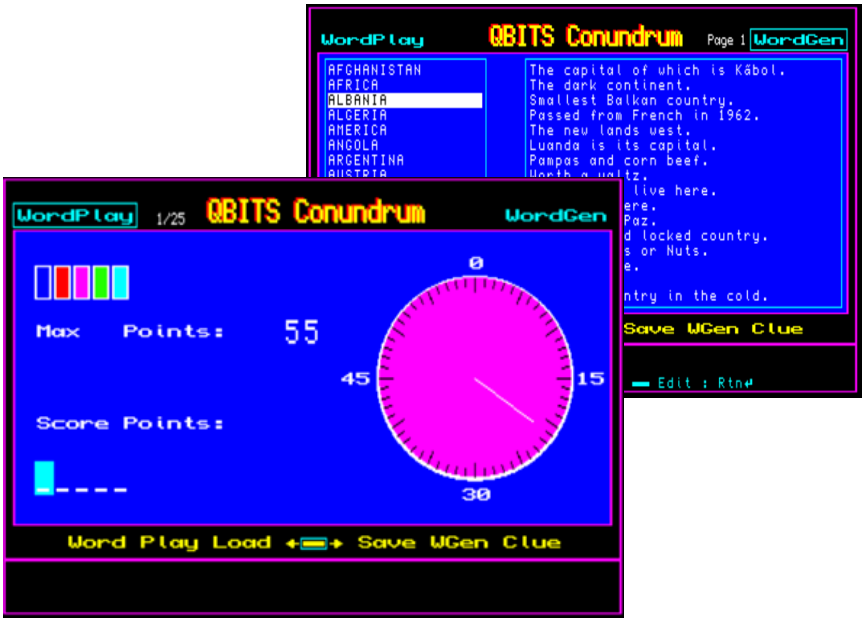
```
1283 DEFine PROCedure ML4(x,y)              :REMark Lamp Stand
1284 INK#ch,0:LINE#ch,x+9,y+1 TO x+9,y+39
1285 LINE#ch,x+12,y+1 TO x+12,y+39:LINE#ch,x+14,y+1 TO x+14,y+39
1286 END DEFine
```

```
1268 DEFine PROCedure ML5(x,y)              :REMark Lamp Base
1289 INK#ch,5:FILL#ch,1:CIRCLE#ch,x+12,y+20,10,.8,PI/2:FILL#ch,0
1290 INK#ch,0:CIRCLE#ch,x+12,y+20,10,.8,PI/2:CIRCLE#ch,x+12,y+21,10,.6,PI/2
1291 LINE#ch,x+9,y+21 TO x+9,y+39:LINE#ch,x+12,y+20 TO x+12,y+39
1292 LINE#ch,x+14,y+21 TO x+14,y+39
1293 END DEFine
```

```
1295 DEFine PROCedure MG1(x,y)              :REMark Guitar Top
1296 FILL#ch,1:INK#ch,3:LINE#ch,x+17,y+1 TO x+16,y+2 TO x+18,y+9
1297 LINE#ch TO x+24,y+8 TO x+22,y+1 TO x+17,y+1:FILL#ch,0:INK#ch,0
1298 LINE#ch TO x+18,y+1 TO x+19,y+6:LINE#ch,x+20,y+1 TO x+21,y+6
1299 END DEFine
```

```
1301 DEFine PROCedure MG2(x,y)              :REMark Guitar Middle
1302 FILL#ch,1:INK#ch,6:LINE#ch,x+1,y+28 TO x+1,y+24 TO x+15,y+14
1303 LINE#ch,x+18,y+14 TO x+1,y+28:FILL#ch,0:INK#ch,0
1304 LINE#ch,x+1,y+22 TO x+15,y+14:LINE#ch,x+1,y+29 TO x+16,y+18
1305 FILL#ch,1:INK#ch,3:LINE#ch, x+10,y+10 TO x+16,y+39 TO x+20,y+39
1306 LINE#ch TO x+14,y+10 TO x+10,y+10:FILL#ch,0
1307 INK#ch,3:FILL#ch,1:CIRCLE#ch,x+12,y+10,9,.7,PI/2.2:FILL#ch,0
1308 FILL#ch,1:ARC#ch,x+18,y TO x+2,y,PI:LINE#ch TO x+2,y:FILL#ch,0
1309 INK#ch,0:FILL#ch,1:CIRCLE#ch,x+11,y+10,3:FILL#ch,0
1310 ARC#ch,x+18,y+12 TO x+16,y+4,-PI/2 TO x+20,y+1,PI
1311 LINE#ch,x+8,y+1 TO x+17,y+39:LINE#ch,x+10,y+1 TO x+19,y+39
1312 FILL#ch,1:CIRCLE#ch,x+18,y+20,4:FILL#ch,0
1313 END DEFine
```

```
1315 DEFine PROCedure MG3(x,y)              :REMark Guitar Bottom
1316 INK#ch,3:FILL#ch,1:LINE#ch,x+2,y+39 TO x+18,y+39
1317 ARC#ch TO x+1,y+32,-PI TO x+2,y+39,-PI/4:FILL#ch,0
1318 INK#ch,0:ARC#ch,x+15,y+38 TO x+12,y+29,-PI/1.5
1319 LINE#ch,x+7,y+36 TO x+8,y+39:LINE#ch,x+9,y+36 TO x+10,y+39
1320 LINE#ch,x+5,y+32 TO x+11,y+31:LINE#ch,x+5,y+34 TO x+11,y+33
1321 END DEFine
```



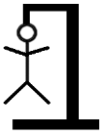
QBITS Conundrum Menu

To Select from the Menu use Left/Right Cursor keys to Highlight an item then action with the Spacebar. The Central Symbol has two functions, in **WordPlay** mode it allows changes to the Countdown Timer and (Q)uit from the Game. In **WordGen** mode it can Reset Arrays for creating a (**N**)ew Word File and again (Q)uit from Game.

WordPlay displays the Countdown Timer, the Conundrum Word and Score Points. **Play** requires a **Load**ed Word File and each Game randomly selects 25 of the entries. The number completed is shown top left next to the Highlighted **WordPlay** (see above). Toggle **Word** On/Off to choose between Coloured Blocks or Jumbled Letters. Similarly Toggle **Clue** On/Off to display if present.

WordGen displays two columns, the first for the **Word** list and second for the **Clue**s. The current **Page** is shown top right next the Highlighted **WordGe**n. Each word File can have 6 pages of 16 rows adding up to total of 96 entries. If a **Word File** hasn't been previously loaded, use **Load** or start the creation of a **New** word file. From **WGen** to select an entry use Up/Down cursors and tab to switch between **Word** and **Clue** then press spacebar to invoke the **Line Editor**. Select **Save** to store Word File to default device, create a new or edit the Filename shown.

QBITS Conundrum Beginnings



The early beginnings began was a variation on a popular word game ‘hangman’, the origins of which are unknown although is has been around for more than a century. A word is represented by a row of dashes and the player tries to guess the missing letters before the drawing of a gallows and hung matchstick man are completed.

QBITS Conundrum Development

Wanting to take this to another level when solving the Word Riddle the display included a Timer to use as a countdown. Characters are typed in from the keyboard in a trial-and-error guess to reveal the hidden word. The Word is represented by Coloured Blocks or by Jumbled Letters with Clues added as a further aid in solving the riddle.

Word when selected **Off**, the **Coloured Blocks** will change to show any correctly typed in character in whatever order they are typed. When **Word** is selected **On**, the hidden word is shown as a **Jumbled Letters**, here characters typed in from the Keyboard must be in the correct order to reveal the Word. By turning **On/Off Word** and **Clues** and changing the length of **Countdown Time**, wide range of difficulty can be achieved to service players of different ages and abilities. This is further defined by the difficulty of the **Word Lists** used.

QBITS Conundrum WordGen

To create Word files **WordGe**n began as a separate program, but is now combined as part of Conundrum. Selecting **WGen** from the Menu switches the main screen to **WordGen** from **WordPlay**, **Play** will switch it back. The **Word** and **Clue** rows displayed in **WordGen** and be selected with **Up/Down** cursors and switched between with the **Tab** key. In either column pressing spacebar invoke the **Line Editor**. For **Word** a maximum of 18 Upper-case Alphabet characters with no spaces are permitted. For **Clue** up to 36 Alphanumeric Characters including spaces and punctuation marks. Each Word File can contain 96 entries. A minimum of 25 entries are required for **WordPlay** ie. the number of random choices for a full Game.

Note: Code Line 1002 cmax%=25 controls the number of rounds in a Game.

QBITS Conundrum Strategy

If **Word** is turned **Off** and Coloured Block are Displayed, then the twelve most commonly occurring letters in the English language are e-t-a-o-i-n-s-h-r-d-l-u. Another possibility is to try the vowels a-e-i-o-u with possibly y. Selecting **Word On** and displaying the Conundrum as Jumbled letters might seem an easier task, but they must be typed in the correct order and without **Clues,** it might still be a challenge against the Set Countdown Time.

QBITS Conundrum Code

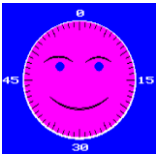
```
1000 REMark QBConundrum_bas [QBITS  1992 Conumdrum 2023 Review – QPC"]

1002 dev$='win1_':MODE 4:gx=0:gy=0      :REMark basic settings

1004 WHEN ERRor :CONTINUE:END WHEN

1006 REMark Import QBITSConfig Settings – QPC2
1007 OPEN_IN#9,dev$&'QBITSConfig':iINPUT#9,gx\gy\dn$\dev$:CLOSE#9

1010 Init_Win:cmax%=25:WordMenu

1012 DEFine PROCedure Init_Win
1013 DIM Word$(96,18),Clue$(96,36),WChk(96),str$(36),str$(36),SDR$(5),Wfn$(24)
1014 OPEN#4,scr_:WINDOW#4,288,160,gx+206,gy+37:PAPER#4,1:CSIZE#4,1,0
1015 OPEN#3,scr_:WINDOW#3,144,160,gx+18,gy+37 :PAPER#3,1:CSIZE#3,1,0
1016 WINDOW#2,512,224,gx,gy           :PAPER#2,0:BORDER#2,1,3:CLS#2
1017 WINDOW#1,496,174,gx+8,gy+30 :PAPER#1,1:BORDER#1,1,3:CLS#1:SCALE#1,100,0,0
1018 WINDOW#0,512, 32,gx,gy+224    :PAPER#0,0:BORDER#0,1,3:CLS#0
1019 SDR$=dev$:max_score%=0:per_score%=0:count%=0:key$=''
1020 CSIZE#2,2,1:OVER#2,1
1021 INK#2,2:FOR i=0 TO 1:CURSOR#2,164+i,8:PRINT#2,'QBITS Conundrum'
1022 INK#2,6:FOR i=0 TO 1:CURSOR#2,166+i,9:PRINT#2,'QBITS Conundrum'
1023 CSIZE#2,2,0:OVER#2,0
1024 QBold 2,5,12,1,400,16,'WordGen':QBold 2,5,12,1,-4,16,'WordPlay'
1025 QBold 2,6,12,1,38,208,'Word Play Load ⬅      ➡ Save WGen Clue'
1026 BLOCK#2,22,7,244,210,5:BLOCK#2,20,5,245,211,0:BLOCK#2,18,3,246,212,6
1027 AT#2,1,1:CSIZE#2,0,0:INK#2,5:CSIZE#1,2,0::CSIZE#0,1,0:INK#0,5
1028 Time%=180:Sech%=180:Sec%=180:CSrn=0:mc%=2:F=0
1029 END DEFine

1031 DEFine PROCedure PlayScrn
1032 HGL 2,5,34,6,  2,88:HGL 2,0,30,6,137,88
1033 BLOCK#2,40,10,370,16,0:CSrn=0:CLS#1
1034 INK#1,7:FILL#1,1:CIRCLE#1,164,50,35:FILL#1,0:CSIZE#1,2,0
1035 INK#1,3:FILL#1,1:CIRCLE#1,164,50,34:FILL#1,0:INK#1,0
1036 FOR j=3 TO 5 STEP 2
1037   FOR i=0 TO 360 STEP j*12-30
1038     x=34*SIN(RAD(i)):y=34*COS(RAD(i))
1039     x1=(34-j)*SIN(RAD(i)):y1=(34-j)*COS(RAD(i))
1040     LINE#1,x+164,y+50 TO x1+164,y1+50
1041   END FOR i
1042 END FOR j
1043 QBold 1,7,12,1,362, 14, '0':QBold 1,7,12,1,450,82,'15'
1044 QBold 1,7,12,1,356,150,'30':QBold 1,7,12,1,256,82,'45'
1045 QBold 1,7,12,1,4,  54,'Max    Points:'
1046 QBold 1,7,12,1,4,108,'Score Points:':CSIZE#1,2,1
1047 END DEFine

1049 DEFine PROCedure HGL(ch%,col%,w%,d%,x%,y%)
1050 INK#ch%,col%:LINE#ch%,x%,y% TO x%+w%,y% TO x%+w%,y%+d% TO x%,y%+d% TO x%,y%
1051 END DEFine
```

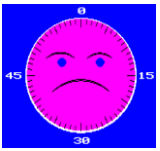

```
1053 DEFine PROCedure WordMenu
1054 F=0:ac%=0:aw%=0:cw%=0:count%=1:PlayScrn
1055 REPeat Comm_lp
1056  x%=mc%*20+15.6:y%=1:HGL 2,5,18,5.8,x%,y%
1057  CLS#0:QBold 0,7,8,1,64,6,'Select ¼ ½ Then Press SpaceBar to Continue...'
1058  k=CODE(INKEY$(-1)) :HGL 2,0,18,5.8,x%,y%:INK#2,5:CLS#0
1059  SELect ON k
1060   =192:mc%=mc%-1:IF mc%<0:mc%=6
1061   =200:mc%=mc%+1:IF mc%>6:mc%=0
1062   = 27:CSIZE#0,0,0:INK#0,7:CLS#2:STOP
1063   = 32:SELect ON mc%
1064      =0:CURSOR#2,28,204:IF aw%=0:aw%=1:PRINT#2,'ON':ELSE aw%=0:PRINT#2,'  '
1065      =1:WordPlay
1066      =2:WordList :IF CSrn=0:mc%=1:ELSE mc%=5
1067      =3:IF CSrn=0:Time_chg :CLS#0:mc%=1:ELSE Word_chg:CLS#0:mc%=5
1068      =4:WordSave :CLS#0
1069      =5:WordGen
1070      =6:CURSOR#2,466,204:IF ac%=0:ac%=1:PRINT#2,'ON':ELSE ac%=0:PRINT#2,'  '
1071     END SELect
1072  END SELect
1073 END REPeat Comm_lp
1074 END DEFine
```


```
1076 DEFine PROCedure WordPlay
1077 IF F=0 OR wc%<cmax%;mc%=2:RETurn
1078 IF CSrn=1:CSIZE#1,2,0:PlayScrn:CSIZE#1,3,1
1079 CNT=DATE:Sec%=Time%:Sech%=Time%:RANDOMISE
1080 REPeat Rnd_lp
1081    n%=RND(1 TO wc%):IF WChk(n%)=0:WChk(n%)=1:WordRND:EXIT Rnd_lp
1082 END REPeat Rnd_lp
1083 Countdown:chr%=1:pos%=1:k=0:str$=FILL$(' ',wl%)
1084 REPeat Wrd_lp
1085    STRIP#1,5:CURSOR#1,16*chr%,135:PRINT#1,key$(chr%):STRIP#1,1
1086    k=CODE(INKEY$(20))
1087    SELect ON k
1088       =192:chr%=chr%-1:IF chr%<1  :chr%=1
1089       =200:chr%=chr%+1:IF chr%>wl%:chr%=wl%
1090       =65 TO 90,97 TO 122:WordChk
1091    END SELect
1092    CURSOR#1,16*pos%,135:PRINT#1,key$(pos%):pos%=chr%
1093    IF key$=Word$(n%) OR str$=Word$(n%) OR Sec%=0:WordScore:EXIT Wrd_lp
1094    IF CNT<>DATE:Sec%=Sec%-(6*(DATE-CNT)):Countdown:CNT=DATE
1095 END REPeat Wrd_lp
1096 END DEFine

1098 DEFine PROCedure WordChk
1099 IF k>96:k=k-32
1100 key$(chr%)=CHR$(k):chr%=chr%+1:IF chr%>wl%:chr%=1
1101 IF aw%=0
1102    FOR i=1 TO wl%
1103       IF key$(pos%)=Word$(n%,i)
1104          str$(i)=CHR$(k):CURSOR#1,16*i,20:PRINT#1,str$(i)
1105       END IF
1106    END FOR i
1107 END IF
1108 END DEFine
```

```
1110 DEFine PROCedure WordRND
1111 CLS#0:BLOCK#1,290,20,16,20,1:BLOCK#1,290,20,16,135,1
1112 INK#1,3:FILL#1,1:CIRCLE#1,164,50,28:FILL#1,0:CSIZE#1,3,1:INK#1,7
1113 Sort$=Word$(n%):wl%=LEN(Sort$):cl%=LEN(Clue$(n%))
1114 REPeat Sort_lp
1115  FOR i=1 TO wl%-1
1116   r1=RND(1 TO wl%):Chr1$=Sort$(r1):r2=RND(1 TO wl%):Chr2$=Sort$(r2)
1117   Sort$(r1)=Chr2$:Sort$(r2)=Chr1$
1118  END FOR i
1119  IF Sort$<>Word$(n%):EXIT Sort_lp
1120 END REPeat Sort_lp
1121 IF aw%=0
1122   FOR blk=1 TO wl%
1123    BLOCK#1,14,20,blk*16,20,7:BLOCK#1,10,18,2+blk*16,20+1,(blk MOD 8)
1124   END FOR blk
1125 END IF
1126 IF aw%=1:CURSOR#1,16,20:PRINT#1,Sort$
1127 IF ac%=1:CURSOR#0,112,10:PRINT#0,FILL$(' ',18-cl%/2)&Clue$(n%)
```


```
1128 key$=FILL$('_',wl%):CURSOR#1,16,135:PRINT#1,key$
1129 max_score%=max_score%+5&wl%
1130 CURSOR#1,190,50:PRINT#1,FILL$(' ',4-LEN(max_score%))&max_score%:INK#2,7
1131 CURSOR#2,112,18:PRINT#2,FILL$(' ',3-LEN(count%))&count%&'/'&cmax%
1132 END DEFine

1134 DEFine PROCedure Countdown
1135 INK#1,3:LINE#1,164,50 TO 164+26*SIN(RAD(Sech%)),50+26*COS(RAD(Sech%))
1136 Sech%=Sec%:BEEP 2000,10,0,0,0,0,0,0
1137 INK#1,7:LINE#1,164,50 TO 164+26*SIN(RAD(Sech%)),50+26*COS(RAD(Sech%))
1138 END DEFine

1140 DEFine PROCedure WordScore
1141 INK#1,3:FILL#1,1:CIRCLE#1,164,50,28:FILL#1,0:INK#1,1
1142 FILL#1,1:CIRCLE#1,152,58,2.5:FILL#1,0
1143 FILL#1,1:CIRCLE#1,176,58,2.5:FILL#1,0:INK#1,0
1144 ARC#1,142,60 TO 160,60,-PI/2:ARC#1,142,60 TO 160,60,-PI/2.3
1145 ARC#1,168,60 TO 184,60,-PI/2:ARC#1,168,60 TO 184,60,-PI/2.3
1146 IF Sec%<6:ARC#1,146,40 TO 182,40,-PI/2:ARC#1,146,40 TO 182,40,-PI/2.2
1147 IF Sec%>0:ARC#1,146,40 TO 182,40, PI/2:ARC#1,146,40 TO 182,40, PI/2.2
1148 INK#1,7:IF Sec%>0:per_score%=per_score%+5*wl%
1149 CURSOR#1,190,104:PRINT#1,FILL$(' ',4-LEN(per_score%))&per_score%
1150 CURSOR#1,16,20:PRINT#1,Word$(n%):count%=count%+1
1151 IF aw%=0:BLOCK#1,216,20,16,135,1
1152 IF count%>cmax%
1153   CLS#0:CURSOR#1,12,78:PRINT#1,'Game End':PAUSE
1154   max_score%=0:per_score%=0:count%=1:mc%=1:PlayScrn
1155   FOR i=1 TO 96:WChk(i)=0
1156 END IF
1157 END DEFine
```





```
1159 DEFine PROCedure Time_chg
1160 CLS#0:INK#1,3:FILL#1,1:CIRCLE#1,164,50,28:FILL#1,0
1161 CURSOR#0,80,6:PRINT#0,'Set Countdown Time:←  → or (Q)uit';
1162 BLOCK#0,14,3,248,10,5:Sec%=Time%:Countdown
1163 REPeat Time_lp
1164  Countdown:k=CODE(INKEY$(-1))
1165  SELect ON k
1166   =192:Sec%=Sec%+30:IF Sec%>360:Sec%=360
1167   =200:Sec%=Sec%-30:IF Sec%< 30:Sec%= 30
1168   = 32:Time%=Sec%:CLS#0:EXIT Time_lp
1169   =69,101:PRINT#0,' Y/N':IF INKEY$(-1)=='Y':LRUN dn$:ELSE RETurn
1170  END SELect
1171 END REPeat Time_lp
1172 END DEFine
```

```
1200 REMark QBITS WordGen

1202 DEFine PROCedure EditScrn
1203 HGL 2,0,34,6,2,88: HGL 2,5,30,6,137,88
1204 BLOCK#2,48,10,110,16,0 :CLS#1:INK#1,5:l=1:pn%=1:CSrn=1
1205 LINE#1,2,2 TO 2,98 TO 67,98 TO 67,2 TO 2,2
1206 LINE#1,83,2 TO 83,98 TO 210,98 TO 210,2 TO 83,2
1207 END DEFine
```


```
1209 DEFine PROCedure WordGen
1210 IF CSrn=0:EditScrn
1211 IF F=1:pn%=1:sr%=0
1212 ch%=3:cp%=1:sr%=0:Str_Chk:Pg_Prn:BCol%=7:ICol%=0
1213 INK#4,7:STRIP#4,1:INK#3,7:STRIP#3,1:l=1:Str_Clr
1214 REPeat Edit_lp
1215  CURSOR#0,92,20:PRINT#0,'LINE ¾¿ : WORD<TAB>CLUE :    Edit : Rtn¼'
1216  BLOCK#0,16,3,300,24,5:BLOCK#0,2,4,410,22,5
1217  Str_Clr:Str_Prn:k=CODE(INKEY$(-1)):sl%=LEN(str$)
1218  SELect ON k
1219   =  9:Ln_Clr:IF ch%=3:ch%=4:ELSE ch%=3:END IF :Str_Chk:Str_Prn
1220   = 10:BCol%=1:ICol%=7:Pg_Prn:Str_Clr:INK#2,5:RETurn            :REMark End
1221   = 32:Str_ED ch%,1,cp%,cs%,sr%,sm%,sx%,sy%,str$:CLS#0     :REMark Edit str$
1222   =208:Ln_Clr:WordUP:cp%=1:Str_Chk                                :REMark Cursor Up
1223   =216:Ln_Clr:WordDn:cp%=1:Str_Chk                               :REMark Cursor Down
1224  END SELect
1225 END REPeat Edit_lp
1226 END DEFine

1228 DEFine PROCedure Word_chg
1229 CURSOR#0,80,6:PRINT#0,'Create a (N)ew Word File or (E)xit';:PAUSE
1230 IF KEYROW(7)=64
1231    F=0:wc%=0:fn$='':CURSOR#0,270,6:CLS#0,4:CLS#3:CLS#4
1232    FOR i=1 TO 96:Word$(i)='':Clue$(i)='':PAUSE 1:CURSOR#0,280,6:PRINT#0,i
1233 END IF
1234 IF KEYROW(6)=16:PRINT#0,' Y/N':IF INKEY$(-1)=='Y':LRUN dn$
1235 END DEFine
```

```
1237 DEFine PROCedure WordUP
1238 IF sr%= 0 AND pn%>1:pn%=pn%-1:Pg_Prn:ELSE IF sr%> 0:sr%=sr%-1:l=l-1
1239 END DEFine

1241 DEFine PROCedure WordDn
1242 IF sr%=15 AND pn%<6:pn%=pn%+1:Pg_Prn:ELSE IF sr%<15:sr%=sr%+1:l=l+1
1243 END DEFine

1300 REMark WordGen Line Editor
```

Line editor Str_Ed (ch% channel, cm% chars max, cp% char position, cs% Char set, sr% str$ row, sl% str$ length, sm% str$ max, sx% & sy% str$ pixel start coordinates, str$ charterer string)
BCol% Background colour, ICol% Ink colour, CCol% Cursor Colour

```
1302 DEFine PROCedure Str_ED(ch%,cm%,cp%,cs%,sr%,sm%,sx%,sy%,str$)
1303 CLS#0:CURSOR#0,80,20:PRINT#0,'Add Chr Del[Ctrl ← /→ ] [← Cursor →] Rtn[← ]'
1304 BLOCK#0,80,10,0,20,0:BLOCK#0,2,4,374,22,5:sl%=LEN(str$)
1305 REPeat Ed_lp
1306   Str_Clr:Str_Prn:CCol%=2:Str_Cur
1307   k=CODE(INKEY$(-1)):sl%=LEN(str$)
1308   SELect ON k
1309     =10          :CCol%=BCol% :Str_Cur :EXIT Ed_lp
1310     =32 TO 127:Str_Prn:k$=' '  :Sel_chr :IF k$>'':Add_chr
1311     =194,232   :CCol%=BCol% :Str_Cur: IF cp%>cm%:cp%=cp%-1:Del_chr
1312     =202,236   :CCol%=BCol% :Str_Cur :Del_chr
1313      =192      :CCol%=BCol% :Str_Cur :IF cp%>cm%:cp%=cp%-1   :REMark ←Left
1314       =200     :CCol%=BCol% :Str_Cur :IF cp%<sl%+1:cp%=cp%+1 :REMark→ Right
1315   END SELect
1216 END REPeat Ed_lp
1317 END DEFine

1319 DEFine PROCedure Pg_Prn              :REMark Print Page
1320 BCol%=1:ICol%=7:sr%=-1
1321 FOR l=pn%*16-15 TO pn%*16
1322   sr%=sr%+1:ch%=3:Str_Chk:Str_Prn:ch%=4:Str_Chk:Str_Prn
1323 END FOR l
1324 INK#2,7:CURSOR#2,370,16:PRINT#2,'Page ';pn%:l=l-15:sr%=0
1325 ch%=3:BCol%=7:ICol%=0:Str_Chk
1326 END DEFine

1328 DEFine PROCedure Ln_Clr              :REMark Clear Line
1329 Str_Clr:BCol%=1:ICol%=7:CCol%=0:Str_Prn:BCol%=7:ICol%=0:cp%=1
1330 END DEFine

1332 DEFine PROCedure Str_Prn
1333 STRIP#ch%,BCol%:INK#ch%,ICol%
1334 CURSOR#ch%,sx%,sy%+sr%*10:PRINT#ch%,str$&FILL$(' ',sm%-LEN(str$))
1335 END DEFine

1337 DEFine PROCedure Str_Cur
1338 IF cp%>=sm%:cp%=sm%:sl%=sm%
1339 BLOCK#ch%,8,1,sx%+cp%*8-8,sy%+sr%*10+9,CCol%
1340 END DEFine
```

```
1342 DEFine PROCedure Str_Chk
1343 IF ch%=0:cs%=2:cp%=1:sl%=LEN(fn$)     :cm%=5:sm%=16:str$=fn$
1344 IF ch%=3:cs%=1:cp%=1:sl%=LEN(Word$(l)):cm%=1:sm%=18:str$=Word$(l)
1345 IF ch%=4:cs%=3:cp%=1:sl%=LEN(Clue$(l)):cm5=1:sm%=36:str$=Clue$(l)
1346 END DEFine

1348 DEFine PROCedure Str_Clr
1349 IF LEN(str$)>=sm%:str$=str$(1 TO sm%)
1350 IF ch%=0:fn$=str$:sr%=0
1351 IF ch%=3:Word$(l)=str$
1352 IF ch%=4:Clue$(l)=str$
1353 END DEFine

1255 DEFine PROCedure Sel_chr
1356               SELect ON k=65 TO 90                    :k$=CHR$(k)
1357 IF cs%=1 :SELect ON k=97 TO 122                    :k$=CHR$(k-32)
1358 IF cs%>1 :SELect ON k=48 TO 57,95,97 TO 122  :k$=CHR$(k)
1359 IF cs%=3 :SELect ON k=32 TO 47,123 TO 127     :k$=CHR$(k)
1360 END DEFine

1362 DEFine PROCedure Add_chr
1363 IF cp% = 1 AND sl%=0:str$=str$&k$
1364 IF cp%>=1 AND cp%<sl%:str$=str$(1 TO cp%-1)&k$&str$(cp% TO sl%)
1365 IF cp%>=1 AND cp%=sl%:str$=str$(1 TO cp%-1)&k$&str$(cp%)
1366 IF cp% > 1 AND cp%>sl%:str$=str$&k$
1367 IF cp%=sm%:str$(cp%)=k$
1368 IF sl%<sm%:sl%=sl%+1:ELSE sl%=sm%
1369 IF cp%<sm%:cp%=cp%+1:ELSE cp%=sm%
1370 END DEFine

1372 DEFine PROCedure Del_chr
1373 IF cp%=sl%:str$=str$(1 TO sl%-1):sl%=sl%-1
1374 IF cp%>=1 AND cp%<sl%:str$=str$(1 TO cp%-1)&str$(cp%+1 TO sl%):sl%=sl%-1
1375 IF cp%=sm%:str$=str$(1 TO sm%-1):cp%=cp%-1:sl%=sl%-1
1376 IF cp%>1 AND sl%=0:str$=""
1377 END DEFine

1379 DEFine PROCedure QBold(ch%,col%,w%,d%,x%,y%,str$)
1380 OVER#ch%,1:INK#ch%,col%:sl%=LEN(str$)
1381 FOR a=1 TO sl%
1382   FOR b=0 TO d%:CURSOR#ch%,x%+b+a*w%,y%:PRINT#ch%,str$(a)
1383 END FOR a:OVER#ch%,0
1384 END DEFine
```

```
1400 REMark QBITS Conundrum Word Files
```


```
1402 DEFine PROCedure WordList
1403 DIM Volumn$(11),Sector$(24),file$(20,36),df$(36)
1404 CLS#0:QBold 0,5,9,1,60,10,'Files being Selected...'
1405 f%=1:ft%=0:fm%=20:DELETE SDR$&'FList'
1406 OPEN_NEW#6,SDR$&'FList':DIR#6,SDR$:CLOSE#6
1407 OPEN_IN#6, SDR$&'FList':INPUT#6,Volumn$,Sector$
1408 REPeat DIR_lp
1409  IF EOF(#6) OR f%>fm%:ft%=f%-1:CLOSE#6:EXIT DIR_lp
1410  INPUT#6,df$:IF 'WGen_' INSTR df$>0:file$(f%)=df$:f%=f%+1:PAUSE 2
1411 END REPeat DIR_lp
1412 IF ft%<1
1413    CLS#0:QBold 0,5,9,1,60,10,'No Word Files Found...'
1414    F=0:mc%=4:PAUSE 50:RETurn
1415 END IF
1416 PAUSE 20:WordFile:mc%=5
1417 END DEFine
```


```
1419 DEFine PROCedure WordFile
1420 CLS#0:QBold 0,5,9,1,60,10,'Select File <↑  ↓>:':BLOCK#0,14,3,200,14,5:f%=1
1421 REPeat File_lp
1422  CURSOR#0,260,10:PRINT#0,f%;'  ';file$(f%,5+('WGen_' INSTR file$(f%)) TO)
1423  CLS#0,4:k=CODE(INKEY$(-1))
1424  SELect ON k
1425   =208:f%=f%-1:IF f%<1:f%=ft%
1426   =216:f%=f%+1:IF f%>ft%:f%=1
1427   = 32:fn$=file$(f%):WLoad :EXIT File_lp
1428  END SELect
1429 END REPeat File_lp
1430 END DEFine
```


```
1432 DEFine PROCedure WLoad
1433 FOR i=1 TO 96:Word$(i)='':Clue$(i)=''
1434 CLS#0:QBold 0,5,9,1,100,10,'Loading Word File...'
1435 OPEN_IN#6,SDR$&fn$:wc%=1
1436 REPeat Ld_lp
1437  INPUT#6,Word$(wc%),Clue$(wc%):CURSOR#0,292,10:PRINT#0,wc%:CLS#0,4
1438  IF EOF(#6) OR wc%=96:CLOSE#6:EXIT Ld_lp:ELSE wc%=wc%+1:PAUSE 1
1439 END REPeat Ld_lp
1440 PAUSE 20:CLS#0:F=1:l=1:IF CSrn=1:pn%=1:Pg_Prn
1441 END DEFine
```


```
1443 DEFine PROCedure WordSave
1444 sm%=16:IF LEN(fn$)>sm% OR fn$='':fn$='WGen_'
1445 CLS#0:CURSOR#0,80,10
1446 PRINT#0,'Save ';SDR$&fn$;FILL$(' ',sm%-LEN(fn$));' Y/N or (E)dit'
```


```
1447 k=CODE(INKEY$(-1)):CURSOR#0,286,10:CLS#0,4
1448 IF k=69 OR k=101
1449    BCol%=0:ICol%=7:Str_ED 0,5,6,2,0,16,160,6,fn$
1450    IF LEN(fn$)<6 OR k=32:mc%=2:RETurn :ELSE WSave
1451 END IF
1452 IF k=89 OR k=121
1453    IF LEN(fn$)<6:BCol%=0:ICol%=7:Str_ED 0,5,6,2,0,16,160,6,fn$
1454    IF LEN(fn$)<6 OR k=32:mc%=2:RETurn :ELSE WSave
1455 END IF
1456 END DEFine

1458 DEFine PROCedure WSave
1459 CURSOR#0,300,10:PRINT#0,' Save Y/N ';:PAUSE
1460 IF KEYROW(5)<>64:RETurn
1461 DELETE SDR$&fn$:OPEN_NEW#8,SDR$&fn$:fm%=96
1462 FOR n=1 TO 96
1463   IF Word$(n)<>'':PRINT#8,Word$(n)\Clue$(n):fm%=fm%-1
1464   CURSOR#0,380,10 :PRINT#0,'Chk:';96-fm%:PAUSE 1
1465 END FOR n
1466 CLOSE#8:CLS#0:mc%=1:F=1
1467 IF fm%=96:DELETE SDR$&fn$:F=0
1468 END DEFine
```

The Game of Darts

The Game of Darts originated in England as a military pastime during the medieval period of the 1300's. Bored soldiers would compete by throwing spearheads and other sharp objects at upturned wine casks. This was encouraged so as to practice their aim and throwing skills.

The natural structure of cross-sectioned tree trunks with their growth rings and radial cracks allowed competitors to further demonstrate their skills. The first purpose-made Darts were manufactured from solid wood, wrapped with a strip of lead for weight and fitted with flights made from split turkey feathers.

Various scoring systems have been used over the centuries, nevertheless it wasn't until the 1800's that Darts reached such a level of popularity a wave of innovations and developments fostered the beginnings of the modern numbering system.

The Modern Dart Board

All though there were many claimants to the inventor of the modern Dart Board, the consensus is given to a Lancashire carpenter named Brian Gamlin who in 1896 produced a Dart Board with the numbering scheme we know today.

At first view the numbering seems quite random, but is in fact a meticulously picked order, the design of which reduces the incidents of a 'lucky shot' and diminishes the element of chance, so that only skill and accuracy will attain the best scores.

Computer Game of Darts

Compared with other sports very few computer-based Darts Games have been released. The first notable one being Metronics 180 released in 1986 for the commadore 64, ZX Spectrum, Amstrad and Atari 8-bit. Indoor Sports released a game set in 1987 which included Darts and in 1991 Seta Corporation released Magic Darts!

In 2006 a version of a Darts Game was released for PlayStation 2 and PC. Fans were finally able to play a modern-day version based on the PDC World Championships. In 2011 Microsoft Studios added Darts to the Games included in their Kinect Sport sequel.

Darts is a pure sport with simple rules that belies the depth of tactical and technical ability required. While being a Professional shooting sport, darts is also a traditional Pub Game.

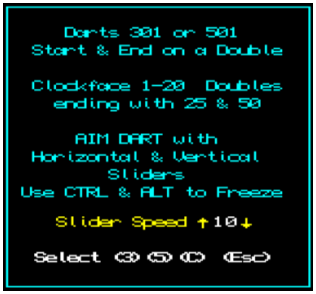
Arrows is a slang term for darts or darting, it refers to the thin stick weighted and pointed at one end with feathers at the other. Average is the score achieved after three arrows (darts) are thrown, which is a player's turn.

QBITS Darts

The most common Dart Board game is no doubt 301, where two people or teams compete to reduce a fixed score of 301 to zero. However, each player or side must start and end by throwing a double. This was my starting point, and then to add a 501 option. Being a little more ambitious I decided to add the clock face game. For this you throw a double for each number in sequence 1-20, then a 25 & 50 bullseye to finish.

QBITS Darts Intro

The Intro screen states a simple review of the options available and a means by which the player's choice can be made.



QBITS Darts Options

For the 301 and 501 options **Red** and **Green** teams, or individual players, can play against each other, the first to finish is the winner.

The Clock-face option is for a single player to complete in as few throws as possible.

QBITS Darts End of Game

At End of a Game the board is scrolled up with results displayed, and shows the number of Darts thrown.

QBITS Dartboard

The modern Dartboard is a disc with twenty segments, each divided into Single, Double, and Treble areas. The circumference being **2πr** each segment i**s** therefore **2πr/20 or** 18 degrees. I thought what could be simpler in writing the code for a Dartboard display, a circle divided into segments with sections for single, double and treble areas. When I first attempted this back in the eighties, I soon realised a little refreshment in tribometry wouldn't go amiss.

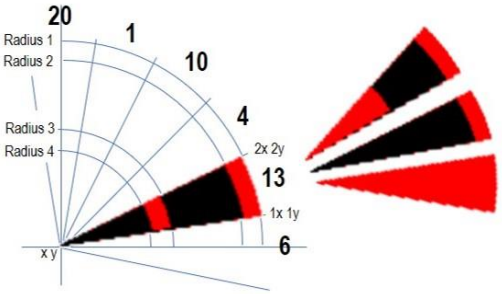
The centre of the circle gives the primary coordinates x, y, then each side of the segment require a straight LINE drawn from x, y TO 1x,1y and another from x, y TO 2x, 2y between then is drawn an ARC 1x,1y TO 2x,2y its radial angle applicable to the portion of the Dartboards circumference. That's the cleaver bit calculating the outer points to draw the LINE's and what is the angle of ARC.

QBITS Dartboard Segments

For a starting point the segment representing the number 6 is 9 degrees above and 9 degrees below the horizontal zero line. If we start at zero then add **π/20** this provides an Offset to begin the drawing of a Dartboard segment. By adding a further **π/10** (18 degrees) this give our second Offset. These are the angles, which with the length of radius and COS & SIN we can calculate 1x, 1y and 2x, 2y coordinates.

Using LINE and ARC to draw a segment then INK & FILL commands a coloured segment is drawn. Reducing the radius and change of colour this creates the Double, Treble and Single sections of a segment.

By adding multiples of **π/10** to the angle, we can then process the next segment and so on around the board.



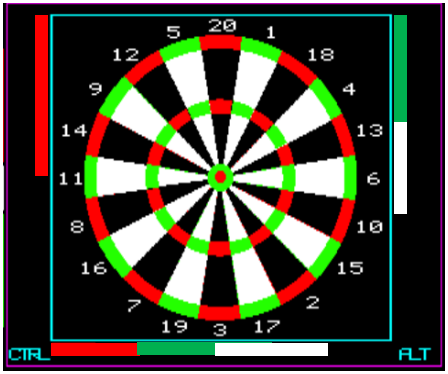
Last but not least we need a couple of FILL'd CIRCLES for the 25 and 50 at the centre.

QBITS Dartboard Numbering

My 1985 QL ROM had Cursor Graphical Coordinate limited to Window#1, so for numbering my Dartboard I used the Pixel Coordinates system and fine-tuned it mostly by trial error. This built up an array of numbers and their x, y coordinates for use with the CURSOR command. Then it was a simple act to store them as DATA lines and use them in a FOR loop to display.

QBITS Dart Throws

In today's world a computer Game of Darts might use a motion detector to determine the accuracy of our throw. Back in the eighties this was not an option.



I don't know when or who introduced the Slider or Track bar as a graphical control in computing, but with a combination of Horizontal and Vertical Sliders a player can evaluate a cross point to aim their Dart.

The output from this gives a dx,dy coordinate for the Dart position with respect to the Dartboard centre coordinates x, y. The dart radius dr is calculated using Pythagoras theorem and the angle da with ACOS.

```
dr=SQRT((dy-y)^2+(dx-x)^2)          dr (dart radius)

da=ACOS((dx-x)/dr)                  da (segment angle)
```

Identifying the relevant segment number was achieved by taking the angle then adding the first Offset and dividing this by **π/10.** The only problem being the angle reduces once passed 180 degrees, to cater for this I add a **π** and subtract the angle from **π.** Using the I**NT**eger of the segment it is then simply a FOR loop to read through a list until the right number is reached.

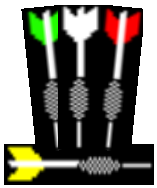
As for the **Double** and **Treble** or centre **Bullseye** circles these can be checked against the radius values set up for the dartboard.

```
1185 DEFine PROCedure dnum
1186 RESTORE 1185
1187 IF dy<50:da=PI+(PI-da)
1188 dt=INT((da+PI/20)/(PI/10))+1          calculates Dart positions in terms
1189 FOR seg=1 TO dt:READ num              of segment number 1 to 20
1190 dm=1                                  and dart multiplier
1191 IF dr > 44 :dm=0                      single
1192 IF dr<=44 AND dr>40:dm=2              double
1193 IF dr<=24 AND dr>20:dm=3              treble
1194 IF dr<=  4 :num=25                    or  25
1195 IF dr<1.7:num=50                      or  50
1196 REMark Dartboard numbers / segment
1197 DATA 6,13,4,18,1,20,5,12,9,14,11,8,16,7,19,3,17,2,15,10,6
1198 END DEFine
```

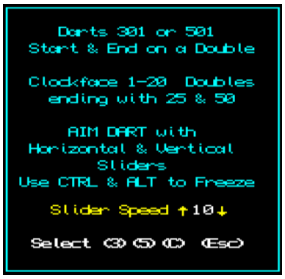
Note: Adjustments can be made to the **Slider** Speed (sp) - see the opening lines of the following Program code.

QBITS Darts code

```
1000 REMark QBITS_Darts_bas [QBITS Darts 2023 Review - QPC2]

1002 dev$='win2_':MODE 8:gx=0:gy=0    :REMark basic settings

1004 WHEN ERRor :CONTINUE:END WHEN

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


1010 Init_win:sp=10:Darts_Intro:QBITS_Darts

1012 DEFine PROCedure Init_win
1013 WINDOW#2,512,224,gx,gy       :PAPER#2,0:CLS#2:BORDER#2,1,3 :SCALE#2,100,0,0
1014 WINDOW#0,512,32,gx,gy+224    :PAPER#0,0:CLS#0:BORDER#0,1,3
1015 ch=6:OPEN#ch,scr_:WINDOW#ch,144,26,gx+4,gy+100:BORDER#ch,1,7:INK#ch,7
1016 ch=5:OPEN#ch,scr_:WINDOW#ch,144,66,gx+4,gy+ 32:BORDER#ch,1,2:INK#ch,2
1017 ch=4:OPEN#ch,scr_:WINDOW#ch,144,66,gx+4,gy+128:BORDER#ch,1,4:INK#ch,4
1018 ch=3:OPEN#ch,scr_:WINDOW#ch,354,210,gx+1542,gy+6
1019 WINDOW#1,280,190,gx+186,gy+12:PAPER#1,0:CLS#1:BORDER#1,1,5:SCALE#1,100,0,0
1020 CSIZE#2,2,1:OVER#2,1:Arrow 2,6,14,7:Arrow 2,6,38,7
1021 INK#2,2:FOR i=0 TO 1:CURSOR#2,4+i,8:PRINT#2,'QBITS DARTS'
1022 INK#2,6:FOR i=0 TO 1:CURSOR#2,6+i,9:PRINT#2,'QBITS DARTS'
1023 OVER#2,0:INK#2,3:LINE#2,50,2 TO 50,98 TO 168,98 TO 168,2 TO 50,2
1024 CSIZE#2,2,0:QBold 2,5,0,8,144,204,'CTRL':QBold 2,5,0,8,460,204,'ALT'
1025 END DEFine

1027 DEFine PROCedure Arrow(ch,col,x,y)
1028 INK#ch,col:FILL#ch,1:LINE#ch,x-6,y TO x-8,y+2 TO x-11,y+2
1029 LINE#ch TO x-9,y TO x-11,y-2 TO x-8,y-2 TO x-6,y:FILL#ch,0
1030 INK#ch,7  :FILL#ch,1:LINE#ch,x-11,y+.2 TO x+9,y TO x-11,y-.2:FILL#ch,0
1031 INK#ch,248:FILL#ch,1:CIRCLE#ch,x+2,y,3,.3,PI/2:FILL#ch,0
1032 END DEFine

1034 DEFine PROCedure QBold(ch,col,d,w,x,y,str$)
1035 OVER#ch,1:INK#ch,col:sl=LEN(str$)
1036 FOR a=1 TO sl
1037     FOR b=0 TO d:CURSOR#ch,b+x+a*w,y:PRINT#ch,str$(a)
1038 END FOR a:OVER#ch,0
1039 END DEFine

1042 DEFine PROCedure Clockface
1043 ch=3:CSIZE#ch,2,0:INK#ch,2:FOR cn=1 TO 20:clk(cn)=cn
1044 n$=25 :QBold ch,2,1,9,-9,176,n$:n$=50    :QBold ch,2,1,9,314,176,n$
1045 FOR cn=1 TO 10:n$=cn+10                  :QBold ch,2,1,9,314,cn*16,n$
1046 FOR cn=1 TO  9:n$=cn                     :QBold ch,2,1,9,0  ,cn*16,n$
1047 n$=10                                    :QBold ch,2,1,9,-9,160,n$:cn=1
1048 CSIZE#6,2,1:CURSOR#6,2,3:PRINT#6,'CLOCK':Arrow 2,7,38,49
1049 END DEFine
```

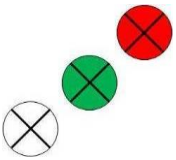


```
1051 DEFine PROCedure Darts_Intro
1052 DIM str$(10,35),dxy(6),clk(29):ch=1:CLS#ch:CSIZE#ch,2,0:INK#ch,7
1053 str$(  1)="    Darts 301 or 501"
1054 str$(  2)=" Start & End on a Double"
1055 str$(  4)=" Clockface 1-20  Doubles"
1056 str$(  5)="   ending with 25 & 50"
1057 str$(  7)="    AIM DART with"
1058 str$(  8)=" Horizontal & Vertical"
1059 str$(  9)="       Sliders"
1060 str$(10)="Use CTRL & ALT to Freeze"
1061 FOR lp=1 TO 10:QBold 1,5,0,10,0,lp*12,str$(lp)
1062 QBold 1,6,0,10,32,140,"Slider Speed ↑     ↓"
1063 QBold 1,7,1,10,12,164,"Select (3)(5)(C) (Q)uit":CURSOR#2,0,0
1064 REPeat key
1065   CURSOR#1,186,140:PRINT#1,FILL$(' ',2-LEN(sp))&sp : k=CODE(INKEY$(-1)):pcol=0
1066   SELect ON k
1067     =208:sp=sp+1:IF sp>15:sp=15
1068     =216:sp=sp-1:IF sp< 5:sp= 5
1069     =51:score1=301:score=301:score2=301:EXIT key
1070     =53:score1=501:score=501:score2=501:EXIT key
1071     =67, 99:pcol=7:EXIT key                    Note: (C)lockFace
1072     =69,101:LRUN dn$:STOP
1073     =27:MODE 4:CSIZE#2,0,0:CSIZE#0,0,0:CLS#0:PRINT#0,'Bye...':STOP
1074   END SELect
1075 END REPeat key
1076 CLS#1:dartbd:bdcnt:bdnums
1077 IF pcol=7
1078    Clockface
1079 ELSE
1080    ch=5:CSIZE#ch,2,1:CURSOR#ch,6,2:PRINT#ch,'RED Team'  :Arrow 2,2,38,71
1081    ch=4:CSIZE#ch,2,1:CURSOR#ch,6,2:PRINT#ch,'GREEN Team':Arrow 2,4,38,27
1082    pcol=0:teamscr:pcol=2
1083 END IF
1084 ch=1:INK#2,7
1085 END DEFine
```

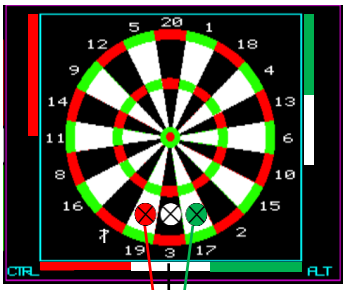



```
1087 DEFine PROCedure Game_End
1088 BEEP 20000,1,10,300,30:CSIZE#1,2,1:PAUSE 50
1089 BLOCK#3,32,194,1,2,0  :BLOCK#3,36,194,314,2,0
1090 BLOCK#3,280,8,34,196,0:CLS#6:CLS#4:CLS#5
1091 ch=1:FOR up=1 TO 50:SCROLL#ch,-4:PAUSE 1
1092 IF pcol=2:shots=shot1:mes$='Winning Team':win$='  REDS  '
1093 IF pcol=4:shots=shot2:mes$='Winning Team':win$=' GREENS '
1094 IF pcol=7:shots=shot3:mes$=' Clock-Face ':win$='Complete'
1095 QBold 1,pcol,1,12,54,60,mes$:QBold 1,pcol,1,14,68,90,win$
1096 QBold 1,7,0,12,32,130,'Darts Thrown: '&shots:PAUSE:Darts_Intro
1097 END DEFine
```

```
1099 DEFine PROCedure QBITS_Darts
1100 dp1=0:shot1=0:dp2=0:shot2=0:dp3=0:shot3=0
1101 REPeat Darts
1102  FOR p=1 TO 6 STEP 2
1103   sliders:dxy(p)=dx:dxy(p+1)=dy
1104   IF pcol=2
1105    dp1=num*dm:shot1=shot1+1
1106    IF score1=score AND dm<>2:dp1=0
1107    IF score1-dp1=0 AND dm=2:Game_End:RETurn
1108    IF score1-dp1<=1 OR score1<dp1:dp1=0:EXIT p
1109    score1=score1-dp1:teamscr
1110   END IF
1111   IF pcol=4
1112    dp2=num*dm:shot2=shot2+1
1113    IF score2=score AND dm<>2:dp2=0
1114    IF score2-dp2=0 AND dm=2:Game_End:RETurn
1115    IF score2-dp2<=1 OR score2<dp2:dp2=0:EXIT p
1116    score2=score2-dp2:teamscr
1117   END IF
1118   IF pcol=7
1119    IF cn<21 AND dm=2 AND num=clk(cn)
1120      IF cn<10:n$=cn:QBold 3,7,1,9,  0,cn*16,n$
1121      IF cn=10:n$=cn:QBold 3,7,1,9, -9,cn*16,n$
1122      IF cn>10:n$=cn:QBold 3,7,1,9,317,(cn-10)*16,n$
1123      cn=cn+1:PAUSE 20:BEEP 2000,5,10
1124    END IF
1125    IF cn=21 AND num=25:cn=cn+1:n$=25:QBold 3,7,1,9, -9,176,n$
1126    IF cn=22 AND num=50:cn=cn+1:n$=50:QBold 3,7,1,9,317,176,n$
1127    shot3=shot3+1:IF cn=23:Game_End:RETurn
1128   END IF
1129  END FOR p
1130  ch=1:PAUSE 20
1131  FOR n=1 TO 6 STEP 2
1132   dx=dxy(n):dy=dxy(n+1):dc=0:dart
1133  END FOR n
1134  dartbd:bdcnt:bdnums:dp1=0:dp2=0:dp3=0:IF pcol<>7:pcol=6-pcol:ELSE pcol=7
1135 END REPeat Darts
1136 END DEFine

1138 DEFine PROCedure teamscr
1139 ch=5:INK#ch,2:CURSOR#ch,6,24:PRINT#ch,score1;"  "
1140 IF pcol=2:CURSOR#ch,p*18,44 :PRINT#ch,FILL$(' ',2-LEN(dp1))&dp1:CLS#ch,4
1141 ch=4:INK#ch,4:CURSOR#ch,6,24:PRINT#ch,score2;"  "
1142 IF pcol=4:CURSOR#ch,p*18,44 :PRINT#ch,FILL$(' ',2-LEN(dp2))&dp2:CLS#ch,4
1143 END DEFine

1145 DEFine PROCedure dart
1146 IF dc>0:BEEP 1000,1,200,60,50,60
1147 ch=1:INK#ch,dc
1148 FILL#ch,1:CIRCLE#ch,dx,dy,2.5:FILL#ch,0
1149 INK#ch,0:LINE#ch,dx-2,dy-2 TO dx+2,dy+2
1150 LINE#ch,dx-2,dy+2 TO dx+2,dy-2
1151 END DEFine
```



```
1153 DEFine PROCedure sliders
1154 ch=2:BLOCK#ch,8,190,174,11,0:BLOCK#ch,8,190,466,11,0
1155 BLOCK#ch,278,6,185,202,0
1156 REPeat lp_x
1157  FOR h=0 TO 278 STEP 4
1158   BLOCK#ch,h,6,185,202,pcol:PAUSE sp/10
1159   IF KEYROW(7)=2:EXIT lp_x
1160  END FOR h
1161  FOR h=278 TO 0 STEP -4
1162   BLOCK#ch,278-h,6,h+185,202,0:PAUSE sp/10
1163   IF KEYROW(7)=2:EXIT lp_x
1164  END FOR h
1165 END REPeat lp_x
1166 dx=(h*.383)                          dx dart x coordinate
1168 REPeat lp_y
1169  FOR v=4 TO 190 STEP 2
1170   BLOCK#ch,8,v,xp,11,pcol:PAUSE sp/10
1171   IF KEYROW(7)=4:EXIT lp_y
1172  END FOR v
1173  FOR v=180 TO 4 STEP -2
1174   BLOCK#ch,8,190-v,xp,11+v,0:PAUSE sp/10
1175   IF KEYROW(7)=4:EXIT lp_y
1176  END FOR v
1177 END REPeat lp_y
1178 dy=100-(v*.521)                      dy dart y coordinate
1179 BLOCK#ch,278,6,185,202,0:BLOCK#ch,8,190,xp,11,0
1180 dr=SQRT((dy-50)^2+(dx-54)^2)         dr dart radius
1181 da=ACOS((dx-54)/dr)                  da dart angle
1182 dc=pcol:dart:dnum                    dc dart colour
1183 END DEFine

1185 DEFine PROCedure dnum
1186 RESTORE 1185
1187 IF dy<50:da=PI+(PI-da)
1188 dt=INT((da+PI/20)/(PI/10))+1
1189 FOR seg=1 TO dt:READ num
1190 dm=1
1191 IF dr > 44 :dm=0
1192 IF dr<=44 AND dr>40:dm=2
1193 IF dr<=24 AND dr>20:dm=3
1194 IF dr<=  4 :num=25
1195 IF dr<1.7:num=50
1196 REMark Dartboard numbers / segment
1197 DATA 6,13,4,18,1,20,5,12,9,14,11,8,16,7,19,3,17,2,15,10,6
1198 END DEFine
```



Sliders used to aim Dart

Dart Positions

calculates Dart positions in terms
of segment number 1 to 20
and dart multiplier
single
double
treble
or 25
or 50

```
1200 DEFine PROCedure dartbd
1201 x=54:y=50:an=PI/20:dx=0:dy=0
1202 FOR f=1 TO 10
1203  c1=2:c2=0:anseg:bdseg
1204  c1=4:c2=7:anseg:bdseg
1205 END FOR f
1206 END DEFine

1208 DEFine PROCedure anseg
1209 x1=COS(an):y1=SIN(an)
1210 an=an+PI/10
1211 x2=COS(an):y2=SIN(an)
1212 END DEFine

1214 DEFine PROCedure bdseg
1215 r=44:c=c1:dwseg
1216 r=40:c=c2:dwseg
1217 r=24:c=c1:dwseg
1218 r=20:c=c2:dwseg
1219 END DEFine
```


```
1221 DEFine PROCedure dwseg
1222 ch=1:FILL#ch,1:INK#ch,c
1223 ARC#ch,x+x1*r,y+y1*r TO x+x2*r,y+y2*r,PI/10
1224 LINE#ch TO x,y TO x+x1*r,y+y1*r:FILL#ch,0
1225 END DEFine

1227 DEFine PROCedure bdcnt
1228 INK#ch,4:FILL#ch,1:CIRCLE#ch,x,y,4  :FILL#ch,0
1229 INK#ch,2:FILL#ch,1:CIRCLE#ch,x,y,1.7:FILL#ch,0
1230 END DEFine

1232 DEFine PROCedure bdnums
1233 RESTORE 1139:ch=1:OVER#ch,1:CSIZE#ch,2,0:INK#ch,7
1234 FOR n=1 TO 20
1235    READ num,nx,ny:CURSOR#ch,nx,ny:PRINT#ch,num
1236 END FOR n
1237 OVER#ch,0                                :REMark Board nums,x,y coordinates
1238 DATA 1,172,5,18,206,18,4,236,38,13,246,62,6,256,90
1239 DATA 10,246,118,15,230,142,2,206,162,17,162,176,3,130,178
1140 DATA 19,86,176,7,60,164,16,20,142,8,13,118,11,2,90
1141 DATA 14,4,62,9,28,38,12,46,18,5,92,5,20,126,1
1142 END DEFine
```

Introduction

While the game's ancient origins are unclear, some historians believe the Roman game of Paganica spread as they conquered much of Europe during the 1st century BC, eventually evolving into the modern game. Others cite Chiwan played in the Ming Dynasty (13688-1644) and introduced to Europe during the Middle Ages. Another contender is Apocryphally, where the Dutch played a game with a stick and leather ball. The winner was whoever hit the ball with the fewest strokes into a target several hundred yards away.

However, it is generally accepted that the modern game of Golf was developed in Scotland from the Middle Ages onwards. The game gained international popularity in the late 19th century when it spread across the rest of the UK and to the then British Empire and the United States.

Computer Golf Games

The first Golf platform game was released in 1979 and appropriately for the first commercial home video games console the Magnavox Odyssey. The Game Leaderboard was released in 1986 for the Amiga, Amstrad, Commodore 64, ZX Spectrum and featured four different water-based courses. Nintendo released Grand Slam in 1991 and Greg Norman's Golf Power in 1992.

As computing power expanded Golf games became more graphically sophisticated as with the PGA Tour Golf games. In 2002 SimGolf challenged players to design their own Golf courses and play them with the in-built PGA pro, Gary Golf.

QBITS Golf Concepts

This began with the decision to create an 18 Hole Golf course with Fairways of varying lengths and difficulty. Graphics to show Power and Direction applied to a Club Drive or Putt. The Fairway created in sections with direction changes and bordered with Trees and Rough ground to the side. Some Fairways to have a Lake and each Green bordered with Bunkers. Then a Wind Speed / Direction indicator used in the calculations of a new ball position.

QBITS Golf Welcome Screen & Menu

An opening screen uses Background and image graphics to introduce the QBITS Golf Game with options for starting a **NEW** game or **LOAD** a previously Saved one or simply **Quit!**

Use the Spacebar to **Tee Off** which will present a new Fairway, but only after a (**N**)ew or (**L**)oad has been triggered. (**S**)ave likewise is inactive until at least one hole has been played. Use (**Q**)uit to leave the Game. Save and Load allows return to the Game at a later time to complete the 18-hole course.

QBITS Golf Wind Speed/Direction

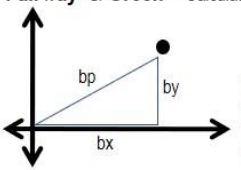
The Wind Speed/Direction indicator is based on a simple eight-point compass design. The current direction shown in Red the others in Cyan with the Speed printed in the centre.

QBITS Golf Power & Direction

Club Power is graphically presented as a Slider raised and lowered by the Up/Down Cursors. The Angle of direction is shown by a rotating bar within an Ellipse below the Golfers feet and direction moved with the Left/Right Cursors. To action the selected Power & Direction press Spacebar.

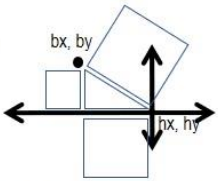
Distance is calculated as a ratio between Power and Fairway length. This is then used to evaluate ball position bx by to which the wind and speed wdx wdy values are included.

Fairway & Green – Calculation of Ball Position



Wind speed/Direction = wdx,wdy

```
1041  IF club=40:wx=0:wy=0:ELSE wx=wdx:wy=wdy
1042  bp=INT((79-ny)*1.3)*(196/Gf(h,2)*cmax/100)
1043  bx=bx+bp*COS(RAD(ang))+wx/Gf(h,2)*bp
1044  by=by+bp*SIN(RAD(ang))+wy/Gf(h,2)*bp
```



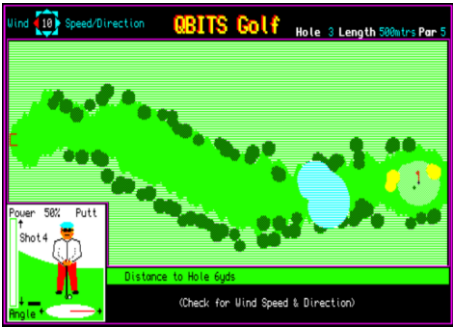
Calculation of distance to hole

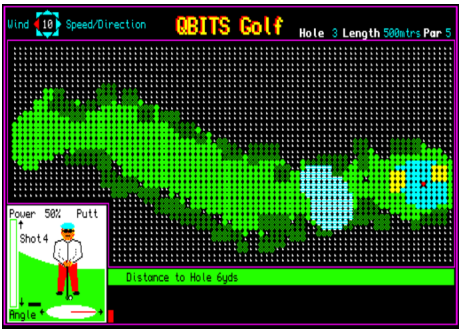
```
1045  lgth=INT(Gf(h,2)/hx*INT(SQRT((hx-bx*2)^2 +(hy-by*2)^2)))
1046  CLS#5:CURSOR#5,20,2:PRINT#5,'Distance to Hole ';lgth;'yds '
```

QBITS Golf Hazards
Club Drive for Fairway is set at 100%, whereas a Putt when on the Green is set to only 50% and the max limit is further reduced. Hitting a Boundary incurs a Penalty Shot, landing in the Rough reduces Drive Power to 50%, landing in a Bunker reduces Power to 25%. If the ball hits a Tree the ball is randomly bounced back until clear. Landing in Water returns the ball to the Tee. The total number of player shots per Fairway are limited to 9.

QBITS Golf Fairway
The challenge was for each Fairway to be represented with differing difficulties, taking into account course Par and Distance. This turned out to be more problematic than originally expected and required playing around with randomised variables to create the change in layouts.

The Fairway is divided into sections starting with the Tee and progressing through three more sections to the fifth the Green. Each section required a border of trees and Bunkers are placed at the edges of the Green. For a few of the longer Fairways a Lake is added. Location of all these are held in an array so that when the ball lands on any area on or off the Fairway any hazards encountered are acted upon.





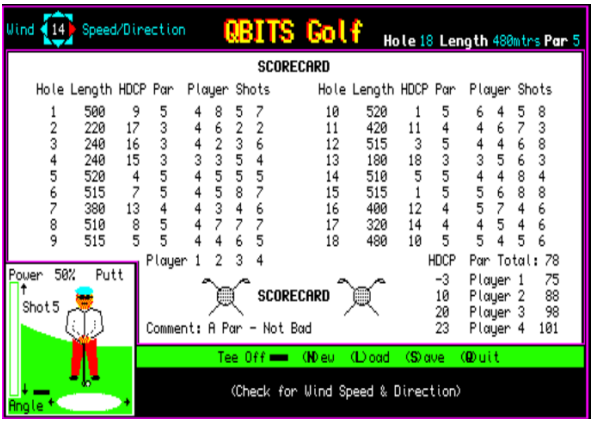
GTest used in the code development displays the Fairway grid((84,40) array configuration with the colour patterns that identify the different elements. Fairway, Green, Hole, Trees, Bunkers, Lake and Rough. The array is used by procedure **Chaz** to process a new ball position.

```
2000 DEFine PROCedure GTest
2001 BLOCK#3,500,129,0,0,0:BLOCK#3,386,52,114,128,0
2002 FOR b=1 TO 40
2003  FOR a=1 TO 84
2004     IF b<13 AND a<20:NEXT a
2005     IF grid(a,b)=   4:INK#3,   4:FILL#3,1:CIRCLE#3,a*2,b*2,.7:FILL#3,0      Fairway
2006     IF grid(a,b)=223:INK#3,223:FILL#3,1:CIRCLE#3,a*2,b*2,.7:FILL#3,0      Green
2007     IF grid(a,b)=224:INK#3,224:FILL#3,1:CIRCLE#3,a*2,b*2,.7:FILL#3,0      Trees
2008     IF grid(a,b)=   6:INK#3,   6:FILL#3,1:CIRCLE#3,a*2,b*2,.7:FILL#3,0      Bunker
2009     IF grid(a,b)=  85:INK#3,  85:FILL#3,1:CIRCLE#3,a*2,b*2,.7:FILL#3,0      Lake
2010     IF grid(a,b)=   0:INK#3,   7:FILL#3,0:CIRCLE#3,a*2,b*2,.7:FILL#3,0      Rough
2011     IF grid(a,b)=255:INK#3,   2:FILL#3,1:CIRCLE#3,a*2,b*2,.4:FILL#3,0      Hole
2012   END FOR a
2013 END FOR b
2014 END DEFine
```

QBITS Golf SCORECARD

At the end of each hole played a SCORECARD is shown with the player results together with three other computer simulated opponents. The Hole order of play 1-18 is shown with Fairway length, a Handicap HDCP (1) being most difficult to easiest (18) with a Par for expected number of strokes. To continue - Tee/Off with Spacebar.

Alternatively, you can (**S**)ave for later or simply (**E**)xit which will not save the present Game.



Score Ratings

"Hole in One! – Superb Shot"
"An Albatross... Incredible"
"Fantastic shot…an Eagle"
"Well played…a Birdie"
"A Par - Not bad!"
"A Single Bogey"
"A Double Bogey"
"Not so good on this hole"
"You are out of shots"

Upon completion SCORECARD calculates and displays your Handicap [HDCP] for the course.

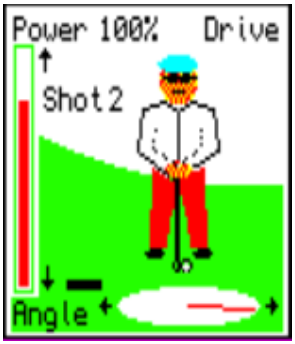
QBITS Golf_Procedures

Init_win	Sets Window sizes etc.
Int_Tee	Draws Power slider, Angle Direction Ellipse etc.
Init_Golfer(x,y)	Golfer Graphics
Init_Wind	Draws Wind – Speed/Direction indicator
Init_Holes	Set Random entries for Fairway and Green
QBGolf_Welcome	Introduction screen
Golf_Trolley(x,y)	Draws Golfers Trolly
GClub(x,y)	Draws Club Symbol
Golf_Game	Main Loop
Golf_Sel	Main Menu:
Commds	Display Menu: Tee Off (N)ew (L)oad (S)ave (E)xit
GFairway(h)	Sets Fairway Sections Tee - Green
Haz(gh,col,hx,hy,hd)	Sets Hazards Trees/ Bunkers
Lake(ch,x,y)	Sets Lake
GDraw(ch,col,fx,fy,fr,fe)	Draws Fairway Tee – Green / Hazards /Lake
GMark(col,mw,md,fx,fy)	Marks grid(84,40) with Fairway Configuration.
CPower	Set / Calculate - Power :Angle : Distance to Hole
CShoot(bx,by)	Moves location of Ball
CHaz(bx,by)	Checks on Boundaries & Hazards
BPlay(str$)	Prints message: Boundaries & Hazards
Scorecard	Displays Course Info & Player Results
GBold(ch,col,cs,cs,cy,str$)	Prints Str$ in Bold

GSave GLoad QExit Confirm Y/N **GTest** Displays Fairway grid(84,40) elements

QBITS Golf Code

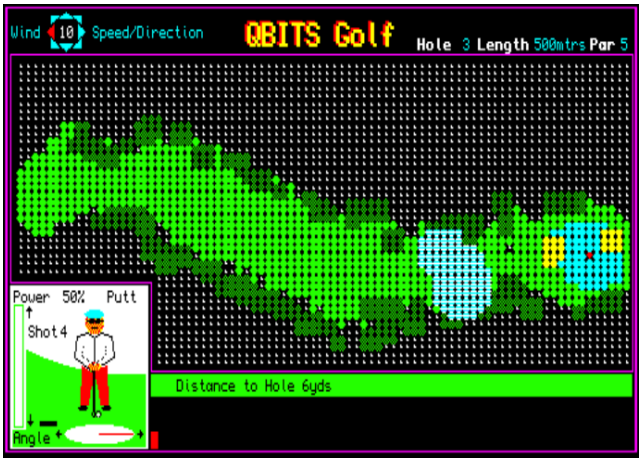
```
1000 REMark QBGolf_v3 [QBITSGolf v3 2021 Revised] WIP

1002 dev$='win1_':MODE 4:gx=0:gy=0 REMark Basic Settings

1004 WHEN ERRor :eck=1:CONTINUE:END WHEN

1006 REMark Import QBITSconfig Settings - QPC2
1007 OPEN_IN#9,dev$&'QBITSConfig' :INPUT#9,gx\gy\dn$\dev$:CLOSE#9

1010 REMark Os-Open screen Gf-Fairway Gn-Green Gs-shots Gp-Golfer Wn-Wind  pw-Par
1011 DIM Gf(18,7),HDCap(18),Gn(18,6),Gs(18,4),Gp(18,6),Wd(8,8),pw(18)
1012 DevFName$=dev$&"QBGolf_v3Dat"

1014 Init_win:h=0:QBITS_Game

1016 DEFine PROCedure QBITS_Golf
1017 REPeat hole
1018   eck=0:Golf_Sel:h=h+1:GFairway h:CWind:CLS#5
1019   bx=1:by=23:sht=0:CPower:Scorecard
1020 END REPeat hole
1021 END DEFine

1023 DEFine PROCedure CPower
1024 cmax=100:club=13:club$='Drive':ang=0:lgth=Gf(h,2)
1025 REPeat h_lp
1026 IF sht>8 OR grid(bx,by)=255 OR lgth<1:EXIT h_lp
1027 INK#4,0:CURSOR#4,36,2:PRINT#4,FILL$(' ',3-LEN(cmax));cmax;'%'
1028 ny=79  :CURSOR#4,78,2:PRINT#4,club$:CURSOR#4,40,22:PRINT#4,sht+1
1029  REPeat T_lp
1030   INK#4,2:LINE#4,57,8.5 TO 57+22*COS(RAD(ang)),8.5+6*SIN(RAD(ang))
1031   k=CODE(INKEY$(-1))
1032   INK#4,7:LINE#4,57,8.5 TO 57+22*COS(RAD(ang)),8.5+6*SIN(RAD(ang))
1033   SELect ON k
1034    =192:ang=ang+7.5:IF ang>=360:ang=0
1035    =200:ang=ang-7.5:IF ang<=0:ang=360
1036    =208:BLOCK#4,4,1,5,ny,2:ny=ny-1:IF ny<club:ny=club
1037    =216:BLOCK#4,4,1,5,ny,7:ny=ny+1:IF ny>79:ny=79
1038    = 32:BLOCK#4,4,68,5,13,7:sht=sht+1:EXIT T_lp
1039   END SELect
1040  END REPeat T_lp
1041   IF club=40:wx=0:wy=0:ELSE wx=wdx:wy=wdy
1042   bp=INT((79-ny)*1.3)*(196/Gf(h,2)*cmax/100)
1043   bx=bx+bp*COS(RAD(ang))+wx/Gf(h,2)*bp
1044   by=by+bp*SIN(RAD(ang))+wy/Gf(h,2)*bp
1045   lgth=INT(Gf(h,2)/px*INT(SQRT((px-bx*2)^2 +(py-by*2)^2)))
1046   CLS#5:CURSOR#5,20,2:PRINT#5,'Distance to Hole ';lgth;'yds  '
1047   CShoot bx,by
1048 END REPeat h_lp
1049 IF sht>9:Gs(h,1)=9:ELSE Gs(h,1)=sht
1050 END DEFine
```



```
1052 DEFine PROCedure CShoot(bx,by)
1053 bcol=grid(x0,y0):IF bcol=0:bcol=95
1054 INK#3,bcol:FILL#3,1:CIRCLE#3,x0*2,y0*2,.4:FILL#3,0
1055 IF club=40:Flag px,py
1056 CHaz bx,by:x0=bx:y0=by
1057 FOR i=1 TO 4
1058  INK#3,bcol:FILL#3,1:CIRCLE#3,bx*2,by*2,.9:FILL#3,0
1059  INK#3,0   :PAUSE 5 :CIRCLE#3,bx*2,by*2,.4:PAUSE 5
1060 END FOR i
1061 END DEFine
```

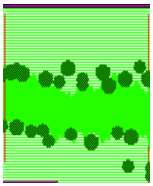
Note: Club Shoot

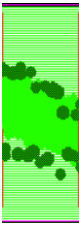
Note: Test display showing hazards

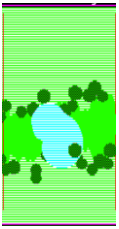



```
1063 DEFine PROCedure CHaz(bx,by)
1064 xmin=1:xmid=20:xmax=84:ymin=1:ymid=13:ymax=40:bck=0:cmax=100
1065 IF bx>xmax:bx=xmax:bck=1
1066 IF bx<xmin:bx=xmin:bck=1
1067 IF by>ymax:by=ymax:bck=1
1068 IF by<ymin:by=ymin:bck=1
1069 IF bx<xmid AND by<ymid:by=ymid:bck=1
1070 IF bck=1:BPlay 'Out of Bounds / Penalty Shot':sht=sht+1:RETurn
1071 bcol=grid(bx,by)
1072 IF bcol=223:club$='Putt ':club=40:cmax=50      :RETurn
1073 IF bcol=  0:BPlay 'Ball in the Rough!'        :cmax=50:bcol=95
1074 IF bcol=  6:BPlay 'Ball Bunkered'            :cmax=25
1075 IF bcol= 85:CLS#5:BPlay 'Water! Back to Tee':bx=1:by=23:bcol=4
1076 IF bcol=224
1077   BPlay 'Ball hit a Tree':PAUSE 30:BPlay ''
1078   x0=bx:y0=by:bx=bx-1:by=by+RND(-1 TO 1):PAUSE 20:CShoot bx,by
1079 END IF
1080 END DEFine

1082 DEFine PROCedure BPlay(str$)
1083 CURSOR#5,200,2:PRINT#5,str$:CLS#5,4
1084 END DEFine
```

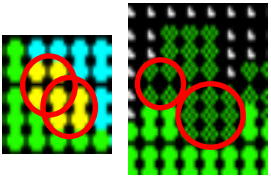
```
1086 DEFine PROCedure GFairway(h)
1087 DIM grid(84,40):RANDOMISE:par=Gf(h,1):dist=Gf(h,2):INK#2,5
1088 CURSOR#2,362,16:PRINT#2,FILL$(' ',2-LEN(h))&h
1089 CURSOR#2,426,16:PRINT#2,dist;'mtrs':CURSOR#2,496,16:PRINT#2,par
1090 BLOCK#3,500,129,0,0,95:BLOCK#3,386,52,114,128,95
1091 REMark Fairway Tee
1092 RESTORE 1086:FOR fx=1 TO 10:READ fy,fd:Haz 9,4,fx,fy,fd
1093 DATA 23,3,23,4,23,5,23,5,24,6,25,5,26,6,26,6,26,5,26,5
1094 INK#3,2:LINE#3,3,44 TO .5,44 TO .5,48 TO 3,48
1095 fy=26:x0=1:y0=23:grid(x0,y0)=4:CIRCLE#3,x0*2,y0*2,.4
1096 REMark Fairway Sect1
1097 FOR fx=fx+1 TO Gf(h,5)
1098   fy=fy+(-par+RND(1 TO 3))/10:fd=9-par+RND(0 TO 1):fr=fd+1
1099   IF fy+fd>38:fy=36-fd
1100   IF fy-fd<18:fy=20+fd
1101   Haz 9,4,fx,fy,fd
1102   Haz RND(0 TO 3),224,fx-RND(2 TO 4),fy+fd+RND(0 TO 2),fd
1103   Haz RND(0 TO 4),224,fx-RND(2 TO 3),fy-fd+RND(0 TO 1),fd
1104 END FOR fx
1105 REMark Fairway Sect2
1106 FOR fx=fx+1 TO Gf(h,6)
1107   fy=fy+(-par+RND(-1 TO 2))/10:fd=9+RND(2 TO 3)-par:fr=fd+1
1108   IF fy+fd>38:fy=36-fd
1109   IF fy-fd< 5:fy= 7+fd
1110   Haz 9,4,fx,fy,fd
1111   Haz RND(0 TO 2),224,fx-RND(2 TO 4),fy+fd+RND( 1 TO 2),fd
1112   Haz RND(0 TO 3),224,fx-RND(3 TO 5),fy-fd+RND(-1 TO 0),fd
1113 END FOR fx
1114 REMark Fairway Sect3
1115 FOR fx=fx+1 TO 72
1116   fy=fy+(-par+RND(4 TO 8))/10:fd=9-par+RND(0 TO 2):fr=fd+1
1117   IF fy+fd>38:fy=36-fd
1118   IF fy-fd< 5:fy= 7+fd
1119   Haz 9,4,fx,fy,fd
1120   Haz RND(0 TO 3),224,fx-RND(2 TO 5),fy+fd+RND( 0 TO 1),fd
1121   Haz RND(0 TO 4),224,fx-RND(3 TO 5),fy-fd+RND(-2 TO 0),fd
1122 END FOR fx
1123   IF par>3 AND RND(1 TO 3)=3:Lake 3,fx-10,fy+RND(-2 TO +2)
1124 REMark Fairway Green
1125 y2=fy+Gn(h,1):RESTORE 1119:FOR fx=73 TO 84:READ fd:Haz 9,4,fx,fy,fd
1126 DATA 5,5,6,6,6,7,7,6,6,6,5,4
1127 GDraw 3,223,158,fy*2,8,1:FOR fx=76 TO 83:READ fd:GMark 223,fx,fy-fd,fx,fy+fd
1128 DATA 3,3,4,4,4,4,3,3
1129 GDraw 3,223,154,y2*2,5,1:FOR fx=74 TO 77:READ fd:GMark 223,fx,y2-fd,fx,y2+fd
1130 DATA 2,2,3,3
1131 grid(79,fy)=255:px=79*2:py=fy*2:Flag px,py
1132 FOR fx=64 TO 82
1133   Haz RND(0 TO 5),224,fx,fy+6+RND( 0 TO 1)
1134   Haz RND(0 TO 4),224,fx,fy-5+RND(-2 TO 0)
1135 END FOR fx
1136 Haz 1,6,73,fy+Gn(h,2),1:Haz 2,6,74,fy+Gn(h,3),1:Haz 1,6,73,fy+Gn(h,4),1
1137 Haz 1,6,81,fy+Gn(h,5),0:Haz 2,6,82,fy+Gn(h,6),1
1138 END DEFine
```





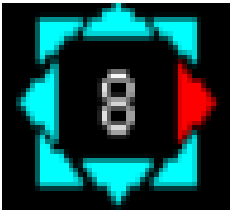


```
1140 DEFine PROCedure Haz(gh,col,hx,hy,hd)
1141 IF gh=1:GMark col,hx,hy-1,hx+1,hy      :GDraw 3,col,hx*2+1,hy*2-1,2,1
1142 IF gh=2:GMark col,hx-1,hy-1,hx+1,hy+1:GDraw 3,col,hx*2  ,hy*2  ,2.5,1
1143 IF gh=9:GMark col,hx,hy-hd,hx,hy+hd  :GDraw 3,col,hx*2  ,hy*2  ,hd*2,.2
1144 :END DEFine
```

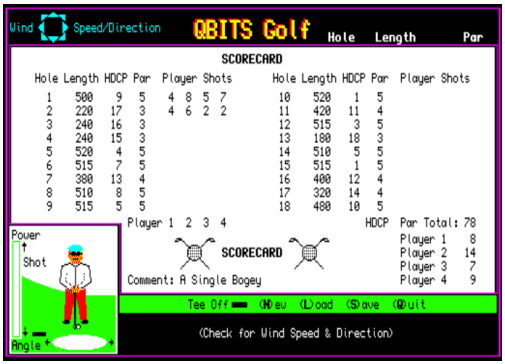


```
1146 DEFine PROCedure Lake(ch,x,y)
1147 INK#3,85:FILL#3,1:ARC#3,x*2,y*2 TO x*2-12,y*2-6,PI TO x*2-8,y*2-10,PI/2
1148 ARC#3 TO x*2+6,y*2-16,PI TO x*2,y*2,PI /2:FILL#ch,0
1149 RESTORE 1143:x=x-6:y=y+1:FOR i=0 TO 9:READ y1,y2:GMark 85,x+i,y+y1,x+i,y+y2
1150 DATA -4,-1,-5,0,-9,0,-10,0,-11,0,-11,0,-11,-2,-11,-2,-10,-3,-9,-5
1151 END DEFine

1153 DEFine PROCedure GDraw(ch,col,fx,fy,fr,fe)
1154 INK#ch,col:FILL#ch,1:CIRCLE#ch,fx,fy,fr,fe,PI:FILL#ch,0
1155 END DEFine

1157 DEFine PROCedure GMark(col,mw,md,fx,fy)
1158 FOR my=md TO fy:FOR mx=mw TO fx:grid(mx,my)=col:END FOR mx:END FOR my
1159 END DEFine

1161 DEFine PROCedure Flag(px,py)
1162 INK#3,2:LINE#3,px,py TO px,py+4 TO px-1,py+3.5 TO px,py+3
1163 INK#3,0:FILL#3,1:CIRCLE#3,px,py,.4:FILL#3,0
1164 END DEFine
```

```
1166 DEFine PROCedure CWind
1167 wdx=0:wdy=0:wnd=Gf(h,3):wns=Gf(h,4):IF wns<10:cp=44:ELSE cp=41
1168 INK#2,7:CURSOR#2,41,9:PRINT#2,'  ':CURSOR#2,cp,9:PRINT#2,wns
1169 IF h>1
1170   a=Gf(h-1,3):INK#2,5:FILL#2,1:LINE#2,Wd(a,1),Wd(a,2) TO Wd(a,3),Wd(a,4)
1171   LINE#2 TO Wd(a,5),Wd(a,6) TO Wd(a,7),Wd(a,8):FILL#2,0
1172 END IF
1173   a=Gf(h,3)  :INK#2,2:FILL#2,1:LINE#2,Wd(a,1),Wd(a,2) TO Wd(a,3),Wd(a,4)
1174   LINE#2 TO Wd(a,5),Wd(a,6) TO Wd(a,7),Wd(a,8):FILL#2,0
1175 SELect ON wnd
1176   ON wnd=1:wdx=0          :wdy=wns     :REMark north
1177   ON wnd=2:wdx=wns/2      :wdy=wns/2   :REMark n/e
1178   ON wnd=3:wdx=wns        :wdy=0       :REMark east
1179   ON wnd=4:wdx=wns/2      :wdy=-wns/2  :REMark s/e
1180   ON wnd=5:wdx=0          :wdy=-wns    :REMark south
1181   ON wnd=6:wdx=-wns/2     :wdy=-wns/2  :REMark s/w
1182   ON wnd=7:wdx=-wns       :wdy=0       :REMark west
1183   ON wnd=8:wdx=-wns/2     :wdy=wns/2   :REMark n/w
1184 END SELect
1185 END DEFine
```



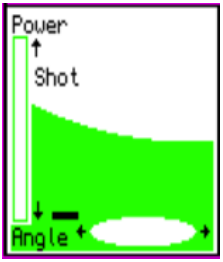
```
1187 DEFine PROCedure QBold(ch,col,cs,cx,cy,str$)
1188 INK#ch,col:OVER#ch,1
1189 FOR a=1 TO LEN(str$)
1190   FOR b=0 TO cs:CURSOR#ch,cx+b+a*(6+cs),cy:PRINT#ch,str$(a)
1191 END FOR a:OVER#ch,0
1192 END DEFine

1194 DEFine PROCedure Scorecard
1195 BLOCK#3,500,129,0,0,7:BLOCK#3,386,52,114,128,7 :STRIP#3,7
1196 QBold 3,0,1,210,4,'SCORECARD':QBold 3,0,1,210,146,'SCORECARD'
1197 CURSOR#3,  24,18:PRINT#3,"Hole Length HDCP Par  Player Shots"
1198 CURSOR#3,270,18:PRINT#3,"Hole Length HDCP Par  Player Shots"
1199 cpar=0:FOR hp=1 TO 18:cpar=cpar+Gf(hp,1)
1200 CURSOR#3,366,124:PRINT#3,'HDCP  Par Total: ';cpar
1201 CURSOR#3,120,124:PRINT#3,'Player 1  2  3  4'
1202 FOR c=1 TO 18
1203  IF c<10:CURSOR#3,36,22+c*10:ELSE CURSOR#3,276,22+(c-9)*10
1204 PRINT#3,c;"    ";Gf(c,2);FILL$(' ',5-LEN(Gf(c,7)));Gf(c,7);"    "; Gf(c,1)
1205 END FOR c
1206 Sc=0:FOR i=1 TO 4:pw(i)=0
1207 FOR hs=1 TO h
1208  FOR i=1 TO 4
1209   IF hs<10:CURSOR#3,144+i*18,22+hs*10:ELSE CURSOR#3,390+i*18,22+(hs-9)*10
1210   PRINT#3,Gs(hs,i):pw(i)=pw(i)+Gs(hs,i):n$=pw(i)
1211   CURSOR#3,402,126+i*10:PRINT#3,'Player ';i;FILL$(' ',5-LEN(n$))&n$
1212   IF hs=18:CURSOR#3,366,126+i*10:n$=n$-cpar:PRINT#3,FILL$(' ',3-LEN(n$))&n$
1213  END FOR i
1214  Sc=Sc+Gs(hs,1)-Gf(hs,1):sht=Gs(h,1):par=Gf(h,1)
1215 END FOR hs
1216 GClub 64,13:GClub 104,13:CURSOR#3,120,166:PRINT#3,'Comment: ';
1217 IF sht=1        :PRINT#3,"Hole in One! - Superb Shot"     :RETurn
1218 IF sht=par-3  :PRINT#3,"An Albatross... Incredible"
1219 IF sht=par-2  :PRINT#3,"Fantasitc shot... An Eagle"
1220 IF sht=par-1  :PRINT#3,"Well played... A birdie"
1221 IF sht=par      :PRINT#3,"A Par - Not Bad"
1222 IF sht=par+1  :PRINT#3,"A Single Bogey"
1223 IF sht=par+2  :PRINT#3,"A Double Bogey"
1224 IF sht>8        :PRINT#3,"You are out of Shots"              :RETurn
1225 IF sht>par+2 :PRINT#3,"Not so good on this Hole"
1226 END DEFine
```



```
1228 DEFine PROCedure Golf_Sel
1229 REPeat lp
1230  Commds:k=CODE(INKEY$(-1))
1231  IF k=32 AND chk=1 AND h<18:EXIT lp
1232  IF k=69 OR k=101 :QExit:BLOCK#5,20,10,230,2,4
1233  IF k=78 OR k=110 :chk=1:Init_Holes:h=0:EXIT lp
1234  IF k=83 OR k=115 :IF chk=1:GSave
1235  IF k=76 OR k=108 :GLoad: IF eck=0:chk=1:Scorecard
1236 END REPeat lp
1237 END DEFine

1239 DEFine PROCedure Commds
1240 CLS#5:BLOCK#5,16,4,116,5,0:OVER#5,1
1241 CURSOR#5, 70,2:PRINT#5,'Tee Off     (N)ew  (L)oad  (S)ave  (E)xit'
1242 CURSOR#5,149,2:PRINT#5,'N      L       S       E ' :OVER#5,0
1243 END DEFine

1245 DEFine PROCedure QExit
1246 CURSOR#5,320,2:PRINT#5,'Y/N':PAUSE:IF KEYROW(5)=64:LRUN dn$:STOP
1247 END DEFine

1249 DEFine PROCedure GSave
1250 CLS#5:CURSOR#5,76,2:PRINT#5,"Please wait - Saving..."
1251 DELETE DevFName$:OPEN_NEW#9,DevFName$:PRINT#9,h
1252 FOR a=1 TO 18
1253   FOR b=1 TO 7:PRINT#9,Gf(a,b):END FOR b
1254   FOR b=1 TO 6:PRINT#9,Gn(a,b):END FOR b
1255   FOR b=1 TO 4:PRINT#9,Gs(a,b):END FOR b
1256   CURSOR#5,196+a*6,2:PRINT#5,'.':PAUSE 1
1257 END FOR a:CLOSE#9
1258 END DEFine

1260 DEFine PROCedure GLoad
1261 CLS#5:CURSOR#5,76,2:PRINT#5,"Please wait - Loading..."
1262 OPEN_IN#9,DevFName$:INPUT#9,h
1263 FOR a=1 TO 18
1254   FOR b=1 TO 7:INPUT#9,Gf(a,b):END FOR b
1255   FOR b=1 TO 6:INPUT#9,Gn(a,b):END FOR b
1256   FOR b=1 TO 4:INPUT#9,Gs(a,b):END FOR b
1267   CURSOR#5,196+a*6,2:PRINT#5,'.':PAUSE 1
1268 END FOR a
1269 CLOSE#9:Init_wind:BLOCK#4,10,10,38,22,7:BLOCK#4,76,10,36,2,7
1270 BLOCK#2,14,10,362,16,0:BLOCK#2,42,10,426,16,0:BLOCK#2,8,10,496,16,0
1271 END DEFine
```

```
1273 DEFine PROCedure Init_win
1274 OPEN#6,scr_:WINDOW#6,512,256,gx,gy:BORDER#6,1,3:PAPER#6,0:CLS#6
1275 OPEN#5,scr_:WINDOW#5,390,13,gx+118,gy+211:PAPER#5,4:INK#5,0:CLS#5
1276 OPEN#4,scr_:WINDOW#4,112,94,gx+4,gy+160  :PAPER#4,7:CLS#4:SCALE#4,100,0,0
1277 OPEN#3,scr_:WINDOW#3,500,180,gx+6,gy+29  :SCALE#3,82,0,0
1278 WINDOW#2,508,212,gx+2,gy+1:PAPER#2,0:SCALE#2,140,0,0
1279 WINDOW#1,246,166,gx+250,gy+36:
1280 WINDOW#0,390,30,gx+118,gy+224:PAPER#1,0:CSIZE#0,0,0
1281 CSIZE#2,2,1:OVER#2,1
1282 INK#2,2:FOR i=0 TO 1:CURSOR#2,190+i,5:PRINT#2,'QBITS Golf'
1283 INK#2,6:FOR i=0 TO 1:CURSOR#2,192+i,6:PRINT#2,'QBITS Golf'
1284 CSIZE#2,0,0:OVER#2,0
1285 INK#2,3:LINE#2,1,36 TO 1,123 TO 248,123 TO 248,2 TO 57,2 TO 57,36 TO 1,36
1286 QBold 2,7,1,324,16,'Hole   Length      Par'
1287 CURSOR#0,80,10:PRINT#0,"(Check for Wind Speed & Direction)"
1288 Init_Tee:Init_Golfer 54,74:Init_wind:QBGolf_Welcome:chk=0:Commds
1289 END DEFine
```

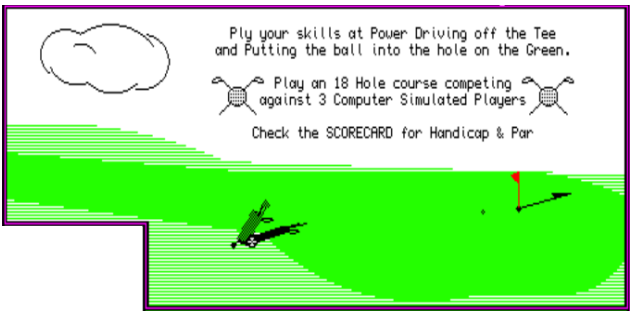
```
1291 DEFine PROCedure Init_Tee
1292 FILL#4,1:INK#4,4:CSIZE#4,0,0
1293 ARC#4,10,60 TO 86.5,45,PI/6:LINE#4 TO 86.5,1  TO 10,1 TO 10,60
1294 FILL#4,0:INK#4,0:CURSOR#4,14,22:PRINT#4,'Shot':STRIP#4,4
1295 CURSOR#4, 2,83:PRINT#4,'Angle':CURSOR#4,36,80:PRINT#4,'¼        ½'
1296 CURSOR#4,13,72:PRINT#4,'¿':STRIP#4,7:BLOCK#4,14,3,24,78,0
1297 CURSOR#4,12,10:PRINT#4,'¾':CURSOR#4,2,2:PRINT#4,'Power'
1298 INK#4,7:FILL#4,1:CIRCLE#4,57,8,22,.25,PI/2:FILL#4,0                :REMark Angle
1299 INK#4,4:LINE#4,2,13 TO 2,88 TO 8,88 TO 8,13 TO 2,13                :REMark Power
1300 END DEFine
```




```
1302 DEFine PROCedure Init_Golfer(x,y)
1303 INK#4,226:FILL#4,1:CIRCLE#4,x,y+3,7,.8,PI:FILL#4,0                 :REMark Head
1304 INK#4,0:LINE#4,x-5,y+4 TO x+6,y+4:LINE#4,x-2,y TO x+2,y            :REMark Mouth
1305 CIRCLE#4,x-2,y+4,2,.6,PI/2:CIRCLE#4,x+2,y+4,2,.6,PI/2              :REMark Specs
1306 INK#4,5:FILL#4,1:CIRCLE#4,x,y+8.5,6,.4,PI/2:FILL#4,1               :REMark Cap
1307 FILL#4,1:INK#4,2                                                  :REMark Left Leg
1308 LINE#4,x,y-29 TO x-3,y-40 TO x-4,y-47 TO x-9,y-47 TO x-8,y-23 TO x,y-29
1309 FILL#4,0:FILL#4,1                                                 :REMark Right leg
1310 LINE#4,x,y-27 TO x+3,y-40 TO x+4,y-47 TO x+9,y-47 TO x+8,y-23 TO x,y-27
1311 FILL#4,0:INK#4,0
1312 FILL#4,1:CIRCLE#4,x-7,y-49,3,.4,PI/2:FILL#4,0                     :REMark Left Shoe
1313 FILL#4,1:CIRCLE#4,x+8,y-49,3,.4,PI/2:FILL#4,0                     :REMark Right Sh
1314 REMark Left / Right Body
1315 LINE#4,x,y-6 TO x-4,y-3 TO x-11,y-5 TO x-13,y-18 TO x,y-28 TO x,y-6
1316 LINE#4,x,y-6 TO x+4,y-3 TO x+11,y-5 TO x+13,y-18 TO x,y-26 TO x,y-6
1317 LINE#4,x,y-22 TO x,y-55:ARC#4 TO x-1,y-51,-PI                      :REMark Club
1318 LINE#4 TO x-1,y-22 TO x,y-22:FILL#4,0:CIRCLE#4,x+2,y-53,1.8
1319 INK#4,7:FILL#4,1:CIRCLE#4,x+2,y-53,1:FILL#1,0                      :REMark Golf Ball
1320 INK#4,226:FILL#4,1:CIRCLE#4,x,y-24,4,.6,PI/2:FILL#4,0              :REMark Hands
1321 INK#4,248:LINE#4,x-7,y-11 TO x-8,y-17 TO x,y-22  TO x+7,y-17 TO x+7,y-11
1322 END DEFine
```

```
1324 DEFine PROCedure Init_wind
1325 RESTORE 1327:x=23:y=131.5
1326 FOR a=1 TO 8
1327   FOR b=1 TO 8 STEP 2:READ dat1,dat2:Wd(a,b)=x+dat1:Wd(a,b+1)=y+dat2
1328 END FOR a
1329 INK#2,5:CURSOR#2,2,9:PRINT#2,'Wind       Speed/Direction':INK#2,5
1330 FOR a=1 TO 8
1331   FILL#2,1:LINE#2,Wd(a,1),Wd(a,2) TO Wd(a,3),Wd(a,4)
1332   LINE#2 TO Wd(a,5),Wd(a,6) TO Wd(a,1),Wd(a,2):FILL#2,0
1333 END FOR a
1334 DATA 0,7.5,-2.5,5,2.5,5,0,7.5,6,6,6,3,3,6,6,6          :REMark North N/E
1335 DATA 7.5,0,5,2.5,5,-2.5,7.5,0,6,-6,6,-3,3,-6,6,-6     :REMark East  S/E
1336 DATA 0,-7.5,2.5,-5,-2.5,-5,0,-7.5,-6,-6,-3,-6,-6,-3,-6,-6   :REMark south
1337 DATA -7.5,0,-5,-2.5,-5,2.5,-7.5,0,-6,6,-3,6,-6,3,-6,6       :REMark west
1338 END DEFine


1340 DEFine PROCedure Init_Holes
1341 RANDOMISE:CLS#5:CURSOR#5,48,2:PRINT#5,"Please wait - Initialising..."
1342 FOR h=1 TO 18
1343   Gf(h,1)=RND(3 TO 5):par=Gf(h,1)                  :REMark Par
1344   Gf(h,2)=140*(par-2)+20*(par+RND(-2 TO 3))        :REMark Distance
1345   IF Gf(h,2)<180:Gf(h,2)=180+RND(1 TO 3)*5         :REMark Check lght
1346   IF Gf(h,2)>525:Gf(h,2)=525-RND(1 TO 3)*5         :REMark Check lgth
1347   Gf(h,3)=RND(1 TO  8)                             :REMark Wind Direction
1348   Gf(h,4)=RND(1 TO 15)                             :REMark Wind Strength
1349   Gf(h,5)=RND(20 TO 40)                            :REMark fxe Sect 1
1350   Gf(h,6)=RND(48 TO 52)                            :REMark fxe Sect 2
1351   Gf(h,7)=Gf(h,2)+h:HDCap(h)=Gf(h,7)               :REMark HDCP
1352   Gn(h,1)=RND(-1 TO 1)                             :REMark Green Angle
1353   Gn(h,2)=RND(-1 TO 0)                             :REMark Bunker 1
1354   Gn(h,3)=RND( 0 TO 1)                             :REMark Bunker 1
1355   Gn(h,4)=RND(-2 TO 0)                             :REMark Bunker 1
1356   Gn(h,5)=RND( 3 TO 4)                             :REMark Bunker 2
1357   Gn(h,6)=RND( 2 TO 3)                             :REMark Bunker 2
1358   REMark Player Score Gs(h,1) + Gs(h,2 to 4) Computer Players
1359   Gs(h,1)=0:FOR p=2 TO 4:Gs(h,p)=Gf(h,1)+RND(-1 TO 3):END FOR p
1360   CURSOR#5,210+h*6,2:PRINT#5,".":PAUSE 1
1361 END FOR h
1362 FOR h=17 TO 1 STEP -1
1363   FOR i=1 TO h
1364     num1=HDCap(i):num2=HDCap(i+1)                  :REMark Sort HDCP
1365     IF num2>num1:HDCap(i+1)=num1:HDCap(i)=num2
1366   END FOR i
1367 END FOR h
1368 FOR h=1 TO 18
1369   FOR i=1 TO 18:IF Gf(h,7)=HDCap(i):Gf(h,7)=i:END IF
1370 END FOR h
1371 END DEFine
```

```
1373 DEFine PROCedure QBGolf_Welcome
1374 BLOCK#3,500,100,0,0,7:INK#3,0:STRIP#3,7
1375 ARC#3,40,72 TO 20,75,PI/2 TO 12,64,PI:ARC#3,24,70 TO 18,72,PI/2
1376 ARC#3,12,66 TO 22,62,PI/2 TO 36,62,PI/2:ARC#3,35,60 TO 38,75,PI
1377 AT#3,1,30:PRINT#3,"Ply your skills at Power Driving off the Tee"
1378 AT#3,2,28:PRINT#3,"and Putting the ball into the hole on the Green."
1379 AT#3,4,36:PRINT#3,"Play an 18 Hole course competing"
1380 AT#3,5,34:PRINT#3,"against 3 Computer Simulated Players"
1381 AT#3,7,33:PRINT#3,"Check the SCORECARD for Handicap & Par"
1382 INK#3,95:FILL#3,1:LINE#3,170,.5 TO 170,36 TO 60,40:ARC#3 TO 0,50,PI/6
1383 LINE#3 TO 0,24 TO 39,24 TO 39,.5 TO 170,.5:FILL#3,0
1384 INK#3,4:FILL#3,1:LINE#3,0,42 TO 115,33:ARC#3 TO 170,30,-PI/4 TO 140,8,-PI/1.2
1385 ARC#3 TO 60,35,-PI/2:LINE#3 TO 115,14 TO 0,30 TO 0,42:FILL#3,0
1386 INK#3,2:FILL#3,1:LINE#3,140,27 TO 140,37 TO 138,36 TO 140,34 TO 140,27
1387 INK#3,0:CIRCLE#3,140,27,.6:FILL#3,0
1388 FILL#3,1:LINE#3,140,27 TO 154,31 TO 149,31 TO 150,30:FILL#3,0
1389 CIRCLE#3,130,26,.4:Golf_Trolley 70,25:GClub 63,58:GClub 148,58
1390 END DEFine
```




```
1392 DEFine PROCedure Golf_Trolley(x,y)
1393 INK#3,0:LINE#3,x-6,y-6 TO x-8,y-8:CIRCLE#3,x+8,y-4.5,2,.2,PI/2
1394 FILL#3,1:LINE#3,x-7,y-7 TO x-4,y-8 TO x+8,y-4 TO x+5,y-3 TO x-7,y-7:FILL#3,0
1395 LINE#3,x+4,y-4 TO x+12,y-2:LINE#3,x+3,y-4 TO x+10,y-2
1396 CIRCLE#3,x+1,y-1,2,.2,PI/2 :LINE#3,x,y+.4 TO x-3,y+1.4
1397 INK#3,224:FILL#3,1:LINE#3,x,y TO x-3,y+1 TO x-7,y-6 TO x-4,y-7 TO x,y:FILL#3,0
1398 INK#3,  0:FILL#3,1:CIRCLE#3,x-8,y-8,.6 :FILL#3,0
1399 INK#3,  0:FILL#3,1:CIRCLE#3,x-3,y-7,1.6:FILL#3,0
1400 INK#3,  7:FILL#3,1:CIRCLE#3,x-3,y-7,.9 :FILL#3,0
1401 INK#3,  0:LINE#3,x-4,y-6 TO x-2,y-8:LINE#3,x-4,y-8 TO x-2,y-6
1402 LINE#3,x-1.5,y TO x-.5,y+3 :CIRCLE#3,x,y+3,.4
1403 LINE#3,x-1,y TO x,y+4       :CIRCLE#3,x+.5,y+4,.4
1404 LINE#3,x-.5,y TO x+1,y+3   :CIRCLE#3,x+1.2,y+3,.4
1405 END DEFine
```

```
1407 DEFine PROCedure GClub(x,y)
1408 LINE#3,x-5,y-5 TO x+5,y+5:ARC#3 TO x+7,y+4,-PI/2:LINE#3 TO x+4,y+4.5
1409 LINE#3,x+5,y-5 TO x-5,y+5:ARC#3 TO x-7,y+4, PI/2:LINE#3 TO x-4,y+4.5
1410 INK#3,63:FILL#3,1:CIRCLE#3,x,y,2:FILL#3,0:INK#3,0:CIRCLE#3,x,y,2.5
1411 END DEFine
```