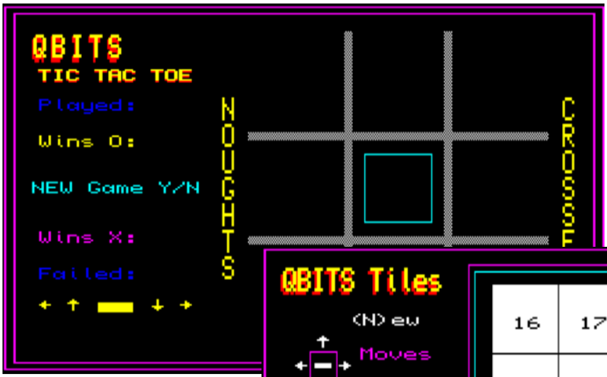
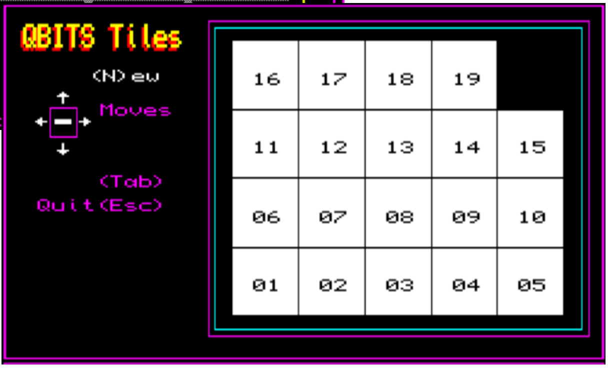
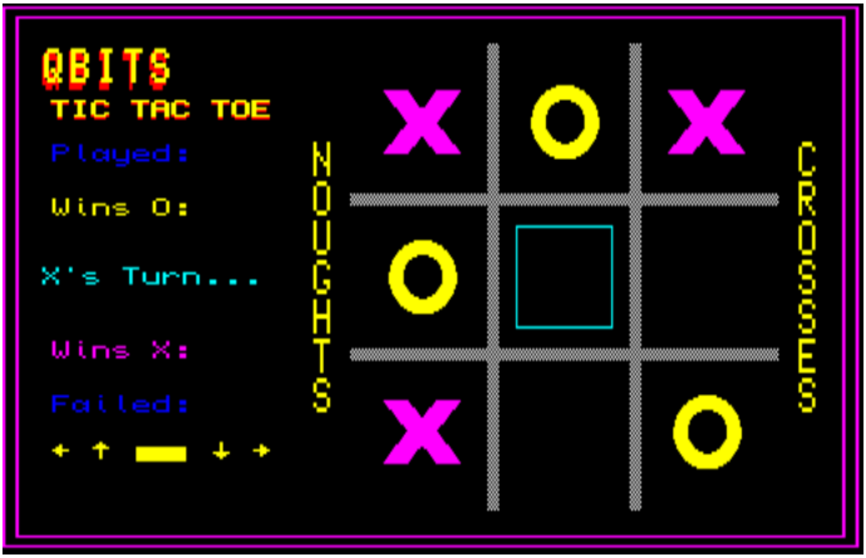
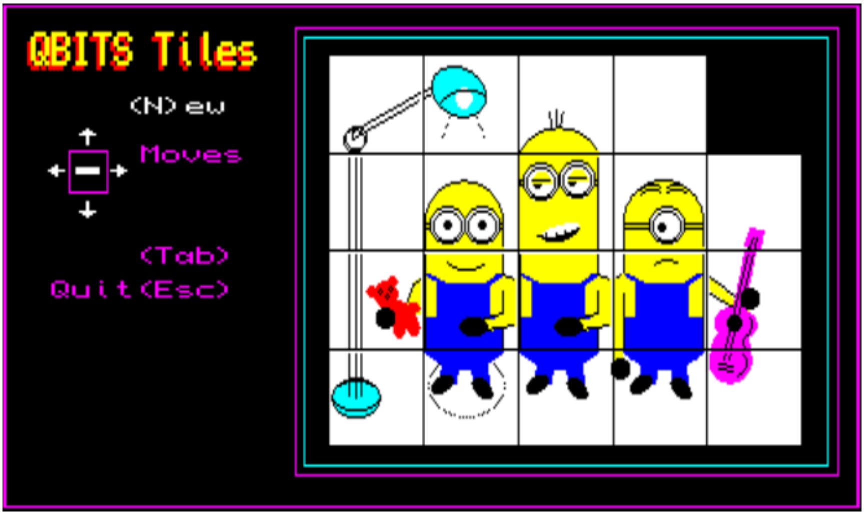
Sinclair QL Retro Gaming





Sinclair QL Retro Gaming


QBITS
TIC TAC TOE
Played:
Wins O:
X's Turn...
Wins X:
Failed:
NOUGHTS
CROSSES


QBITS Tiles
(N)ew
Moves
(Tab)
Quit(Esc)

QBITS Introduction

Nought & Crosses (Tic Tac Toe) or Sliding Tile Games were very much part of my Childhood. The children of today still enjoy playing such games even if they have been ported to tablets and the like.

When I began writing S/SuperBASIC code back in the eighties, these Games were what I considered to be my Coffee Cup Challenges. They began as doddles drawing the screen layouts and imagining the coding to achieve my graphical displays.

The handling of keyed Inputs and the actions that might deriving from such was an intriguing challenge to my limited understanding of computer programming. Little by little with the reward of seeing the resultant screen displays of my creations my confidence grew encouraging me on to ever more difficult tasks.

Presented here are two updated versions of my Coffee Cup Challenges, Noughts and Crosses and a variant of the Sliding Tiles game.

QBITS Tic Tac Toe

The S/SuperBASIC code presents a three-by-three Grid and by use of the Cursor keys a Cell is selected in which to draw a Nought (zero) or Cross. The trick is then defining the status of play, deciding if three of a kind has occurred either in a horizontal, vertical or diagonal three cells across the grid. Each game has three possible outcomes, a win by either the Noughts or Crosses or undecided, in which neither side can win.

QBITS Tic Tac Toe alternates with each Game the first player casting a Nought or a Cross. When a Players turn is indicated, they move the highlight with the Cursor keys to the grid position selected, hitting the spacebar will then draw the Nought or Cross.

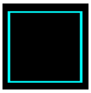
QBITS Tic Tac Toe Strategy

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 will give you the best chance of a win. However, if your opponent places their Nought or Cross in the centre, this will make it harder. The game is too short for any initiative for the second player to force a win they must rely on the first player making a minimum of two mistakes.

Games with two equally matched players invariably ends in a draw (undecide).

```
1000 REMark QBTTT_v2  (QBITS Tic Tac Toe v02 2021 Revision)

1002 MODE 8:gx=20:gy=30:Init_Scrn:chk=0:QBTTT                    :REMark BBQL gx=0 gy=0

1004 DEFine PROCedure Init_Scrn
1005 WINDOW#2,512,220,gx,gy    :PAPER#2,0:BORDER#2,1,3:CLS#2
1006 WINDOW#1,496,212,gx+8,gy+4:PAPER#1,0:BORDER#1,1,3::CLS#1
1007 WINDOW#0,512, 30,gx,gy+220:PAPER#0,0:CLS#0
1008 CSIZE#1,3,1:OVER#1,1:INK#1,6
1009 INK#1,2:FOR i=1 TO 3:CURSOR#1,12,10+i:PRINT#1,'QBITS'
1010 INK#1,6:FOR i=1 TO 2:CURSOR#1,12+i,10:PRINT#1,'QBITS'
1011 Str$='NOUGHTS':FOR i=1 TO 7:CURSOR#1,174,32+i*16:PRINT#1,Str$(i)
1012 Str$='CROSSES':FOR i=1 TO 7:CURSOR#1,464,32+i*16:PRINT#1,Str$(i)
1013 CSIZE#1,2,0:INK#1,2:FOR i=1 TO 2:CURSOR#1,16+i,33:PRINT#1,'TIC TAC TOE'
1014 INK#1,6:FOR i=1 TO 2:CURSOR#1,16+i,32:PRINT#1,'TIC TAC TOE'
1015 OVER#1,0:CURSOR#1,18,170:PRINT#1,'←↑     ↓→':BLOCK#1,30,6,70,174,6
1016 INK#1,1:CURSOR#1,18, 50:PRINT#1,"Played:":PG=0
1017 INK#1,6:CURSOR#1,18, 72:PRINT#1,"Wins O:":OG=0
1018 INK#1,3:CURSOR#1,18,130:PRINT#1,"Wins X:":XG=0
1019 INK#1,1:CURSOR#1,18,152:PRINT#1,"Failed:":FG=0
1020 RESTORE 1025:INK#1,248:SCALE#1,100,0,0:m=0:c=2:r=2
1021 FOR i=1 TO 4
1022   READ x1,y1,x2,y2,x3,y3,x4,y4
1023   FILL 1:LINE x1,y1 TO x2,y2 TO x3,y3 TO x4,y4 TO x1,y1:FILL 0
1024 END FOR i
1025 DATA 70,36,160,36,160,34,70,34, 70,66,160,66,160,64,70,64
1026 DATA 99,5,99,95,101,95,101,5, 129,5,129,95,131,95,131,5
1027 END DEFine

1029 DEFine PROCedure QBTTT
1030 REPeat loop
1031   x=45+c*30:y=r*30  :Tile x,y
1032   IF chk=0:CURSOR#1,12,100:PRINT#1,"NEW Game Y/N"
1033   IF chk=1:CURSOR#1,12,100:PRINT#1,"O's Turn... "
1034   IF chk=2:CURSOR#1,12,100:PRINT#1,"X's Turn... "
1035   k=CODE(INKEY$(-1)):Tile x,y
1036   SELect ON k                                                New Game Y/N
1037    =78,110:IF chk=0:EXIT loop                                :REMark (N) Exit Game
1038    =69,121:IF chk=0:Tile_CLS:chk=RND(1 TO 2):c=2:r=2         :REMark (Y) Play On
1039    =192:c=c-1:IF c<1:c=1                                     :REMark Left
1040    =200:c=c+1:IF c>3:c=3                                     :REMark Right
1041    =208:r=r+1:IF r>3:r=3                                     :REMark Up
1042    =216:r=r-1:IF r<1:r=1                                     :REMark Down
1043    = 32:RESTORE 1071:IF chk=1:Nought:ELSE IF chk=2:Cross     :REMark Mark (Tile)
1044    =232:SGame                                                :REMark Game Simulation
1045   END SELect
1046 END REPeat loop
1047 MODE 4:CLS#1:PRINT#0,'Bye...'
1048 END DEFine
```

```
1050 DEFine PROCedure Tile_Hgl(x,y)
1051 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
1052 END DEFine
```



```
1054 DEFine PROCedure Nought
1055 IF Grid(c,r)=0:Grid(c,r)=79:chk=2:c=2:r=2:x=x+10:y=y-10:ELSE c=2:r=2:RETurn
1056 INK 6:FILL 1:CIRCLE x,y,7:FILL 0:INK 0:FILL 1:CIRCLE x,y,4:FILL 0:Tile_Chk 79
1057 END DEFine
```

```
1059 DEFine PROCedure Cross
1060 IF Grid(c,r)=0:Grid(c,r)=88:chk=1:c=2:r=2:ELSE c=2:r=2:RETurn
1061 FILL 1:INK 3:x=x+10:y=y-10:LINE x-8,y+6 TO x-3,y+6 TO x+2,y TO x-3,y-6
1062 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
1063 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:Tile_Chk 88
1064 END DEFine
```

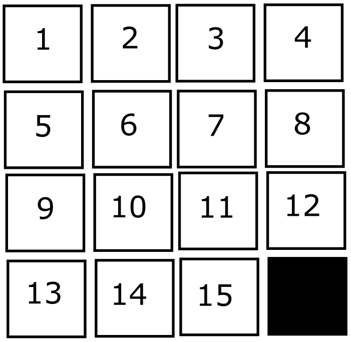
This next PROCedure checks for a possible win, the different combinations of three cells in a line.

```
1066 DEFine PROCedure Tile_Chk(n)
1067 FOR i=1 TO 8
1068   READ a,b,c,r,d,e:IF Grid(a,b)=n AND Grid(c,r)=n AND Grid(d,e)=n:Win:RETurn
1069 END FOR i
1070 m=m+1:IF m=9:m=0:n=0:Win
1071 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
1072 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
1073 END DEFine

1075 DEFine PROCedure Win
1076 INK 5  :PG=PG+1:INK#1,1:CURSOR#1,100 ,50:PRINT#1,FILL$(' ',3-LEN(PG))&PG:chk=0
1077 IF n=79:OG=OG+1:INK#1,6:CURSOR#1,100, 72:PRINT#1,FILL$(' ',3-LEN(OG))&OG
1078 IF n=88:XG=XG+1:INK#1,3:CURSOR#1,100,130:PRINT#1,FILL$(' ',3-LEN(XG))&XG
1079 IF n=0 :FG=FG+1:INK#1,1:CURSOR#1,100,152:PRINT#1,FILL$(' ',3-LEN(FG))&FG
1080 END DEFine

1082 DEFine PROCedure Tile_CLS
1083 DIM Grid(3,3):m=0:c=2:r=2:RESTORE 1087:INK 0
1084 FOR i=1 TO 9
1085   READ x,y,x1,y1:FILL 1:LINE x,y TO x1,y TO x1,y1 TO x,y1 TO x,y:FILL 0
1086 END FOR i
1087 DATA 75,30,95,10,105,30,125,10,135,30,155,10,75,60,95,40,105,60,125,40
1088 DATA 135,60,155,40,75,90,95,70,105,90,125,70,135,90,155,70
1089 END DEFine
```

Game Simulation

This awaits some future coding.

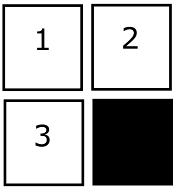
Sliding Tiles Puzzle



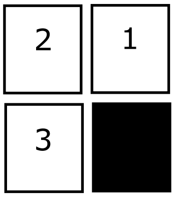
The puzzle represented on the right was introduced by Sam Loyd in 1880. It consists of a square grid with 15 tiles and one blank space. You rearrange the tiles by sliding a tile into the blank square. The most common goal of the game is to start with the tiles in some random order and then rearrange them into the sequence shown.

There are many variations of the game. One challenge Loyd gave was to take the original arrangement of the squares, interchange two of them, and then rearrange the squares to recover the original arrangement. This game can be played with any size grid, not just a 4 by 4 grid as in the original puzzle.

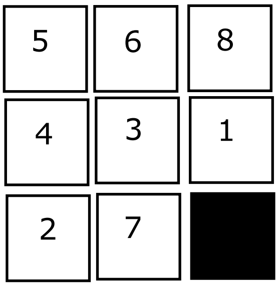
The main mathematical idea in solving Loyd's challenge is recursion, performing a task by repeatedly carrying out a basic procedure. The key to solving the puzzle is by turning the problem into a smaller puzzle.



Starting with a 2 by 2 Tile grid in the standard arrangement. Firstly, can you identify all the possible patterns of the three numbers and the moves to obtain the arrangement of Loyd's challenge, which is shown on the right?



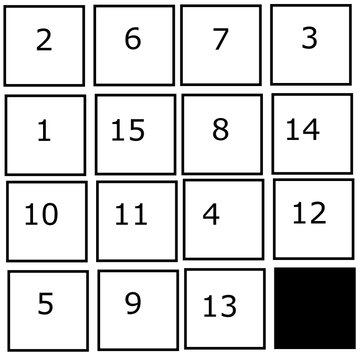
Solving the 3 by 3 Game with the following arrangement.



Start by arranging the top row in correct order. Next, correctly arrange the column on the left side. Now you are left with resulting arrangement of a 2 by 2 game. The most difficult thing is to correctly arrange the final two squares of a row or column. One thing to try is to make moves to get these into place more or less simultaneously.

To **S**olve a 4 by 4 Game with the following arrangement.

Use a similar strategy as before arrange the top row, then arrange the left column. Reducing the puzzle to a 3 by 3 to solve. Continue as before reducing the puzzle to a 2 by 2 as in the previous problem to finish the puzzle.

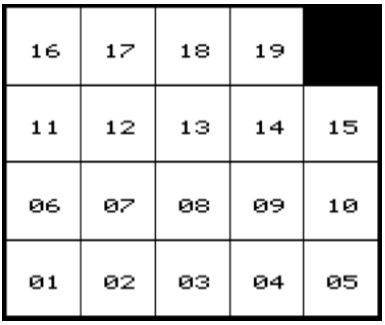


2	6	7	3
1	15	8	14
10	11	4	12
5	9	13	

QBITS Sliding Tiles
The first step was to identify each Tile with a number, because I decided to use S/SuperBASIC Graphic coordinates, you will perhaps understand why my sequence begins bottom left, progressing to top right. Next was to use a Random Shuffle routine so the order of Tiles are displaced. Then to replace the Tile numbers with Vector Graphics that when correctly sequenced create a picture.

In my **QBITS Tile Game** when all the Tiles are correctly placed the **Blank Tile** is top right or position twenty. 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.

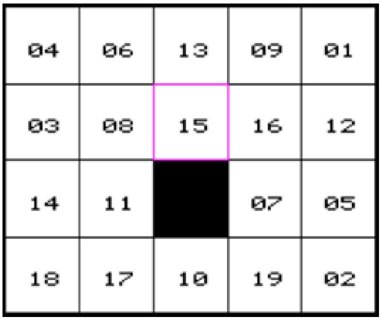
QBITS Tile Grid



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

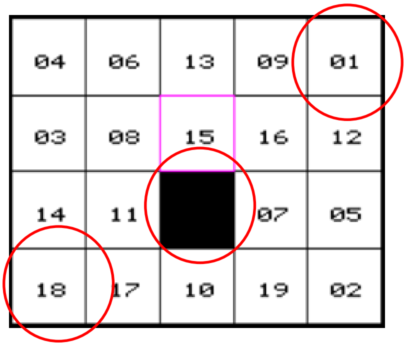
Just to add a bit more of a challenge the **QBITS Tile Grid** is a Five by Four. The Tiles are set with one to five on the lowest row and ending on row four with the Blank Tile number twenty at the top most right.

For each Game the Tiles are randomly shuffled to set a new arrangement.



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

QBITS Tile Strategy
The basics are one identify the first Tile, two the Tile's correct location and three the Blank Tile. Then rotate the Tiles moving the Blank in a clockwise or anti clockwise direction until the selected Tile is in its correct position. This then is applied to the next Tile and so on. Following previous strategies, aim to complete the left most column first and reduce the puzzle to a 4 by 4. Then continue until you have reduced the puzzle to the final 2 by 2 Grid in the top right-hand corner.



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

Example: Move the Blank to current position of Tile (09) or (12) then rotate the Tiles around the outer columns and rows until (01) occupies its correct position.

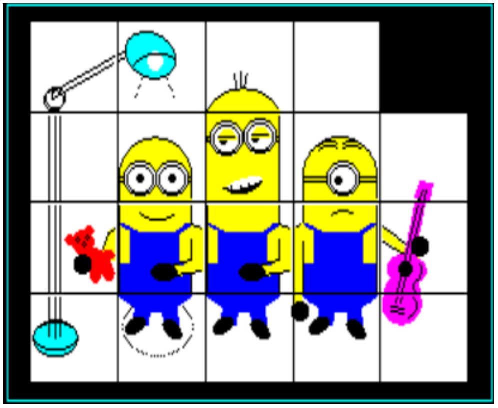
As (01) is in the diagonally opposite corner an alternative move would be to zig-zag diagonally across the board.

Either way continue until the first column (01), (06), (11), (16) is complete. The Board is now a 4 by 4 Grid. Complete the next left most column and the bottom (first) row. What is left is a 3 by 3 Grid. Now reduce again to be left with a 2 by 2 Grid.

QBITS Tiles Picture
Sliding Tiles puzzles need not be just numbered, they are quite often depicted as pictures. First thoughts was using Bitmap designs, but then I warmed to the idea of using vector Graphics and making it more of a challenge. These had to be based on familiar object or characters.

The Graphics were made up of repeatable designs and assembled so each Tile was different and its position could be correctly located.

However, an alternative solution presented itself by including the ability to swap from Picture mode to Tile Number mode and back. This was achieved through use of the <**Tab**> key and an IF check Statement.



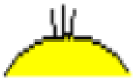
```
1000 REMark QBTiles_v2 (QBITS Tiles 2021 v2)

1002 gx=20:gy=30                    :REMark BBQL gx=0 gy=0
1003 DIM Tile(20,3)                 :REMark (tn, 1-20 ,tp) 1-20 ,tx) ,ty)

1005 MODE 8:Init_Scrn:QBITS_Tiles

1007 DEFine PROCedure Init_Scrn
1008 WINDOW#2,512,256,gx,gy:PAPER#2,0:CLS#2:SCALE#2,100,0,0
1009 WINDOW#2,496,208,gx+9,gy+0:BORDER#2,1,3:QB_Title
1010 INK#2,3:Tile_Hgl 2,113,90,60,6:INK#2,5:Tile_Hgl 2,109,86,62,8:INK#2,7
1011 WINDOW#1,280,167,gx+196,gy+16:SCALE#1,164,0,0:CSIZE#1,2,0
1012 WINDOW#0,496, 40,gx+9,gy+210 :BORDER#0,1,3:PAPER#0,0:INK#0,7
1013 CURSOR#2,66,36:PRINT#2,'(N)ew'
1014 CURSOR#2,40,48:PRINT#2,'🡅'   :CURSOR#2,40,78:PRINT#2,'🡇'
1015 CURSOR#2,22,62:PRINT#2,'🡄   🡆  ' :BLOCK#2,14,3,40,66,7
1016 INK#2,3:Tile_Hgl 2,8,8,13,63      :CURSOR#2,76,56:PRINT#2,'Moves'
1017 CURSOR#2,66,78:PRINT#2,'(Tab)':CURSOR#2,16,112:PRINT#2,'Quit(Esc)'
1018 END DEFine

1020 DEFine PROCedure QB_Title
1021 CSIZE#2,2,1:OVER#2,1
1022 INK#2,2:FOR i=0 TO 1:CURSOR#2,10+i,10:PRINT#2,'QBITS Tiles'
1023 INK#2,6:FOR i=0 TO 1:CURSOR#2,12+i,8:PRINT#2,'QBITS Tiles'
1024 CSIZE#2,2,0:OVER#2,0
1025 END DEFine

1027 DEFine PROCedure QBITS_Tiles
1028 t=0:Tsel=0:Init_Tiles:Set_Tiles:PAUSE:Sort_Tiles:Set_Tiles:tc=2:tr=2
1029 REPeat G_lp
1030  INK#1,3:Tile_Hgl 1,40,40,tc*40,tr*40
1031  k=CODE(INKEY$(-1))
1032  INK#1,0:Tile_Hgl 1,40,40,tc*40,tr*40
1033  SELect ON k
1034   =  9:IF Tsel=0:Tsel=1:Set_Tiles:ELSE Tsel=0:Set_Tiles
1035   = 27:MODE 4:CSIZE#2,0,0:INK#2,7:EXIT G_lp
1036   = 32:Tile_Chg:IF chk=1:Tile_Draw 1,ts:Tile_Draw 1,tn
1037   = 78,110 :t=0:Init_Tiles:Set_Tiles:PAUSE:Sort_Tiles:Set_Tiles:tc=2:tr=2
1038   =192:tc=tc -1:IF tc<0:tc=0
1039   =200:tc=tc+1:IF tc>4:tc=4
1040   =208:tr=tr +1:IF tr>3:tr=3
1041   =216:tr=tr  -1:IF tr<0:tr=0
1042  END SELect
1043 END REPeat G_lp
1044 END DEFine
```

```
1046 DEFine PROCedure Tile_Hgl(ch,w,d,x,y)
1047 LINE#ch,x,y TO x+w,y TO x+w,y+d TO x,y+d TO x,y
1048 END DEFine

1050 DEFine PROCedure Tile_Chg
1051 tn=1+tc+tr*5:chk=0        :IF Tile(tn,1)=20  :RETurn
1052 IF tc>0:ts=tn-1            :Tile_Chk ts:IF chk=1:RETurn
1053 IF tc<4:ts=tn+1            :Tile_Chk ts:IF chk=1:RETurn
1054 IF tr>0:ts=1+tc+(tr-1)*5 :Tile_Chk ts:IF chk=1:RETurn
1055 IF tr<3:ts=1+tc+(tr+1)*5:Tile_Chk ts:IF chk=1:RETurn
1056 END DEFine

1058 DEFine PROCedure Tile_Chk(ts)
1059 IF Tile(ts,1)=20:Tile(ts,1)=Tile(tn,1):Tile(tn,1)=20:chk=1
1060 END DEFine

1062 DEFine PROCedure Tile_Draw(ch,ts)
1063 IF Tile(ts,1)<20:STRIP#ch,7:INK#ch,7:ELSE STRIP#ch,0:INK#ch,0
1064 t$=Tile(ts,1):x=Tile(ts,2):y=Tile(ts,3):FILL#ch,1
1065 LINE#ch,x,y TO x+40,y+1 TO x+40,y+40 TO x,y+40 TO x,y:FILL#ch,0
1066 INK#ch,0:Tile_Hgl 1,40,40,x,y:IF Tsel=1:Tile_Pic ts,x,y
1067 IF Tsel=0:CURSOR#ch,x,y,16,-22:PRINT#ch,FILL$('0',2-LEN(t$))&t$
1068 END DEFine

1070 DEFine PROCedure Init_Tiles
1071 FOR r=0 TO 3
1072   FOR c=0 TO 4:t=t+1:Tile(t,1)=t:Tile(t,2)=c*40:Tile(t,3)=r*40
1073 END FOR r
1074 END DEFine

1076 DEFine PROCedure Set_Tiles
1077 ch=1:PAPER#ch,0:CLS#ch:FOR t=1 TO 20:Tile_Draw 1,t
1078 END DEFine

1080 DEFine PROCedure Sort_Tiles
1081 FOR t=20 TO 3 STEP -1
1082   ran=RND(1 TO t-1):temp=Tile(t,1):Tile(t,1)=Tile(ran,1):Tile(ran,1)=temp
1083 END FOR t
1084 END DEFine
```

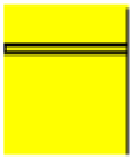
QBITS Minions Graphics

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.

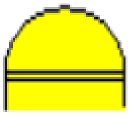
```
1086 DEFine PROCedure Tile_Pic(ts,x,y)
1087 ch=1:tp=Tile(ts,1)
1088 SELect ON tp
1089  =  1:ML5 x,y
1090  =  2:ML1 x,y:MB2 x,y
1091  =  3:MB2 x,y
1092  =  4:MB2 x+4,y:MA3 x,y
1093  =  5:MG3 x,y
1094  =  6:MT x+10,y-6:ML4 x,y
1095  =  7:MB1 x,y:MS1 x,y:MA2 x,y
1096  =  8:MB1 x,y:MA2 x,y
1097  =  9:MB1 x+4,y:MS2 x,y:MA1 x,y
1098  =10:MG2 x,y
1099  =11:ML4 x,y
1100  =12:MH3 x,y:ME1 10,10,x,y:ME2 10,10,x,y:ME1 25,10,x,y:ME2 25,10,x,y
1101  =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
1102  =14:MH3 x+4,y:ME1 19,10,x+4,y:ME2 17,10,x+4,y-1:MH4 x+4,y+1
1103  =15:MG1 x,y
1104  =16:ML3 x,y
1105  =17:ML2 x,y
1106  =18:MH1 x,y
1107 END SELect
1108 END DEFine

1110 DEFine PROCedure MH1(x,y)             :REMark Head Top
1111 INK#ch,6:FILL#ch,1:LINE#ch,x+1,y+1 TO x+34,y+1
1112 ARC#ch TO x+1,y+1,PI/1.5:FILL#ch,0:INK#ch,0
1113 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
1114 LINE#ch,x+14,y+10 TO x+13,y+17:LINE#ch,x+18,y+10 TO x+19,y+17
1115 END DEFine
```



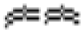
```
1117 DEFine PROCedure MH2(x,y)             :REMark Head Long
1118 INK#ch,6:FILL#ch,1:LINE#ch,x+1,y+1 TO x+34,y+1 TO x+34,y+39
1119 LINE#ch TO x+34,y+39 TO x+1,y+39 TO x+1,y+1:FILL#ch,0
1120 INK#ch,0:LINE#ch,x+34,y+1 TO x+34,y+39
1121 LINE#ch,x+1,y+27 TO x+34,y+27 TO x+34,y+29 TO x+1,y+29 TO x+1,y+27
1122 FILL#ch,0:REMark MS3 x,y
1123 END DEFine
```



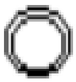
```
1125 DEFine PROCedure MH3(x,y)             :REMark Head Average
1126 INK#ch,6:FILL#ch,1:ARC#ch,x+1,y+12 TO x+34,y+12,-PI
1127 LINE#ch TO x+34,y+1 TO x+1,y+1 TO x+1,y+12:FILL#ch,0:INK#ch,0
1128 LINE#ch,x+1,y+1 TO x+1,y+12:ARC#ch TO x+34,y+12,-PI:LINE#ch TO x+34,y+1
1129 LINE#ch,x+1,y+11 TO x+34,y+11 TO x+34,y+9 TO x+1,y+9 TO x+1,y+11
1130 END DEFine
```



```
1132 DEFine PROCedure MH4(x,y)                              :REMark Hair
1133 INK#ch,0
1134 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
1135 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
1136 END DEFine
```



```
1138 DEFine PROCedure ME1(w,d,x,y)                          :REMark Eye Cover
1139 INK#ch,7:FILL#ch,1:CIRCLE#ch,x+w,y+d,7:FILL#ch,0
1140 INK#ch,0:CIRCLE#ch,x+w,y+d,7.4:CIRCLE#ch,x+w,y+d,6
1141 END DEFine
```



```
1143 DEFine PROCedure ME2(w,d,x,y)                          :REMark Eye
1144 FILL#ch,1:CIRCLE#ch,x+w,y+d,1.6:FILL#ch,0
1145 END DEFine
```

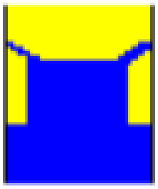
```
1147 DEFine PROCedure ME3(w,d,x,y)                          :REMark Eye Lid
1148 INK#ch,0:ME2 w,d-1,x-2,y:INK#ch,6:FILL#ch,1
1149 ARC#ch,x+w-4,y+d TO x+w+4,y+d,-PI:LINE#ch TO x+w-4,y+d:FILL#ch,0
1150 INK#ch,0:LINE#ch,x+w-4,y+d+1 TO x+w+4,y+d+1
1151 END DEFine
```

```
1153 DEFine PROCedure MS1(x,y)                    :REMark Smile
1154 INK#ch,0:ARC#ch,x+10,y+35 TO x+26,y+35,PI/2
1155 END DEFine
```

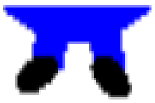
```
1157 DEFine PROCedure MS2(x,y)                    :REMark Frown
1158 INK#ch,0:ARC#ch,x+18,y+34 TO x+28,y+34,-PI/2
1159 END DEFine
```

```
1161 DEFine PROCedure MS3(x,y)                    :REMark Teeth
1162 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
1163 INK#ch,7:FILL#ch,1:CIRCLE#ch,x+18,y+8,8,.2,-PI/2.4:FILL#ch,0
1164 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
1165 LINE#ch,x+12,y+7 TO x+12,y+4
1166 END DEFine
```

```
1168 DEFine PROCedure MB1(x,y)                    :REMark Body Trunk
1169 FILL#ch,1:INK#ch,6
1170 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
1171 FILL#ch,0:FILL#ch,1:INK#ch,1
1172 LINE#ch,x+1,y+1 TO x+34,y+1 TO x+34,y+12 TO x+1,y+12 TO x+1,y+1
1173 FILL#ch,0:FILL#ch,1:INK#ch,1
1174 LINE#ch,x+6,y+12 TO x+28,y+12 TO x+28,y+26 TO x+6,y+26 TO x+6,y+12
1175 FILL#ch,0:FILL#ch,1:INK#ch,1
1176 LINE#ch,x+1,y+30 TO x+2,y+30 TO x+10,y+26 TO x+8,y+26 TO x+1,y+30
1177 FILL#ch,0:FILL#ch,1:INK#ch,1
1178 LINE#ch,x+31,y+30 TO x+34,y+30 TO x+28,y+26 TO x+26,y+26 TO x+32,y+30
1179 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
1180 END DEFine
```

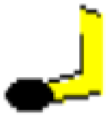


```
1182 DEFine PROCedure MB2(x,y)                       :REMark Body Feet
1183 INK#ch,1:FILL#ch,1:LINE#ch,x+1,y+39 TO x+34,y+39:ARC#ch TO x+1,y+39,-PI/2
1184 FILL#ch,0:FILL#ch,1
1185 LINE#ch,x+8,y+34 TO x+15,y+33 TO x+14,y+26 TO x+7,y+26 TO x+8,y+34
1186 FILL#ch,0:FILL#ch,1
1187 LINE#ch,x+27,y+34 TO x+20,y+33 TO x+22,y+26 TO x+28,y+26 TO x+27,y+34
1188 FILL#ch,0:FILL#ch,1:INK#ch,0:CIRCLE#ch,x+8,y+25,5,.6,-PI/3:FILL#ch,0
1189 FILL#ch,0:FILL#ch,1:INK#ch,0:CIRCLE#ch,x+25,y+24,5,.6,PI/4:FILL#ch,0
1190 END DEFine
```



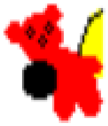
```
1192 DEFine PROCedure MA1(x,y)                       :REMark Arm Straight
1193 INK#ch,6:FILL#ch,1:LINE#ch,x+1,y+28 TO x+1,y+1 TO x+5,y+1 TO x+5,y+24
1194 LINE#ch TO x+5,y+24 TO x+5,y+30 TO x+1,y+28:FILL#ch,0
1195 INK#ch,0:LINE#ch,x+3,y+30 TO x+1,y+28:LINE#ch,x+5,y+22 TO x+5,y
1196 END DEFine
```

```
1198 DEFine PROCedure MA2(x,y)                       :REMark Arm Left
1199 INK#ch,6:FILL#ch,1
1200 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
1201 LINE#ch TO x+34,y+12 TO x+34,y+30:FILL#ch,0:FILL#ch,1:INK#ch,0
1202 CIRCLE#ch,x+21,y+9,6,.6,PI/2:FILL#ch,0:LINE#ch,x+34,y+30 TO x+39,y+28
1203 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
1204 END DEFine
```



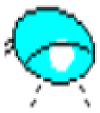
```
1206 DEFine PROCedure MA3(x,y)                       :REMark Arm Right
1207 INK#ch,6:FILL#ch,1:LINE#ch,x+1,y+39 TO x+4,y+39 TO x+4,y+36
1208 LINE#ch TO x+1,y+36 TO x+1,y+39:FILL#ch,0
1209 INK#ch,0:FILL#ch,1:CIRCLE#ch,x+3,y+32,4:FILL#ch,0
1210 END DEFine
```

```
1212 DEFine PROCedure MT(x,y)                        :REMark Teddy Bear
1213 INK#ch,6:FILL#ch,1
1214 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
1215 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
1216 FILL#ch,1:CIRCLE#ch,x+14,y+28,5:FILL#ch,0               :REMark head
1217 FILL#ch,1:CIRCLE#ch,x+ 8,y+29,2:FILL#ch,0               :REMark ear 1
1218 FILL#ch,1:CIRCLE#ch,x+17,y+33,2:FILL#ch,0               :REMark ear 2
1219 FILL#ch,1:CIRCLE#ch,x+20,y+20,8,.6,PI/3:FILL#ch,0       :REMark body
1220 FILL#ch,1:CIRCLE#ch,x+26,y+22,3,.6,PI/2:FILL#ch,0       :REMark arm 1
1221 FILL#ch,1:CIRCLE#ch,x+20,y+14,4,.6,PI:FILL#ch,0         :REMark leg 1
1222 FILL#ch,1:CIRCLE#ch,x+26,y+14,3,.8,PI:FILL#ch,0:INK#ch,0 :REMark hand
1223 FILL#ch,1:CIRCLE#ch,x+14,y+18,4:FILL#ch,0               :REMark leg 2
1224 CIRCLE#ch,x+12,y+29,1:CIRCLE#ch,x+16,y+30,1:CIRCLE#ch,x+15,y+27,1     eyes  nose
1225 END DEFine
```



```
1227 DEFine PROCedure ML1(x,y)                       :REMark Lamp Highlight
1228 INK#ch,248:CIRCLE#ch,x+18,y+25,16,.8,PI/2
1229 END DEFine
```

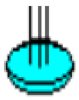
```
1231 DEFine PROCedure ML2(x,y)                :REMark Lamp Head
1232 INK#ch,0:LINE#ch,x+1,y+25 TO x+10,y+30 TO x+10,y+27 TO x,y+22
1233 INK#ch,5:FILL#ch,1:CIRCLE#ch,x+15,y+25,12,.8,PI/3:FILL#ch,0
1234 INK#ch,7:FILL#ch,1:CIRCLE#ch,x+17,y+22,4,.8,PI:FILL#ch,0:INK#ch,0
1235 CIRCLE#ch,x+15,y+25,12,.8,PI/3:ARC#ch,x+7,y+19 TO x+26,y+23,-PI/2
1236 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
1237 END DEFine
```



```
1239 DEFine PROCedure ML3(x,y)                :REMark Lamp Arm
1240 INK#ch,0:LINE#ch,x+39,y+24 TO x+10,y+8 TO x+10,y+5 TO x+39,y+21
1241 CIRCLE#ch,x+11,y+5,4.5:CIRCLE#ch,x+12,y+6,4.5
1242 END DEFine
```

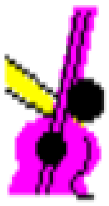
```
1244 DEFine PROCedure ML4(x,y)                :REMark Lamp Stand
1245 INK#ch,0:LINE#ch,x+9,y+1 TO x+9,y+39
1246 LINE#ch,x+12,y+1 TO x+12,y+39:LINE#ch,x+14,y+1 TO x+14,y+39
1247 END DEFine
```

```
1249 DEFine PROCedure ML5(x,y)                :REMark Lamp Base
1250 INK#ch,5:FILL#ch,1:CIRCLE#ch,x+12,y+20,10,.8,PI/2:FILL#ch,0
1251 INK#ch,0:CIRCLE#ch,x+12,y+20,10,.8,PI/2:CIRCLE#ch,x+12,y+21,10,.6,PI/2
1252 LINE#ch,x+9,y+21 TO x+9,y+39:LINE#ch,x+12,y+20 TO x+12,y+39
1253 LINE#ch,x+14,y+21 TO x+14,y+39
1254 END DEFine
```

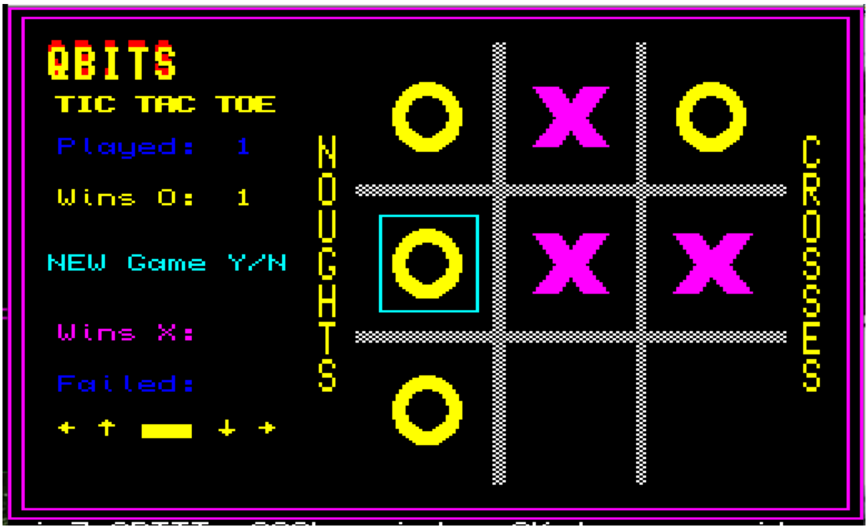


```
1256 DEFine PROCedure MG1(x,y)                :REMark Guitar Top
1257 FILL#ch,1:INK#ch,3:LINE#ch,x+17,y+1 TO x+16,y+2 TO x+18,y+9
1258 LINE#ch TO x+24,y+8 TO x+22,y+1 TO x+17,y+1:FILL#ch,0:INK#ch,0
1259 LINE#ch TO x+18,y+1 TO x+19,y+6:LINE#ch,x+20,y+1 TO x+21,y+6
1260 END DEFine
```

```
1262 DEFine PROCedure MG2(x,y)                :REMark Guitar Middle
1263 FILL#ch,1:INK#ch,6:LINE#ch,x+1,y+28 TO x+1,y+24 TO x+15,y+14
1264 LINE#ch,x+18,y+14 TO x+1,y+28:FILL#ch,0:INK#ch,0
1265 LINE#ch,x+1,y+22 TO x+15,y+14:LINE#ch,x+1,y+29 TO x+16,y+18
1266 FILL#ch,1:INK#ch,3:LINE#ch, x+10,y+10 TO x+16,y+39 TO x+20,y+39
1267 LINE#ch TO x+14,y+10 TO x+10,y+10:FILL#ch,0
1268 INK#ch,3:FILL#ch,1:CIRCLE#ch,x+12,y+10,9,.7,PI/2.2:FILL#ch,0
1269 FILL#ch,1:ARC#ch,x+18,y TO x+2,y,PI:LINE#ch TO x+2,y:FILL#ch,0
1270 INK#ch,0:FILL#ch,1:CIRCLE#ch,x+11,y+10,3:FILL#ch,0
1271 ARC#ch,x+18,y+12 TO x+16,y+4,-PI/2 TO x+20,y+1,PI
1272 LINE#ch,x+8,y+1 TO x+17,y+39:LINE#ch,x+10,y+1 TO x+19,y+39
1273 FILL#ch,1:CIRCLE#ch,x+18,y+20,4:FILL#ch,0
1274 END DEFine
```



```
1276 DEFine PROCedure MG3(x,y)                :REMark Guitar Bottom
1277 INK#ch,3:FILL#ch,1:LINE#ch,x+2,y+39 TO x+18,y+39
1278 ARC#ch TO x+1,y+32,-PI TO x+2,y+39,-PI/4:FILL#ch,0
1279 INK#ch,0:ARC#ch,x+15,y+38 TO x+12,y+29,-PI/1.5
1280 LINE#ch,x+7,y+36 TO x+8,y+39:LINE#ch,x+9,y+36 TO x+10,y+39
1281 LINE#ch,x+5,y+32 TO x+11,y+31:LINE#ch,x+5,y+34 TO x+11,y+33
1282 END DEFine
```



QBITS

TIC TAC TOE

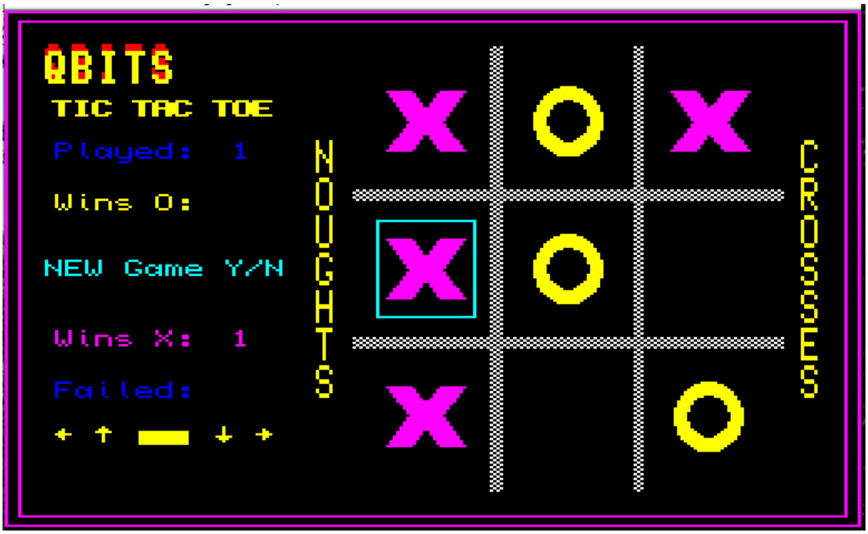
Played: 1

Wins O: 1

NEW Game Y/N

Wins X:

Failed:



QBITS

TIC TAC TOE

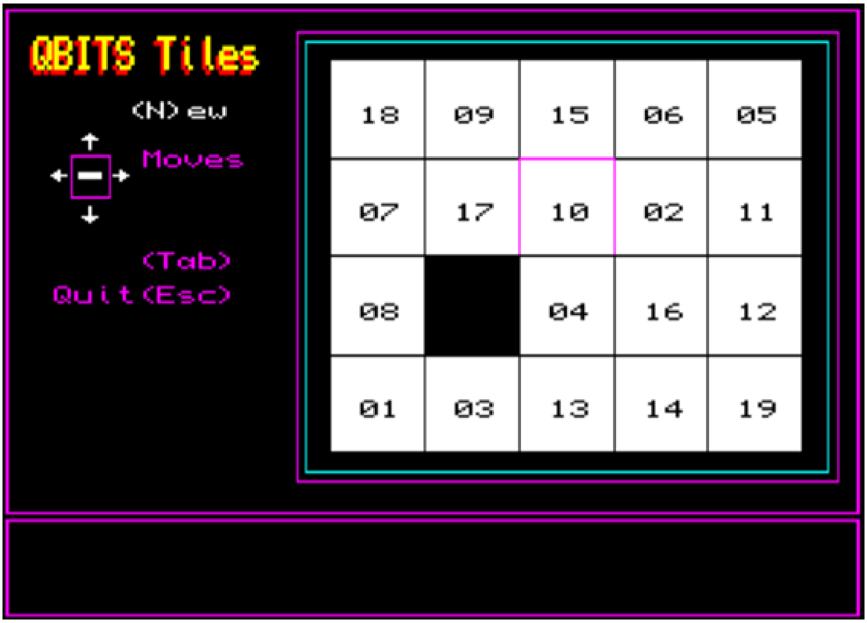
Played: 1

Wins O:

NEW Game Y/N

Wins X: 1

Failed:



QBITS Tiles

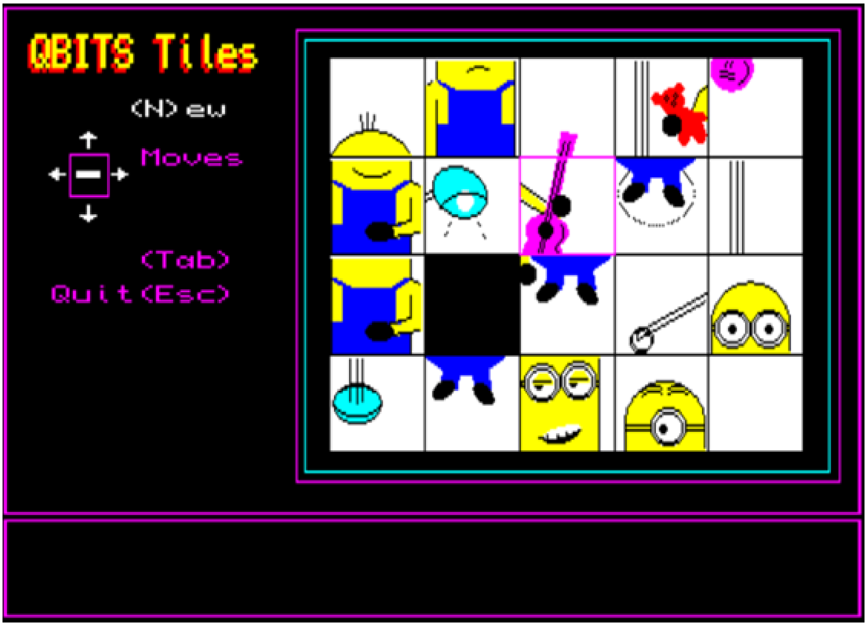
(N)ew

Moves

(Tab)

Quit(Esc)

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



QBITS Tiles

(N)ew

Moves

(Tab)

Quit(Esc)