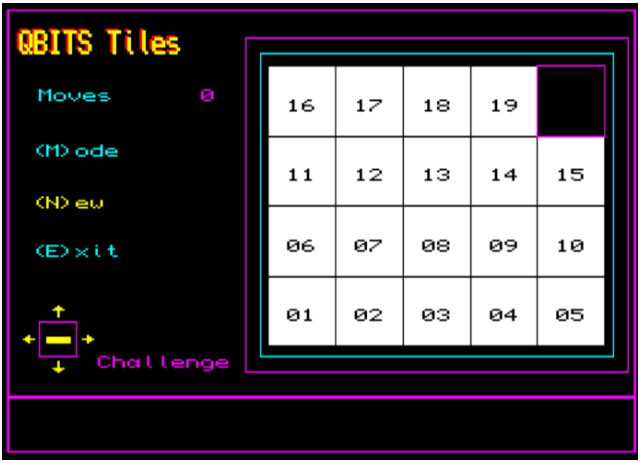


Sliding Tile Puzzle

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

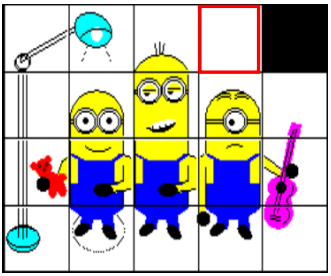
Blank

Tile String: 06 09 13 01 05 10 14 00 04 10 16 17 12 02 11 07 08 15 19 03
Inversions: +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 (**Q**) to Quit 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 $\mathbf{n}^2(\mathbf{n}-1)$. 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.

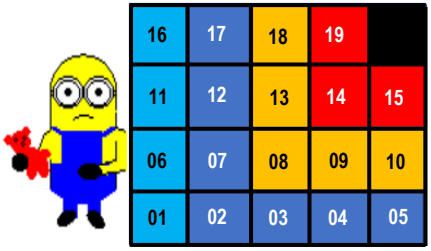
After reviewing a series of games, the average number of Inversion was around 200, it suggested an inexperienced player might take twice this number of moves to solve the Puzzle. Therefore, the smallest number of moves taken will be dependent on the number of Inversions and Skill of the player.

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.



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

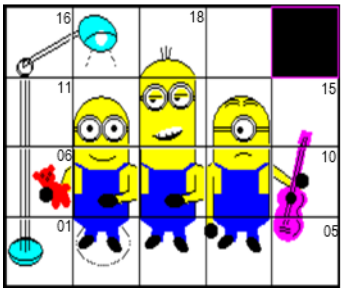
```
1000 REMark QBITS_Tiles_bas [QBITS Tiles 2023 Review - QPC2]

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     = 81,113:QQuit: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 QQuit
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
```



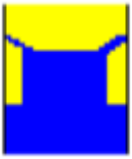
Note: Sliding Tiles presented as a Picture. The Vector Graphics are draw as a combination of individual parts.

.

```
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
```