

Introduction

The 1980's appeal and accessibility of Computer Games with home computers expanded rapidly as with their growing graphics capabilities. Character Fonts evolved into 2D creations named Sprites. These Images represented by a matrix of tiny dots, pixels displayed to screen were stored in memory as Bitmaps.

Creating Pixel Characters and Background Screens soon became a recognised Art form. Screens of the 1980's were typically 256 by 192 pixels, limiting detail and where the best Sprite Designers made recognisable characters within a 24x24 or less Pixel Grid.



QBITS PIXEL Art

The 1980's ambition was to write a Program in SuperBASIC to create simple RETRO Games for the Sinclair QL Platform. Unfortunately, early performances of Programs run with the QL Interpreter was slow and Sprite Bitmaps required lots of memory. Today with the extended range of QL Platforms and enhanced Software, Speed and available Memory this may not be the issue, and code compatibility becomes a primary concern.

QBITS Prog Concept

A Program that Builds in Stages the necessary elements for a Retro Game. For **Stage One** the SPRITE Designer began with a rewrite of code taken from earlier **QBITS Progs BITMap Designer** and **QLFont Editor**. Functions were extended to Import and Resize a QL 9x8 Font sets used in Retro Games and provide the ability to Select Edit areas within a Grid to ReColour.

Screen backgrounds use Sprites designed as Tiles. **Stage Two** copies these across from the Spite Designer to form a TILE Library, and deploying them to build sections of a screen background. Each Tile being set as a Wall, Floor, Hazard, Reward etc. and identifying actions. Then Link Map to manage multiple Background Screens.

Stage Three Control settings for SPRITE moved by a Player or under Program control. A GAME Title that can be edited. Display of Score & Lives counters and Settings for Hazard or Reward Gains and Losses. Include a limited Test to explore SPRITE actions.

Stage Four Play Checks on RETRO Game elements and SAVE as standalone code.

QBITS PIXELArt - Palette Choice

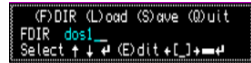
Developed with QPC2 environment with two Colour Modes. The COLOUR_QL displays 0-7 as **Black Red Magenta Green Cyan Yellow White** with 8...255 as a Colour, Contrast and Stipple combination. COLOUR_PAL uses 0...255 as colours and shades from a Palette based on 24Bit RGB values (sixteen million variations).



Select with Left [QL8 or PAL] Right Cursor and Enter.
Note: Default is QL8 [Reverting to Mode 4 for BBQL]

QBITS PIXArt - File Management

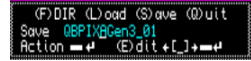
Press (F) to change File **DIR**ectory Select Drive and Enter.
Press (E) to Edit **Drive/SubDIR**ectory name. Enter to Action.



Press (L) for **Load**. Select Drive with Up/Down Cursors and Enter. Program checks for QBITS **_bmp _fnt** files. Select file with Up/Down Cursors. Abort with Spacebar Load with Enter.



Press (S) to **Save** Current **_bmp** File. Press (E) to Edit filename. SPRITE Mode has a basic header followed by SPRITE BITMaps. SCREEN Mode has an extended header with Screen MAPing information followed by TILE Library BITMaps. A constructed RETRO Game will Save as a standalone Code listing.

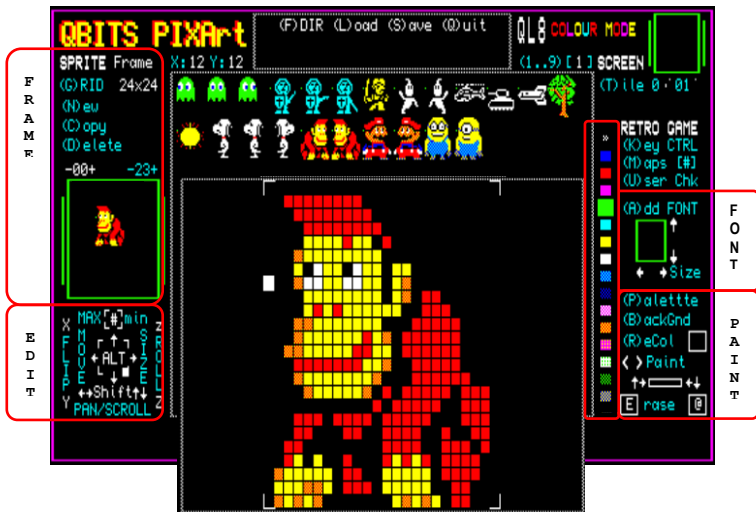


If file exists an Overwrite Y/N prompt is displayed.

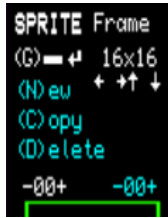


QBITS PIXELArt - SPRITE Mode

Loading a SPRITE_bmp file will set the Grid Size. If Frame number is set to **-00+** the SPRITES are displayed in Pixel Grid Size as a group across central screen. Load a QL FONT_fnt file set Size and Import to a PIXEL Art Frame.



QBITS PIXEL Art - (G)RID



Press 'G' for **SPRITE Mode** or to change **Frame** Gride use Cursors to size Columns and Rows. Abort **Spacebar** or Action **Enter**. Then you can use **-00+** to select a **SPRITE Frame**.

QBITS PIXEL Art - Frame

Press 'N' to add a **New Frame**.

Press 'C' to **Copy** present Frame to Another

Press 'D' to **Delete** current Frame.



QBITS PIXEL Art - Paint

Select **Paint** Colour with Left < > Right Chevron keys. Colour is shown as Enlarged Square that moves Up/Down the Palette Bar.

(P)aLETTE 0..7 **Black Red Magenta Green Cyan Yellow White** colours 8..15 hold default QL8 or PAL colours. Press 'P' to change a Palette Colour and use **↑ → NNN ← ↓** Cursor keys to change and display the Changed Palette colour [Abort with **Spacebar** or Action **Enter**].

(B)ackGnd sets the background colour for all frames. Press 'B' Select a Colour from Palette Bar. Abort with **Spacebar** or **Enter** to Action.

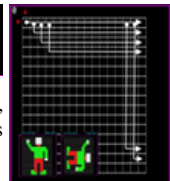
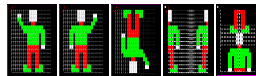


Select Grid position with **↑ → ← ↓** Press **Spacebar** to toggle **Paint ON/OFF** moving the cursor extends this into other columns and/or rows. Press 'E' and move over Painted Cells to restore them to Background colour. Press '@' to Fill Frame Edit area with chosen colour or use with [E]rase to Reset cells to Background colour. To use (R)eCol - Set a New Colour < > and Press 'R' then select the existing Frame colour you wish to change. Press **Enter** or **Spacebar** to Abort.

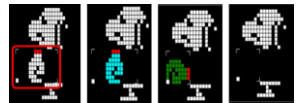


QBITS PIXEL Art - Edit

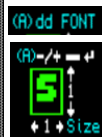
Press 'xX' to **FLIP** on Horizontal or 'yY' on Vertical axis, 'zZ' to **ROLL** 90° clockwise or anticlockwise. To move cells within Frame Grid use **Shift** Cursors to **PAN** or **SCROLL**.



Press '#' to MAX[#]min the Edit area of Frame Grid. The Edit Area can be Re-Positioned or Re-Sized (set Screen Cursor on a corner and drag) with **ALT** Cursors **FLIP**, **ROLL**, **PAN**, **SCROLL**, **FILL** [E] or (R)eColour within the Edit Area or all of Grid Frame.



Note: First Load a FONT_fnt File.



Press (A) Select **-/+** Font and use cursors **← → ↑ ↓** to resize. Press **Enter** to Add to Frame Sequence or Abort with **Spacebar**.

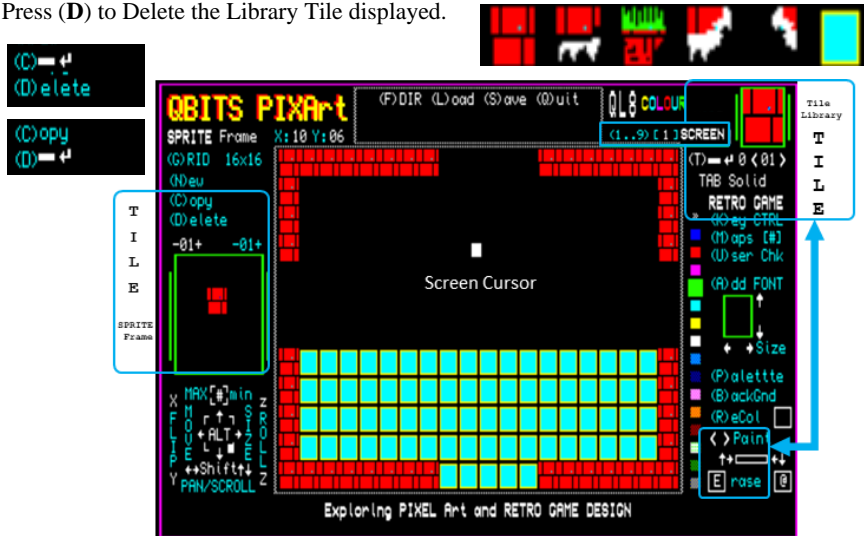
Use Edit Tools with All or ReSized Grid Area and Paint Options, ReColour etc together with Frame Copy to build colourfully designed **SPRITES**.

QBITS PIXEL Art - SCREEN Mode

Load a SCREEN/TILE_bmp file. Press 'T' for SCREEN Mode and set TILE Attribute that determines action when a moving Sprite comes into contact. The Tile Library used to build Background Screens.. [Use TAB Key Enter to Action or Spacebar to abort]

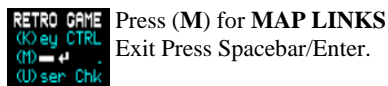
QBITS PIXELArt - (T)ILE

Adding a TILE. Select SPRITE Frame with -/+ then Press (C) to Copy to TILE Library. Object is shown in TILE Frame top right with number below < 01 > incremented. Press (D) to Delete the Library Tile displayed.

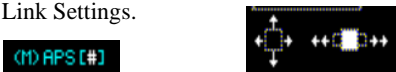


Select a Library Tile with < > Chevron keys and use ↑ ↓ ← → to Position screen Cursor. Toggle Spacebar **ON/OFF** to Paint individual Tile to SCREEN or extend as a row or column. Use [E]rase to Remove.

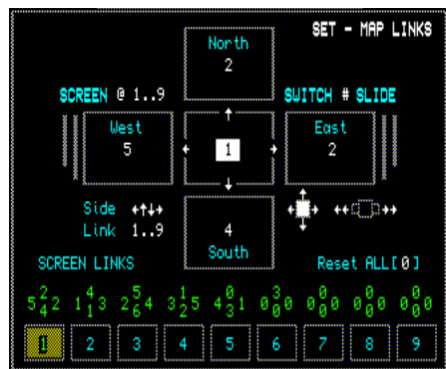
QBITS PIXEL Art - Maps

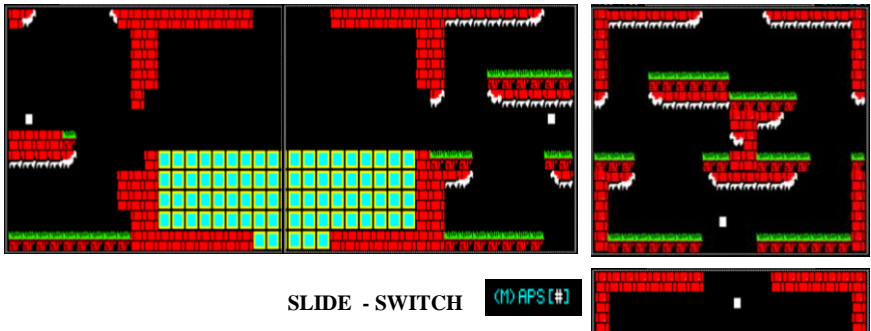


Press @ to Select a Primary Screen 1..9. Then use Cursors to Select a Compass direction and Set a Screen Link 1..9. The bottom screen MAP displays changed Link Settings.



Select SWITCH or SLIDE with # Back in SCREEN Mode Press [#] and move Cursor to an EXIT point and Test SCREEN Links.

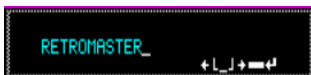




QBITS PIXELArt - Key CTRL



Press 'K' for SPRITE Setup 1&2 as Player and 3..6 Computer controlled. SPRITES 3..6 use as Hazard or Reward and set Score and/or Live Charges. A SPRITE is inactive if default Frame @ is set to [00].



Press 'N' to change Title Name, use ◀[_]▶ to position cursor. Add/Delete characters. [] Abort or ⏎ Action]



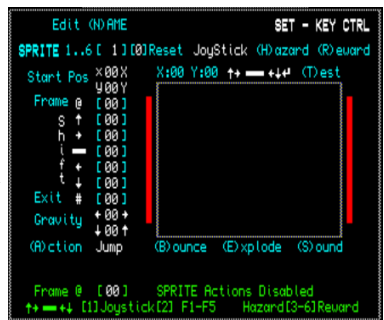
QBITS PIXEL Art - Action SPRITES



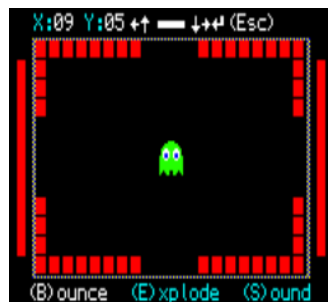
Note: First Load SPRITE_bmp File



Select 1..6 []
Screen Start Position x00X y00Y
Set Default Frame @ []
Direction SPRITES ↑→ [] ↓←
(A)ttack Spacebar []
ie. Explode Exit # []
Prog CTRL Gravity X00X Y00Y
(A)ction toggle between JUMP : FIRE



QBITS PIXEL Art - Test SPRITES



Press 'T' to Test movement and reaction. The screen is set with border Tiles and Exits on each side. Tiles act as solid objects, Test checks for Spacebar (A)ttack settings and Collision Responses dictated by (B)ounce and (E)xplore.

Program-run Sprites move according to Gravity settings. Player Controlled Sprites use cursors and can speed up or slow down dependant on key presses in any one direction.

Press 'S' On/Off for accompanying (S)ound settings.

QBITS PIXEL Art - Review of SPRITE Moves

These start with a default Image displayed to screen with further images for directional changes made by Cursor Keys or Spacebar actions. Speed of movement is defined by Horizontal and Vertical Gravity values xx yy . Reaction to other screen Objects static or moving are set with (A)ction (B)ounce (E)xplode. The Program Controlled Sprites need to be set as (H)azard or (R)eward. The accompanying Sound setting can be set On/Off.

QBITS PIXEL Art - Review of (A)ttack Setting

Actions for Player Sprites are governed by the four Cursor keys and Spacebar that mimic a Joysticks direction and fire button. The Spacebar [Fire button] is used to make a Jump or Fire Shells dependant on choice. Platform Games that Slide between Screens tend to disable Player controlled vertical $\uparrow\downarrow$ movement and activate Jump [Spacebar] as the means of Ascending Levels. Vertical Gravity only applies to computer controlled falls.

Jump action includes a stationary position and moving one. A Jump is calculated as Tile height th with Pixel coordinates set from top left of screen then $yy = -th$ can be used to calculate the new Jump screen position. To calculate for a moving Jump $yy = -ABS(xx)$ can be deployed. The Decent Speed or vertical Gravity is then set as $yy = +th/2$. Collision Detection checks are required for both Ascent and Decent to Landing Area.

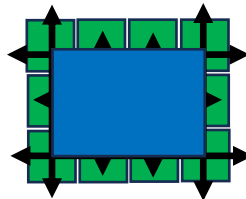
Fire action is determined by the last direction of movement. A Firing point is calculated from the Sprite Centre $xc = sx + cm/2 + (2 + cm/2) * cf$ & $yr = sy + rm/2 + (1 + rm/2) * rf$. Where cm & rm are the Sprite width and height and cf & rf provide positive or negative direction of sx sy vales. The 1 & 2 Constants are added to avoid any XOR pixel debris. The Target position is then calculated as $xc = xc + 4 * cf$ $yr = yr + 4 * rf$ and incremented within a FOR loop to mimic a fired projectile with results from any Collision Detection actioned.

QBITS PIXEL Art - Collision Detection

In two-dimensional Games Collision Detection occurs when a Sprite moves into areas occupied by other moving Sprites or a Background Tile. What then takes place can be absorbed or destroyed by the other or with one or both continuing with altered direction and speed. Dependant on the encountered Object(s) a Players Status is updated.

As Sprites and Screen Tiles might be of different sizes so any Collision Detection must resolve all possible encounters The Screen Tile size is used as a default so xx yy Speed is added to a Sprites x,y position. A FOR loop checks for any possible overlaps with other Object in range. The Y axis is checked first this helps the X axis moves appear smoother.

Check for YY Axis Collisions
Resolve & Set Response
Check for XX Axis Collision
Resolve & Set Response
Clear Old Sprite Position
Update Gravity xx yy
Draw Sprite Position with Response



WORK IN
PROGRESS

QBITS PIXEL Art - BITMaps

The 1980's Games utilised character fonts with one colour. **QBITS Font Edit** Special Edition has three Demo Games that explored this approach. **QBITS BITMap Design** explored multi colour Objects with various Grid sizes. The BITMaps saved contained Headers to describe file Type and info such as Grid Size of Bitmap Data that followed.

QL FONT _fnt Files

For the QL default Font Character set the First Byte holds the Start Character code and second Byte the number of Character Bitmaps to follow ie 32/96

PIXEL Art _bmp Files [type sz]

IF **sz=0:bm=frt**:mlth= 8+**bm*cm*rm** :addr=ALCHP(mlth):ptr=addr+8
IF **sz=1:bm=frt**:mlth=16+**bm*cm*rm** :addr=ALCHP(mlth):ptr=addr+16
IF **sz=2:bm=tm**:mlth=52+2160+**bm***256 :addr=ALCHP(mlth):cm=16:rm=16
IF
sz=3:bm=tm:mlth=52+2160+200+**bm***256+**bm**:addr=ALCHP(mlth):cm=16:rm=16

bm BITmaps Max : **cm** Columns Max : **rm** Rows Max: **bg** BackGnd Colour :**sz** Type

SPRITE _bmp	frt Frame total	Bytes
Bytes 0... 7	QL8 or PAL bm cm rm bg sz	8
Bytes 8...15	Changeable Palette Colours	16
	bm*cm*rm SPRITE Bitmaps ???	
TILE _bmp	tm Tile Max	
Bytes 0.....15	QL8 PAL bm cm rm bg sz + Palette	16
Bytes 16.....51	Screen Links 9x4 = 36	52
Bytes 52.....212	Screen Tiles 9x20x12 = 2160	2112
Bytes 212.....2412	Key CTRL 9x20 = 180 +	
	bm*cm*rm TILES bitmaps ??? + bm TILE Action	

QBITS PIXEL Art uses **sz=1** for **SPRITE** and **sz=3** for **SCREEN/Tile _bmp** Files

WORK IN
PROGRESS

QBITS PIXEL Art - RETRO Play

RGen Part One - The constructed Game is accessed with (U)ser Check for RETRO PLAY Mode to confirm if performing as expected. [Esc] Returns to PIXEL Art for further changes.

UNDER
CONSTRUCTION

RGen Part Two - If all is Satisfactory Press 'S' to (S)ave as standalone code, which can be Loaded and Run independently of QBITS PIXEL Art...

QBITS PIXEL Art Prog Code [S/SuperBASIC]

1000 REMark **QBITS_PIXELArt_RGen** [QBITS PIXEL Art Game Design 2024 - QPC2] vM30

1002 dev\$='dos1_' :MODE 4:gx=0:gy=0 :REMark Basic Settings

1004 **WHEN ERror** :eck=1:CONTINUE:**END WHEN** :COLOUR_QL

1006 REMark **Import QBITSConfig Settings - QPC2**

1007 OPEN _IN#9,dev\$&'QBITSConfig':INPUT#9,gx\gy\dn\$dev\$\dn%\dm%

1008 DIM drv\$(dm%,16):FOR d=0 TO dm%:INPUT#9,drv\$(d):END FOR d:CLOSE#9

1010 REMark **PIXEL Array Settings**

1011 DIM FG(32,64,64),TG(64,64),File\$(50,20),CP(15),BEEP\$(4,34)

1012 DIM Tile(120,15,15),TAss(120),TScn(9,20,12),WScn(20,12),TMap(9,4),SK(9,20)

1013 DIM TAS\$(9,6),RGN\$(13),fnt\$(9,8):Fntaddr=ALCHP(1164):sg%=0:cg%=0:ch%=0

1014 PFile\$="":TFile\$="":SFile\$="":k=0:lks=0:RGN\$="RETROMASTER":csx=1:csy=1

1016 **COLOUR_QL:Init Screens:Init Palette:Init Layout:PIXArt_Menu**

1018 REMark **PIXArt SetUP**

1020 **DEFine PROCEDURE Init Screens**

1021 WINDOW#0,512,32,gx,gy+224 :PAPER#0, 0:BORDER#0,1,3 :CLS#0

1022 WINDOW#1,512,256,gx,gy :PAPER#1, 0:

1023 WINDOW#2,512,224,gx,gy :PAPER#2, 0:BORDER#2,1,3 :CLS#2

1024 OPEN#3,scr_ :WINDOW#3, 328,196,gx+ 92,gy+ 35 :BORDER#3,1,248:CLS#3

1025 OPEN#4,scr_ :WINDOW#4, 72, 68,gx+ 12,gy+ 96 :CLS#4

1026 OPEN#5,scr_ :WINDOW#5, 200, 32,gx+156,gy+ 2 :BORDER#5,1,248:CLS#5

1027 OPEN#6,scr_ :WINDOW#6, 90,196,gx+ 2,gy+ 36 :CLS#6

1028 OPEN#7,scr_ :WINDOW#7, 36, 34,gx+466,gy+ 2 :CLS#7

1029 OPEN#8,scr_ :WINDOW#8, 90,196,gx+420,gy+ 36 :CLS#8

1030 OPEN#10,scr_ :WINDOW#10, 80, 72,gx+ 8,gy+170

1031 OPEN#11,scr_ :WINDOW#11, 24, 24,gx+453,gy+119:CSIZE#11,3,1

1033 QBT\$='QBITS PIXArt':QTitle 1,2,4,4,QBT\$:QGT\$='TITLE'

1034 **END DEfine**

1036 **DEFine PROCEDURE Init Palette**

1037 CSIZE#5,0,1:CSIZE 0,0:pmb\$='COLOUR MODE':CP(7)=7

1038 CSIZE 1,1:pm\$='QL8':QBold 1,7,360,2,pm\$:**RESTORE 1051**

1039 IF VER\$(1)>=2.98

1040 COLOUR_QL

1041 INK#5,5:CUSOR#5,12,6:PRINT#5,'Select Palette ';

Note: Palette Settings

1042 INK#5,7:PRINT#5,'QL8 ◀ ▶ PAL ◀ ▶':BLOCK#5,2,6,181,12,7

1043 **REPeat P_lp**

1044 QBold 1,7,360,2,pm\$:k=CODE(INKEY\$(-1))

1045 IF k=192:pm\$='QL8':**RESTORE 1051**:tool%=255

1046 IF k=200:pm\$='PAL':**RESTORE 1052**:tool%=10

1047 IF k= 10:IF pm\$='PAL':COLOUR_PAL:END IF :CLS#5:**EXIT P_lp**

1048 **END REPeat P_lp**

1049 **END IF**

1050 CSIZE#5,0,0:CSIZE 0,0:FOR i=0 TO 15:**READ CP(i)**

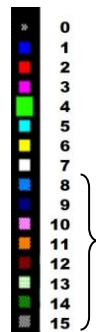
8..15 Default QL8 or PAL

1051 DATA 0,1,2,3,4,5,6,7,225,200,227,230,210,31,224,254

1052 DATA 0,4,2,5,3,7,6,1,25,50,31,22,46,36,61,10

1053 FOR i=1 TO 11:QBold 1,RND(2 TO 7),380+i*6,6,pmb\$(i)

1054 **END DEfine**:



1056 DEFINE PROCEDURE Init_Layout

```

1057 BLOCK 2,26,462,6,CP(4) :BLOCK 2,26,504, 6,CP(4) :BORDER#7,1,CP(4)
1058 BLOCK 2,50,8,106,CP(4) :BLOCK 2,50,86,106,CP(4) :BORDER#4,1,CP(4)
1059 BLOCK#8,14,12,21,182,CP(7) :BLOCK#8,14,12,71,182,CP(7) :CLS#3:CLS#4
1060 QBold 1,7, 6, 24,'SPRITE' :QBold 1,7,420,24,'SCREEN' :CLS#5:CLS#7
1061 QBold 8,7,22,44,'RETRO GAME':QBold 8,7,19,153,'<>':QBold 8,7,22,94,'[?]'
1062 QBold 8,7,58,80,'%':QBold 8,7,58,98,'¿':QBold 8,7,20,159,'<>'
1063 FOR i=0 TO 15:BLOCK#8,8,6,36+10*i,CP(i):END FOR i
1064 RESTORE 1062:FOR i=1 TO 19:QPrnt:END FOR i
1065 DATA 6,5,2,2,'(G)RID',1,7,48,25,'Frame',1,5,92,25,'X: Y:'
1066 DATA 8,5,2,2,'(T)ile',1,5,360,25,'(1..9)[ J',8,5,20,72,'(A)dd FONT'
1067 DATA 6,5,4,14,'(N)ew',6,5,4,25,'(C)opy',6,5,4,36,'(D)elete',8,5,58,108,'Size'
1068 DATA 8,5,20,56,'(U)ser Chk',8,5,20,36,'(K)ey CTRL',8,5,20,46,'(M)aps [#]'
1069 DATA 8,5,20,124,'(P)alette',8,5,20,135,'(B)ackGnd',8,5,20,146,'(R)eColour'
1070 DATA 8,7,28,170,'↑→',8,5,40,159,'Paint',8,5,38,183,'rase'
1071 BLOCK#8,14,12,73,158,CP(7):BLOCK#8,12,10,74,159,0
1072 BLOCK#8,26,5,42,173,CP(7) :BORDER#11,1,4:INK#8,CP(7):cur=0:rub=0:GCErase
1073 RESTORE 1074:FOR i=0 TO 7:READ TASS(i)
1074 DATA 'Solid','Floor','Hazard','Reward','???' '???' '???' '???'
1075 END DEFINE

```



1077 DEFine PROCEDURE Init_Guide

```

1078 ch=10:INK#ch,CP(7):RESTORE 1088
1079 CURSOR#ch,0, 4:PRINT#ch,'X z:M1$='FLIPROLL'
1080 CURSOR#ch,0,48:PRINT#ch,'Y Z:M2$='MOVESIZE':INK#ch,CP(5)
1081 FOR i=0 TO 3:CURSOR#ch, 0,14+i*8:PRINT#ch,M1$(i+1),' 'M1$(i+5)
1082 FOR i=0 TO 3:CURSOR#ch,12,10+i*8:PRINT#ch,M2$(i+1),' 'M2$(i+5)
1083 BLOCK#ch,24,20,28,17,CP(7):BLOCK#ch,22,18,29,18,0:BLOCK#ch,16,20,32,17,0
1084 CURSOR#ch,12,0:PRINT#ch,'MAX min'
1085 BLOCK#ch,14,11,33,0,CP(7):BLOCK#ch,12, 9,34,1,0
1086 BLOCK#ch,14, 3,33,4,0:BLOCK#ch, 6,11,37,0,0:BLOCK#ch,5,5,47,33,CP(7)
1087 FOR i=1 TO 7:READ pc,px,py,p$:INK#ch,CP(pc):CURSOR#ch,px,py:PRINT#ch,p$
1088 DATA 5,9,52,'PAN/SCROLL',7,12,42,'←↑Shift ↓→',7,37,1,'#'
1089 DATA 7,37,12,'↑',7,22,22,'← →',7,31,23,'ALT',7,37,33,'↓'
1090 END DEFINE

```



1092 DEFINE PROCEDURE QTitle(ch,chz,tx,ty,str\$)

```

1093 OVER#ch,1:CSIZE#ch,chz,1
1094 INK#ch,2:FOR i=0 TO 1:CURSOR#ch,tx+i,ty :PRINT#ch,str$
1095 INK#ch,6:FOR i=2 TO 3:CURSOR#ch,tx+i,ty+1:PRINT#ch,str$
1096 OVER#ch,0:CURSOR#ch,0,0:CSIZE#ch,0,0
1097 END DEFINE

```



1099 DEFine PROCEDURE QBold(ch,bc,bx,by,B\$)

```

1100 INK#ch,CP(bc):CURSOR#ch,bx,by :PRINT#ch,B$
1101 OVER#ch,1:CURSOR#ch,bx,by+1:PRINT#ch,B$:OVER#ch,0
1102 END DEFINE

```



1104 DEFINE PROCEDURE QPrnt

```

1105 LOCAL ch,i,x,y,p$:READ ch,i,x,y,p$:INK#ch,CP(i):CURSOR#ch,x,y:PRINT#ch,p$
1106 END DEFINE

```



1108 REMark PIXEL Art Menu

1110 DEFine PROCEDURE PIXArt_Menu

```

1111 INK#8,CP(7):rub=0:GCErase:k=0:ic=4:oc=4:bc=0:pn=0:
1112 tmax=96:tn=0:tm=0:sn=1:obj=0:frm=31:fr=0:frt=0:cm=16:rm=16
1113 Init_Guide:PCol:DIRFile:FrSet:Info:GFrame
1114 REPEAT Main_Ip
1115 IF fed=1 AND fr>0
1116     IF rub=7:c=x:r=y:FG(fr,c,r)=255:Gbit
1117     IF cur=7:c=x:r=y:FG(fr,c,r)=ic:Gbit
1118 END IF
1119 IF fed=2 AND obj=0
1120     IF rub=7:tx=2+x*16:ty=1+y*16:BLOCK#3,16,16,tx,ty,CP(bg):TScn(sn,x,y)=0
1121     IF cur=7:tx=2+x*16:ty=1+y*16:TDraw 0:TScn(sn,x,y)=tn
1122 END IF
1123 BLOCK#8,24,3,41,166,CP(cur):INK CP(7):INK#8,CP(7):GErase
1124 CURSOR 106,25:PRINT FILL$( '0',2-LEN(x+1));x+1
1125 CURSOR 136,25:PRINT FILL$( '0',2-LEN(y+1));y+1
1126 GPos:k=CODE(INKEY$(-1)):ox=x:oy=y:GPos:GLoc
1127 SELECT ON k
1128     =32:IF cur=0:cur=7:rub=0:ELSE cur=0:REMark SB ON/OFF
1129     =69,101:IF rub=0:rub=7:cur=0:ELSE rub=0:REMark (E)rase
1130     =85,117:REtro_PlayREMark (U)ser Check
1131     =70,102:SelDrv 1,'Drive ':DIRFileREMark (F)DIR
1132     =76,108:BMLoad:DIRFileREMark (L)oad
1133     =83,115:BMSave:DIRFileREMark (S)ave
1134     =81,113:QExit:BLOCK#5,18,10,160,12,0REMark (Q)uit
1135     =47,63:col%=CP(6):Info:PAUSE:IF fed=1:GFrame:ELSE GTile:REMark (?)Help
1136 END SELECT
1137 IF fed=1:SGen
1138 IF fed=2:TGen
1139 END REPEAT Main_Ip
1140 END DEFine

```

Grde Mode

Erase - Cell restored to Background

Paint ON Set Cell colour

Tile Mode

Remove Screen Tile

Paint ON Set Screen Tile

GRID Cell / Screen TILE XX Position

GRID Cell / Screen TILE YY Position

X: 12 Y: 12

Edit Functions for SPRITE Frame

Edit Functions For SCREEN TILE MAPS & SPRITE Key-CTRL

1142 DEFine PROCEDURE GCErase

```

1143 BLOCK#8,12,10,22,183,CP(rub):OVER#8,-1:CURSOR#8,24,183:PRINT#8,'E':OVER#8,0
1144 BLOCK#8,12,10,72,183,CP(cur):OVER#8,-1:CURSOR#8,74,183:PRINT#8,'@':OVER#8,0
1145 END DEFine

```

1147 DEFine PROCEDURE GPos

Grid Cursor Position

```

1148 OVER#3,-1:BLOCK#3,gpw,gph,gpx,gpy,CP(7):OVER#3,0
1149 END DEFine

```

1151 DEFine PROCEDURE GLoc

Grid Location XX/YY change

```

1152 SELECT ON k
1153     =192:IF x> 0:x=x-1:REMark ← x \
1154     =200:IF x<xm:x=x+1:REMark → x Grid Tile
1155     =208:IF y> 0:y=y-1:REMark ↑ y Position
1156     =216:IF y<ym:y=y+1:REMark ↓ y /
1157 END SELECT
1158 IF fed=1:gpw=cw+1:gph=rh+1:gpx=-1+zx+x*cw:gpy=-1+zy+y*rh
1159 IF obj=1 AND lks=0:Scn_Switch
1160 IF obj=1 AND lks=1:Scn_Slide
1161 IF fed=2:gpw=8:gph=8:gpx=6+x*16:gpy=5+y*16
1162 END DEFine

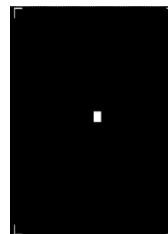
```

Settings for Grde Mode

Check for Screen Switch [Tile Mode]

Check for Screen Slide [Tile Mode]

Settings Check for Tile Mode



1164 **DEFine PROCEDURE DIRFile**

1165 CLS#5:INK#5,CP(7):CURSOR#5,16,1:PRINT#5,'(F)DIR (L)oad (S)ave (Q)uit'

1166 OVER#5,1:CURSOR#5,23,1:PRINT#5,'F L S Q' :OVER#5,0

1167 **END DEFine**

Note: Press 'F' then 'E' to Edit Drive/SubDIR :

To Load Press 'L' Select a Drive then Select File from List presented

To Save Press 'S' Select a Drive then 'E' to Edit Filename if required

Note: Switches to GRID Mode – Set TILE Mode Inactive - Change Settings and Show SPRITE Frame

1169 **DEFine PROCEDURE GFrame**

1170 BLOCK#8,30,10,56,4,0:TNum 5:obj=0:CLSOBJ 5:CLS#7 :REMark Disable Tile Mode

1171 INK#6,CP(7):CURSOR#6,52,4:PRINT#6,cm;'x';rm :fed=1:sz=1:k=0

1172 **FrSet:FRNumf rt,5,56,50**

1173 **END DEFine**

Note: Switches to TILE Mode – Set GRID Mode Inactive - Change Settings and Show SCREEN/TILE Background

1175 **DEFine PROCEDURE GTile**

1176 INK#6,CP(5):CURSOR#6,52,4:PRINT#6,cm;'x';rm :REMark Disable Grid Mode

1177 **QBold 8,7,55,4,'< >':TNum 7:tmax=120:obj=0 :fed=2 :sz=3:k=0**

1178 x=9:xm=19:y=5:ym=11:lks=0:cur=0:rub=0::GLoc: HLObj 5 :Tile_MAP

1179 **END DEFine**

Note: Switches to RETRO Mode – Sets SCREEN/Tile Background and Shows Active SPRITES

1181 **DEFine PROCEDURE Retro_Play**

1182 INK#8,CP(7):CURSOR#8,20,56:PRINT#8,'(U)[Esc] ':Set_Score

1183 BLOCK#3,170,48,80,46,CP(7):BLOCK#3,168,46,81,47,0

1184 **QTitle 3,2,132,50,'UNDER':QTitle 3,2,90,70,'CONSTRUCTION':PAUSE**

1185 INK#8,CP(5):CURSOR#8,20,56:PRINT#8,'(U)ser Chk'

1186 CLS#3:CLS#5:DIRFile:IF fed=1:GFrame:ELSE GTile

1187 **END DEFine**

Note: Exit Prog:- CLOSE Open Channels and Restore QL_COLOUR Setting - LRUN QBITSProgs_bas

1189 **DEFine PROCEDURE QExit**

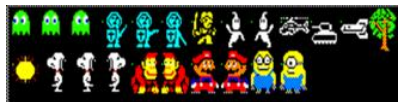
1190 CURSOR#5,160,12:PRINT#5,'Y/N':IF GAns=0:RETurn

1191 CLOSE#10:CLOSE#8:CLOSE#7:CLOSE#6:CLOSE#5:CLOSE#4:CLOSE#3:CLS#2:COLOUR_QL

1192 CURSOR#0,0,0:PRINT#0,'Bye...':LRUN dn\$

1193 **END DEFine**

1200 REMark SPRITE Generator



Note: SPRITE Frames 1...32 max - Saved as _bmp File

Bytes: [0 to 2] **QL8** or **PAL** ID Header followed by

[3] **bm** number of SPRITE Bitmaps GRID Size [4] **cm** columns [5] **rm** rows

[6] **bg** BackGnd colour [7] **sz** File Loading Reference

[8 to 15] BackGnd Colours range 8...255 of selected Palette QL or PAL

[16.....nn.] SPRITE BITMaps [bm * cm * rm]

1202 DEFINE PROCEDURE SGen

1203 SELECT ON k

1204 =84,116:GTile

1205 =45,95:IF fr>0 :fr=fr-1:FrChk

1206 =43,61:IF fr<frt:fr=fr+1:FrChk

1207 =10,32,44,46,60,62,9,253::oic=ic:PCol

1208 =65,97:IF frt<frm AND cg%>0:FontAdd

1209 =78,110:IF frt<frm:FrNEW :CURSOR#6,21,14:PRINT#6,'ew' :REMark (N)ew

1210 =67,99:IF frt>1 :FrCOPY:CURSOR#6,21,25:PRINT#6,'opy' :REMark (C)opy

1211 =68,100:IF frt>0 :FrDEL :CURSOR#6,21,36:PRINT#6,'elete' :REMark (D)elete

1212 =39,64:IF frt>0 :FrChg 1:GDraw :REMark (@) Grid Fill

1213 =82,114:IF frt>0 :FrReCol :REMark (R)eColour

1214 =70,103 :GSize :REMark (G)rid

1215 =66,98:IF frt>0 :GBGnd :REMark (B)kGnd

1216 =80,112 :GPalette :REMark (P)alette

1217 = 35:IF fr>0 :EMaxMin :REMark # Edit Area

1218 =193:EMove:ELeft :EMove :REMark ALT ←

1219 =201:EMove:ERight :EMove :REMark ALT → Move

1220 =209:EMove:EUp :EMove :REMark ALT ↑ SIZE

1221 =217:EMove:EDown :EMove :REMark ALT ↓

1222 =196:dm=0:pa=cs+cn-1:pb=cs:pc=cs-1:pd=cs :GSlid :REMark Shift ← PAN

1223 =204:dm=0:pa=cs:pb=cs+cn-1:pc=cs:pd=cs-1 :GSlid :REMark Shift → PAN

1224 =212:dm=1:sa=rs+m-1:sb=rs:sc=rs-1:sd=rs :GSlid :REMark Shift ↑ SCROLL

1225 =220:dm=1:sa=rs:sb=rs+m-1:sc=rs:sd=rs-1 :GSlid :REMark Shift ↓ SCROLL

1226 =88,120:xf=cs+cn-1:yf=rs:xz=-1:yz=1 :GFlip :REMark X FLIP

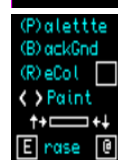
1227 =89,121:yf=rs+m-1:xf=cs:yz=-1:xz=1 :GFlip :REMark Y FLIP

1228 =122:rxm=m-1:rym=0:xt=-1:yt=1 :GRoll :REMark Z ROLL Anti-CW

1229 = 90:rym=m-1:rxm=0:yt=-1:xt=1 :GRoll :REMark Z ROLL ClockWise

1230 END SELECT

1231 END DEFINE



1233 DEFINE PROCEDURE PCol

1234 BLOCK#8,12,10,4,34+oic*10,0:BLOCK#8,8,6,6,36+oic*10,CP(oic)

1235 SELECT ON k=44,60,253:ic=ic-1:IF ic<pn:ic=15:END IF :END SELECT

1236 SELECT ON k=46,62, 9:ic=ic+1:IF ic>15:ic=pn:END IF :END SELECT

1237 BLOCK#8,12,10,4,34+ic*10,CP(ic):BLOCK#8, 4,3,8,37,255

1238 IF ic=0:BLOCK#8,12,10,4,34,CP(7):BLOCK#8,10,8,5,35,0

1239 END DEFINE

Highlighted Palette Colour

1241 DEFINE FuNction GAns(ax,ay,a\$)

1242 REPEAT lp

1243 k=CODE(INKEY\$(-1))

1244 SELECT ON k=10,89,121:RETURN 1

1245 SELECT ON k=32,78,110:RETURN 0

1246 END REPEAT lp

1247 END DEFINE



1250 REMark **SPRITE** Frame GRID Edit

1252 **DEFine PROCEDURE EMaxMin**

1253 IF cn<cm OR m<rm

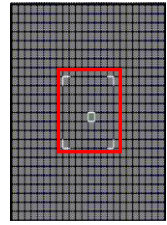
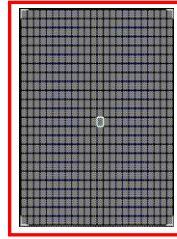
1254 **EMove**:cs=0:cn=cm:rs=0:rn=rm:**EMove**

1255 ELSE

1256 **EMove**:cs=cm/2:cn=8:rs=rm/2:rn=8:**EMove**

1257 END IF

1258 **END DEFine**



MAX  min Edit Area

1260 **DEFine FuNction Epos**

1261 IF cs=x AND rs=y :RETurn 1

:REMark Top Left

1262 IF cs=x AND rs=m-1=y :RETurn 2

:REMark Bottom Left

1263 IF cs+cn-1=x AND rs=y :RETurn 3

:REMark Top Right

1264 IF cs+cn-1=x AND rs+m-1=y :RETurn 4

:REMark Bottom Right

1265 RETurn 0

1266 **END DEFine**

Position Edit Area
ReSIZE Edit Area

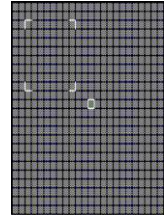
1268 **DEFine PROCEDURE EUop**

1269 IF rs>0 AND EPos=0:rs=rs-1

1270 IF rs>0 AND EPos=1 OR rs>0 AND EPos=3:m=m+1:y=y-1:rs=rs-1

1271 IF m>4 AND EPos=2 OR m>4 AND EPos=4:m=m-1:y=y-1

1272 **END DEFine**



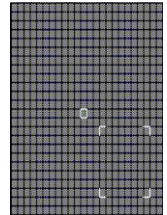
1274 **DEFine PROCEDURE EDown**

1275 IF rs+m<rm AND EPos=0:rs=rs+1

1276 IF m>4 AND EPos=1 OR m>4 AND EPos=3:m=m-1:y=y+1:rs=rs+1

1277 IF rs+m<rm AND EPos=2 OR rs+m<rm AND EPos=4:m=m+1:y=y+1

1278 **END DEFine**



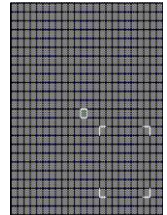
1280 **DEFine PROCEDURE ELeft**

1281 IF cs>0 AND EPos=0:cs=cs-1

1282 IF cs>0 AND EPos=1 OR cs>0 AND EPos=2:cn=cn+1:x=x-1:cs=cs-1

1283 IF cn>4 AND EPos=3 OR cn>4 AND EPos=4:cn=cn-1:x=x-1

1284 **END DEFine**



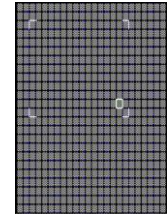
1286 **DEFine PROCEDURE ERight**

1287 IF cs+cn<cm AND EPos=0:cs=cs+1

1288 IF cn>4 AND EPos=1 OR cn>4 AND EPos=2:cn=cn-1:x=x+1:cs=cs+1

1289 IF cs+cn<cm AND EPos=3 OR cs+cn<cm AND EPos=4:cn=cn+1:x=x+1

1290 **END DEFine**



1292 **DEFine PROCEDURE EMove**

1293 x1=-1+zx+cs*cw:x2=-1+zx+(cs+cn)*cw:y1=-1+zy+rs*rh:y2=-1+zy+(rs+m)*rh

1294 OVER#3,-1:col%=CP(7)

1295 BLOCK#3,cw+1,1,x1+1,y1,col% :BLOCK#3,1,rh+1,x1,y1 ,col%

1296 BLOCK#3,cw ,1,x2-cw, y1,col% :BLOCK#3,1,rh+1,x2,y1 ,col%

1297 BLOCK#3,cw+1,1,x1+1,y2,col% :BLOCK#3,1,rh+1,x1 ,y2-rh,col%

1298 BLOCK#3,cw ,1,x2-cw, y2,col% :BLOCK#3,1,rh+1,x2 ,y2-rh,col%

1299 OVER#3,0

1300 **END DEFine**

:

Repositioning Grid Cells

GSlide	PAN Left	dm=0:pa=cs+cn-1:pb=cs:pc=cs-1:pd=cs	:
GSlide	PAN Right	dm=0:pa=cs:pb=cs+cn-1:pc=cs:pd=cs-1	
GSlide	SCROLL Up	dm=1:sa=rs+m-1:sb=rs:sc=rs-1:sd=rs	
GSlide	SCROLL Down	dm=1:sa=rs:sb=rs+m-1:sc=rs:sd=rs-1	
GFlip	FLIP X	xf=cs+cn-1:yf=rs:xz=-1:yz=1	
GFlip	FLIP Y	yf=rs+m-1:xf=cs:yz=-1:xz=1	
GRoll	ROLL z	rxm=m-1:rym=0:xt=-1:yt=1	Anti-CW
GRoll	ROLL Z	rym=m-1:rxm=0:yt=-1:xt=1	Clockwise

Note: Edit Area reduced to Square of shortest side

1302 DEFINE PROCEDURE GSlid

Grid Slide

1303 IF dm=0

1304 FOR r=rs TO rs+m-1

1305 FOR c=1 TO cn-1: TG(pa,r)=FG(fr,pb,r):TG(c+pc,r)=FG(fr,c+pd,r)

Left/Right

1306 END FOR r

1307 ELSE

1308 FOR r=1 TO m-1

1309 FOR c=cs TO cs+cn-1: TG(c,sa)=FG(fr,c,sb):TG(c,r+sc)=FG(fr,c,r+sd)

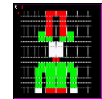
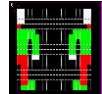
Up/Down

1310 END FOR r

1311 END IF

1312 cur=0:GTSet:GDraw

1313 END DEFINE



1315 DEFINE PROCEDURE GFlip

Grid Flip XX YY

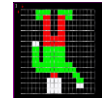
1316 FOR r=0 TO m-1

1317 FOR c=0 TO cn-1: TG(xf+c*xz,yf+r*yz)=FG(fr,cs+c,rs+r)

1318 END FOR r

1319 cur=0:GTSet:GDraw

1320 END DEFINE



1322 DEFINE PROCEDURE GRoll

Grid Rotate 90°

1323 IF cs+m>cm: RETURN :ELSE EMove:cn=rm:EMove

1324 FOR r=0 TO m-1

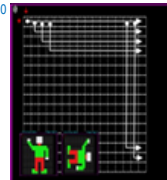
1325 rx=rxm+(r*xt):ry=rym

1326 FOR c=0 TO m-1: TG(cs+c,rs+r)=FG(fr,cs+rx,rs+ry):ry=ry+yt

1327 END FOR r

1328 cur=0:GTSet:GDraw

1329 END DEFINE



1331 DEFINE PROCEDURE GTSet

Grid Temp

1332 FOR r=rs TO rs+m-1: FOR c=cs TO cs+cn-1: FG(fr,c,r)=TG(c,r):END FOR c:END FOR r

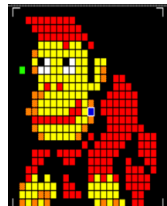
1333 END DEFINE

1335 DEFINE PROCEDURE GDraw

Grid Draw

1336 FOR r=rs TO rs+m-1: FOR c=cs TO cs+cn-1: GBit:END FOR c:END FOR r

1337 END DEFINE



1339 DEFINE PROCEDURE GBit

Grid Pixel Bit

1340 IF FG(fr,c,r)=255: pcol%=CP(bg):ELSE pcol%=CP(FG(fr,c,r))

1341 IF fed=1: BLOCK#3,cw-1,rh-1,zx+c*cw,zy+r*rh,pcol%

1342 BLOCK#4,1,1,px+c,py+r,pcol%

1343 END DEFINE

1350 REMark SPRITE Frame Actions

1352 DEFine PROCEDURE GSize

1353 ocm=cm:orm=rm:INK#6,CP(7):CURSOR#6,2,2:PRINT#6,'(G)':FrMes 6,20,2

1354 CURSOR#6,56,10:PRINT#6,' ← ↑ ↓ → ':zx=66:zy=1

1355 REPEAT Size Ip

1356 CURSOR#6,50,2:PRINT#6,cm;'x':rm:k=CODE(INKEY\$(-1))

1357 SElect ON k

1358 =192:IF cm>=24:cm=cm-8:cw=192 DIV rm:rh=cw

1359 =200:IF cm<=56:cm=cm+8 :IF cm>rm:cw=192 DIV cm:rh=cw

1360 =208:IF rm<=56:rm=rm+8 :IF rm>cm:cw=192 DIV rm:rh=cw

1361 =216:IF rm>=24:rm=rm-8:cw=192 DIV cm:rh=cw

1362 = 32:cm=ocm:rm=orm:EXIT Size Ip

1363 = 10:FrSet:EXIT Size Ip

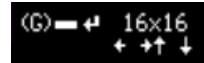
1364 END SElect

1365 END REPEAT Size Ip

1366 BLOCK#6,30,10,46,10,0:INK#6,CP(5):

1367 CURSOR#6,2,2:PRINT#6,'(G)RID ':INK#6,CP(7):PRINT#6,cm;'x':rm;'

1368 END DEFine



Note: Grid Size multiples of 8
Range 16 to 64

1370 DEFine PROCEDURE FrMes(ch,mx,my)

1371 INK#ch,CP(7):CURSOR#ch,mx,my:PRINT#ch,' ← '

1372 BLOCK#ch,10,3,mx,my+4,CP(7):BLOCK#ch,2,4,mx+18,my+2,CP(7)

1373 END DEFine



Note: Frame Message Prompt
Abort Spacebar or Action Enter

1375 DEFine PROCEDURE FrSet

Note: Frame Set - Parameters

1376 cw=192 div rm:rh=cw:xm=cm-1:ym=rm-1:zx=162-(cm*cw)/2:zy=98-(rm*rh)/2

1377 :px=43-cm/2:py=33-tm/2:x=cm/2:y=rm/2:cs=0:cn=cm:rs=0:rn=rm:CLS#3:CLS#4:FrChk

1378 END DEFine

1380 DEFine PROCEDURE FrChk

Note: Frame Change Check

1381 IF fed=1 AND fr=0:FrShow:FrNum fr,7,8,50:RETurn

1382 IF fed=1:FrNum fr,7,8,50:cs=0:cn=cm:rs=0:rn=rm:CLS#3:CLS#4:GDraw:EMove

1383 IF fed=2:FrNum fr,7,8,50:CLS#4:GDraw

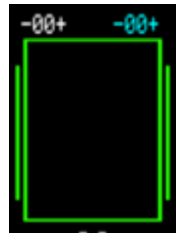
1384 END DEFine

1386 DEFine PROCEDURE FrNum(fr%,fi%,fx%,fy%)

1387 INK#6,CP(fi%):CURSOR#6,fx%,fy%:PRINT#6,'-':FILL\$(0',2-LEN(fr%));fr%:'+'

1388 END DEFine

Note: Frame Numbering displays Current and Total SPRITES



1390 DEFine PROCEDURE FrShow

1391 PAPER 0:CLS#3:nx=320 DIV cm:ny=192 DIV rm:max=nx*ny

1392 fx=2:fy=1:IF max>frt:max=frt

1393 FOR f=1 TO max

1394 FOR r=0 TO rm-1

1395 FOR c=0 TO cm-1

1396 IF FG(f,c,r)=255:pcol%=0:ELSE pcol%=CP(FG(f,c,r))

1397 BLOCK#3,1,1,fx+c,fy+r,pcol%

1398 END FOR c

1399 END FOR r

1400 fx=2+(f MOD nx)*cm:fy=1+(f DIV nx)*rm

1401 END FOR f

1402 END DEFine

Note: -00+ Frame Option for Display of SPRITE Gp



Note: QBITS PIXEL Art Import of QL FONT_fnt File BITMaps and ReSize into SPRITE Frames
First Load a FONT_fnt File. Press 'A' and select with -/+ set to '0' to display Set.

```

1404 DEFine PROCEDURE FntAdd
1405 CHAR_USE#11,Fntaddr,0:QBold 8,7,36,72,'/+ ' :FrMes 8,58,72:ocsn=csn:ofr=fr
1406 REPEAT char_lp
1407 CURSOR#11,3,0:PRINT#11,CHR$(ch%+sg%):IF ch%=0:CLS#3:FntShow
1408 CURSOR#8,41,108:PRINT#8,csx:CURSOR#8,58,90:PRINT#8,csy
1409 k=CODE(INKEY$(-1))
1410 SElect ON k
1411 =45,95:ch%=ch%-1:IF ch%<=0:ch%=0 :REMark - Chr$(n)
1412 =43,61:ch%=ch%+1:IF ch%>=cg%:ch%=cg% :REMark + Chr$(n)
1413 =192:csx=csx-1 :IF csx<=1:csx=1
1414 =200:csx=csx+1 :IF csx>=6:csx=6
1415 =208:csy=csy+1 :IF csy>=6:csy=6
1416 =216:csy=csy-1 :IF csy<=1:csy=1
1417 = 32:csn=ocsn:fr=ofr:EXIT char_lp
1418 = 10:csn=ch%:FntSize csn,csx,csy:EXIT char_lp
1419 END SElect
1420 END REPEAT char_lp
1421 INK#8,CP(5):CURSOR#8,20,72:PRINT#8,'(A)dd FONT':FrChk
1422 END DEFine

```

```

(F)DIR (L)oad (S)ave (Q)uit
Load Aliens.fnt
Select + - =

```



```

1424 DEFine PROCEDURE FntShow
1425 WINDOW#3,260,180,gx+150,gy+38:CSIZE#3,3,1:CHAR_USE#3,Fntaddr,0
1426 PRINT#3:FOR i=1+sg% TO sg%+cg%:PRINT#3,CHR$(i);
1427 WINDOW#3,324,194,gx+ 94,gy+36:CSIZE#3,0,0:CHAR_USE#3,0,0
1428 END DEFine

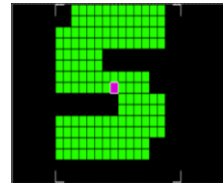
```



```

1430 DEFine PROCEDURE FntSize(csn,csx,csy)
1431 frt=frt+1:fr=frt:FrCLS fr
1432 FOR r=0 TO 8:fnt$(r)=BIN$(PEEK(Fntaddr+2+r+csn*9),8)
1433 FOR r=0 TO 8
1434 yz=csy*r:yr=0
1435 FOR c=0 TO 7
1436 xz=csx*c:xc=0
1437 REPEAT clp
1438 IF fnt$(r,c+1)='1':FG(fr,xz+xc,yz)=ic:ELSE FG(fr,xz+xc,yz)=bg
1439 xc=xc+1:IF xc>=csx:EXIT clp
1440 END REPEAT clp
1441 END FOR c
1442 REPEAT rlp
1443 yr=yr+1:IF yr=csy:EXIT rlp
1444 FOR c=0 TO 8*csx:FG(fr,c,yz+yr)=FG(fr,c,yz)
1445 END REPEAT rlp
1446 END FOR r
1447 FrNum fr,7,8,50:FrNum frt,5,56,50:rs=0:rn=rm:cs=0:cn=cm
1448 END DEFine

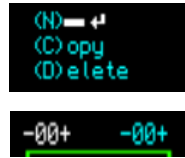
```




```

1450 DEFine PROCEDURE FrNew
1451 INK#6,CP(7):FrMes 6,21,14:IF GAns=0:INK#6,CP(5):RETurn
1452 IF fr=frt:frt=frt+1:fr=fr+1:FrCLS fr
1453 IF fr<frt:FOR f=frt TO fr STEP -1:FrChg 4:END FOR f:frt=frt+1:FrCLS fr
1454 FrNum fr,7,8,50:FrNum frt,5,56,50:INK#6,CP(5)
1455 CLS#3:cs=0:cn=cm:rs=0:rn=rm:xm=cm-1:ym=rm-1:fed=1:ch=3:GDraw:EMove
1456 END DEFine

```



```

1458 DEFine PROCEDURE FrCOPY
1459 INK#6,CP(7):FrMes 6,21,25:FrNum fr,5,8,50:f2=fr
1460 REPeat Copy_LP
1461 FrNum f2,7,56,50:GDraw
1462 k=CODE(INKEY$(-1))
1463 SElect ON k
1464 =45,95:IF f2>1:f2=f2-1
1465 =43,61:IF f2<frt:f2=f2+1
1466 =78,110,32:EXIT Copy_LP
1467 =89,121,10:FrChg 6:fr=f2:GDraw:fed=1:EXIT Copy_LP
1468 END SElect
1469 END REPeat Copy_LP
1470 FrNum fr,7,8,50:FrNum frt,5,56,50
1471 INK#6,CP(5):EMove:cs=0:cn=cm:rs=0:rn=rm:EMove
1472 END DEFine

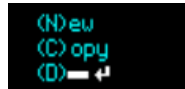
```



```

1474 DEFine PROCEDURE FrDEL
1475 BLOCK#6,42,10,21,36,0:INK#6,CP(7):FrMes 6,21,36:IF GAns=0:INK#6,CP(5):RETurn
1476 IF frt=1:fr=1:FrCLS :frt=0:fr=0
1477 IF frt>1 AND fr=frt:FrCLS frt:frt=frt-1:fr=frt
1478 IF frt>1 AND fr<frt:FOR f=fr TO frt:FrChg 3:END FOR f:FrCLS frt:frt=frt-1
1479 FrNum fr,7,8,50:FrNum frt,5,56,50:INK#6,CP(5)
1480 CLS#3:cs=0:cn=cm:rs=0:rn=rm:ch=3:GDraw:EMove
1481 END DEFine

```



```

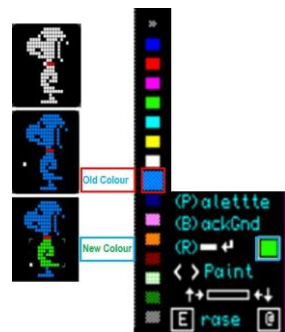
1483 DEFine PROCEDURE FrCLS(f%)
1484 FOR r=0 TO 63:FOR c=0 TO 63:FG(f%,r,c)=255:END FOR c:END FOR r
1485 END DEFine

```

```

1487 DEFine PROCEDURE FrReCol
1488 orub=rub:sic=ic:FrMes 8,38,146:BLOCK#8,10,8,75,160,CP(sic)
1489 REPeat P_lp
1490 k=CODE(INKEY$(-1))
1491 SElect ON k
1492 =44,46,60,62,9,253:oi=ic:PCol
1493 =10:IF oi=bg:EXIT P_lp:ELSE FrChg 2:GDrawEXIT P_lp
1494 =32:oi=sic:EXIT P_lp
1495 END SElect
1496 END REPeat P_lp
1497 INK#8,CP(5):CURSOR#8,380,146:PRINT#8,'eColour ':k=0
1498 rub=0:cur=0:sic=ic:PCol:BLOCK#8,10,8,75,160,CP(sic)
1499 END DEFine

```



Note: First Select **New Colour** from **< > Palette**. Press **[R]** then Select **Old** [existing] Grid Colour Press **Enter** and Colours are exchanged within Whole Grid or Resized Edit Area.

1501 DEFINE PROCEDURE FChg(ck)

```
1502 FOR r=rs TO rs+m-1
1503   FOR c=cs TO cs+cn-1
1504     IF ck=1 AND rub=0:FG(fr,c,r)=ic:cur=7
1505     IF ck=1 AND rub=7:FG(fr,c,r)=255
1506     IF ck=2:IF FG(fr,c,r)=ic:FG(fr,c,r)=sic
1507     IF ck=3:FG(fr,c,r)=FG(fr+1,c,r)
1508     IF ck=4:FG(fr+1,c,r)=FG(fr,c,r)
1509     IF ck=5 AND FG(fr,c,r)=bg:FG(fr,c,r)=255
1510     IF ck=6:FG(fr+2,c,r)=FG(fr,c,r)
1511   END FOR c
1512 END FOR r
1513 END DEFINE
```

Note: Frame Grp Array Change of Settings

```
:REMark Set Edit Area New Colour
:REMark Set Edit Area BackGnd
:REMark Chg Edit Area Colours
:REMark Delete Frame
:REMark Insert New Frame
:REMark Restore BackGnd
:REMark Copy Frame f2 – fr
```

Note: Press 'B' and Select Palette Colour with < > chevron keys shown as →

1515 DEFINE PROCEDURE GBGnd

```
1516 tic=ic:obg=bg:oic=ic:ic=bg:INK#8,CP(7)
1517 CURSOR#8,20,135:PRINT#8,'(B)<>' :FrMes 8,58,136
1518 REPEAT BGnd_ip
1519   PCol:k=CODE(INKEY$(-1))
1520   SELECT ON k
1521     =44,46,60,62,9,253:oic=ic
1522     =10:IF ic=bg:EXIT BGnd_ip:ELSE bg=ic:FrChg 5:GDraw:EXIT BGnd_ip
1523     =32:EXIT BGnd_ip
1524   END SELECT
1525 END REPEAT BGnd_ip
1526 INK#8,CP(5):oic=ic:tic=ic:PCol:CURSOR#8,22,130:PRINT#8,'(B)ackGnd':FrChg
1527 END DEFINE
```

1529 DEFINE PROCEDURE GPalette

```
1530 INK#8,CP(7):pn=8:IF ic<8:oic=ic:ic=8:PCol:END IF
1531 CURSOR#8,22,120:PRINT#8,'(P)<>' :FrMes 8,60,120
1532 CURSOR#8,22,139:PRINT#8,'↑ → ↓ ←':tic=CP(ic):oic=ic
1533 REPEAT Col_ip
1534   PCol:col%=CP(ic):CURSOR#8,43,130:PRINT#8,FILL$(0',3-LEN(col%)):col%
1535   k=CODE(INKEY$(-1)):oic=ic
1536   SELECT ON k
1537     =192:IF col%> 8:col%=col% -1:CP(ic)=col%
1538     =200:IF col%<255:col%=col% +1:CP(ic)=col%
1539     =208:IF col%<192:col%=col%+64:CP(ic)=col%
1540     =216:IF col%> 64:col%=col%-64:CP(ic)=col%
1541     = 32:CP(ic)=tic :PCol:EXIT Col_ip
1542     = 10:CP(ic)=col%:PCol:FrChk:EXIT Col_ip
1543   END SELECT
1544 END REPEAT Col_ip
1545 BLOCK#8,60,20,20,120,0:INK#8,CP(5):CURSOR#8,22,120:PRINT#8,'(P)alette ':
1546 CURSOR#8,220,130:PRINT#8,'(B)ackGnd':pn=0
1547 END DEFINE
```

Enlarged Square
on Palette Bar.

(B) < > ← →

(P) < > ← →
↑ → 225 ↓ ←

Note: Only Palette Colours 8.5 are changeable. Select colour and change Number displayed with Cursor Keys. The New Colour is shown in Enlarged Square on the Palette Bar.

1550 REMark SCREEN/TILE Generator

1552 DEFine PROCEDURE TGen

1553 SElect ON k

```

1554 =70,103:GFrame                                :REMark (G)rid
1555 =45,95:IF fr>0 :fr=fr-1                        :FrChk          :REMark - Frame
1556 =43,61:IF fr<frt:fr=fr+1                      :FrChk          :REMark + Frame
1557 =60,44:IF tn>1 :tn=tn-1                        :TNum 7         :REMark <
1558 =62,46:IF tn<tm :tn=tn+1                      :TNum 7         :REMark >
1559 =75,107:IF frt>0:AGen                          :DIRFile        :REMark (K)ey CTRL
1560 =49 TO 57:sn=k-48:IF tm>0                      :Tile_MAP       :REMark (1...9)
1561 =67,99:IF fr>0 AND tm<tmx :Tile_ADD:CURSOR#6,29,26:PRINT#6,'opy '
1562 =68,100:IF tm>0                                :Tile_DEL:CURSOR#6,29,36:PRINT#6,'elete'
1563 =66,98:GBGnd:IF k=10                          :Tile_MAP       :REMark (B)kGnd
1564 =84,116:IF tn>0                                :Tile_Ass       :REMark (T)ile Asset
1565 =77,109:CLS#3:INK#3,CP(6) :Scn_MAP:Scn_Links :REMark (M)AP Scn Links
1566 =35:IF obj=0:obj=1:HObj 7:ELSE obj=0:HObj 5    :REMark # Check Links
1567 END SElect
1568 END DEFine

```

1570 DEFine PROCEDURE Scn_Num(sn)

```

1571 INK CP(7):CURSOR 405,25:PRINT sn:cur=7
1572 END DEFine

```

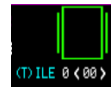
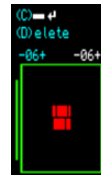
Note: SCREEN Num (1..9) [1] SCREEN

1574 DEFine PROCEDURE Tile_ADD

```

1575 FrMes 6,21,25:IF frt=0 OR GAns=0:INK#6,CP(5):RETURN
1566 IF tn=0 OR tn=tm:tn=tn+1
1577 IF tn<tm:FOR td=tm TO tn STEP -1:Tile_Chg 2:END FOR td
1578 td=tn:Tile_Chg 1:tm=tm+1:TNum 7:INK#6,CP(5)
1579 END DEFine

```



Copy SPRITE Frame to TILE Library

1581 DEFine PROCEDURE Tile_DEL

```

1582 BLOCK#6,30,10,32,36,0:FrMes 6,21,36:IF GAns=0:INK#6,CP(5):RETURN
1583 IF tn<tm:FOR td=tn TO tm:Tile_Chg 3:END FOR td:td=tm:Tile_Chg 4
1584 tm=tm-1:IF tn>tm:tn=tm:END IF :TNum 7:INK#6,CP(5)
1585 END DEFine

```



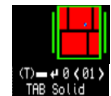
Delete TILE

1587 DEFine PROCEDURE TNum(ni)

```

1588 INK#8,CP(ni):CURSOR#8,44,2:PRINT#8,TAss(tn)
1589 TDraw 2 :CURSOR#8,62,2:PRINT#8,FILL$( '0',2-LEN(tn))&tn
1590 END DEFine

```



Note: Tile Status & Library number

1592 DEFine PROCEDURE Tile_Chg(tck)

```

1593 FOR r=0 TO 15
1594   FOR c=0 TO 15
1595     IF tck=1:Tile(td,c,r)=FG(fr,cs+c,rs+r)      :REMark Add New Tile
1596     IF tck=2:Tile(td+1,c,r)=Tile(td,c,r)        :REMark Insert Tile Space
1597     IF tck=3:Tile(td,c,r)=Tile(td+1,c,r)        :REMark Cut Tile from List
1598     IF tck=4:Tile(td,c,r)=255                   :REMark CLS Tile
1599   END FOR c
1600 END FOR r
1601 END DEFine

```

Note: tic TILE Check

1603 DEFine PROCEDURE HLObj(oi)

```

1604 INK#8,CP(oi):CURSOR#8,62,26:PRINT#8,/#
1605 END DEFine

```

(M) ops [#]

Note: ON/OFF # hash highlight for SWITCH - SLIDE

```

1607 DEFINE PROCEDURE Tile_Ass
1608 CURSOR#8,2,2:PRINT#8,'(T)':FrMes 8,20,2:atn=TAss(tn):atn=TAss(tn)
1609 REPEAT Ass_lp
1610   CURSOR#8,44,2:PRINT#8,atn:CURSOR#8,12,14:PRINT#8,'TAB ':TAS$(atn)
1611   k=CODE(INKEY$(-1))
1612   IF k=9 :atn=atn+1:IF atn>9:atn=0
1613   IF k=32:atn=TAss(tn):EXIT Ass_lp
1614   IF k=10:TAss(tn)=atn:EXIT Ass_lp
1615 END REPEAT Ass_lp
1616 CURSOR#8,44,2:PRINT#8,atn:BLOCK#8,80,12,2,14,0
1617 INK#8,CP(5):CURSOR#8,4,4:PRINT#8,'(T)ILE '
1618 END DEFINE

```



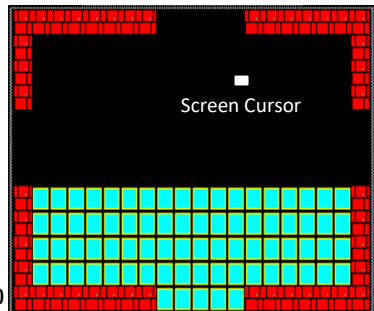
Note: 'TAB' through Assets :
Abort with Spacebar
Action with Enter

```

1620 DEFINE PROCEDURE Tile_Chk
1621   tn=TScn(0,x,y)
1622   IF TAss(tn)=0:x=ox:y=oy :REMark Solid Blocks Move
1623   IF TAss(tn)=1:x=x:y=y :REMark Floor No Action
1624   IF TAss(tn)=2:x=x:y=y :Hazard :REMark Hazard WIP Score / Lives
1625   IF TAss(tn)=3:x=x:y=y :Reward :REMark Reward WIP Score / Lives
1626   IF TAss(tn)=4:x=x:y=y :REMark ???
1627   IF TAss(tn)=5:x=x:y=y :REMark ???
1628   IF TAss(tn)=6:x=x:y=y :REMark ???
1629   IF TAss(tn)=7:x=x:y=y :REMark ???
1630 END DEFINE

```

Note: The Screen Background is a Matrix of 20x12 Tiles the **Tile_MAP** searches TScn array to identify occupied Cells by its Library **TILE** number and uses **TDraw** to display.



```

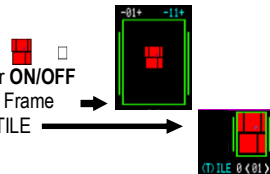
1632 DEFINE PROCEDURE Tile_MAP
1633 PAPER#3,CP(bg):CLS#3:Scn_Num sn
1634 FOR tr=0 TO 11
1635   FOR tc=0 TO 19
1636     TScn(0,tc,tr)=TScn(sn,tc,tr) MOD 64
1637     tn=TScn(0,tc,tr):IF tn>0:tx=2+tc*16:ty=1+tr*16:TDraw 0
1638   END FOR tc
1639 END FOR tr
1640 cur=0:rub=0:snp=sn:snw=TMap(snp,4):sne=TMap(snp,2):ec=0:wc=19
1641 END DEFINE

```

```

1643 DEFINE PROCEDURE TDraw(ver)
1644 LOCAL x,y
1645 IF ver=0:ch=3:w=1:h=1:x=bx:y=ty :REMark Paint cur ON/OFF
1646 IF ver=1:ch=4:w=1:h=1:x=26:y=25:CLS#4 :REMark SPRITE Frame
1647 IF ver=2:ch=7:w=2:h=2:x= 0:y= 0:CLS#7 :REMark Library TILE
1648 FOR r=0 TO 15:FOR c=0 TO 15:PDRAW:END FOR c:END FOR r
1649 END DEFINE

```



```

1651 DEFINE PROCEDURE PDraw
1652 pcol%=Tile(tn,c,r):IF pcol%=255:pcol%=bg
1653 BLOCK#ch,w,h,x+c*w,y+r*h,CP(pcol%)
1654 END DEFINE

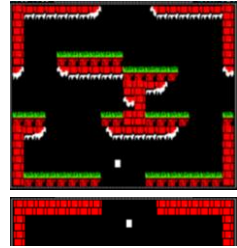
```

Note: Checks for Background. Fill Block at
TILE x y coordinates with TILE Colour

```

1656 Define PROCEDURE Scn_Switch
1657 ns=0:IF TScn(0,x,y)>0:Tile_Chk:END IF :REMark Check IF Move Blocked
1658 IF y=0 :ns =TMap(sn,1):IF ns>0:SEnt ns,1,11,18,11:x=c+1:y=10 :END IF
1659 IF x=xm:ns =TMap(sn,2):IF ns>0:SEnt ns,0, 1, 0,10 :x=1 :y=r+1 :END IF
1660 IF y=ym:ns =TMap(sn,3):IF ns>0:SEnt ns,1, 0,18, 0 :x=c+1:y=1 :END IF
1661 IF x=0 :ns =TMap(sn,4):IF ns>0:SEnt ns,19,1,19,10 :x=18 :y=r+1:END IF
1662 IF ns>0:sn=ns:Tile_MAP
1663 END Define

```



Note: See Maps for Links Set between Screens. Test by Pressing [#]

```

1665 Define PROCEDURE SEnt(ns,ax,by,cx,dy)
1666 FOR r=by TO dy:FOR c=ax TO cx:IF TScn(ns,c,r)=0:x=c:y=r:RETurn
1667 END Define

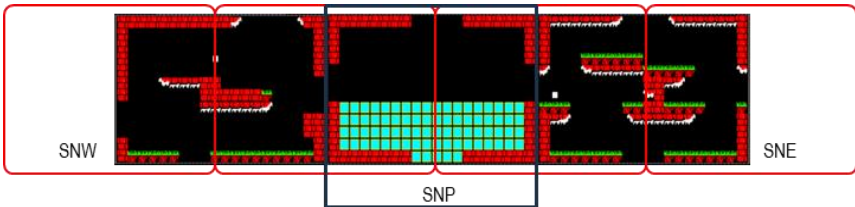
```

Note: This Option for Platform Games that use Sliding Screens and where Player's change in Levels are accomplished with a **Jump** action from pressing the **Spacebar**.

```

1669 Define PROCEDURE Scn_Slide
1670 IF TScn(0,x,y)>0:Tile_Chk:END IF :REMark Check IF Move Blocked
1671 IF x+1>xm:PAN#3,-16:x=ox:PANEast:ec=ec+1:wc=wc+1:IF wc>19:wc=19
1672 IF x-1< 0:PAN#3, 16:x=ox:PANWest:wc=wc-1:ec=ec-1:IF ec<0:ec=0
1673 END Define

```



```

1675 Define PROCEDURE PANEast
1676 IF ec>19:snp=sne:ec=0:wc=19:sne=TMap(snp,2):Scn_Num snp
1677 IF wc<19:sne=snp:ec=wc+1
1678 FOR tr=0 TO 11
1679 FOR tc=1 TO 19:TScn(0,tc-1,tr)=TScn(0,tc,tr):END FOR tc
1680 TScn(0,19,tr)=TScn(sne,ec,tr):tn=TScn(0,19,tr)
1681 IF tn>0:tx=2+19*16:ty=1+tr*16:TDRAW 0
1682 END FOR tr
1683 END Define

```

```

1685 Define PROCEDURE PANWest
1686 IF wc<0:snp=snw:wc=19:ec=0:snw=TMap(snp,4):Scn_Num snp
1687 IF ec>0:snw=snp:wc=ec-1
1688 FOR tr=0 TO 11
1689 FOR tc=18 TO 0 STEP -1:TScn(0,tc+1,tr)=TScn(0,tc,tr):END FOR tc
1690 TScn(0,0,tr)=TScn(snw,wc,tr):tn=TScn(0,0,tr)
1691 IF tn>0:tx=2:ty=1+tr*16:TDRAW 0
1692 END FOR tr
1693 END Define

```

1700 REMark MAPS SCREEN Links

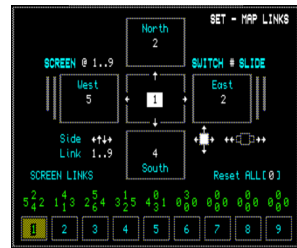
1702 DEFINE PROCEDURE Scn_MAP

```

1703 obg=bg:PAPER#3,0:CLS#3:QBold 3,7,224,4,'SET - MAP LINKS'
1704 QBold 3,5,34,40,'SCREEN':QBold 3,5,203,40,'SWITCH SLIDE'
1705FOR i=0 TO 1:BLOCK#3,68,40,52+i*152,54,CP(15):BLOCK#3,64,38,54+i*152,55,0
1706FOR i=4 TO 5:BLOCK#3,3,i*7,5+i*8,56,CP(15):BLOCK#3,3,i*7,317-i*8,56,CP(15)
1707 FOR i=0 TO 2:BLOCK#3,68,40,128,8+i*46,CP(15):BLOCK#3,64,38,130,9+i*46,0
1708 FOR i=1 TO 9
1709   mx=-27+i*35:BLOCK#3,30,20,mx,170,CP(15):BLOCK#3,28,18,mx+1,171,0
1710   CURSOR#3,-16+i*35,176:PRINT#3,i
1711 END FOR i
1712 RESTORE 1716:FOR i=1 TO 26:QPrnt
1713 BLOCK#3,11,10,212,102,CP(15):BLOCK#3,9,8,213,103,0
1714 BLOCK#3,21,6,254,104,CP(15):BLOCK#3,19,4,255,105,0
1715 BLOCK#3,11,10,259,102,CP(15):BLOCK#3,9,8,260,103,0
1716 DATA 3,7,214,94,'↑',3,7,214,110,'↓',3,7,205,102,'←' →'
1717 DATA 3,7,240,102,'←' →' →' →',3,7,246,40,'#'
1718 DATA 3,7,158,24,TMap(sn,1),3,7,158,114,TMap(sn,3)
1719 DATA 3,7,82,70,TMap(sn,4),3,7,236,70,TMap(sn,2)
1720 DATA 3,0,155,52,' ',3,7,126,69,'←' →',3,0,155,90,' '
1721 DATA 3,7,158,50,'↑',3,5,146,12,'North',3,5,226,58,'East'
1722 DATA 3,7,158,88,'↓',3,5,146,126,'South',3,5,72,58,'West'
1723 DATA 3,7,77,40,'@ 1..9',3,5,18,132,'SCREEN LINKS'
1724 DATA 3,5,52,102,'Side',3,7,88,102,'←' →' ↑ ↓'
1725 DATA 3,5,52,114,'Link',3,7,88,114,'1..9'
1726 DATA 3,5,228,132,'Reset ALL [0]',3,7,291,132,'0'
1727 BLOCK#8,40,10,38,26,0:FrMes 8,38,26:ScnTie:mx=158:my=70:sl=0
1728 END DEFINE

```

RETRO GAME
(O) ay CTRL
(M) ← →
(W) ser Chk



Map of SCREEN Connections

Note: Use @ and 1...9 to Select SCREEN. Use Cursor Keys to Select Side and 1..9 to set Link. A LINK will connect on the opposite side of SCREEN Chosen: ie WEST Side Links to the EAST Side of selected SCREEN.

1730 DEFINE PROCEDURE Scn_Links

```

1731 REPEAT Scn_Lp
1732   PrntScn 7,7,0:PrntMap:k=CODE(INKEY$(-1)):PrntScn 0,0,7
1733   SELECT ON k
1734     =35 :IF lks=0:lks=1:ScnTie:ELSE lks=0:ScnTie
1735     =39,64 :mx=158:my= 70:sl=0 :REMark @ Select Screen
1736     =192 :mx= 82:my= 70:sl=4:so=2 :REMark West - East Entry
1737     =200 :mx=236:my= 70:sl=2:so=4 :REMark East - West Entry
1738     =208 :mx=158:my= 24:sl=1:so=3 :REMark North - South Entry
1739     =216 :mx=158:my=114:sl=3:so=1 :REMark South - North Entry
1740     = 48 :DIM TMap(9,4):RESTORE 1718:FOR i=1 TO 4:QPrnt
1741       =49 TO 57:IF sl=0
1742         PrntCLS:sn=k-48:RESTORE 1718:FOR i=1 TO 4:QPrnt
1743         ELSE
1744           TMap(sn,sl)=k-48:TMap(k-48,so)=sn
1745         END IF
1746       =32,10:bg=obg:PAPER#3,CP(bg):CLS#3:EXIT Scn_Lp Note: Bg BackGnd obg Old BGnd Colour
1747   END SELECT
1748 END REPEAT Scn_Lp
1749 INK#8,CP(5):CURSOR#8,38,26:PRINT#8,'aps[#]':obj=0:Tile_MAP
1750 END DEFINE

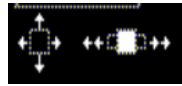
```

1752 **DEFINE PROCEDURE** ScnTie

1753 IF lks=0:BLOCK#3,9,8,213,103,CP(7):BLOCK#3,9,8,260,103,0

1754 IF lks=1:BLOCK#3,9,8,213,103,0:BLOCK#3,9,8,260,103,CP(7)

1755 **END DEFINE**

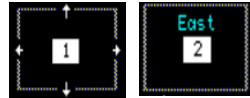


1757 **DEFINE PROCEDURE** PrntScn(bink,sink,pink)

1758 BLOCK#3,18,12,mx-6,my-1,CP(bink):STRIP#3,CP(sink):INK#3,CP(pink)

1759 CURSOR#3,mx,my:IF sl=0:PRINT#3,sn:ELSE PRINT#3,TMap(sn,sl)

1760 **END DEFINE**



1762 **DEFINE PROCEDURE** PrntMap

1763 STRIP#3,0:INK#3,CP(4)

1764 BLOCK#3,26,16,-25+sn*35,172,240:CURSOR#3,-16+sn*35,176:PRINT#3,sn

1765 FOR i=1 TO 9:CURSOR#3,-25+i*35,154:PRINT#3,TMap(i,4),' ',TMap(i,2)

1766 FOR i=1 TO 9:CURSOR#3,-16+i*35,148:PRINT#3,TMap(i,1)

1767 FOR i=1 TO 9:CURSOR#3,-16+i*35,158:PRINT#3,TMap(i,3)

1768 STRIP#3,2:INK#3,0

1769 **END DEFINE**



1771 **DEFINE PROCEDURE** PrntCLS

1772 BLOCK#3,26,16,-25+sn*35,172,0

1773 STRIP#3,0:INK#3,CP(5):CURSOR#3,-16+sn*35,176:PRINT#3,sn

1774 **END DEFINE**



1776 **REMARK** Key ConTRL Action Generator

1778 **DEFINE PROCEDURE** Init_Agen

Note: Screen Layout Action Generator

1779 obg=bg:PAPER#3,0:CLS#3:INK#3,CP(7): BLOCK#8,38,10,42,36,0:FrMes 8,38,36

1780 str\$="[]":INK#3,CP(5):FOR i=1 TO 7:CURSOR#3,73,44+i*10:PRINT#3,str\$

1781 str\$=Shift:INK#3,CP(7):FOR i=1 TO 5:CURSOR#3,41,56+i*9:PRINT#3,str\$(i)

1782 OPEN#12,scl:WINDOW#12,164,98,gx+222,gy+81:BORDER#12,1,CP(15)

1783 BLOCK#3,5,80,119,54,CP(2):BLOCK#3,5,80,296,54,CP(2)

1784 **QBold 3,5,6,20,'SPRITE':QBold 3,7,232,4,'SET - KEY CTRL'**

1785 **RESTORE 1741**:FOR i=1 TO 28:QPrnt:END FOR i

1786 DATA 3,5,36,4,'Edit (N)AME',3,5,47,20,'1.6',3,5,73,20,'[] Reset'

1787 DATA 3,7,109,20,'0',3,5,14,37,'Start Pos',3,5,20,52,'Frame',3,7,56,54,'@'

1788 DATA 3,7,109,20,'0',3,7,74,33,'x',3,7,96,34,'X',3,7,74,42,'y',3,7,96,44,'Y'

1789 DATA 3,7,56,63,'↑',3,7,56,73,'→',3,7,56,93,'←',3,7,56,103,'↓'

1790 DATA 3,5,20,113,'Exit',3,7,56,114,'#',3,5,14,144,'(A)ttack'

1791 DATA 3,5,20,127,'Gravity',3,7,73,123,'←',3,7,73,133,'↑',3,7,73,143,'↓'

1792 DATA 3,5,128,34,'X:00 Y:00 (T)est',3,7,190,34,'↑',3,7,190,44,'↓',3,7,190,54,'←',3,7,190,64,'→'

1793 DATA 3,4,20,172,'Frame @ [] SPRITE Actions Disabled',3,7,83,172,'00'

1794 DATA 3,4,12,182,'↑',3,7,12,192,'←',3,7,12,202,'↓',3,7,12,212,'→',3,7,12,222,'[1]Joystick[2] F1-F5 Hazard[3-6]Reward'

1795 BLOCK#3,12,3,54,87,CP(7):BLOCK#3,12,3,28,186,CP(4)

1796 BLOCK#3,16,3,207,38,CP(7):BLOCK#3, 2,4,244,36,CP(7)

1797 DIM Actn\$(1,5):Actn\$(0)=Jump:Actn\$(1)=Fire:Snum=10:LNum=4:osn=sn

1798 ka=1:kb=0:kx=0:kxm=176:ky=0:kym=96:k7=0:k8=0:k9=0:k10=0:k11=0

1799 **END DEFINE**



Note: Settings for Action SPRITE 1 & 2 Player and Program 3 .6 actions [Gravity Controlled].

```

1801 DEFine PROCEDURE AGen
1802 IF fr<1:BMLoad:END IF :Init_AGen:Set_Score:INK#3,CP(7):SP_Chg:SP_Actn
1803 REPEAT CTRL_LP
1804 SP_Type:SP_Actn:IF kb>3 AND kb<16:SP_Prt ka,kb      Note: Action Generator Menu
1805 k$=(INKEY$(-1)):k=CODE(k$)
1806 SELECT ON k
1807 =84,116 :IF SK(ka,6)>0:CTRL_Test:CUSOR#3,247,34:PRINT#3,'(T)est'
1808 =78,110 :Chg_Title :REMark Edit (N)ame
1809 =48 :FOR i=1 TO 20:SK(ka,i)=0:END FOR i:SP_Chg :REMark [0] Reset
1810 =49 TO 54 :ka=k-48:SP_Chg :REMark Sprite 1..6
1811 =67,99 :IF ka<3:SK(ka,1)=1 :REMark (C)onTRL Sprite
1812 =72,104 :IF ka>2:SK(ka,1)=2:SP_Score :REMark (H)azard Sprite
1813 =82,114 :IF ka>2:SK(ka,1)=3:SP_Score :REMark (R)eward Sprite
1814 =45,95 :IF fr> 0:fr=fr-1:FrChk :REMark - Frame
1815 =43,61 :IF fr<fr:fr=fr+1:FrChk :REMark + Frame
1816 =120 :IF SK(ka,4)>0 :SK(ka,4)=SK(ka,4)-1:kb=4 :REMark x - Pos
1817 =88 :IF SK(ka,4)<kxm/4:SK(ka,4)=SK(ka,4)+1:kb=4 :REMark X + Pos
1818 =121 :IF SK(ka,5)>0 :SK(ka,5)=SK(ka,5)-1:kb=5 :REMark y - Pos
1819 =89 :IF SK(ka,5)<kym/4:SK(ka,5)=SK(ka,5)+1:kb=5 :REMark Y + Pos
1820 =39,64 :SK(ka,6)=fr:kb=6 :REMark @ Default Frame
1821 =212 :IF ka<6:SK(ka,7)=fr:kb=7 :REMark Shift UP Sprite
1822 =204 :IF ka<6:SK(ka,8)=fr:kb=8 :REMark Shift Right Facing Sprite
1823 =252 :IF ka<6:SK(ka,9)=fr:kb=9 :REMark Shift SB Jump/Fire Sprite
1824 =196 :IF ka<6:SK(ka,10)=fr:kb=10 :REMark Shift Left Facing Sprite
1825 =220 :IF ka<6:SK(ka,11)=fr:kb=11 :REMark Shift Down Sprite
1826 =35 :IF ka<6:SK(ka,12)=fr:kb=12 :REMark # Clear
1827 =192 :IF ka<6 AND SK(ka,13)>8:SK(ka,13)=SK(ka,13)-1:kb=13 :REMark ←
1828 =200 :IF ka<6 AND SK(ka,13)<8:SK(ka,13)=SK(ka,13)+1:kb=13 :REMark →
1829 =208 :IF ka<6 AND SK(ka,14)<8:SK(ka,14)=SK(ka,14)+1:kb=14 :REMark ↑
1830 =216 :IF ka<6 AND SK(ka,14)>8:SK(ka,14)=SK(ka,14)-1:kb=14 :REMark ↓
1831 =65,97 :IF ka<3:kb=15:SP_Toggle :REMark (A)ction
1832 =66,98 :IF ka<6:kb=16:SP_Toggle :REMark (B)ounce 0<>1
1833 =69,101 :IF ka<6:SK(ka,16)=2 :REMark (E)xplode
1834 =83,115 :kb=18:SP_Toggle :REMark (S)ound
1835 =10,32:CLOSE#12:bg=obg:PAPER#3,CP(bg):CLS#5:EXIT CTRL_LP
1836 END SELECT
1837 END REPEAT CTRL_LP
1838 INK#8,CP(5):CURSOR#8,38,36:PRINT#8,'ey CTRL':CLS#3:sn=osn:Tile_MAP
1839 END DEFine

```

```

1841 DEFine PROCEDURE SP_Type
1842 INK#3,CP(5):CURSOR#3,160,20:PRINT#3,'JoyStick (H)azard (R)eward'
1843 CURSOR#3,124,144:PRINT#3,'(B)ounce (E)xplode (S)ound':INK#3,CP(7)
1844 IF ka=1 OR ka=2:CURSOR#3,160,20:PRINT#3,'JoyStick':SK(1,1)=1:SK(2,1)=1
1845 IF SK(ka,1)=2 :CURSOR#3,214,20:PRINT#3,'(H)azard'
1846 IF SK(ka,1)=3 :CURSOR#3,268,20:PRINT#3,'(R)eward'
1847 IF SK(ka,16)=1 :CURSOR#3,124,144:PRINT#3,'(B)ounce'
1848 IF SK(ka,16)=2 :CURSOR#3,184,144:PRINT#3,'(E)xplode'
1849 IF SK(ka,18)=1 :CURSOR#3,250,144:PRINT#3,'(S)ound'
1850 END DEFine

```

```

1752 DEFine PROCEDURE SP_Actn

```

```

1753 CURSOR#3,74,144:PRINT#3,Actn$(SK(ka,15))

```

```

1754 END DEFine

```

Note: Press 'A' to Toggle (A)ction

= 0	= 1
(A)ttack Jump	(A)ttack Fire


```

1856 DEFINE PROCEDURE SP_Prnt(ka,kb)
1857 CURSOR#3,82,-6+kb*10:PRINT#3,FILL$(0',2-LEN(SK(ka,kb)))&SK(ka,kb)
1858 END DEFINE

```

```

1860 DEFINE PROCEDURE Chg_Title
1861 CLS#5:EditName 5,0,20,12,13,RGN$:Snum=0:LNum=4:Set_Score
1862 END DEFINE

```



```

1864 DEFINE PROCEDURE SP_Chg
1865 INK#3,CP(7):CURSOR#3,88,20:PRINT#3,ka:SP_Type:SP_Actn
1866 FOR i= 4 TO 14:SP_Prnt ka,i:END FOR i:kb=4
1867 END DEFINE

```

```

1869 DEFINE PROCEDURE SP_Score
1870 BLOCK#5,100,30,0,0,0:Snum=0:LNum=4
1871 CURSOR#5,46,6:PRINT#5,'Set ↓ ↑':CURSOR#5,58,20:PRINT#5,'Set ← →'
1872 REPEAT Score_ip
1873 CURSOR#5,80, 6:PRINT#5,FILL$( ' ',4-LEN(Snum))&Snum
1874 CURSOR#5,92,20:PRINT#5,FILL$( ' ',2-LEN(LNum))&LNum:k=CODE(INKEY$(1))
1875 SELECT ON k
1876 =192:IF LNum>-4:LNum=LNum-1
1877 =200:IF LNum< 4:LNum=LNum+1
1878 =208:IF Snum< 100:Snum=Snum+10
1879 =216:IF Snum>-100:Snum=Snum-10
1880 = 32:Set_Score:EXIT Score_ip
1881 = 10:SK(ka,2)=Snum:SK(ka,3)=LNum:Set_Score:EXIT Score_ip
1882 END SELECT
1883 END REPEAT Score_ip
1884 END DEFINE

```



```

1886 DEFINE PROCEDURE Set_Score
1887 CLS#5:QTitle 5,1,1,1,RGN$:Snum=0:LNum=4
1888 QBold 5,7,118,5,'SCORE':QBold 5,7,118,19,'LIVES': Chg_Score
1889 END DEFINE

```



```

1891 DEFINE PROCEDURE Chg_Score
1892 CURSOR#5,160,1:CSIZE#5,1,1
1893 IF Snum>-1 AND Snum<9999:PRINT#5,FILL$(0',4-LEN(Snum))&Snum:ELSE Game_End
1894 CURSOR#5,160,1:CSIZE#5,0,0:IF LNum<1:Game_End:END IF: IF LNum>4:RETURN
1895 F BLOCK#5,32,8,160,20,0:OR i=1 TO LNum:CURSOR#5,152+i*8,20:PRINT#5,'#';
1896 END DEFINE

```

```

1898 DEFINE PROCEDURE SP_Toggle
1899 IF SK(ka,kb)=0:SK(ka,kb)=1:ELSE SK(ka,kb)=0:END IF :kb=4
1900 END DEFINE

```

```

1902 DEFINE PROCEDURE SP_BEEP(bs)
1903 SELECT ON bs
1904 =1:BEEP 0,1,2,3,4,5,6,7 :REMark d,p,h,t,s,w,f,r
1905 =2:BEEP 4000,0,200,2,3,0,0,0
1906 =3:BEEP 15000,12,48,30,-8,15,15,15
1907 =4:BEEP 5000,32,12,8,0,0,0,0
1908 END SELECT
1909 END DEFINE

```

```

1911 DEFine PROCEDURE CTRL_Test
1912 INK#3,CP(7):CURSOR#3,242,34:PRINT#3,'(Esc)  ':Test_Scn
1913 chk=0:Snm=0:LNum=4:Score_chg:kb=6:tsz=4:tw=8:th=8:sw=cm:sh=rm
1914 sx=SK(ka,4)*4 :sy=SK(ka,5)*4:osx=sx:osy=sy:kxm=160:kym=96
1915 xx=SK(ka,13) :IF xx>4:xx=4:END IF :IF xx<4:xx=-4:END IF
1916 yy=SK(ka,14) :IF yy>4:yy=4:END IF :IF yy<4:yy=-4:END IF
1917 IF SK(ka,1)=1 OR SK(ka,1)=2:ix=2:xx=0:iy=1:yy=0 :END IF
1918 IF SK(ka,15)=1:Sk(ka,16)=0:Sp_Type:END IF
1919 REPEAT Test_Ip
1920 SP_Draw sx,sy:PAUSE 5:SP_Draw osx,osy
1921 IF ka=1 OR ka=2:Ctrl_Moves:END IF :Chk_Move
1922 IF KEYROW(1)=8:CLS#12:INK#3,CP(5):EXIT Test_Ip
1923 END REPEAT T_ip
1924 END DEFINE

1926 DEFine PROCEDURE SP_Draw(px,py)
1927 OVER#12,-1:ch=12:fr=SK(ka,kb)
1928 FOR r=rs TO rs+rm-1:FOR c=cs TO cs+cn-1:PBit:END FOR c:END FOR r
1929 OVER#12,0
1930 END DEFINE

1932 DEFine PROCEDURE PBit
1933 IF FG(fr,c,r)=255:pcol%=CP(bg):ELSE pcol%=CP(FG(fr,c,r))
1934 BLOCK#ch,1,1,px+c,py+r,pcol%
1935 END DEFINE

1937 DEFine PROCEDURE Ctrl_Moves
1938 k=KEYROW(1):tmx=xx:tmy=yy:IF SK(ka,15)=0:SElect ON k=4,6,20:RETurn
1939 SElect ON k
1940 = 32:IF SK(ka,15)=0:SP_Jump:ELSE SP_Fire
1941 =128:yy=yy+iy :cr= 0:rf=+1 ↓ :REMark Down
1942 =144:yy=yy+iy:xx=xx+ix:cf=+1:rf=+1 ↘ :REMark Down/Right
1943 = 16:xx=xx+ix :cf=+1:rf= 0 → :REMark Right
1944 = 20:yy=yy-iy:xx=xx+ix :cf=+1:rf= -1 ↗ :REMark Up/Right
1945 = 4:yy=yy-iy :cf= 0:rf= -1 ↑ :REMark Up
1946 = 6:yy=yy-iy:xx=xx-ix :cf= -1:rf= -1 ↙ :REMark UP/Left
1947 = 2:xx=xx-ix :cf= -1:rf= 0 ← :REMark Left
1948 =130:yy=yy+iy:xx=xx-ix :cf= -1:rf=+1 ↖ :REMark Down/Left
1949 END SElect
1950 IF xx>4 OR xx<-4:xx=tmx:END IF :IF yy>4 OR yy<-4:yy=tmy:END IF
1951 END DEFINE

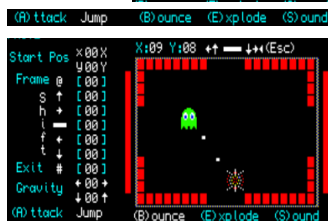
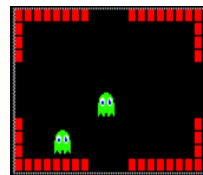
```

Note Speed <= Tile width
Speed <= Tile height

Note [Esc Key] KEYROW(1)=8

Note: ■ SpaceBar Actions

Note Max Set for CTRL Test



```

1968 DEFine PROCEDURE Chk_Move
1969 IF osx+xx< 0 :osx=160-sw:sy=40:PAUSE 10
1970 IF osx+xx>160-sw :osx= 0 :sy=40:PAUSE 10
1971 IF osy+yy< 0 :osy= 96-sh :sx=70:PAUSE 10
1972 IF osy+yy> 96-sh :osy= 0 :sx=70:PAUSE 10
1973 tc=osx DIV tw:tr=(osy+yy) DIV th:Chk_Hit:IF chk=1:YChg:chk=0
1974 tc=(osx+xx) DIV tw:tr=osy DIV th:Chk_Hit:IF chk=1:XChg:chk=0
1975 CURSOR#3,140,34:PRINT#3,FILL$('0',2-LEN(tc));tc
1976 CURSOR#3,170,34:PRINT#3,FILL$('0',2-LEN(tr));tr
1977 sx=osx+xx:sy=osy+yy:osx=sx:osy=sy
1978 END DEFine

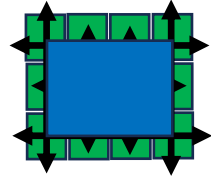
```

```

1980 DEFine FunCedure Chk_Hit
1981 FOR row=0 TO sh DIV th
1982   FOR col=0 TO sw DIV tw
1983     IF TTile(tc+col,tr+row)>0:chk=1:RETurn
1984   END FOR col
1985 END FOR row
1986 END DEFine

```

Note: sh screen th Tile height
sw Screen tw Tile width



```

1988 DEFine PROCEDURE XChg
1989 IF SK(ka,16)=0:xx=0:yy=0 :IF SK(ka,18)=1:SP_BEEP 2 :REMark Stop
1990 IF SK(ka,16)=1:xx=xx :IF SK(ka,18)=1:SP_BEEP 2 :REMark Bounce
1991 IF SK(ka,16)=2:SP_Ex :IF SK(ka,1)<5:xx=0:yy=0 :REMark Explode
1992 END DEFine

```

```

1994 DEFine PROCEDURE YChg
1995 IF SK(ka,16)=0:xx=0:yy=0 :IF SK(ka,18)=1:SP_BEEP 2
1996 IF SK(ka,16)=1:yy=yy :IF SK(ka,18)=1:SP_BEEP 2
1997 IF SK(ka,16)=2:SP_Ex :IF SK(ka,1)<5:xx=0:yy=0
1998 END DEFine

```

```

2000 DEFine PROCEDURE SP_Ex
2001 IF SK(ka,18)=1:SP_BEEP 3
2002 kb=12:FOR i=1 TO 6:SP_Draw osx,osy:PAUSE 5:SP_Draw osx,osy:PAUSE 5
2003 sx=SK(ka,4)*tw:sy=SK(ka,5)*th:xx=0:yy=0:kb=6:BEEP
2004 Snum=Snum+SK(ka,2)::LNum=LNum+SK(ka,3):Chg_Score
2005 END DEFine

```



```

2007 DEFine PROCEDURE Game_End
2008 CLS#12:INK#12,7:CSIZE#12,1,1
2009 CURSOR#12,48,20:PRINT#12,'GAME END':CSIZE#12,0,0:PAUSE:chk=1
2010 END DEFine

```

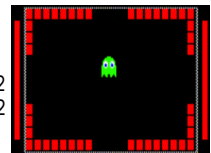


```

2012 DEFine PROCEDURE Test_Scn
2013 LOCAL tc,tr :DIM TTile(20,12)
2014 FOR tr=0 TO 11
2015   FOR tc=0 TO 19
2016     IF tr = 0 OR tr = 11:IF tc<8 OR tc>11:TTile(tc,tr)=1:BLOCK#12,6,7,tc*8,tr*8,2
2017     IF tc=0 OR tc = 19:IF tr<4 OR tr > 7:TTile(tc,tr)=1:BLOCK#12,6,7,tc*8,tr*8,2
2018   END FOR tc
2019 END FOR tr
2020 END DEFine

```

Note: Test Screen Border Tile display



2050 REMark File Management

2052 **DEFine PROCEDURE SelDrv(act%,Act\$)**

2053 INK#5,CP(7):CURSOR#5,6,20:PRINT#5,'Select ↑↓←→';BLOCK#5,2,4,78,22,CP(7)

2054 IF act%=1:PRINT#5,'(E)dit':END IF :INK CP(5)

2055 **REPEAT drv_ip**

2056 CURSOR#5,6,11:PRINT#5,Act&drv\$(dn%):CLS#5,4

2057 k=CODE(INKEY\$(-1))

2058 **SElect ON k**

2059 =208:dn%=dn% -1:IF dn%<0:dn%=dm%

2060 =216:dn%=dn%+1:IF dn%>dm%:dn%=1

2061 =101,69:IF act%=1:**EditName 5,act%,42,11,16,drv\$(dn%)**

```
(F)DIR (L)oad (S)ave (Q)uit
Load win1_
Select ↑ ↓ ← →
```

```
(F)DIR (L)oad (S)ave (Q)uit
Drive dos1_
Select ↑ ↓ ← → (E)dit +[F]→=
```

:REMark Drive/SubDIR

2062 = 10:**EXIT drv_ip**

2063 **END SElect**

2064 **END REPEAT drv_ip**

2065 **END DEFine**

2067 **DEFine PROCEDURE Selfie**

2068 CURSOR#5,6,20:PRINT#5,'Select ↑↓':INK#5,CP(5):**FrMes 5,78,20**

2069 **REPEAT FSel_ip**

2070 CURSOR#5,6,11:PRINT#5,'Load 'File\$(sf%):CLS#5,4

2071 k=CODE(INKEY\$(-1))

2072 **SElect ON k**

2073 =208:sf%=sf% -1:IF sf%<1:sf%=ft% :REMark Up

2074 =216:sf%=sf%+1:IF sf%>ft%:sf%=1 :REMark Dn

2075 =110,78,32:chk=0:**EXIT FSel_ip** :REMark Y/N Spacebar Abort chk=0

2076 =121,89,10:chk=1:**EXIT FSel_ip** :REMark Y/N Enter Action chk=1

2077 **END SElect**

2078 **END REPEAT FSel_ip**

2079 **END DEFine**

```
(F)DIR (L)oad (S)ave (Q)uit
Load PIXELArt05_bmp
Select ↑ ↓ ← →
```

2081 **DEFine PROCEDURE DIR_File(typ\$)**

Note: typ\$ ' _bmp' or ' _fnt'

2082 **DIR_List**:CURSOR#5,6,20:PRINT#5,'Checking...':CLS#5,4:PAUSE 20

2083 chk=1:OPEN_IN#9,drv\$(dn%)&'FList':dl%=LEN(drv\$(dn%))

2084 **REPEAT dir_ip**

2085 IF EOF(#9) OR sf%>fm%:sf%=sf% -1:ft%=sf%:CLOSE#9:**EXIT dir_ip**

2086 INPUT#9,F\$:F\$=F\$(dl%-4 TO):fl%=LEN(F\$)

2087 IF fl%<=20 AND typ\$ INSTR F\$>0:File\$(sf%)=F\$:sf%=sf%+1

2089 **END REPEAT dir_ip**

2090 IF ft%<1:chk=0:CURSOR#5,6,11:PRINT#5,'No Files Found...':PAUSE 30

2091 **END DEFine**

```
(F)DIR (L)oad (S)ave (Q)uit
No Files Found...
Checking...
```

2093 **DEFine PROCEDURE Dir_List**

2094 DELETE drv\$(dn%)&'FList':OPEN_NEW#9,drv\$(dn%)&'FList':DIR#9,drv\$(dn%):CLOSE#9

2095 **END DEFine**

2097 **DEFine PROCEDURE Dir_Chk**

2098 OPEN_IN#9,drv\$(dn%)&'FList':dl%=LEN(drv\$(dn%))

2099 **REPEAT dir_ip**

2100 IF EOF(#9):CLOSE#9:chk=0:**EXIT dir_ip**

2101 INPUT#9,F\$:F\$=F\$(dl%-4 TO)

2102 IF F\$==SFile\$:CLOSE#9:chk=1:**EXIT dir_ip**

2103 **END REPEAT dir_ip**

2104 **END DEFine**

2106 **DEFine PROCEDURE BMLoad**

2107 **SeI**Drv 2,'Load ':LNK CP(7):sf%=1:ft%=0:fm%=20

2108 **DIR_File** ' _bmp':IF chk=0:**RETurn**:END IF :**Self**File:IF chk=0:**Return**:END IF

2109 BLOCK#5,190,20,2,10,0:CURSOR#5,6,11:PRINT#5,'Loading.':CLS#5,4:ID\$="

2110 OPEN _In#9,drv\$(dn%)&File\$(sf%):ID\$=INKEY\$(#9)&INKEY\$(#9)&INKEY\$(#9)

2111 IF ' _fnt' INSTR File\$(sf%)

2112 sg%=CODE(ID\$(1)):cg%=CODE(ID\$(2)):CLOSE#9

2113 LBYES drv\$(dn%)&File\$(sf%),Fntaddr:GFrame

2114 END IF

2115 IF ID\$='QL8' OR ID\$='PAL'

2116 bm=CODE(INKEY\$(#9)):cm=CODE(INKEY\$(#9)):rm=CODE(INKEY\$(#9))

2117 bg=CODE(INKEY\$(#9)):sz=CODE(INKEY\$(#9)):CLOSE#9

2118 IF sz=0:frt=bm:base=**ALCHP(8+frt*cm*rm):ptr=base+8** :PFile\$=File\$(sf%)

2119 IF sz=1:frt=bm:base=**ALCHP(8+8+frt*cm*rm):ptr=base+16** :PFile\$=File\$(sf%)

2120 IF sz=2:tm=bm:base=**ALCHP(8+8+36+2160+tm*256)** :TFile\$=File\$(sf%)

2121 IF sz=3:tm=bm:base=**ALCHP(8+8+36+2160+200+tm*256+bm)** :TFile\$=File\$(sf%)

2122 LBYES drv\$(dn%)&File\$(sf%),base

2123 IF sz=2:**SnLoad base+16,base+52:ptr=base+2212**

2124 IF sz=3:**SnLoad base+16,base+52:AnLoad base+2212:ptr=base+2412**

2125 FOR i=8 TO 15:BLOCK#8,8,6,6,36+10*i,CP(i)

2126 OVER#5,1:**FrLoad bm,cm,rm,ptr**:OVER#5,0

2127 **RECHP base**:IF sz<2:fr=1:**GFrame**:ELSE tn=1:sn=1:**GTile**

2128 END IF

2129 **END DEFine**

2131 **DEFine PROCEDURE SnLoad(ptr1,ptr2)**

2132 FOR sn=1 TO 9

2133 FOR sd=1 TO 4:TMap(sn,sd)=PEEK(ptr1):ptr1=ptr1+1:END FOR sd

2134 FOR tr=0 TO 11

2135 FOR tc=0 TO 19:TScn(sn,tc,tr)=PEEK(ptr2):ptr2=ptr2+1:END FOR tc

2136 END FOR tr

2137 END FOR sn

2138 **END DEFine**

2140 **DEFine PROCEDURE AnLoad(ptr4)**

2141 FOR i=1 TO 13:RGN\$(i)=CHR\$(PEEK(ptr4)):ptr4=ptr4+1:END FOR i:ptr4=ptr4+1

2142 FOR ka=1 TO 9

2143 FOR kb=1 TO 20:SK(ka,kb)=PEEK(ptr4):ptr4=ptr4+1:END FOR kb

2144 END FOR ka

2145 **END DEFine**

2147 **DEFine PROCEDURE FrLoad(bm%,cm%,rm%,ptr)**

2148 IF sz<2:DIM FG(32,64,64)

2149 IF sz>0:FOR i=8 TO 15:CP(i)=PEEK(base+i)

2150 FOR f=1 TO bm%

2151 IF f<48:CURSOR#5,48+f*3,11:PRINT#5,'.':PAUSE 1

2152 FOR r=0 TO rm%-1

2153 FOR c=0 TO cm%-1

2154 IF sz<2:FG(f,c,r)=PEEK(ptr):END IF

2155 IF sz>1:Tile(f,c,r)=PEEK(ptr):END IF

2156 IF frt=0:FG(f,c,r)=PEEK(ptr):END IF :ptr=ptr+1

2157 END FOR c

2158 END FOR r

2159 END FOR f

2160 IF sz=3:FOR f=1 TO bm%:TAss(f)=PEEK(ptr):ptr=ptr+1:END FOR f

2161 **END DEFine**

(F)DIR (L)oad (S)ave (W)uit
Load win1_
Select ↑ ↓ ↵

(F)DIR (L)oad (S)ave (W)uit
Loading

2163 **DEFine PROCEDURE BMSave**

2164 IF frt>0 OR tm>0:chk=0:eck=0:**SeIDrv 2,'Save ':ELSE RETURN**

2165 IF sz=1:SFile\$=PFile\$:ELSE SFile\$=TFile\$:END IF

2166 IF SFile\$="":SFile\$=QBDefault_bmp:END IF

2167 IF LEN(SFile\$)>20:SFile\$=SFile\$(1 TO 16) & '_bmp'

2168 INK CP(7):CURSOR#5,6,20:PRINT#5,'Action (E)dit':**FrMes 5,48,20**

2169 **REPEAT chk_lp**

2170 CURSOR#5,42,11:PRINT#5,SFile\$:FILL\$(' ',20-LEN(SFile\$)):k=CODE(INKEY\$(-1))

2171 **SElect ON k=101,69:EditName 5,2,42,11,20,SFile\$**

2172 **SElect ON k=10:Dir_List:Dir_Chk:EXIT chk_lp**

2173 **SElect ON k=32:BLOCK#5,190,20,2,10,0:RETURN**

2174 **END REPEAT chk_lp**

2175 CURSOR#5,6,20

2176 IF eck=1:PRINT#5,'DEVICE ERROR...':CLS#5,4:PAUSE 30:eck=0:**RETURN**

2177 IF chk=1:PRINT#5,'Overwrite':CLS#5,4:**FrMes 5,72,20:IF GAns=0:RETURN**

2178 DELETE drv\$(dn%)&SFile\$:BLOCK#5,190,20,2,10,0

2179 CURSOR#5,6,11:PRINT#5,'Saving':CLS#5,4

2180 IF sz=1:bm=frt:mlth=16+bm*cm*rm :addr=**ALCHP(mlth):ptr=addr+16**

2181 IF sz=3:bm=tm :mlth=52+2160+200+bm*256+bm:addr=**ALCHP(mlth):cm=16:rm=16**

2182 FOR i=0 TO 2:POKE addr+i,CODE(pm\$(i+1)):END FOR i

2183 POKE addr+3,bm:POKE addr+4,cm:POKE addr+5,rm:POKE addr+6,bg:POKE addr+7,sz

2184 IF sz=3:**SnSave addr+16,addr+52:AnSave addr+2212:ptr=addr+2412**

2185 OVER#5,1:**FrSave bm,cm,cm,ptr:OVER#5,0**

2186 SBYTES drv\$(dn%)&SFile\$,addr,mlth:**RECHP addr**

2187 **END DEFine**

```
(F)DIR (L)oad (S)ave (Q)uit  
Save dos1_  
Select ↑ ↓
```

```
(F)DIR (L)oad (S)ave (Q)uit  
Save QBPI\QGen3_01  
Action ← → (E)dit ← ↑ ↓ →
```

```
(F)DIR (L)oad (S)ave (Q)uit  
Save QBPI\QGen4_02_bmp  
DEVICE ERROR...
```

```
(F)DIR (L)oad (S)ave (Q)uit  
Save QBPI\QGen4_02_bmp  
Overwrite ← →
```

```
(F)DIR (L)oad (S)ave (Q)uit  
Saving.....
```

2189 **DEFine PROCEDURE SnSave(ptr2,ptr3)**

2190 FOR sn=1 TO 9

2191 FOR sd=1 TO 4:POKE ptr2,TMap(sn,sd):ptr2=ptr2+1:END FOR sd

2192 FOR tr=0 TO 11

2193 FOR tc=0 TO 19:POKE ptr3,TScn(sn,tc,tr):ptr3=ptr3+1:END FOR tc

2194 END FOR tr

2195 END FOR sn

2196 **END DEFine**

2198 **DEFine PROCEDURE AnSave (ptr4)**

2199 FOR i=1 TO 13:POKE ptr4,CODE(RGN\$(i)):ptr4=ptr4+1:END FOR i

2200 FOR ka=1 TO 9

2201 FOR kb=1 TO 20: ptr4=ptr4+1:POKE ptr4,SK(ka,kb): END FOR kb

2202 END FOR ka

2203 **END DEFine**

2205 **DEFine PROCEDURE FrSave(bm%,cm%,rm%,ptr)**

2206 IF sz>0:FOR i=8 TO 15:POKE addr+i,CP(i)

2207 FOR f=1 TO bm%

2208 IF f<48:CURSOR#5,42+f*3,11:PRINT#5,'.':PAUSE 1

2209 FOR r=0 TO rm%-1

2210 FOR c=0 TO cm%-1

2211 IF sz=1:POKE ptr,FG(f,c,r):END IF

2212 IF sz=3:POKE ptr,Tile(f,c,r):END IF :ptr=ptr+1

2213 END FOR c

2214 END FOR r

2215 END FOR f

2216 IF sz=3:FOR f=1 TO bm%:POKE ptr,TAss(f):ptr=ptr+1:END FOR f

2217 **END DEFine**

```

2222 DEFine PROCEDURE EditName(ch%,act%,sx%,sy%,sm%,str$)
2223 IF act%=2:sl%=(' _bmp' INSTR str$)-1:str$=str$(1 TO sl%)
2224 INK#ch%,CP(7):CURSOR#ch%,124,20:PRINT#ch%, '◀ [ ] ▶'
2225 BLOCK#ch%,10,3,156,24,CP(7):BLOCK#ch%,2,4,172,22,CP(7)
2226 temp$=str$:sl%=LEN(str$):cp%=sl%+1
2227 REPEAT Ed_ip
2228   Ln_Prn:Ln_Cur:k$=INKEY$(#0,-1):k=CODE(k$)
2229   SElect ON k
2230     = 10:IF sl%=0:str$=temp$:END IF :EXIT Ed_ip
2231     = 32:str$=temp$:EXIT Ed_ip
2232     = 48 TO 57,65 TO 90,95, 97 TO 122:Add_chr
2233     =194:IF cp%>1:cp%=cp%-1:Del_chr
2234     =202:Del_chr
2235     =192:IF cp%>1:cp%=cp%-1
2236     =200:IF cp%<sl%+1:cp%=cp%+1
2237   END SElect
2238 END REPEAT Ed_ip
2239 IF act%=1
2240   IF sl%<5:str$=temp$:RETURN
2241   IF str$(sl%)<>'_':IF sl%<sm%:str$=str$&'_' :ELSE str$(sl%)='_'
2242   IF str$(5)<>'_':str$(5)='_'
2243 END IF
2244 k=0:BLOCK#ch%,60,10,122,20,0:IF act%=2:str$=str$&' _bmp'
2245 END DEFine

```

```

(F)DIR (L)oad (S)ave (W)uit
Drive dos1_
Select ↑ ↓ (E)dit +[ ]+←→

```

```

(F)DIR (L)oad (S)ave (W)uit
Save 0BPIXGGen3_01
Action ←→ (E)dit +[ ]+←→

```

```

2247 DEFine PROCEDURE Ln_Prn
2248 IF LEN(str$)>sm%:str$=str$(1 TO sm%):cp%=sm%
2249 INK#ch%,5:CURSOR#ch%,sx%,sy%:PRINT#ch%,str$:CLS#ch%,4
2250 END DEFine

```

```

2252 DEFine PROCEDURE Ln_Cur
2253 BLOCK#ch%,6,1,sx%+cp%*6-6,sy%+8,CP(7)
2254 END DEFine

```

```

2256 DEFine PROCEDURE Add_chr
2257 IF cp% =1 AND sl%=0 :str$=str$&k$
2258 IF cp%>=1 AND cp%<sl%:str$=str$(1 TO cp%-1)&k$&str$(cp% TO sl%)
2259 IF cp%>=1 AND cp%=sl%:str$=str$(1 TO cp%-1)&k$&str$(cp%)
2260 IF cp%> 1 AND cp%>sl%:str$=str$&k$
2261 IF cp%=sm%:str$(cp%)=k$
2262 IF sl%<sm%:sl%=sl%+1:ELSE sl%=sm%
2263 IF cp%<sm%:cp%=cp%+1:ELSE cp%=sm%
2264 END DEFine


```

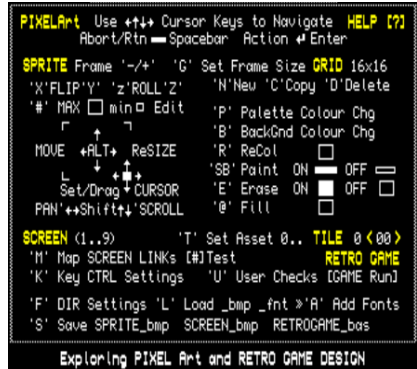
```

2266 DEFine PROCEDURE Del_chr
2267 IF cp%=sl%:str$=str$(1 TO sl%-1):sl%=sl%-1
2268 IF cp%>=1 AND cp%<sl%:str$=str$(1 TO cp%-1)&str$(cp%+1 TO sl%):sl%=sl%-1
2269 IF cp%=sm%:str$=str$(1 TO sm%-1):cp%=cp%-1:sl%=sm%-1
2270 IF cp%=1 AND sl%=1:str$="":sl%=0
2271 END DEFine

```

2300 REMark PIXArt Help Screen

Note: To activate a key function, Press the character identified by brackets ie (G)RID or [#]. The Key will change colour and be shown ie. (G)  Press the Spacebar to Abort or Enter to Action.



2302 DEFine PROCEDURE Info

```

2303 LOCAL ik%,ix%,iy%,ip$ :CLS#3:CSIZE#3,0,0
2304 QBold 3,6,6, 4,'PIXELArt' :QBold 3,6,270, 4,'HELP [?]'
2305 QBold 3,6,8, 29,'SPRITE' :QBold 3,6,242, 29,'GRID'
2306 QBold 3,6,8,129,'SCREEN' :QBold 3,6,242,129,'TILE < >'
2307 QBold 3,6,252,141,'RETRO GAME'
2308 OVER#3,1:INK#3,CP(7):RESTORE 2310
2309 FOR i=1 TO 28:READ ix%,iy%,ip$:CURSOR#3,ix%,iy%:PRINT#3,ip$
2310 DATA 66, 4,"Use <+> <+> Cursor Keys to Navigate"
2311 DATA 54,14,"Abort/Rtn Spacebar Action <+>Enter"
2312 DATA 50,30,"Frame '-/+ 'G' Set Frame Size",272,30,"16x16"
2313 DATA 12,43,"X'FLIP'Y'z'ROLL'Z",160,41,"N'New 'C'Copy 'D'Delete"
2314 DATA 12,54,"# MAX min Edit",18,113,"PAN' <+> Shift <+> SCROLL"
2315 DATA 160,68,"P' Palette Colour Chg",160,57,"B' BackGnd Colour Chg"
2316 DATA 18,79,"MOVE <+> ALT <+> ReSIZE",38,102,"Set/Drag CURSOR"
2317 DATA 65,69,"<+> ,65,87,<+> ,89,87,<+> ,80,93,<+> ,89,99,<+> "
2318 DATA 157,90,"SB",184,90,"Paint ON OFF",160,112,"@ Fill"
2319 DATA 160,101,"E Erase ON OFF",160,79,"R Recolour"
2320 DATA 48,130,"(1..9) 'T Set Asset 0..",275,130,"0 00"
2321 DATA 12,141,"M Map SCREEN LINKS [#]Test
2322 DATA 12,156,"K Key CTRL Settings 'U' User Checks [GAME Run]"
2323 DATA 12,168,"F DIR Settings 'L' Load _bmp _fnt 'A' Add Fonts"
2324 DATA 12,180,"S Save SPRITE_bmp SCREEN_bmp RETROGAME_bas"
2325 OVER#3,1:ik%=CP(7) :BLOCK#3,12,3,112,18,ik% :BLOCK#3,2,4,234,16,ik%
2326 BLOCK#3,12,9,61, 54,ik% :BLOCK#3,10,7,62,55,0 :REMark MAX[ jmin
2327 BLOCK#3,6,5,100, 56,ik% :BLOCK#3,4,3,101,57,0
2328 BLOCK#3,18,4,244,93,ik% :BLOCK#3,16,4,293, 93,ik% :BLOCK#3,14,2,294, 94,0
2329 BLOCK#3,12,9,247,101,ik% :BLOCK#3,12,9,296,101,ik% :BLOCK#3,10,7,297,102,0
2330 BLOCK#3,6,5,40,67,ik% :BLOCK#3,5,4,41,68,0 :REMark [ALT] Edit Area
2331 BLOCK#3,6,5,90,67,ik% :BLOCK#3,5,4,90,68,0
2332 BLOCK#3,6,5,40,96,ik% :BLOCK#3,5,4,41,96,0 :BLOCK#3,7,5,89,96,ik%
2333 BLOCK#3,12,9,296,112,ik% :BLOCK#3,10,7,297,113,0
2334 QBold 0,7,130,10,'Exploring PIXEL Art and RETRO GAME DESIGN':PAUSE:CLS#3
2335 END DEFine

```