Sinclair QL Retro-Computing

Sinclair QL Retro-Computing

QBITS Software
Copyright 2021

Colin Knibb

'Sir Clive Sinclair 30/07/1940 - 16/09/2021'

A special thanks to Tony Tebby, Jan Jones and Marcel Kilgus for
SMSQ/E and the S/SuperBASIC Programming Language
and to others who have contributed down the years.

Thanks to Quanta, QL World mag. Jochen Merz - QL Today
Members of QL Forum. The Sinclair QL Preservation Project
Dylwin Jones - Sinclair QL Homepage
Rich Mellor - RWAP Software
Steve Bourne - QBITS Trader (1987\90's)

Contents

QPC11 Emulator

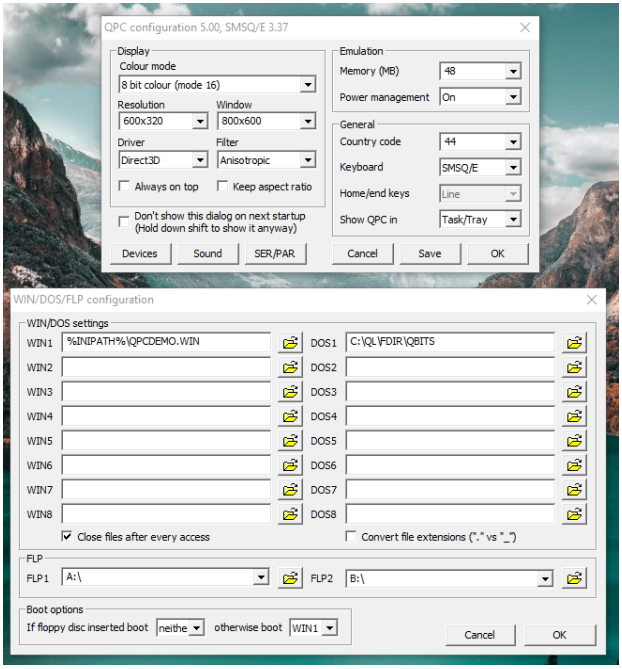
Installed and run on a Windows PC this Emulates a Sinclair QL Computer. However, it has a far more advanced O/S with Tony Tebby's SMSQ/E the successor to his QDOS and an updated expanded SBASIC to the QL SuperBASIC of Jan Jones day.

Downloads: https://www.kilgus.net/qpc/downloads/
Also Check out: http://www.dilwyn.me.uk/emu/index.html

QPC11 Manual

Issued 2021 with the release of QPC11 v 5.00 it explains Installation, Concepts, and SBASIC keywords. QPC Screen resolution and size is extended from the original 512x256 with additional Colour Palettes.

Download and follow the documentation's instructions to Install.
Start **QPC11** and change the configuration to that shown below: -



QBITS Progs

Download and unzip into a New Files Folder. In **QPC configuration** Click on **Devices** and link **dos1** to your **QBITSProgs** Folder, press OK and then **Save**. Press **Start** and with **QPC11** up and running exit from the demo page and in the SuperBasic Interpreter's Command Window type: - **LRUN Dos1_QBITS_Boot**

The QBITS Progs Menu should now be displayed
Select a Program with Cursor Keys and Spacebar.

Progs Introduction

Microcomputers released to the home market in the mid nineteen eighties came with a BASIC Interpreter. The **B**eginner's **A**ll-purpose **S**ymbolic **I**nstruction **C**ode used a FORTRAN style of one-to-a-line statements. Variants evolved among home computer manufactures to become quite sophisticated and yet small enough to fit within memory constraints of the day. Computer Magazines published BASIC code lists for Games and Utilities and for a while BASIC became the de-facto standard for introducing beginners to computer programming.

In nineteen eighty-four during the summer recess, I managed to get some work experience in the computing department of Aberystwyth University. Most of my time was spent mapping and etching circuit boards, but it was also to be my first sighting of the Sinclair QL with its multi-tasking operating system QDOS. The QL was under review in particular the four PSION business programs Quill (word processor), Abacus (spreadsheet), Archive (intelligent database) and Easel (for drawing charts etc.). It had an external ROM with what I believe include a trial release of the SuperBASIC Interpreter.

Sinclair QL

I bought my first QL in 1985 a few months before the price dropped from £399 to £199. An early addition was a Trump card increasing RAM to 640kbytes and expanding my storage capacity with dual 3½" Floppy disk drives. It came with a release of Toolkit II which improved and extended the SuperBASIC list of Keywords.

The drawback of those earlier times, computer platforms weren't fast enough to satisfy the growing demands running BASIC code through the built-in Interpreter. Compiling, writing your program in Assembly or Machine code greatly increased the speed of execution.

My experience of programming at the time was fledgling, an introduction to machine code, instructions on the Forth language commands and a few lessons of BASIC on an BBC micro. At work we had just received our first IBM PC, the display was green characters on a black screen.

The QBITS Name

Steve Bourne and I first met as members attending a QUANTA club held in the old school hall of a Village called Lolworth near Cambridge UK. I gained a lot of programming advice and help from the members. Steve had just begun selling QL second-hand hardware and suggested selling copies of my software. I vaguely recall a conversation discussing QL bit and bobs and from which the name QBITS for his Trader's name and my Software ensued. The QBITS Progs became an added contribution to Steve's wares as he trawled around different QL Club venues and shows back in the late eighties and early nineties.

QBITS Software

Late 1980's display of QBITS Software that Steve carried as part of his Trading stock. As I recall I think they were Compiled with SUPERCHARGE.

QBITS & SuperBASIC

The QL User's Guide introduces the SuperBASIC Language and instructions on programming. My **SuperBASIC** experience began by exploring the variety of ways in which to display Character fonts. I discovered with commands **INK**, **STRIP**, **PAPER** I could **PRINT** in different backgrounds and colours. An even bigger impact was dropping the **AT** line/column Keyword for the more versatile **CURSOR**. Used with **CSIZE** and **OVER** I could create different font sizes and even 3D affects. **CLS** options, **LEN, FILL$, SCROLL, PAN** further added to the changes that could be made with character displays.

QBITS & Character Strings

FILL$ is used to add spaces or Characters. **LEN** returns the number of characters in a string.

CURSOR x,y **:PRINT** String$& **FILL$**(' ',StrLen-**LEN** (String$)) [adds a set number of ' spaces']

Strings for units, ten, hundreds, thousands etc. where numbering expands from right to left.

CURSOR x,y:**PRINT FILL$**('0',4-**LEN**(number%)&number% [0000-9999]

The **CURSOR** option where **PRINT** is positioned relative to the Graphic coordinate system.

CURSOR gx,gy,px,py:**PRINT** String$ (graphic gx,gy coordinates px,py pixel offset)

QBITS Character Graphics

Navigating **QBITS Progs** use Cursor keys ← ↑ ↓ → for Actions ▬ Spacebar and ↵ Enter key. The Spacebar and Enter tail are provided by BLOCK commands.

CURSOR 24,20:**PRINT** 'Select using ← ↑ ↓ → ▬ ↵ ':**BLOCK**#0,12,3,130,24,5:**BLOCK**#0,2,4,198,22,5

More Symbols can be created with the Graphic commands **ARC**, **CIRCLE**, **LINE**, **POINT**:

Info/Help ⓘ INK#ch,col:CIRCLE#ch,x,y,2:LINE#ch,x,y-1 TO x,y+.5:POINT#ch,x,y+1

or **CTRL** INK#ch,col:CIRCLE#ch,x,y+.2,1.2 :LINE#ch,x,y+2.2 TO x,y-2
LINE#ch,x-1.6,y+1.6 TO x+1.6,y-1.6:LINE#ch,x+1.6,y+1.6 TO x-1.6,y-1.6

Other examples would be **ALT** ⎇ **Esc** ⎋ and **Exit** ⎆ Symbols.

QBITS Development
Back in the nineteen eighties the challenge of Graphics encouraged further exploration of the structured coding of SuperBASIC, its PROCedures and FuNctions that made GOTO & GOSUB virtually redundant. Ideas for new QBITS Progs often began as doddles, that migrated to screen layouts, set in **WINDOW**'s, with background **PAPER**, a **BORDER** maybe, then by trial-and-error blocks of code that transformed into a workable program. Having created a few SuperBASIC Progs, I did try my hand at writing them in Assembly Code. Then along came SUPERCHARGE and Compiling SuperBASIC was a much easier route.

QBITS Revival - QL Emulators
By the early nineties my QL involvement was in decline. I think it was in 2004 I downloaded a copy of Jimmy Montesinos **QL2K Emulator** and my interest in SuperBASIC was rekindled. Today there are several choices for QL Emulation for Desktops, Laptops, Tablets using MS, Mac or Linux Operating Systems. For now my preferred choice is **QPC11**.

QBITS Progs
This Collection of **QBITS Progs** is **c**onfigured for use with the **QPC11 Emulator**. In exploring these Progs hopefully, the reader will gain an understanding of the simple to more complex programming aspects of the **S/SuperBASIC Environment**.

QBITS Boot & Config
At start up the S/SuperBASIC Interpreter will Load and Run a program called '**Boot**'. This file is essentially to Load Extensions to the O/S and Executable programs that match the preferred arrangement of the individual user.

Bottom left of the **QPC Configuration - Device** page - are **Boot options** - to run from either floppy disk 1 or 2 or any of the WIN1 to WIN8 drives. The Default is WIN1, where the Boot file will need a link to **LRUN dos1_QBITSBoot**.

```
100 REMark dos1_QBITSBoot  (QBITS Boot 2021 QPCII)
101 gx=40:gy=40:dn$='ram2_QBITSProgs':dev$='dos1_':dn%=4:dm%        \
102 DATA 'mdv1_','mdv2_','flp1_','flp2_','dos1_','dos2_','dos3_','dos4_'      QBITS Progs variables
103 DATA 'win1_','win2_','win3_','win4_','win5_','win6_','ram1_','ram2_'      /
104 FORMAT ram2_100
105 COPY dos1_QBITSProgs TO ram2_QBITSProgs
106 COPY dos1_QBITS_FTidy_v8 TO ram2_QBITS_FTidy_v8
107 ALTKEY 'Q','LRUN ram2_QBITSProgs'&CHR$(10)          :REMark ALTKEY settings
108 ALTKEY ' f ','LRUN ram2_QBITS_FTidy_v8'&CHR$(10)
109 DELETE 'ram2_QBITSConfig'-                          :REMark Config settings
110 OPEN_NEW#9,'ram2_QBITSConfig':PRINT#9,gx\gy\dn$\dev$\dn%dm%
111 RESTORE 102:FOR d=1 TO dm%+1:READ d$:PRINT#9,d$
112 CLOSE#9:LRUN ram2_QBITSProgs                        :REMark Progs Menu
```

QBITSBoot creates **QBITSConfig**, which is used to manage common variables used by the **QBITS Programs**. Settings for gx, gy locate the QBITS backward compatible 512x256 screen size to sit within the higher screen resolutions of the QPC11 Emulator. When Exiting a QBITS Programs dn$ is set to 'ram2_QBITSProgs', LRUN dn$ Returns to QBITSProgs.

For Progs that use **Load**/**Save:** Settings are provided for Dev$, dn%, dm% and a list of Storage Device names, 'mdv1_, flp1_, win1_, etc. These can be changed.

QBITSProgs Code

This First Program is to show what is on offer a List of Progs which are Selected by use of the Cursor keys and Spacebar. Each Program when highlighted has a brief description displayed in the lower window. Having Selected and LRUN one of the Progs the (**E**)xit key from that Program Returns to QBITSProgs.

```
1000 REMark QBITSProgs [QBITS Progs 2021 QPCII]

1002 OPEN_IN#9,'ram2_QBITSConfig':INPUT#9,gx,gy,dn$,dev$:CLOSE#9

1004 QBITSProgs:QBITSInfo:QBITSMenu

1006 DEFine PROCedure QBITSProgs
1007 DIM Sel$(16,20)
1008 OPEN#3,scr_:WINDOW#3,512,256,gx,gy:PAPER#3,0:BORDER#3,1,3:CLS#3
1009 FOR i=0 TO 2:PAPER#i,0:CSIZE#i,0,0:INK#i,7:STRIP#i,0:SCALE#i,100,0,0
1010 WINDOW#2,500,204,gx+6,gy+4
1011 WINDOW#1,500,204,gx+6,gy+4  :BORDER#1,1,5:CLS#1
1012 WINDOW#0,500, 42,gx+6,gy+210:BORDER#0,1,5:CLS#0
1013 OVER#1,1:CSIZE#1,2,1
1014 INK#1,2:FOR i=0 TO 1:CURSOR#1,44+i,12:PRINT#1,'QBITS SuperBASIC Progs'
1015 INK#1,6:FOR i=0 TO 1:CURSOR#1,46,13+i:PRINT#1,'QBITS SuperBASIC Progs'
1016 INK#1,2:FOR i=0 TO 1:CURSOR#1,44+i,150:PRINT#1,'QPCII Emulator'
1017 INK#1,6:FOR i=0 TO 1:CURSOR#1,46,151+i:PRINT#1,'QPCII Emulator'
1018 OVER#1,0:CSIZE#1,2,0:QBold 1,5,12,324,22,'ALT Q':RESTORE 1022
1019 FOR a=1 TO 16
1020   READ x,y,str$,Prog$:QBold 1,5,12,x,y,str$:Sel$(a)=Prog$
1021 END FOR a
1022 DATA  50, 40,'FileTidy          ','FTidy_v8'
1023 DATA  50, 52,'TicTacToe         ','TTT_v3'
1024 DATA  50, 64,'MineDetect        ','MDETR_v3'
1025 DATA  50, 76,'TileSlider        ','Tiles_v3'
1026 DATA  50, 88,'Conundrum         ','Conundrum_v3'
1027 DATA  50,100,'Darts             ','Darts_v3'
1028 DATA  50,112,'Golf              ','Golf_v3'     :REMark 2021
1029 DATA  50,124,'Organiser *       ','Org_v3'      :REMark 2020
1030 DATA 270, 40,'BITMAPS QL        ','BITMAPS_v3'
1031 DATA 270, 52,'BITMAPS 24        ','BITMAPS_vCP24'
1032 DATA 270, 64,'3DGraphics        ','3DGraphics_v3'
1033 DATA 270, 76,'QLSound           ','QLSounds_v3'
1034 DATA 270, 88,'Warehouse         ','WH21_v3'
1035 DATA 270,100,'Karnak Maze       ','Maze_v4'
1036 DATA 270,112,'Pandemic          ','Pandemic_v3'
1037 DATA 270,124,'Trader            ','Trader_v3'
1038 DATA 270,136,'AD2375 ***        ','AD2375_v3'
1039 DATA 270,148,'COP50  ***        ','COP50_v3'
1040 END DEFine

1042 DEFine PROCedure QBold(ch,col,w,x,y,str$)
1043 OVER#ch,1:INK#ch,col
1044 FOR i=1 TO LEN(str$):CURSOR#ch,x+w*i,y  :PRINT#ch,str$(i)
1045 FOR i=1 TO LEN(str$):CURSOR#ch,1+x+w*i,y:PRINT#ch,str$(i)
1046 OVER#ch,0
1047 END DEFine
```

```
1049 DEFine PROCedure QBITSMenu
1050 CSIZE#1,1,0:KAlt 1,7,154,87.6:QBold 1,7,7,420,22,'+Q':KExit 2,7,168,9
1051 QBold 1,7,7,40,180,'Select with ←↑↓→ Cursors and Action with ▬ Spacebar'
1052 CSIZE#1,3,0:KEsc 1,7,160,8.5:BLOCK#1,12,3,336,184,7:x=260:y=1:max=11:c$=' →'
1053 REPeat Menu_lp
1054   IF x=260:RESTORE 1067+y:ELSE RESTORE 1078+y
1055   READ str$:CURSOR#0,48,10:PRINT#0,str$:CURSOR#1,x,28+y*12:PRINT#1,c$
1056   k=CODE(INKEY$(-1)):CURSOR#1,x,28+y*12:PRINT#1,' ':CLS#0
1057   SELect ON k
1058     =192:IF x=260:x=200:max=6 :col=10:c$=' →':IF y>max:y=max
1059     =200:IF x=200:x=260:max=10:col= 0:c$=' ← '
1060     =208:y=y-1:IF y<1  :y=max
1061     =216:y=y+1:IF y>max:y=1
1062     =  32:CLS#3:LRUN dev$&'QBProgs_'&Sel$(y+col)
1063     =  27,69,101:CSIZE#2,0,0:INK#2,7:EXIT Menu_lp
1064   END SELect
1065 END REPeat Menu_lp
1066 END DEFine

1067 DEFine PROCedure QBITSInfo
1069 DATA 'A File Tidy Program - to Review Files and SuperBASIC Progs'
1070 DATA 'The Coffee Break Challenge Classic Noughts & Crosses Game'
1071 DATA 'Clear Mines - Based on Classic Mine Sweeper Game of the 1980s'
1072 DATA 'A Sliding Tile Puzzle Game with Numbers or Minions Picture'
1073 DATA 'Select the Correct Order of Letters and Find the Hidden Word'
1074 DATA 'Classic Darts - Play 301/501 or Around the Clock Face Game'
1075 DATA 'Compete over an 18 Hole Course - SCORECARD with Par & HandiCap'
1076 DATA 'Exploring a 1980s Style Personal Organiser - WIP 2022'
1077 DATA 'Exploring BITMap Design - A Prog for 1980s Style Sprites etc.'
1078 DATA 'Exploring BITMap Design - Using a 24 Bit Colour Palette.'
1079 DATA 'Exploring 3D Rotation Graphics - Its all in the Coding'
1080 DATA 'Exploring the Quirky Nature of the QL SuperBASIC BEEP Command'
1081 DATA 'Manage a WareHouse - Handle Invoice Requests & Stock Deliveries'
1082 DATA 'Solve the Maze - Your Mission Travel back in Time to Save Humanity'
1083 DATA 'As a Specialist - Lead a Team to Contain & Eradicate a Deadly Virus'
1084 DATA 'As a Market Trader - Manage a Portfolio of Company Stocks & Shares'
1085 DATA 'AD2375 Galaxy Adventure - WIP - Release Date 2022'
1086 DATA 'COP50 Saving the Planet - New - Release Date 2022'
1087 END DEFine

1089 DEFine PROCedure KAlt(ch,col,x,y)
1090 INK#ch,col:CURSOR#ch,x,y,0,-5:PRINT#ch,'→'
1091 LINE#ch,x+.5,y-2 to x+2,y-2::LINE#ch,x-1.5,y-2 TO x-1,y-2 TO x+.2,y
1092 END DEFine

1094 DEFine PROCedure KEsc(ch,col,x,y)
1095INK#ch,col:FILL#ch,1:LINE#ch,x-1,y+2 TO x+1,y TO x+.8,y-.5
1096 LINE#ch TO x-1.4,y+1.6 TO x-1,y+2:FILL#ch,0:ARC#ch,x-1.5,y TO x,y+1.8,30
1097 END DEFine

1099 DEFine PROCedure KExit(ch,col,x,y)
1100 INK#ch,col:CURSOR#ch,x,y,1,-5:PRINT#ch,'→':LINE#ch,x,y TO x,y-.4
1101 LINE#ch,x+1.2,y+1.8 TO x-1,y+1.8 TO x-1,y-2 TO x+1.6,y-2
1102 END DEFine
```

File Tidy Introduction

The Sinclair QL came with two **microdrive** storage devices, **'mdv1_** & **mdv2_**' but it wasn't long before external **floppy drives** '**flp1_**' became available. And soon to follow were **Hard Drives** '**win1_**' etc, opening up the possibility of even larger storage capacity.

The **QL Technical Guide** identifies a **QL filename** as being up to **36 bytes** in length or the equivalent number in **ASCII characters.** Viewing files using the original SuperBASIC **DIR** command displays a single vertical list. This soon spread over several pages. Mistyping a file name for **LOAD** or **LRUN** became a frustrating exercise in using the **DIR** command to review any missspelling of filenames. The **QBITS** approach was to develop a **Directory Handler**.

QBITS FTidy Concepts

By 1987 a collection of SuperBASIC routines to keep track of filenames evolved into an early File Management Tool called File Tidy later shortened to **FTidy**. It accessed a Source Device and held up to 160 Filenames. The screen showed four columns of 18 characters. However, when a file was selected the full Filename of up to 36 characters were shown in the window below the Menu. Early versions had a Print command to printout File lists. This was dropped for **LOAD** and later to just **LRUN** so **SubDIR**ectories could be added to the Menu options. The **COPY** and **DELETE** commands allow a single file or batch processing of multiple files.

The **QBITS FTidy** displays the **Device name, Volume/Sectors** and when showing the full **Filename** the **Bytes** size and **Date/Time** stamp. The Filenames are sorted Alphanumerically, with the **SubDIR**ectories listed first. The simple **Line Editor** has been improved over the years.

Directory Menu: **FDIR**-[Drive DIR] **MDIR**-[Make SubDIR] **SDIR**-[SubDIR Access]

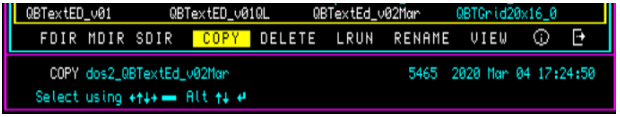
Filename Menu: COPY\DELETE\LRUN\RENAME\VIEW

QBITS FTidy Menu

On start up a **Help** screen of Commands with a brief description of their functions. First choose a Source device (🡅🡇) **DIR**ectory, filenames are then displayed in four columns across the screen. Highlight Commands with 🡄 🡆 Cursors then Press ▬ Spacebar or **F M S C D L R V I E** keys.

FTidy File Directory

FDIR displays the **DIRectory** of the Default or last **Device** chosen. Select **MDIR** to create a new SubDirectory. If SubDIRirectories are shown then Select **SDIR** to access SubDIR levels. Action with <↵> **ENTER:** The Filenames of the selected **Device** or **SubDIR**ectory are written to **FList** which is Read and Sorted to generate the columns to screen.

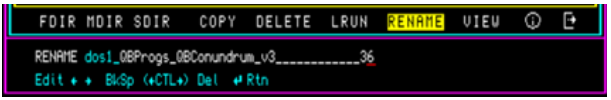


FTidy COPY / DELETE

Select single or multiple files. The Filename(s) are identified by moving through the files listed and highlighting with the Spacebar. For **COPY** select a destination **Target** device then a second pass is made through the selected list with a **'y/n'** for each file before any action is carried out. The selected Filenames are then copied from **Source** to **Target** device. For **DELETE,** the highlighted Filename(s) are confirmed with '**y/n**' before Deleting.

FTidy LRUN

Select a Filename, the full Filename up to 36 characters is displayed together with its Byte length and Time/Date stamp. You are then prompted with **LRUN** filename '**y/n**'.



FTidy RENAME

Select an existing Filename (**file$**) and edit the string (**str$**) with the simple **L**ine **E**ditor. Checks are made that the Filename doesn't already exist, if not a COPY of the file is made with new Filename to source and then the old file is Deleted.

FTidy VIEW

Being able to read the contents, especially the first few lines of a program was seen as a necessary addition to help in recognising a file for what it was. Opening a selected file and reading it requires a little fineness. If a Byte file, a wraparound new line is required after so many characters. For a SuperBASIC or ASCII Character file then acknowledgment of an Enter for each new line.

Note: Executable Files

For this one method would be to write a **SuperBASIC** Program that loads the Byte Program Code into memory and activate its operation. That load Prog can then be **LRUN** from **FTidy**.

QBITS File Tidy Procedures

Before witting a New Program its good practice to set out your objectives and breakdown the codding into manageable sections.

S/SuperBASIC File Directory Management COPY/DELETE/LRUN/RENAME/VIEW.

Init_win	Initialises Program setting the screen display
F_Title	Displays QBITS Title / DIR headings
F_Info	Displays Info/Help screen
Cmd_Menu	Main Program Loop
F_Exit	Closes Channels & Clears Screen to default
SelDrv	Selects Source or Target Devices
FileDIR	Generates File List of Device DIRectory or a Sub DIRectory
F_Sort	Arranges Filenames AlphaNumerically with SubDIR(s) first
F_Check	Returns y/n answer [yes=1 no=0]
MakeDIR	Uses Line Editor to create a New SubDIRectory
SubDIR	Selects and displays the Filenames of a SubDIRectory
Sub_up	Move to Drive DIR or Higher-Level SubDIR
Sub_dn	Move to a Lower-Level SubDIR
F_Select	Use Cursor keys to Select a SubDIR or Filename
Fscr_posn	Calculate screen position of SubDIR or Filename
Fscr_up	Scroll up one row
Fscr_dn	Scroll down one row
F_write	Print SubDIR or Filename to screen position
F_Stat	Print Selected Filename - Byte length and Date/Time Stamp
F_clear	Prints updated File List to screen
F_Batch	Selects and Confirms Filename(s) y/n? from Source Device
F_Target	Selects destination - Targeted Device DIR/SubDIR
F_Copy	Copies File(s) to Target Device DIR SubDIR with overwrite y/n?
F_Delete	Deletes Selected Files(s) from Source Device y/n?
F_Lrun	Load and Run selected File from list
F_Rename	Uses Line editor to Rename a selected Filename
F_View	Prints Code of selected Filename to screen
Ln_Ed	Line Editor
Str_chk	Checks if last Character of string is '**_**' and deletes
Ln_Prn	Prints Filename to Screen
Ln_Cur	Prints Current Cursor Position to Screen
Add_chr	Adds a character at any Position in or at end of string
Del_chr	Deletes a character anywhere in string
KExit	Key Graphic Image created for Exit
KInfo	Key Graphic Image created for Info/Help

QBITS FTidy Code

```
1000 REMark QBITS_FTidy_v8 (QBITS File Tidy version 8 2021 QPC11)

1002 OPEN_IN#9,'ram2_QBITSConfig':INPUT#9,gx\gy\dn$\dev$\dn%\dm%
1003 DIM Drv$(dm%,5):FOR d=0 TO dm%:INPUT#9,Drv$(d):END FOR d:CLOSE#9

1005 REMark DIRectory Actions - FDIR(File DIR):MDIR(Make Sub):SDIR(Sub DIR)
1006 REMark [SubDIR Levels 1-24 Characters from 36 of Filename]
1007 REMark FILE Actions - COPY DELETE LRUN RENAME VIEW
1008 REMark [Single/Batch file(s) – COPY DELETE]

1010 WHEN ERRor
1011   CLS#3:CURSOR#3,12,3:PRINT#3,'DEVICE ERROR':PAUSE 20:CLS#3:CONTINUE
1012 END WHEN
```

Note: If an error such as Device Access is denied the use of **WHEN ERR**or will **CONTINUE** with the next statement instead of the Interpreter halting the program.

```
1014 num%=300          :REMark Set Max Number of Files / Display - Device
1015 DIM DFile$(num%,2,36),fink%(num%),CFile$(num%,2,36),cink%(num%)
1016 DIM Comd$(58),key$(3,52),help$(9,48),Time$(20)
1017 DIM DD$(5),DDIR$(24),SD$(5),SDIR$(24),TD$(5),TDIR$(24),str$(36)

1019 Mode 4:Init_win:Cmd_Menu

1021 DEFine PROCedure Init_win
1022 OPEN#5,scr_:WINDOW#5,512,256,gx,gy          :PAPER#5,0:BORDER#5,1,3:CLS#5
1023 OPEN#4,scr_:WINDOW#4,504,214,gx+4,gy+7      :PAPER#4,0:BORDER#4,1,5:CSIZE#4,1,0
1024 OPEN#3,scr_:WINDOW#3,280,26,gx+114,gy+2     :PAPER#3,0:BORDER#3,1,5:CLS#3
1025 WINDOW#2,504,220,gx+4,gy+2 :PAPER#2,0                    :CSIZE#2,0,0:INK#2,7
1026 WINDOW#1,496,162,gx+8,gy+42:PAPER#1,0       :BORDER#1,1,6:CSIZE#1,0,0:INK#1,7
1027 WINDOW#0,512, 34,gx,gy+222 :PAPER#0,0       :BORDER#0,1,3:CSIZE#0,0,0:INK#0,7
1028 INK#4,3:CURSOR#4,2,12:PRINT#4,'VOLUME' :CURSOR#4,458,12:PRINT#4,'Files'
1029 Time$=DATE$:CURSOR#2,4,7:PRINT#2,Time$(10 TO 11)&Time$(5 TO 9)&Time$(1 TO 4)
1030 SCALE#2,100,0,0:SCALE#1,100,0,0:F_Title 2,42,'QBITS File Tidy':F_Info
1031 END DEFine
```

QBITS FTidy Title

For the opening screen F_Title is used to display the full program name ‘QBITS File Tidy’. When accessing the File Directories, it displays the Drive Directory or SubDIRectory name of the File Folder being viewed.

```
1033 DEFine PROCedure F_Title (cs%,x%,Title$)
1034 CLS#3:CSIZE#3,cs%,1:OVER#3,1
1035 INK#3,2 :FOR i=0 TO 1:CURSOR#3,x%+i,1:PRINT#3,Title$
1036 INK#3,6 :FOR i=1 TO 2:CURSOR#3,x%+i,2:PRINT#3,Title$
1037 OVER#3,0
1038 END DEFine
```

cs% - Character size : x% - horizontal offset : Title$ - Character string.

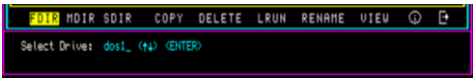
QBITS FTidy Info

The program opens with a Help screen and prompts for a Drive selection - either open the default or choose an alternative **DIR**ectroy.

```
1040 DEFine PROCedure F_Info
1041 key$(1)='Select Menu with  ← →CURSOR keys and ←ENTER'
1042 key$(2)='Select File(s) ← ↑ ↓ → Mark   Page Up/Down ALT+↑↓'
1043 key$(3)="To show Info   Panel press 'i' to Exit   press 'e'"
1044 help$(0)='  FDIR     - Display DIRectory Files of Source Drive'
1045 help$(1)='  MDIR     - Make   SubDIRectory on Source Drive'
1046 help$(2)='  SDIR     - Display SubDIRectory of Source Drive'
1047 help$(3)='                '
1048 help$(4)='  COPY     - MARKed File(s) from Source to Target'
1049 help$(5)='DELETE  - MARKed File(s) from Source Drive'
1050 help$(6)='  LRUN     - SuperBASIC Program'
1051 help$(7)='RENAME - Selected File from Source Device'
1052 help$(8)='  VIEW     - Selected File Information and Content'
1053 Comd$=' FDIR MDIR SDIR   COPY  DELETE  LRUN  RENAME  VIEW '
1054 OVER#1,1:CSIZE#1,1,0:CLS#1
1055 FOR hp=0 TO 8
1056   INK#1,7:FOR i=0 TO 1:CURSOR#1,56+i,21+hp*11:PRINT#1,help$(hp,1 TO 9);
1057   INK#1,3:PRINT#1,help$(hp,10 TO)
1058 END FOR hp
1059 OVER#1,0:CSIZE#1,0,0:INK#1,5
1060 CURSOR#1,112,    8:PRINT#1,key$(1):BLOCK#1,2,4,340,10,5
1061 CURSOR#1,108,124:PRINT#1,key$(2):BLOCK#1,12,3,256,128,5:INK#1,6
1062 CURSOR#1,  96,142:PRINT#1,key$(3):KInfo 1,6,82,8:KExit 1,6,154,8
1063 mp=0:mm%=0:mx%=7:lptr%=0:INK#0,7:px%=108:DD$='':SelDrv:CLS#0
1064 END DEFine
```

QBITS FTidy Storage Device

The Device Drive is selected at the beginning of the program and subsequently by actioning the **FDIR** Command from the Programmes main Menu. Array Drv$(n) holds sixteen different Drive names. The default drive is set on line 1014: dn=n (n = 0 TO 15) ie. ‘dos1_’.

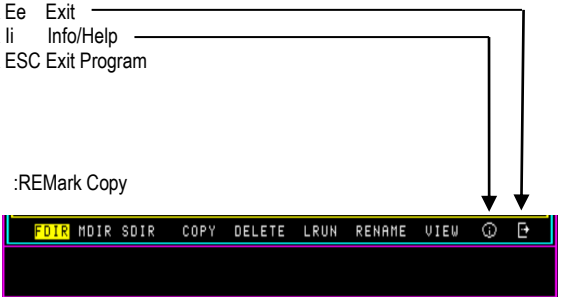



```
1066 DEFine PROCedure SelDrv
1067 DIM SubDIR$(6,24):dl%=0:OD$=DD$                                  Old Drive$ - DIR Drive$
1068 IF mp<2:INK#0,7:CURSOR#0,18,6:PRINT#0,'Select Drive: ':px%=108   px% horizontal coordinate
1069 REPeat Dr_lp
1070   INK#0,5:CURSOR#0,px%,6:PRINT#0,Drv$(dn%)&' (↑↓) <ENTER>':CLS#0,4  dn% drive number
1071   k=CODE(INKEY$(#0,-1))
1072   SELect ON k
1073     =  10:DD$=Drv$(dn%):EXIT Dr_lp        :REMark Enter  Select Drive
1074     =208:dn%=dn%-1:IF dn%< 0:dn%=dm% :REMark Up
1075     =216:dn%=dn%+1:IF dn%>dm%:dn%= 0:REMark Down
1076   END SELect
1077 END REPeat Dr_lp
1078 IF OD$=DD$:RETurn :ELSE DDIR$='':FileDIR                          DDIR$ Drive DIR
1079 END DEFine
```

QBITS FTidy Command Menu

Select using <Left & Right Cursors> or the first (**L**)etter of Menu Command to highlight the name then press <SPACEBAR>.

```
1081 DEFine PROCedure Cmd_Menu
1082 KInfo 2,7,150,4.6:KExit 2,7,160,4.4
1083 REPeat Cmd_lp
1084   STRIP#4,0:INK#4,7:CURSOR#4,16,200:PRINT#4,Comd$
1085   IF mp<3:mf%=1+mp*5:ml%=mf%+4:ELSE mf%=-4+mp*7:ml%=mf%+6
1086   STRIP#4,6:INK#4,0:CURSOR#4,16+(mf%*8),200:PRINT#4,Comd$(1+mf% TO ml%)
1087   k=CODE(INKEY$(#0,-1))
1088   SELect ON k
1089     =192:mp=mp-1:IF mp<mm%:mp=mx%    :REMark ← left cursor min-max
1090     =200:mp=mp+1:IF mp>mx%:mp=mm%    :REMark → right cursor
1092     =70,102:mp=0                     :REMark Ff FDIR
1093     =77,109:mp=1                     :REMark Mm  MDIR
1094     =83,115:mp=2                     :REMark Ss  SDIR
1095     =67,  99:mp=3                    :REMark Cc  COPY
1096     =68,100:mp=4                     :REMark Dd  DELETE
1097     =76,108:mp=5                     :REMark Ll  LRUN
1098     =82,114:mp=6                     :REMark Rr  RENAME
1099     =86,118:mp=7                     :REMark Vv  VIEW
1100     =69,101:LRUN dn$                 :REMark Ee  Exit
1091     =73,105:h%=1:F_Info:h%=0         :REMark Ii  Info/Help
1101     =27 :F_Exit                      :REMark ESC Exit Program
1102     =10,32:SELect ON mp
1103           =0:px%=102:SelDrv          :CLS#0
1104           =1:MakeDIR                 :CLS#0
1105           =2:SubDIR                  :CLS#0
1106           =3:F_Batch                 :CLS#0   :REMark Copy
1107           =4:F_Delete                :CLS#0
1108           =5:F_Lrun                  :CLS#0
1109           =6:F_Rename                :CLS#0
1110           =7:F_View                  :CLS#0
1111         END SELect
1112   END SELect
1113 END REPeat Cmd_lp
1114 END DEFine
```




```
1116 DEFine PROCedure F_Exit
1117 CLS#5:CLOSE#5:CLOSE#4:CLOSE#3
1118 WINDOW#2,504,212,gx+4,gy+2   :PAPER#2,0:INK#2,7:CLS#2
1119 WINDOW#1,504,212,gx+4,gy+2   :PAPER#1,0:INK#1,7:CLS#1
1120 WINDOW#0,504, 40,gx+4,gy+214:PAPER#0,0:INK#0,7:CLS#0:STOP
1121 END DEFine
```

Note: PROCedure **F_Exit** Restores WINDOW's 0,1,2 to default setting. **STOP** halts the S/SuperBASIC Program currently loaded. The option **NEW** – releases channels other than 0,1,2 and memory used by a S/SuperBASIC Program.

```
1123 DEFine PROCedure FileDIR
1124 CLS#0:DELETE DD$&DDIR$&'FList'
1125 IF mp<2:INK#0,5:CURSOR#0,24,6:PRINT#0,'Files being Selected...'
1126 F_Title 1,4,'DIR '&DD$&DDIR$
1127 OPEN_NEW#9,DD$&DDIR$&'FList':STAT#9,DD$&DDIR$:WSTAT#9,DD$&DDIR$:CLOSE#9
1128 OPEN_IN #9,DD$&DDIR$&'FList':INPUT#9,Dnam$\DSec$:n=1:ftot%=0
1129 REPeat DIR_lp
1130   IF EOF(#9) OR n>num%:ftot%=n-1:CLOSE#9:EXIT DIR_lp
1131   INPUT#9,DFile$(n,1):fink%(n)=5
1132   IF '>' INSTR DFile$(n,1)=0:INPUT#9,DFile$(n,2):n=n+1:ELSE n=n+1
1133 END REPeat DIR_lp
1134 BLOCK#2,480,10,4,29,0:INK#2,6:CURSOR#2,4,29:PRINT#2,Dnam$;'  ';DSec$
1135 CURSOR#2,466,29:PRINT#2,FILL$(' ',5-LEN(ftot%))&ftot%:INK#2,7
1136 BLOCK#2,60,10,440,7,0:IF dl%>0:CURSOR#2,440,7:PRINT#2,'L:';dl%;' SubDIR'
1137 IF ftot%=0:stot%=0:RETurn
1138 F_Sort:nm%=1:nx%=ftot%:lptr%=0:CLS#1:F_clear:n=1
1139 END DEFine
```

QBITS FileDIR & F_Sort

FileDIR copies to **FList** the full Filename, Byte Size & Date/Time stamp entries of the Selected Storage Device. The list is then Read and sorted Alphanumerically. A Sort program is given in the QL Users Guide Chapter 16. A few tweaks with the choice of array and variable names and Voilá! The **FList** of Filenames are selected sequentially by a **FOR loop** and compared within a **REPeat Loop** leaving the output **DFile$(n)** array sorted in Alphanumeric order. A second **FOR loop** selects any **SubDIR**ectory names and lists them in front of the **Filenames**.

```
1141 DEFine PROCedure F_Sort
1142 FOR sn=1 TO ftot%
1143   p=sn:comp$=DFile$(p,1):info$=DFile$(p,2)
1144   REPeat Sort_lp
1145     IF comp$>=DFile$(p-1,1):EXIT Sort_lp
1146     DFile$(p,1)=DFile$(p-1,1):DFile$(p,2)=DFile$(p-1,2):p=p-1
1147   END REPeat Sort_lp
1148   DFile$(p,1)=comp$:DFile$(p,2)=info$
1149 END FOR sn
1150 ntop=1:nsel=1:stot%=0
1151 FOR sn=1 TO ftot%
1152   IF '>' INSTR DFile$(sn,1)
1153     comp$=DFile$(sn,1):info$=DFile$(sn,2):nsel=sn-1
1154     FOR p=nsel TO ntop STEP -1
1155       DFile$(p+1,1)=DFile$(p,1):DFile$(p+1,2)=DFile$(p,2)
1156     END FOR p
1157     DFile$(ntop,1)=comp$:DFile$(ntop,2)=info$:ntop=ntop+1:stot%=stot%+1
1158   END IF
1159 END FOR sn
1160 END DEFine
```

QBITS FTidy SubDIRectories

The full identity of a file location can be 41 Characters. This begins with the Drive Device a five-character identifier, ie. mdv1_ flp1_ win1_ dos1_ the fifth character '**_**' always being an underscore. The next 36 Characters make up the Filename, of which the first twenty-four characters can be considered for **SubDIR**ectory use. For example, 'SubDIR1_' which as with Drive names have to end with an underscore. If they were named with letters of the alphabet, 'A_' to 'L_' we could potentially create twelve SubDIR levels. However, for QBITS File Tidy I limited this to just six sub levels, creating further levels will not be accessed by **SDIR.**

DIM SubDIR$(6,24) where SubDIR$(1) = "**SD1**_" and SubDIR$(6) access = "**SD1_SD2_SD3_SD4_SD5_SD6**_"

```
1162 DEFine PROCedure F_Check                    Used to return a 'y/n' answer (y=1 n=0)
1163 REPeat chk_lp
1164   k=CODE(INKEY$(#0,-1))
1165   SELect ON k=78,110:chk=0:EXIT chk_lp
1166   SELect ON k=89,121:chk=1:EXIT chk_lp
1167 END REPeat chk_lp
1168 END DEFine
```


```
1170 DEFine PROCedure MakeDIR
1171 md%=24-LEN(DDIR$):px%=138+LEN(DDIR$)*6
1172 IF md%<3:CURSOR#0,24,6:PRINT#0,'Lowest Level Reached':PAUSE 50:RETurn
1173 INK#0,7 :CURSOR#0,24,6:PRINT#0,'Create SubDIR ';:INK#0,5:PRINT#0,DD$&DDIR$
1174 cp%=1:sl%=0:sm%=md%:str$="":Ln_Ed:IF LEN(str$)=0:RETurn
1175 CURSOR#0,px%+LEN(str$)*6,6:PRINT#0,' (y/n)':F_Check
1176 IF chk=1
1177   FOR n=1 TO stot%:IF DDIR$&str$ INSTR DFile$(n,1):RETurn
1178   MAKE_DIR DD$&DDIR$&str$:FileDIR                MAKE_DIR - S/SuperBASIC Keyword
1179 END IF
1180 END DEFine
```

```
1182 DEFine PROCedure SubDIR
1183 INK#0,7:CURSOR#0,24,6:PRINT#0,'SubDIR ';:INK#0,5:PRINT#0,DD$&DDIR$;
1184 INK#0,7:PRINT#0,' ⬆⬇':CLS#0,4:k=CODE(INKEY$(#0,-1))
1185 IF k=208 AND dl%>=0  :Sub_up
1186 IF k=216 AND stot%>=0:Sub_dn
1187 END DEFine
```


```
1189 DEFine PROCedure Sub_up
1190 SubDIR$(dl%)='':dl%=dl%-1:DDIR$=SubDIR$(dl%):FileDIR:RETurn
1191 END DEFine

1193 DEFine PROCedure Sub_dn
1194 IF dl%=6:RETurn                                SubDIR Limiter
1195 px%=96:Cmd$='SubDIR ':mark%=5:n=1:nm%=1:nx%=stot%:st%=1:F_select:st%=0
1196 INK#0,5:CURSOR#0,px%,6:PRINT#0,DFile$(n,1)&' (y/n)':CLS#0,4:F_Check
1197 IF chk=1
1198   DDIR$=DFile$(n,1,1 TO flen%-3)&'_':dl%=dl%+1:nm%=stot%:nx%=ftot%:FileDIR
1199   SubDIR$(dl%)=DDIR$:CURSOR#0,px%,6:PRINT#0,DDIR$:CLS#0,4
1200 END IF
1201 nm%=1:nx%=ftot%:F_clear:n=1
1202 END DEFine
```

```
1204 DEFine PROCedure F_select                  filename: select and highlight in screen location
1205 INK#0,7:CURSOR#0,24,6:PRINT#0,Cmd$;:INK#0,5:PRINT#0,DD$&DDIR$
1206 REPeat Sel_lp
1207  Fscr_posn:fink%(n)=7:F_write:fink%(n)=5:k=CODE(INKEY$(#0,-1))
1208  SELect ON k
1209    =192:Fscr_posn:F_write:n=n-1          :REMark back 1
1210    =200:Fscr_posn:F_write:n=n+1          :REMark forward 1
1211    =208:Fscr_posn:F_write:n=n-4          :REMark up 1 row
1212    =216:Fscr_posn:F_write:n=n+4          :REMark down 1 row
1213    =209:Fscr_posn:F_write:n=n-60         :REMark up 1 page
1214    =217:Fscr_posn:F_write:n=n+60         :REMark down 1 page
1215    =  32:fink%(n)=mark%:F_write:n=n+1    :REMark mark filename
1216    =  10:fink%(n)=7:CURSOR#0,0,20:CLS#0,4:RETurn
1217  END SELect
1218 END REPeat Sel_lp
1219 END DEFine

1221 DEFine PROCedure Fscr_posn                 calculate screen position
1222 IF n<nm%:n=nm%                            nm% n min  ftot% DIR File total
1223 IF n>nx%:n=nx%                            nx%  n max
1224 fptr%=n-1:frow%=(fptr% DIV 4)             fptr%  file pointer        frow% file row
1225 IF frow%>(15+lptr%):Fscr_up:n=fptr%+1:Fscr_posn  [facilitates page scroll]
1226 IF frow%<(  0+lptr%):Fscr_dn:n=fptr%+1:Fscr_posn
1227 srow%=frow%-lptr%:scol%=(fptr% MOD 4)*20  srow% screen row      scol% screen column
1228 END DEFine

1230 DEFine PROCedure Fscr_up
1231 lptr%=lptr%+1:SCROLL#1,-10:n=(lptr%+15)*4:srow%=15
1232 FOR i=0 TO 3:scol%=i*20:n=n+1:F_write
1233 END DEFine

1235 DEFine PROCedure Fscr_dn
1236 lptr%=lptr%-1:SCROLL#1,10:n=(lptr%)*4:srow%=0
1247 FOR i=0 TO 3:scol%=i*20:n=n+1:F_write
1238 END DEFine

1240 DEFine PROCedure F_write
1241 IF n>ftot% OR n<1:RETurn
1242 flen%=LEN(DFile$(n,1)):slen%=LEN(DDIR$)                    flen% file length
1243 IF flen%-slen%>18:flen%=18+slen%
1244 INK#1,fink%(n):CURSOR#1,8+scol%*6,srow%*10                  fink% file ink print colour
1245 PRINT#1,DFile$(n,1,1+slen% TO flen%)&FILL$(' ',18+slen%-flen%)   slen% string length
1246 INK#0,5:CURSOR#0,px%,6:PRINT#0,DFile$(n,1):CLS#0,4:IF st%=1:F_Stat   st% stat
1247 END DEFine

1249 DEFine PROCedure F_Stat
1250 INK#0,5:CURSOR#0,24,20:PRINT#0,'Select using ←↑↓→ ▬ Alt ↑↓ ↵':CLS#0,4
1251 BLOCK#0,12,3,130,24,5:BLOCK#0,2,4,198,22,5
1252 CURSOR#0,300,6:PRINT#0,DFile$(n,2)         Use of BLOCK for Spacebar and Return Tail
1253 END DEFine

1255 DEFine PROCedure F_clear                              Clear marked files
1256 FOR sc=1 TO ftot%:fink%(sc)=5
1257 fs%=(lptr%*4)+1:fe%=(lptr%+16)*4:IF fe%>ftot%:fe%=ftot%     fs% file start   fe% file end
1258 FOR n=fs% TO fe%:Fscr_posn:F_write
1259 END DEFine
```

```
1261 DEFine PROCedure F_Batch                                  Select Single/Multiple Files
1262 px%=96:Cmd$='  COPY ':mark%=7:nm%=stot%+1:st%=1:F_select:st%=0
1263 CURSOR#0,px%,6:PRINT#0,SDIR$;'File(s) (y/n)':CLS#0,4:fnum=n:F_Check
1264 IF chk=1:SD$=DD$:SDIR$=DDIR$:cn%=0:ELSE nm%=1:F_clear:n=fnum:RETurn
1265 FOR n=stot%+1 TO ftot%
1266   IF fink%(n)=7
1267     cn%=cn%+1:CFile$(cn%,2)=DFile$(n,2)
1268     CFile$(cn%,1)=DFile$(n,1,1+LEN(SDIR$) TO LEN(DFile$(n,1)))
1269   END IF
1270 END FOR n
1271 TD$=DD$:TDIR$=DDIR$:F_Target
1272 END DEFine
```


```
1274 DEFine PROCedure F_Target                                 Selection of Target Device DIRectory
1275 REPeat tag_lp
1276   CURSOR#0,24, 6:PRINT#0,'  COPY '&SDIR$&' File(s) TO '&TD$&TDIR$:CLS#0,4
1277   CURSOR#0,24,20:PRINT#0,'Change (D)rive (S)ubDIR  (C)OPY File(s)'
1278   k=CODE(INKEY$(#0,-1))
1279   SELect ON k
1280     =68,100:px%=138:SelDrv :TD$=DD$              :REMark D
1281     =83,115:px%=170:SubDIR :TDIR$=DDIR$          :REMark S
1282     =67, 99:EXIT tag_lp                          :REMark C Exit loop
1283   END SELect
1284 END REPeat tag_lp
1285 IF SD$&SDIR$=TD$&TDIR$:nm%=1:F_clear:n=fnum:RETurn :ELSE CLS#0:F_Copy
1286 END DEFine
```


```
1288 DEFine PROCedure F_Copy                                   COPY the selected file(s)
1289 FOR n2=1 TO cn%
1290   str$=CFile$(n2,1):chk=1
1291   CURSOR#0,24,6:PRINT#0,'  COPY '&str$&' TO '&TD$&TDIR$:CLS#0,4
1292   FOR n1=stot%+1 TO ftot%
1293     IF str$==DFile$(n1,1,1+LEN(TDIR$) TO LEN(DFile$(n1,1)))   Check for duplication
1294       INK#0,3:CURSOR#0,6,20:PRINT#0,CFile$(n2,2)&'     '&DFile$(n1,2)   Show Bytes/Date/Time
1295       INK#0,5:CURSOR#0,340,6:PRINT#0,' Overwrite y/n':F_Check
1296       IF chk=0:NEXT n2                                     IF 'NO' move to next file
1297     END  IF
1298     IF chk=1:DELETE TD$&TDIR$&str$:COPY SD$&SDIR$&str$ TO TD$&TDIR$&str$
1299   END FOR n1
1300 END FOR n2
1301 FileDIR                                                    Update Screen Output
1302 END DEFine
```


```
1304 DEFine PROCedure F_Lrun
1305 px%=96:Cmd$='  LRUN ':mark%=5:nm%=stot%+1:st%=1:F_select:st%=0
1306 CURSOR#0,96,6:PRINT#0,DFile$(n,1)&' (y/n)':CLS#0,4:F_Check
1307 IF chk=1:LRUN DD$&DFile$(n,1)                            IF 'Yes' LRUN Filename
1308 fink%(n)=5:Fscr_posn:F_write                              IF 'NO' clear highlighted filename
1309 END DEFine
```

Note: EX, EXEC, EW, EXEC_W or LBYTES or LRESPR Files can be loaded and actioned from a .bas file and then LRUN from FTidy.

```
1311 DEFine PROCedure F_Delete
1312 px%=96:Cmd$='DELETE ':mark%=7:nm%=stot%+1:st%=1:F_select:st%=0
1313 fnum=n:fdel%=ftot%
1314 FOR n=stot%+1 TO ftot%
```

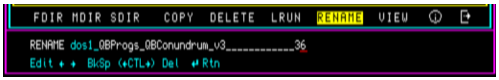

```
1315  IF fink%(n)=7
1316   CURSOR#0,96,6:PRINT#0,DFile$(n,1)&' (y/n)':CLS#0,4:F_Check
1317   IF chk=1:DELETE DD$&DFile$(n,1):fdel%=fdel%-1:fink%(n)=0:Fscr_posn:F_write
1318   IF chk=0:fink%(n)=5:Fscr_posn:F_write
1319  END IF
1320 END FOR n
1321 IF LEN(DDIR$)>0 AND fdel%=0:DELETE DD$&DDIR$:dl%=dl%-1:DDIR$=SubDIR$(dl%)
1322 IF fdel%<ftot%:FileDIR:ELSE nm%=1:F_clear:n=fnum
1323 END DEFine
```


```
1325 DEFine PROCedure F_View
1326 px%=96:Cmd$='  VIEW ':mark%=5:nm%=stot%+1:st%=1:F_select:st%=0
1327 CURSOR#0,96,6:PRINT#0,DFile$(n,1)&' (y/n)';:CLS#0,4:fnum=n:F_Check
1328 IF chk=0:fink%(n)=5:Fscr_posn:F_write:RETurn
1329 CLS#1:BLOCK#0,40,10,100+LEN(DFile$(n,1))*6,6,0:INK#0,5
1330 CURSOR#0,240,20:PRINT#0,'<SPACEBAR> to continue... <ENTER> to Exit'
1331 OPEN_IN#9,DD$&DFile$(n,1):char%=0:fline%=0:fbyts=0
1332 REPeat View_lp
1333   k$=INKEY$(#9,-1):PRINT#1,k$;:char%=char%+1
1334   fbyts=fbyts+1:CURSOR#0,160,20:PRINT#0,'Bytes: ';fbyts;'  '
1335   IF EOF(#9)
1336     CLOSE#9:BLOCK#0,150,10,240,20,0
1337     IF INKEY$(#0,-1)=CHR$(10):EXIT View_lp:ELSE GO TO 1340:END IF
1338   END IF
1339   IF char%>=74 AND k$<>CHR$(10):char%=0:fline%=fline%+1:END IF
1340   IF k$=CHR$(10):char%=0:fline%=fline%+1:END IF
1341   IF fline%>15
1342     fline%=0:IF INKEY$(#0,-1)=CHR$(10):CLOSE#9:EXIT View_lp:END IF
1343   END IF
1344 END REPeat View_lp
1345 nm%=1:CLS#1:F_clear:n=fnum
1346 END DEFine

1348 DEFine PROCedure F_Rename                                  see Ln_Ed
1349 px%=96:Cmd$='RENAME ':mark%=5:nm%=stot%+1:st%=1:F_select:st%=0
1350 CURSOR#0,96,6:PRINT#0,DFile$(n,1)&' (y/n)':CLS#0,4:F_Check
1351 IF chk=0:fink%(n)=5:Fscr_posn:F_write:RETurn              IF NO Clear Highlight & RETurn
1352 INK#0,5:CURSOR#0,px%,6:PRINT#0,DDIR$:CLS#0,4
1353 str$=DFile$(n,1,1+LEN(DDIR$) TO LEN(DFile$(n,1)))
1354 sl%=LEN(str$):cp%=sl%+1:sm%=36-LEN(DDIR$):px%=px%+LEN(DDIR$)*6:Ln_Ed     Edit Filename
1355 IF str$='':str$=DFile$(n%,1):fink%=5:Fscr_posn:F_write:n=fnun:RETurn
1356 FOR n1=1 TO ftot%:IF str$==DFile$(n1,1):F_Chk:RETurn      Check IF Filename already exists
1357 COPY DD$&DFile$(n,1) TO DD$&str$:DELETE DD$&DFile$(n,1):FileDIR
1358 END DEFine

1360 DEFine PROCedure F_Chk
1361 INK#0,5:CURSOR#0,24,20:PRINT#0,'Filename Exists':CLS#0,4:PAUSE 50     Warning Filename Exist
1362 fink%(n)=5:Fscr_posn:F_write                              Clear Highlighted Filename
1363 END DEFine
```

QBITS FTidy Line Editor

This is a simple String or Line Editor, where characters can be entered or deleted where an underline identifies the current string position. This is moved with the Left & Right Cursors keys. Characters are restricted to numeric 0 to 9 [ASCII 48-57], the UPPER/lower-case Alphabet A-z [ASCII 65-90 & 97-122] plus underscore '_' [95].

```
1365 DEFine PROCedure Ln_Ed
1366 INK#0,5:CURSOR#0,24,20
1367 PRINT#0,'Edit ← →  BkSp (←CTL→) Del ↵ Rtn':BLOCK#0,2,4,198,22,5
1368 REPeat Ed_lp
1369   Ln_Prn:Ln_Cur:k$=INKEY$(#0,-1):k=CODE(k$)
1370   SELect ON k
1371     = 10:EXIT Ed_lp
1372     = 48 TO 57, 65 TO 90,95, 97 TO 122:Ln_Prn:Add_chr          ASCII codes
1373     =194:IF cp%>1:cp%=cp%-1:Del_chr                            cp% cursor position
1374     =202:Del_chr
1375     =192:IF cp%>1:cp%=cp%-1
1376     =200:IF cp%<sl%+1:cp%=cp%+1                                sl%  string length
1377   END SELect
1378 END REPeat Ed_lp
1379 END DEFine
```




```
1381 DEFine PROCedure Str_chk                                      SubDIR: Limit '_' in File naming
1382 REPeat str_lp
1383   IF '_' INSTR str$(LEN(str$))=1:cp%=sl%:Del_chr:Ln_Prn       Check & Delete any end of string '_'
1384   IF '_' INSTR str$(LEN(str$))=0:PAUSE 30:Exit str_lp
1385 END REPeat str_lp
1386 END DEFine

1388 DEFine PROCedure Ln_Prn
1389 IF LEN(str$)>sm%:str$=str$(1 TO sm%):cp%=sm%
1390 INK#0,7:CURSOR#0,px%,6:PRINT#0,str$:CLS#0,4
1391 END DEFine

1393 DEFine PROCedure Ln_Cur
1394 BLOCK#0,8,1,px%+cp%*6-6,15,2
1395 END DEFine

1397 DEFine PROCedure Add_chr
1398 IF cp%  =1 AND sl% = 0   :str$=str$&k$
1399 IF cp%>=1 AND cp%<sl%:str$=str$(1 TO cp%-1)&k$&str$(cp% TO sl%)
1400 IF cp%>=1 AND cp%=sl%:str$=str$(1 TO cp%-1)&k$&str$(cp%)
1401 IF cp%>  1 AND cp%>sl%:str$=str$&k$
1402 IF cp%=sm%:str$(cp%)=k$                                       sm% string max in number of characters
1403 IF sl%<sm% :sl% = sl%+1:ELSE sl% =sm%
1404 IF cp%<sm%:cp%=cp%+1:ELSE cp%=sm%
1405 END DEFine

1407 DEFine PROCedure Del_chr
1408 IF cp%=sl%:str$=str$(1 TO sl%-1):sl%=sl%-1
1409 IF cp%>=1 AND cp%<sl%:str$=str$(1 TO cp%-1)&str$(cp%+1 TO sl%):sl%=sl%-1
1410 IF cp%=sm%:str$=str$(1 TO sm%-1):cp%=cp%-1:sl%=sm%-1
1411 IF cp%=1 AND sl%=1:str$="":sl%=0
1412 END DEFine
```

```
1414 REMark QBITS FTidy Graphics

1416 DEFine PROCedure KExit(ch,col,x,y)
1417 INK#ch,col:CURSOR#ch,x,y,1,-5:PRINT#ch,'½':LINE#ch,x,y TO x,y-.4
1418 LINE#ch,x+1.2,y+1.8 TO x-1,y+1.8 TO x-1,y-2 TO x+1.6,y-2
1419 END DEFine
```

Exit Symbol

```
1421 DEFine PROCedure KInfo(ch,col,x,y)
1422 INK#ch,col:CIRCLE#ch,x,y,2:LINE#ch,x,y-1.2 TO x,y:POINT#ch,x,y+.5
1423 END DEFine
```

Info/Help symbol

Note: As an exercise the Graphics below were seen as possible symbols for use with the Pointer Environment.

```
1500 DEFine PROCedure GDisk(col,x,y)
1501 FILL#2,1:INK#2,col
1502 LINE#2,x-2,y+2 TO x+2,y+2 TO x+2,y-2 TO x-2,y-2 TO x-2,y+2
1503 FILL#2,0:INK#2,0:FILL#2,1
1504 LINE#2,x-1.4,y TO x+1.5,y TO x+1.5,y-1.6 TO x-1.4,y-1.6 TO x-1.4,y
1505 FILL#2,0:INK#2,7
1506 END DEFine
```

```
1508 DEFine PROCedure GSave(x,y)
1509 GDisk 5,x,y:INK#2,0:CIRCLE#2,x,y-.8,.6
1510 END DEFine
```

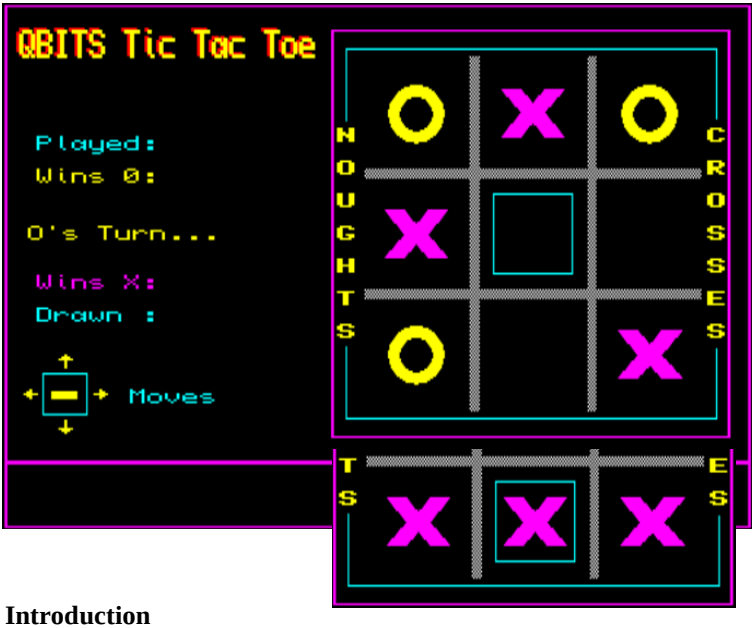
```
1512 DEFine PROCedure GCopy(col,x,y)
1513 GPage col,x,y:INK col:LINE x-1,y-.5 TO x-1,y-5 TO x+2,y-5
1514 END DEFine
```

```
1516 DEFine PROCedure GDelete(x,y)
1517 INK#2,2:FILL#2,1:LINE#2,x-3,y TO x-2.5,y-2.8 TO x,y-2.8
1518 LINE#2 TO x+.5,y TO x-3,y:FILL#2,0:GDisk 2,x,y
1519 END DEFine
```

```
1521 DEFine PROCedure GDrive(col,x,y)
1522 FILL#2,1:INK#2,col
1523 LINE#2,x-4,y TO x,y+2 TO x+4,y+1 TO x+4,y-1 TO x,y-3 TO x-4,y-2 TO x-4,y
1524 FILL#2,0:INK#2,0
1525 LINE#2,x-4,y TO x,y-1.4 TO x+4,y+1:LINE#2,x,y-3 TO x,y-1
1526 LINE#2,x-3.6,y-1.6 TO x-.4,y-2.6:INK#2,7
1527 END DEFine
```

```
1529 DEFine PROCedure GFolder(col,x,y)
1530 FILL#2,1:INK#2,col
1531 LINE#2,x-3,y+2 TO x-2.6,y+2.2 TO x-1,y+2.2 TO x,y+2 TO x+2,y+2
1532 LINE#2 TO x+2,y TO x+3,y TO x+2,y-1.8 TO x-3,y-1.8 TO x-3,y+1.8
1533 FILL#2,0:INK#2,0
1534 LINE#2,x-3,y-1.8 TO x-2,y TO x+3,y:INK#2,7
1535 END DEFine
```

.



TIC TAC TOE Introduction

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

QBITS Tic Tac Toe

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

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

QBITS Tic Tac Toe Strategy

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

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

QBITS TIC TAC TOE Code

```
1000 REMark QBITS_TTT_v3  (QBITS Tic Tac Toe v3 2021 QPCII)

1002 OPEN_IN#9,'ram2_QBITSConfig':INPUT#9,gx\gy\dn$:CLOSE#9
1003 MODE 8:Init_win:chk=0:QBITS_TTT

1005 DEFine PROCedure Init_win
1007 WINDOW#2,512,224,gx,gy         :PAPER#2,0:BORDER#2,1,3:CLS#2
1008 WINDOW#1,274,200,gx+224,gy+12:PAPER#1,0:BORDER#1,1,3:CLS#1
1009 WINDOW#0,512, 32,gx,gy+224   :PAPER#0,0:BORDER#0,1,3:CLS#0
1010 CSIZE#2,2,1:OVER#2,1:SCALE#1,100,0,0
1011 INK#2,2:FOR i=0 TO 1:CURSOR#2,4+i,8:PRINT#2,'QBITS Tic Tac Toe'
1012 INK#2,6:FOR i=0 TO 1:CURSOR#2,6+i,9:PRINT#2,'QBITS Tic Tac Toe'
1013 CSIZE#2,2,0:Str1$='NOUGHTS':Str2$='CROSSES' :c=1:r=2
1014 FOR a=1 TO 7
1015   FOR b=0 TO 2:CURSOR#2,224+b,42+a*16:PRINT#2,Str1$(a)
1016   FOR b=0 TO 2:CURSOR#2,478+b,42+a*16:PRINT#2,Str2$(a)
1017 END FOR a
1018 OVER#2,0:m=0:OG=0:XG=0:INK#1,5:LINE#1,3,22 TO 3,4 TO 98, 4 TO 98,22
1019 RESTORE 1021:PG=0:FG=0:LINE#1,3,78 TO 3,96 TO 98,96 TO 98,78:INK#1,248
1020 FOR i=1 TO 8:READ col,x,y,str$:INK#2,col:CURSOR#2,x,y:PRINT#2,str$
1021 DATA 6,33,168,'⬆',6,33,200,'⬇',6,9,184,'⬅  ➡'
1022 DATA 5,82,186,'Moves',5,18,62,'Played:',6,18,78,'Wins 0:'
1023 DATA 3,18,130,'Wins X:',5,18,146,'Draw  :':BLOCK#2,18,4,30,188,6
1024 FOR i=1 TO 4
1025   READ x1,y1,x2,y2,x3,y3,x4,y4
1026   FILL 1:LINE x1,y1 TO x2,y2 TO x3,y3 TO x4,y4 TO x1,y1:FILL 0
1027 END FOR i
1028 DATA 8,36,94,36,94,34,8,34, 8,66,94,66,94,64,8,64
1029 DATA 35,6,35,94,37,94,37,6, 65,6,65,94,67,94,67,6
1030 INK#2,5: LINE#2,8,10 TO 8,19 TO 18,19 TO 18,10 TO 8,10
1031 END DEFine

1033 DEFine PROCedure QBITS_TTT                                  Game Control
1034 REPeat loop
1035   IF chk=0:INK#2,5:CURSOR#2,12,106:PRINT#2,"(N)EW / (E)xit"
1036   IF chk=1:INK#2,6:CURSOR#2,12,106:PRINT#2,"O's Turn...   "
1037   IF chk=2:INK#2,3:CURSOR#2,12,106:PRINT#2,"X's Turn...   "
1038   x=11+c*30:y=r*30:Tile_Hgl x,y:k=CODE(INKEY$(-1)):Tile_Hgl x,y   calculate x-y from c-r
1039   SELect ON k
1040     =27:MODE 4:CSIZE#2,0,0:CSIZE#0,0,0:CLS#0:PRINT#0,'Bye...':STOP
1041     =69,101:IF chk=0:LRUN dn$:ELSE chk=0  :REMark (E)xit Game
1042     =77,110:Tile_CLS:chk=RND(1 TO 2)      :REMark (N)ew Game
1043     =192:c=c -1:IF c<0:c=0                :REMark Left
1044     =200:c=c+1:IF c>2:c=2                 :REMark Right
1045     =208:r =r+1:IF r >3:r=3               :REMark Up
1046     =216:r =r -1:IF r <1:r=1              :REMark Down
1047     =  32:RESTORE 1076:IF chk=1:Nought:ELSE IF chk=2:Cross
1048   END SELect
1049 END REPeat loop
1050 END DEFine
```

QBITS TicTackToe Graphics

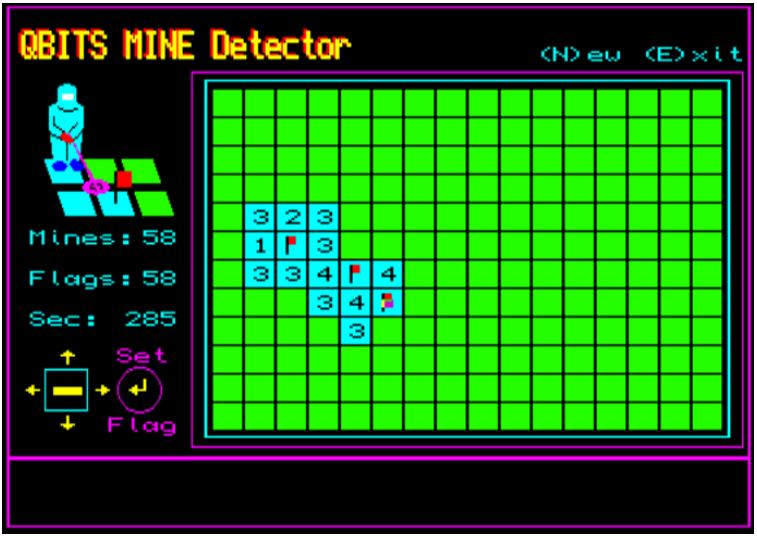
```
1052 DEFine PROCedure Tile_Hgl(x,y)
1053 INK 5:OVER -1:LINE x,y TO x+20,y TO x+20,y-20 TO x,y-20 TO x,y:OVER 0
1054 END DEFine
```

```
1056 DEFine PROCedure Nought
1057 IF Grid(c+1,r)=0:Grid(c+1,r)=79:chk=2:x=x+10:y=y-10:ELSE RETurn
1058 INK 6:FILL 1:CIRCLE x,y,7:FILL 0:INK 0:FILL 1:CIRCLE x,y,4:FILL 0
1059 BEEP 2000,5,10,0,0,0,0,0:Tile_Chk 79
1060 END DEFine
```

```
1062 DEFine PROCedure Cross
1063 IF Grid(c+1,r)=0:Grid(c+1,r)=88:chk=1:ELSE RETurn
1064 INK 3:x=x+10:y=y-10:FILL 1:LINE x-8,y+6 TO x-3,y+6 TO x+2,y TO x-3,y-6
1065 LINE TO x-8,y-6 TO x-3,y TO x-8,y+6:FILL 0:FILL 1:LINE x+8,y+6 TO x+3,y+6
1066 LINE TO x-2,y TO x+3,y-6 TO x+8,y-6 TO x+3,y TO x+8,y+6:FILL 0
1067 BEEP 2000,5,10,0,0,0,0,0:Tile_Chk 88
1068 END DEFine
```

QBITS TicTackToe Game Checks

```
1070 DEFine PROCedure Tile_Chk(n)                    Checks for three in row
1071 FOR i=1 TO 8
1072   READ a,b,c,r,d,e
1073   IF Grid(a,b)=n AND Grid(c,r)=n AND Grid(d,e)=n:Tile_Win:RETurn
1074 END FOR i
1075 m=m+1:IF m=9:m=0:n=0:Tile_Win
1076 DATA 1,1,1,2,1,3, 1,1,2,2,3,3, 1,1,2,1,3,1, 1,2,2,2,3,2
1077 DATA 3,3,3,2,3,1, 3,3,2,3,1,3, 1,3,2,2,3,1, 2,1,2,2,2,3
1078 END DEFine

1080 DEFine PROCedure Tile_Win
1081 chk=0  :PG=PG+1:INK#2,5:CURSOR#2,124, 64:PRINT#2,FILL$(' ',2-LEN(PG))&PG
1082 IF n=79:OG=OG+1:INK#2,6:CURSOR#2,124, 80:PRINT#2,FILL$(' ',2-LEN(OG))&OG
1083 IF n=88:XG=XG+1:INK#2,3:CURSOR#2,124,132:PRINT#2,FILL$(' ',2-LEN(XG))&XG
1084 IF n=0 :FG=FG+1:INK#2,5:CURSOR#2,124,148:PRINT#2,FILL$(' ',2-LEN(FG))&FG
1085 BEEP 5000,20,12,40,8,8,0,0
1086 END DEFine

1088 DEFine PROCedure Tile_CLS
1089 DIM Grid(3,3):m=0:n=0:c=1:r=2:RESTORE 1093:INK 0
1090 FOR i=1 TO 9
1091   READ x,y,x1,y1:FILL 1:LINE x,y TO x1,y TO x1,y1 TO x,y1 TO x,y:FILL 0
1092 END FOR i
1093 DATA 11,30,31,10, 41,30,61,10, 71,30,91,10, 11,60,31,40, 41,60,61,40
1094 DATA 71,60,91,40,11,90,31,70, 41,90,61,70,71,90,91,70
1095 END DEFine
```



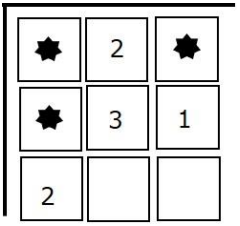
Mine Detector Introduction

Minesweeper was first introduced as a Mainframe Game of the 1960 and 1970's it then became popular as part of the Puzzle Video Game Genre during the 1980's. Each Game starts out with a grid of unmarked squares. You **Click** on a square to reveal its **Status** and some of the surrounding squares or **Mark** it with a **Flag** as one holding a **Mine**.

As a Single-player Puzzle Game, the object is to Locate and **Mark** all the "**Mines**". Success is being able to eliminate all possible positions of the distributed Mines within the shortest time. If a player **Click's** a square that is **mined**, the game ends.

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

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



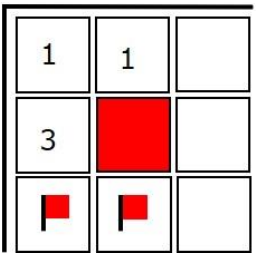
Apart from squares adjacent to the boundaries and comers of the board, a square has the possibility of each of the surrounding squares holding a mine. Counting the possible mines this would be between zero and eight.

Minesweeper Logic or Probability

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

Local Probabilities

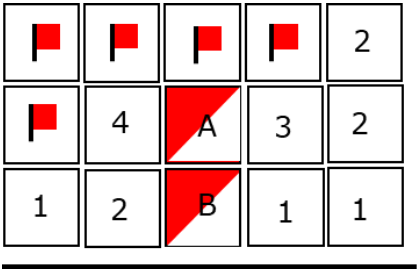
The squares shown have two with **Flags** where mines have been identified. For **Square** with the number **3** only one other **Mine** needs to be identified and this can only be in the Square shown in RED.



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

Local Probability Conflicts

The squares with a **Flag** are those with a **Mine**. Squares in **RED/WHITE** are where the other **Mine(s)** maybe located. This is implied by the numbers 4,3,2,1 shown in the adjacent squares.



Note: Squares surrounding the one containing a 4 already has four Flags, so its true value must be 5. It can't be 6 as this would break with the numbering of the other Squares adjacent to A B.

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

QBITS APM_Sweeper

In the 1980's the QBITS game was called **Anti-Personnel Mine Sweeper.** However today with many Charities and Government Organisation providing schemes for clearing minefields of old and recent conflicts, the Title of **Mine Detector** is perhaps more explicit.

QBITS Mine Detector Game

Press (**N**) to start a New Game. The Aim of the Game is to **Mark** with a **Flag** all of the **Mines** in the grid and complete this within the Time Limit of **300sec** (**GTime**) which can be altered (see code line 1003).

Use the Cursor keys to move the highlight to a selected square. Use <**Enter**> to **Mark** with a **Flag** or <**Spacebar**> to reveal Squares **Status** or it could be a **Mine** which will end your Game**.** To add more difficulty **QBITS Mine Detector** counts surrounding mines to a max of four. Therefore, it is necessary to review several adjacent squares to determine if a mine is present. On a few occasions this will simply come down to a best guess.

QBITS Mine Detector Code

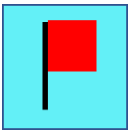
```
1000 REMark QBMDETR_v3 (QBITS Mine Detector v3 2021 QPCII)

1002 OPEN_IN#9,'ram2_QBITSConfig':INPUT#9,gx\gy\dn$:CLOSE#9
1003 MODE 8:GTime=300:Init_win:MF_Game

1005 DEFine PROCedure Init_win
1006 WINDOW#2,512,222,gx,gy              :PAPER#2,0:BORDER#2,1,3:CLS#2
1007 WINDOW#1,362,176,gx+136,gy+36:PAPER#1,0:BORDER#1,1,5
1008 WINDOW#0,512, 34,gx,gy+222        :PAPER#0,0:BORDER#0,1,3:CLS#0
1009 CSIZE#2,2,1:OVER#2,1:SCALE#1,120,0,0:CSIZE#0,2,0
1010 INK#2,2:FOR i=0 TO 1:CURSOR#2,4+i,8:PRINT#2,'QBITS MINE Detector'
1011 INK#2,6:FOR i=0 TO 1:CURSOR#2,6+i,9:PRINT#2,'QBITS MINE Detector'
1012 CSIZE#2,2,0:OVER#2,0
1013 INK#2,3:LINE#2,42,2 TO 42,86 TO 170,86 TO 170,2 TO 42,2:RESTORE
1014 FOR i=1 TO 10:READ col,x,y,str$:INK#2,col:CURSOR#2,x,y:PRINT#2,str$
1015 DATA 6,33,166,'🡅',6,33,198,'🡇',6,9,182,'🡄  🡆',6,80,182,'¼'
1016 DATA 5,360,18,'(N)ew (E)xit',3,72,166,'Set',3,66,200,'Flag'
1017 DATA 5,12,108,'Mines:',5,12,128,'Flags:',5,12,148,'Sec:'
1018 BLOCK#2,20,4,30,186,6:BLOCK#2, 2,6,92,182,6
1019 INK#2,5:LINE#2,8,10 TO 8,19 TO 18,19 TO 18,10 TO 8,10:INK#2,3
1020 CIRCLE#2,30,14.5,5:MF_Detector 2,8,82:INK#2,5
1021 END DEFine

1023 DEFine PROCedure MF_Game
1024 DIM mes$(4,40):MF_Seed:chk=0
1025 REPeat main
1026  IF chk=1
1027    CURSOR#2,78,148:PRINT#2,FILL$(' ',3-LEN(gsec))&gsec
1028    gsec=GTime-(DATE-OldTime):IF gsec=0:End_Game
1029  END IF
1030  x1=x:y1=y:OVER#1,-1  :BLOCK#1,8,4,(a+(x1-1)*w),(b+(y1-1)*h),6
1031  k=CODE(INKEY$(20)) :BLOCK#1,8,4,(a+(x1-1)*w),(b+(y1-1)*h),6:OVER#1,0
1032  SELect ON k
1033    =192:x=x -1:BEEP 400:IF x< 1:x=1
1034    =200:x=x+1:BEEP 400:IF x>16:x=16
1035    =208:y=y -1:BEEP 400:IF y< 1:y=1
1036    =216:y=y+1:BEEP 400:IF y>12:y=12
1037    = 69,101:LRUN dn$                                    :REMark (E)xit
1038    = 77,109:MF_Locate:mines=255:End_Game          :REMark Show Mines
1039    = 78,110:chk=1:CLS#0:MF_Seed:OldTime=DATE      :REMark (N)ew Game
1040    = 10:IF chk=1:MF_Mark:IF mines=0:End_Game      :REMark Mark mines
1041    = 32:IF chk=1:IF Board(x,y)=255:MF_Explode:ELSE  :MF_Show
1042    = 27:MODE 4:CSIZE#2,0,0:INK#2,7:CSIZE#0,0,0:INK#0,7:PRINT#0,'Bye':STOP
1043  END SELect
1044 END REPeat main
1045 END DEFine
```

```
1047 DEFine PROCedure MF_Show
1048 IF y-1>=1 AND x-1>=1   :x1=x-1:y1=y-1   :MF_Status:END IF
1049 IF y-1>=1              :x1=x  :y1=y-1   :MF_Status:END IF
1050 IF y-1>=1 AND x+1<=16 :x1=x+1:y1=y-1   :MF_Status:END IF
1051 IF x-1>0               :x1=x-1:y1=y     :MF_Status:END IF
1052 x1=x:y1=y :MF_Status
1053 IF x+1<=16             :x1=x+1:y1=y     :MF_Status
1054 IF y+1<=12 AND x-1>=1 :x1=x-1:y1=y+1   :MF_Status
1055 IF y+1<=12             :x1=x  :y1=y+1   :MF_Status
1056 IF y+1<=12 AND x+1<=16:x1=x+1:y1=y+1 :MF_Status
1057 END DEFine
```

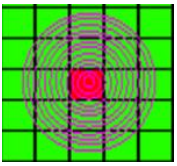
Note: S/SuperBASIC single Line Entries for **IF** Statements or **FOR** loops the **END IF** / **END FOR** are assumed.

```
1059 DEFine PROCedure MF_Status
1060 IF Board(x1,y1)>0 AND Board(x1,y1)<200
1061    BLOCK#1,w-2,h-1,4+(x1-1)*w,4+(y1-1)*h,5
1062    CURSOR#1,7+w DIV 21+(x1-1)*w,-1+h DIV 2+(y1-1)*h
1063    STRIP#1,5:INK#1,0:PRINT#1,Board(x1,y1)
1064 END IF
1065 END DEFine
```

```
1067 DEFine PROCedure MF_Mark
1068 IF Board(x,y)=255
1069    Board(x,y)=200:mines=mines-1:flags=flags-1:x1=10+(x-1)*w:y1=6+(y-1)*h
1070    BLOCK#1,w-2,h-1,x1-6,y1-2,5:BLOCK#1,7,4,x1,y1,2:BLOCK#1,2,9,x1,y1,0
1071 END IF
1072 CURSOR#2,90,108:PRINT#2,FILL$(' ',2-LEN(mines))&mines
1073 CURSOR#2,90,128:PRINT#2,FILL$(' ',2-LEN(flags))&flags
1074 END DEFine
```

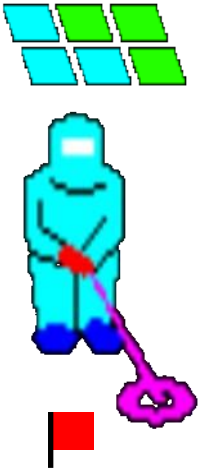


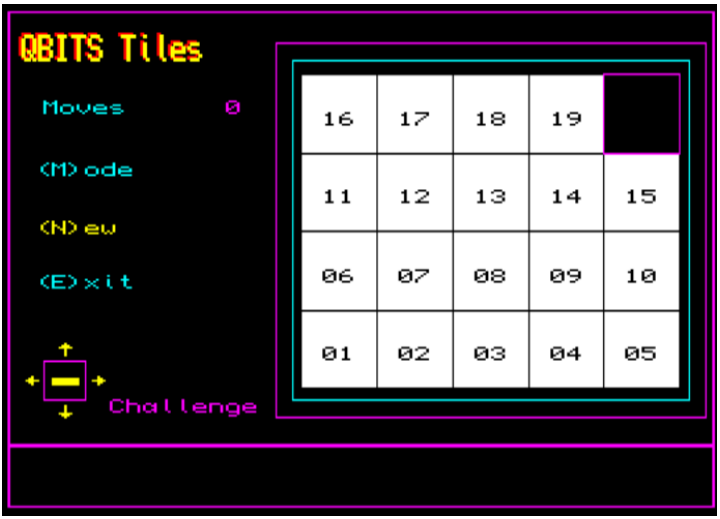
```
1076 DEFine PROCedure MF_Explode
1077 BLOCK#1,w-2,h-1,4+(x-1)*w,4+(y-1)*h,2
1078 BEEP 0,100,50,1,100,6,15,15:hscale=(16/12)*116
1079 INK#1,3:FOR i=1 TO 12:CIRCLE#1,x*11.2-4,(12-y+1)*9.8-4,1.5*i
1080 PAUSE 80:BEEP:End_Game
1081 END DEFine
```



```
1083 DEFine PROCedure End_Game
1084 CLS#0:gsec=GTime-gsec
1085 mes$(1)='You left '&mines&' mines to find...   '
1086 mes$(2)=' after '&gsec&' seconds of play...    '
1087 mes$(3)=' leaving '&flags&' marker Flags...    '
1088 mes$(4)='     Game Over - press any key...    '
1089 IF mines=0:mes$(1)='Well done all Mines Cleared...  '
1090 IF mines<255
1091  FOR m=1 TO 4
1092   FOR i=1 TO LEN(mes$(m)):CURSOR#0,490,5:PRINT#0;mes$(m,i):PAUSE 5:PAN#0,-12
1093  END FOR m
1094 END IF
1095 k$=INKEY$(-1):CLS#0:MF_Seed:OldTime=DATE
1096 END DEFine
```

```
1098 DEFine PROCedure MF_Seed
1099 DIM Board(17,13):w=22:h=14:mct=0:mines=0:flags=0:CLS#1
1100 FOR m=1 TO 72:x=RND(1 TO 16):y=RND(1 TO 12):Board(x,y)=255      Random Mine Distribution
1101 FOR y=1 TO 12                                                  Gather Info to Identify Mine Locations
1102   FOR x=1 TO 16                                                and Number of distributed Mines
1103     IF Board(x-1, y-1 )=255:mct=mct+1
1104     IF Board(x,   y-1 )=255:mct=mct+1
1105     IF Board(x+1,y-1 )=255:mct=mct+1
1106     IF Board(x-1 ,y   )=255:mct=mct+1
1107     IF Board(x+1,y    )=255:mct=mct+1
1108     IF Board(x-1, y+1)=255:mct=mct+1
1109     IF Board(x,   y+1)=255:mct=mct+1
1110     IF Board(x+1,y+1)=255:mct=mct+1
1111     IF mct>4:mct=4
1112     IF Board(x,y)=255:mines=mines+1:flags=flags+1:ELSE Board(x,y)=mct
1113     BLOCK#1,w-2,h-1,4+(x-1)*w,4+(y-1)*h,4:mct=0
1114   END FOR x
1115 END FOR y
1116 y=RND(4 TO 8):x=RND(2 TO 4):a=-1+w DIV 2:b=2+h DIV 2:MF_Mark:k=0
1117 END DEFine

1119 DEFine PROCedure MF_Locate                                     end Game Show Mine distribution
1120 FOR y=1 TO 12
1121   FOR x=1 TO 16:x1=x:y1=y:IF Board(x,y)=255 OR Board(x,y)=200:MF_Mark
1122 END FOR y
1123 END DEFine

1125 DEFine PROCedure MF_Detector(ch,x,y)
1126 RESTORE 1127:FOR i=1 TO 6:READ col,tx,ty:MF_Tile 2,col,7,5,tx,ty
1127 DATA 5,x,y-16,4,x+9,y-16,4,x+18,y-16,5,x+3,y-23,5,x+12,y-23,4,x+21,y-23
1128 MF_PPE 2,5,x+5,y
1129 END DEFine

1131 DEFine PROCedure MF_Tile(ch,col,tw,td,tx,ty)
1132 INK#ch,col:x1=tx:x2=tx+tw:x3=tx+tw+tw/4:x4=tx+tw/4:y1=ty:y2=ty-td
1133 FILL#ch,1:LINE#ch,x1,y1 TO x2,y1 TO x3,y2 TO x4,y2 TO x1,y1:FILL#ch,0
1134 END DEFine

1136 DEFine PROCedure MF_PPE(ch,col,x,y)
1137 INK#ch,col:FILL#ch,1
1138 ARC#ch,x+2,y TO x-2,y,PI/2:LINE#ch TO x-2,y-3 TO x-4,y-4 TO x-4,y-8
1139 LINE#ch TO x-3,y-16 TO x+3,y-16 TO x+4,y-8 TO x+4,y-4 TO x+2,y-3
1140 LINE#ch TO x+2,y:FILL#ch,0:INK#ch,7:FILL#ch,1
1141 LINE#ch,x+1.5,y-1 TO x+1.5,y-2 TO x-1,y-2 TO x-1,y-1 TO x+1.5,y-1:FILL#ch,0
1142 INK#ch,0:LINE#ch,x,y-16 TO x,y-11:LINE#ch,x-4,y-10 TO x-1,y-11
1143 LINE#ch,x-3,y-6 TO x-3,y-8 TO x,y-10:LINE#ch,x+4,y-9 TO x+1,y-12
1144 LINE#ch,x+.5,y-10 TO x+3,y-7:ARC#ch,x-2,y-4 TO x+3,y-4,PI/2
1145 INK#ch,1:FILL#ch,1:CIRCLE#ch,x-1.5,y-17,1.6,.6,PI/2:FILL#ch,0
1146 INK#ch,1:FILL#ch,1:CIRCLE#ch,x+2.5,y-17,1.6,.6,PI/3:FILL#ch,0
1147 INK#ch,3:FILL#ch,1:CIRCLE#ch,x+7,y-22,3,.5,PI/2:FILL#ch,0
1148 INK#ch,0:FILL#ch,0:CIRCLE#ch,x+7,y-22,1.5,.4,PI/2:FILL#ch,0
1149 INK#ch,3:FILL#ch,1:LINE#ch,x+.4,y-10 TO x+7,y-22 TO x+7.4,y-22 TO x+.4,y-10
1150 FILL#ch,0:INK#ch,2:FILL#ch,1:CIRCLE#ch,x,y-11,1.5,.5,PI/3:FILL#ch,0
1151 BLOCK#ch,12,8,72,80,2:BLOCK#ch,2,16,72,80,0
1152 END DEFine
```





Sliding Tiles Introduction

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 introduced to form words or even statements. The 1980's Popularity was further enhanced by the Rubik Cube a Rotational three-dimensional version of the Sliding Tile Puzzle.

Early puzzles were Tokens on a flat board; the Rules prohibited lifting any piece 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 facilitate these restrictions.

Sam Loyd introduced his version of the 15-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, sliding puzzles 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 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 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

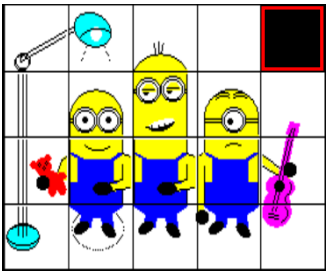
Blank

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

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

QBITS Sliding Tile Picture

For the challenge of replacing Numbered Tiles with an Image I used Vector Graphics. Each Tile is drawn with a Graphic combination that when sequenced correctly creates a picture. By using S/SuperBASIC Graphic coordinates in part explains the reason why QBITS Tile numbering begins bottom left, progressing to top right.



QBITS Tile Moves

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

QBITS Sliding Tile Permutations

The act of changing the arrangement of a given number of elements is a permutation. The number of moves to position a Tile in its correct row and column is **n** the number of tiles in the set, a maximum number of moves for each tile is **n**-1 and each switch involves two Tile positions, this equates to **n2**(**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 moves by an average player should fall somewhere between.

After reviewing a series of games, the average number of Inversion was around 200, and

suggested an inexperienced player would be lucky to complete in less than 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 level of the player.

QBITS Tile Strategy

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

QBITS Solving the Puzzle

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

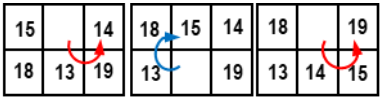
16	17	18	19	
11	12	13	14	15
06	07	08	09	10
01	02	03	04	05

Steps 1: Position Tiles correctly for Column One: 01,06,11,16. Reduces Grid to 4x4
Steps 2: Complete the bottom Row: 01 02 03 04 05 and then Column Two: 02 07 12 17
Steps 3: Reduce to 3x3. Complete Row Two: 06 07 08 09 10 & Column Three: 03 08 13 18
Steps 4: Solve the remaining 2x2 grid at top right of Puzzle.

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

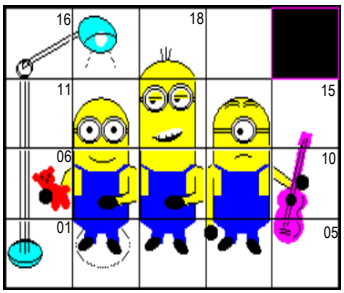
```
1000 REMark QBITS_Tiles_v3 (QBITS Tiles v3 2021 QPCII)

1002 OPEN_IN#9,'ram2_QBITSConfig':INPUT#9,gx,gy,dn$:CLOSE#9
1003 DIM Tile(20,3):cp=3:MODE 8:Init_win:QBITS_Tiles

1005 DEFine PROCedure Init_win
1006 WINDOW#2,512,224,gx,gy          :PAPER#2,0:BORDER#2,1,3:CLS#2:INK#2,3
1007 WINDOW#1,280,167,gx+212,gy+28:SCALE#1,164,0,0:CSIZE#1,2,0
1008 WINDOW#0,512, 32,gx,gy+224   :BORDER#0,1,3:PAPER#0,0:CLS#0
1009 LINE#2,64,6 TO 168,6 TO 168,93 TO 64,93 TO 64,6:INK#2,5
1010 LINE#2,68,10 TO 164,10 TO 164,89 TO 68,89 TO 68,10
1011 CSIZE#2,2,1:OVER#2,1
1012 INK#2,2:FOR i=0 TO 1:CURSOR#2,4+i,8:PRINT#2,'QBITS Tiles'
1013 INK#2,6:FOR i=0 TO 1:CURSOR#2,6+i,9:PRINT#2,'QBITS Tiles'
1014 CSIZE#2,2,0:OVER#2,0:RESTORE 1016
1015 FOR i=1 TO 8:READ col,x,y,str$:INK#2,col:CURSOR#2,x,y:PRINT#2,str$
1016 DATA 5,22,44,'Moves',5,16,76,'(M)ode',6,16,106,'(N)ew',5,16,134,'(E)xit'
1017 DATA 3,70,198,'Challenge',6,33,168,' ↑ ',6,33,200,' ↓ ',6,9,184,' ← → '
1018 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
1019 END DEFine

1021 DEFine PROCedure QBITS_Tiles
1022 t=0:Tsel=0:Init_Tiles:Set_Tiles:tc=4:tr=3:m=0
1023 REPeat G_lp
1024  INK#1,3:Tile_Hgl 1,40,40,tc*40,tr*40
1025  IF m<999:CURSOR#2,130,44:PRINT#2,FILL$(' ',3-LEN(m))&m
1026  k=CODE(INKEY$(-1)):INK#1,0:Tile_Hgl 1,40,40,tc*40,tr*40
1027  SELect ON k
1028   = 27:MODE 4:CSIZE#2,0,0:INK#2,7:CSIZE#0,0,0:CLS#0:PRINT#0,'Bye':STOP
1029   = 32:Tile_Chg:IF chk=1:m=m+1:Tile_Draw 1,ts:Tile_Draw 1,tn:Game_Chk
1030   = 69,101:GExit
1031   = 77,109:IF Tsel=0:Tsel=1:Set_Tiles:ELSE Tsel=0:Set_Tiles
1032   = 78,110 :t=0:Init_Tiles:Set_Tiles:PAUSE 50:Sort_Tiles:Set_Tiles
1033   =192:tc=tc -1:IF tc<0:tc=0
1034   =200:tc=tc+1:IF tc>4:tc=4
1035   =208:tr =tr+1:IF tr>3:tr=3
1036   =216:tr =tr -1:IF tr<0:tr=0
1037  END SELect
1038 END REPeat G_lp
1039 END DEFine

1041 DEFine PROCedure Game_Chk
1042 cnt=0:FOR i=1 TO 20:IF Tile(i,1)=i:cnt=cnt+1
1043 IF m>999 OR cnt=20
1044    CURSOR#2,90,106:PRINT#2,'Game End':PAUSE
1045    BLOCK#2,100,10,90,106,0:BLOCK#2,40,10,132,184,0:m=0:t=0
1046 END IF
1047 END DEFine
```

```
1049 DEFine PROCedure GExit
1050 CURSOR#2,96,134:PRINT#2,'Y/N':PAUSE:IF KEYROW(5)=64:LRUN dn$
1051 BLOCK#2,40,10,96,134,0
1052 END DEFine

1054 DEFine PROCedure Tile_Hgl(ch,w,d,x,y)
1055 LINE#ch,x,y TO x+w,y TO x+w,y+d TO x,y+d TO x,y
1056 END DEFine

1058 DEFine PROCedure Tile_Chg
1059 tn=1+tc+tr*5:chk=0 : IF Tile(tn,1)=20            :RETurn
1060 IF tc>0:ts=tn-1            :Tile_Chk ts:IF chk=1:RETurn
1061 IF tc<4:ts=tn+1            :Tile_Chk ts:IF chk=1:RETurn
1062 IF tr>0:ts=1+tc+(tr-1)*5 :Tile_Chk ts:IF chk=1:RETurn
1063 IF tr<3:ts=1+tc+(tr+1)*5:Tile_Chk ts:IF chk=1:RETurn
1064 END DEFine

1066 DEFine PROCedure Tile_Chk(ts)
1067 IF Tile(ts,1)=20:Tile(ts,1)=Tile(tn,1):Tile(tn,1)=20:chk=1
1068 END DEFine

1070 DEFine PROCedure Tile_Draw(ch,ts)
1071 IF Tile(ts,1)<20:STRIP#ch,7:INK#ch,7:ELSE STRIP#ch,0:INK#ch,0
1072 BEEP 2000,5,10,0,0,0,0,0:t$=Tile(ts,1):x=Tile(ts,2):y=Tile(ts,3)
1073 FILL#ch,1:LINE#ch,x,y TO x+40,y+1 TO x+40,y+40 TO x,y+40 TO x,y:FILL#ch,0
1074 INK#ch,0:Tile_Hgl 1,40,40,x,y:IF Tsel=1:Tile_Pic ts,x,y
1075 IF Tsel=0:CURSOR#ch,x,y,14,-22:PRINT#ch,FILL$('0',2-LEN(t$))&t$
1076 END DEFine

1078 DEFine PROCedure Init_Tiles
1079 FOR r=0 TO 3
1080    FOR c=0 TO 4:t=t+1:Tile(t,1)=t:Tile(t,2)=c*40:Tile(t,3)=r*40
1081 END FOR r
1082 END DEFine

1084 DEFine PROCedure Set_Tiles
1085 BLOCK#2,64,10,120,164,0:ch=1:PAPER#ch,0:CLS#ch:FOR t=1 TO 20:Tile_Draw 1,t
1086 END DEFine

1088 DEFine PROCedure Sort_Tiles
1089 FOR t=20 TO 3 STEP -1
1090   ran=RND(1 TO t-1):temp=Tile(t,1):Tile(t,1)=Tile(ran,1):Tile(ran,1)=temp
1091 END FOR t
1092 m=0:ct=0:Solvable
1093 FOR t=1 TO 20:IF Tile(t,1)=20:tc=Tile(t,2)/40:tr=Tile(t,3)/40
1094 END DEFine

1096 DEFine PROCedure Solvable
1097 FOR t=1 TO 19
1098   IF Tile(t,1)=20:NEXT t:ELSE FOR c=t+1 TO 20:IF Tile(t,1)>Tile(c1,1):ct=ct+1
1099 END FOR t
1100 CURSOR#2,130,184:PRINT#2,INT(ct+ct/cp):IF ct MOD 2>0:Sort_Tiles
1101 END DEFine
```

Challenge - Number of Turns

QBITS Minions Graphics

Creating a Tiled Picture gave the opportunity to try out Vector Graphics instead of using Pixel Bitmaps. The result will give variations of picture quality across the range of QL Platforms and screen changes Performing better with Processors running10x or more than the Original QL.




```
1103 DEFine PROCedure Tile_Pic(ts,x,y)
1104 ch=1:tp=Tile(ts,1)
1105 SELect ON tp
1106  = 1:ML5 x,y
1107  = 2:ML1 x,y:MB2 x,y
1108  = 3:MB2 x,y
1109  = 4:MB2 x+5,y:MA3 x,y
1110  = 5:MG3 x,y
1111  = 6:MT x+11,y-6:ML4 x,y
1112  = 7:MB1 x,y:MS1 x,y:MA2 x,y
1113  = 8:MB1 x,y:MA2 x,y
1114  = 9:MB1 x+5,y:MS2 x,y:MA1 x,y
1115  =10:MG2 x,y
1116  =11:ML4 x,y
1117  =12:MH3 x,y:ME1 10,10,x,y:ME2 10,10,x,y:ME1 25,10,x,y:ME2 25,10,x,y
1118  =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
1119  =14:MH3 x+5,y:ME1 19,10,x+5,y:ME2 17,10,x+5,y-1:MH4 x+5,y+1
1120  =15:MG1 x,y
1121  =16:ML3 x,y
1122  =17:ML2 x,y
1123  =18:MH1 x,y
1124 END SELect
1125 END DEFine
```

```
1127 DEFine PROCedure MH1(x,y)          :REMark Head Top
1128 INK#ch,6:FILL#ch,1:LINE#ch,x+1,y+1 TO x+34,y+1
1129 ARC#ch TO x+1,y+1,PI/1.5:FILL#ch,0:INK#ch,0
1130 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
1131 LINE#ch,x+14,y+10 TO x+13,y+17:LINE#ch,x+18,y+10 TO x+19,y+17
1132 END DEFine
```

```
1134 DEFine PROCedure MH2(x,y)          :REMark Head Long
1135 INK#ch,6:FILL#ch,1:LINE#ch,x+1,y+1 TO x+34,y+1 TO x+34,y+39
1136 LINE#ch TO x+34,y+39 TO x+1,y+39 TO x+1,y+1:FILL#ch,0
1137 INK#ch,0:LINE#ch,x+34,y+1 TO x+34,y+39
1138 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
1139 FILL#ch,0:REMark MS3 x,y
1140 END DEFine
```

```
1142 DEFine PROCedure MH3(x,y)          :REMark Head Average
1143 INK#ch,6:FILL#ch,1:ARC#ch,x+1,y+12 TO x+34,y+12,-PI
1144 LINE#ch TO x+34,y+1 TO x+1,y+1 TO x+1,y+12:FILL#ch,0:INK#ch,0
1145 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
1146 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
1147 END DEFine
```

```
1149 DEFine PROCedure MH4(x,y)              :REMark Hair
1150 INK#ch,0
1151 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
1152 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
1153 END DEFine
```

```
1155 DEFine PROCedure ME1(w,d,x,y)          :REMark Eye Cover
1156 INK#ch,7:FILL#ch,1:CIRCLE#ch,x+w,y+d,7:FILL#ch,0
1157 INK#ch,0:CIRCLE#ch,x+w,y+d,7.4:CIRCLE#ch,x+w,y+d,6
1158 END DEFine
```

```
1160 DEFine PROCedure ME2(w,d,x,y)          :REMark Eye
1161 FILL#ch,1:CIRCLE#ch,x+w,y+d,1.6:FILL#ch,0
1162 END DEFine
```

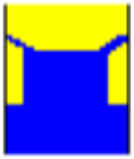
```
1164 DEFine PROCedure ME3(w,d,x,y)          :REMark Eye Lid
1165 INK#ch,0:ME2 w,d-1,x-2,y:INK#ch,6:FILL#ch,1
1166 ARC#ch,x+w-4,y+d TO x+w+4,y+d,-PI:LINE#ch TO x+w-4,y+d:FILL#ch,0
1167 INK#ch,0:LINE#ch,x+w-4,y+d+1 TO x+w+4,y+d+1
1168 END DEFine
```

```
1170 DEFine PROCedure MS1(x,y)              :REMark Smile
1171 INK#ch,0:ARC#ch,x+10,y+35 TO x+26,y+35,PI/2
1172 END DEFine
```

```
1174 DEFine PROCedure MS2(x,y)              :REMark Frown
1175 INK#ch,0:ARC#ch,x+18,y+34 TO x+28,y+34,-PI/2
1176 END DEFine
```

```
1178 DEFine PROCedure MS3(x,y)              :REMark Teeth
1179 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
1180 INK#ch,7:FILL#ch,1:CIRCLE#ch,x+18,y+8,8,.2,-PI/2.4:FILL#ch,0
1181 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
1182 LINE#ch,x+12,y+7 TO x+12,y+4
1183 END DEFine
```

```
1185 DEFine PROCedure MB1(x,y)              :REMark Body Trunk
1186 FILL#ch,1:INK#ch,6
1187 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
1188 FILL#ch,0:FILL#ch,1:INK#ch,1
1189 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
1190 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
1191 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
1192 FILL#ch,0:FILL#ch,1
1193 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
1194 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
1195 END DEFine
```



```
1197 DEFine PROCedure MB2(x,y)              :REMark Body Feet
1198 INK#ch,1:FILL#ch,1:LINE#ch,x+1,y+39 TO x+34,y+39
1199 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
1200 LINE#ch TO x+14,y+26 TO x+8,y+26 TO x+8,y+34:FILL#ch,0:FILL#ch,1
1201 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
1202 FILL#ch,0:FILL#ch,1:INK#ch,0:CIRCLE#ch,x+9,y+25,5,.6,-PI/3
1203 FILL#ch,0:FILL#ch,1:CIRCLE#ch,x+24,y+24,5,.6,PI/4:FILL#ch,0
1204 END DEFine
```

```
1206 DEFine PROCedure MA1(x,y)              :REMark Arm Straight
1207 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
1208 LINE#ch TO x+5,y+24 TO x+5,y+30 TO x+1,y+28:FILL#ch,0
1209 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
1210 END DEFine
```

```
1212 DEFine PROCedure MA2(x,y)              :REMark Arm Left
1213 INK#ch,6:FILL#ch,1
1214 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
1215 LINE#ch TO x+34,y+12 TO x+34,y+30:FILL#ch,0:FILL#ch,1:INK#ch,0
1216 CIRCLE#ch,x+21,y+9,6,.6,PI/2:FILL#ch,0:LINE#ch,x+34,y+30 TO x+39,y+28
1217 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
1218 END DEFine
```

```
1220 DEFine PROCedure MA3(x,y)              :REMark Arm Right
1221 INK#ch,6:FILL#ch,1:LINE#ch,x+1,y+39 TO x+4,y+39 TO x+4,y+36
1222 LINE#ch TO x+1,y+36 TO x+1,y+39:FILL#ch,0
1223 INK#ch,0:FILL#ch,1:CIRCLE#ch,x+3,y+32,4:FILL#ch,0
1224 END DEFine
```

```
1226 DEFine PROCedure MT(x,y)               :REMark Teddy Bear
1227 INK#ch,6:FILL#ch,1
1228 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
1229 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
1230 FILL#ch,1:CIRCLE#ch,x+14,y+28,5:FILL#ch,0                  :REMark head
1231 FILL#ch,1:CIRCLE#ch,x+ 8,y+29,2:FILL#ch,0                  :REMark ear 1
1232 FILL#ch,1:CIRCLE#ch,x+17,y+33,2:FILL#ch,0                  :REMark ear 2
1233 FILL#ch,1:CIRCLE#ch,x+20,y+20,8,.6,PI/3:FILL#ch,0          :REMark body
1234 FILL#ch,1:CIRCLE#ch,x+26,y+22,3,.6,PI/2:FILL#ch,0          :REMark arm 1
1235 FILL#ch,1:CIRCLE#ch,x+20,y+14,4,.6,PI:FILL#ch,0            :REMark leg 1
1236 FILL#ch,1:CIRCLE#ch,x+26,y+14,3,.8,PI:FILL#ch,0:INK#ch,0   :REMark hand
1237 FILL#ch,1:CIRCLE#ch,x+14,y+18,4:FILL#ch,0                  :REMark leg 2
1238 CIRCLE#ch,x+12,y+29,1:CIRCLE#ch,x+16,y+30,1:CIRCLE#ch,x+15,y+27,1
1239 END DEFine
```

```
1241 DEFine PROCedure ML1(x,y)              :REMark Lamp Highlight
1242 INK#ch,248:CIRCLE#ch,x+18,y+25,16,.8,PI/2
1243 END DEFine
```

```
1245 DEFine PROCedure ML2(x,y)          :REMark Lamp Head
1246 INK#ch,0:LINE#ch,x+1,y+25 TO x+10,y+30 TO x+10,y+27 TO x,y+22
1247 INK#ch,5:FILL#ch,1:CIRCLE#ch,x+15,y+25,12,.8,PI/3:FILL#ch,0
1248 INK#ch,7:FILL#ch,1:CIRCLE#ch,x+17,y+22,4,.8,PI:FILL#ch,0:INK#ch,0
1249 CIRCLE#ch,x+15,y+25,12,.8,PI/3:ARC#ch,x+7,y+19 TO x+26,y+23,-PI/2
1250 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
1251 END DEFine
```

```
1253 DEFine PROCedure ML3(x,y)          :REMark Lamp Arm
1254 INK#ch,0:LINE#ch,x+39,y+24 TO x+10,y+8 TO x+10,y+5 TO x+39,y+21
1255 CIRCLE#ch,x+11,y+5,4.5:CIRCLE#ch,x+12,y+6,4.5
1256 END DEFine
```

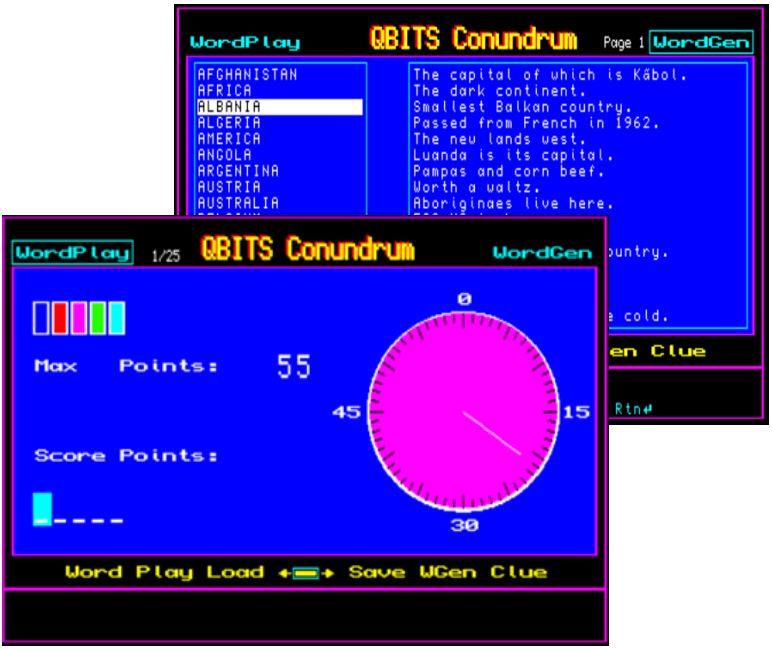
```
1258 DEFine PROCedure ML4(x,y)          :REMark Lamp Stand
1259 INK#ch,0:LINE#ch,x+9,y+1 TO x+9,y+39
1260 LINE#ch,x+12,y+1 TO x+12,y+39:LINE#ch,x+14,y+1 TO x+14,y+39
1261 END DEFine
```

```
1263 DEFine PROCedure ML5(x,y)          :REMark Lamp Base
1264 INK#ch,5:FILL#ch,1:CIRCLE#ch,x+12,y+20,10,.8,PI/2:FILL#ch,0
1265 INK#ch,0:CIRCLE#ch,x+12,y+20,10,.8,PI/2:CIRCLE#ch,x+12,y+21,10,.6,PI/2
1266 LINE#ch,x+9,y+21 TO x+9,y+39:LINE#ch,x+12,y+20 TO x+12,y+39
1267 LINE#ch,x+14,y+21 TO x+14,y+39
1268 END DEFine
```

```
1270 DEFine PROCedure MG1(x,y)          :REMark Guitar Top
1271 FILL#ch,1:INK#ch,3:LINE#ch,x+17,y+1 TO x+16,y+2 TO x+18,y+9
1272 LINE#ch TO x+24,y+8 TO x+22,y+1 TO x+17,y+1:FILL#ch,0:INK#ch,0
1273 LINE#ch TO x+18,y+1 TO x+19,y+6:LINE#ch,x+20,y+1 TO x+21,y+6
1274 END DEFine
```

```
1276 DEFine PROCedure MG2(x,y)          :REMark Guitar Middle
1277 FILL#ch,1:INK#ch,6:LINE#ch,x+1,y+28 TO x+1,y+24 TO x+15,y+14
1278 LINE#ch,x+18,y+14 TO x+1,y+28:FILL#ch,0:INK#ch,0
1279 LINE#ch,x+1,y+22 TO x+15,y+14:LINE#ch,x+1,y+29 TO x+16,y+18
1280 FILL#ch,1:INK#ch,3:LINE#ch, x+10,y+10 TO x+16,y+39 TO x+20,y+39
1281 LINE#ch TO x+14,y+10 TO x+10,y+10:FILL#ch,0
1282 INK#ch,3:FILL#ch,1:CIRCLE#ch,x+12,y+10,9,.7,PI/2.2:FILL#ch,0
1283 FILL#ch,1:ARC#ch,x+18,y TO x+2,y,PI:LINE#ch TO x+2,y:FILL#ch,0
1284 INK#ch,0:FILL#ch,1:CIRCLE#ch,x+11,y+10,3:FILL#ch,0
1285 ARC#ch,x+18,y+12 TO x+16,y+4,-PI/2 TO x+20,y+1,PI
1286 LINE#ch,x+8,y+1 TO x+17,y+39:LINE#ch,x+10,y+1 TO x+19,y+39
1287 FILL#ch,1:CIRCLE#ch,x+18,y+20,4:FILL#ch,0
1288 END DEFine
```

```
1290 DEFine PROCedure MG3(x,y)          :REMark Guitar Bottom
1291 INK#ch,3:FILL#ch,1:LINE#ch,x+2,y+39 TO x+18,y+39
1292 ARC#ch TO x+1,y+32,-PI TO x+2,y+39,-PI/4:FILL#ch,0
1293 INK#ch,0:ARC#ch,x+15,y+38 TO x+12,y+29,-PI/1.5
1294 LINE#ch,x+7,y+36 TO x+8,y+39:LINE#ch,x+9,y+36 TO x+10,y+39
1295 LINE#ch,x+5,y+32 TO x+11,y+31:LINE#ch,x+5,y+34 TO x+11,y+33
1296 END DEFine
```



Conundrum Introduction

'Hangman' once a popular word game, the origins of which are unknown, has been around for more than a century. A word is represented by a row of dashes and the Player tries to guess the missing letters before the drawing of a gallows and hanged man is completed.

Its past use in junior schools as an aid to developing vocabulary and spelling skills was by the nineteen eighties deemed inappropriate. For QBITS and my own children I created **Spellbear**. The game opened with a bear suspended beneath a group of balloons, each wrong guess and the bear lost a balloon, eventually splashing into a muddy pond.

QBITS Conundrum Development

Taking the simple Word riddle and mindful of creating various levels of difficulty I began with a Timer to use as a Countdown. The Word to be represented by Coloured Blocks or by Jumbled Letters and with Clues added as a further aid in solving the riddle. Alphabet Characters are typed in from the keyboard in a trial-and-error number of guesses to reveal the Hidden Word.

QBITS Conundrum Strategy

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

QBITS Conundrum Menu

Use Left/Right Cursor keys to Highlight a Menu item then action with Spacebar. **WordPlay** is the screen displayed at start, selecting **WGen** changes the display to a two columns screen for the creation and edit of WordFiles. Selecting **Play** will switch back to the **WordPlay** screen. The Central Menu Symbol  has two functions, in **WordPlay** it allows changes to the Countdown Timer and an (**E**)xit from the Game. In **WordGen** it Resets Arrays for creating a (**N**)ew Word File and an alternative (**E**)xit from the Game.

In **WordPlay** mode the **Coloured Blocks** change to show any correctly typed in character in whatever order they are typed, but if **Word** is set to **On**, the hidden word is then shown as **Jumbled Letters**, here characters typed in from the Keyboard must be in the correct order to reveal the Word. By turning **Word** and **Clue** ON/OFF and changing the length of **Countdown** a wide range of difficulties can be achieved to service players of different ages and abilities.

The challenge is further defined by the difficulty of the **Word Lists**. The **Word Generator** as part of the code allows **New** lists to be typed in or existing one to be edited with updates.

QBITS WordPlay

This displays the Countdown Timer, the Conundrum Word and Score Points. **Play** requires **Load**ing of a Word File and each Game randomly selects 25 of the entries. The number completed is shown top left next to the Highlighted **WordPlay**. Toggle **Word** On/Off to show Coloured Blocks or Jumbled Letters. Similarly Toggle On/Off for the **Clue** display.

Note: Code Line 1008 **cmax%**=25 controls the number of rounds in a Game.

QBITS WordGen

This displays two columns, the first for the **Word** list and second for the **Clue**s. The current **Page** number is shown top right next the Highlighted **WordGe**n.

If a **Word File** hasn't been previously loaded, use **Load** or start the creation of a **New** word file. The **Word** and **Clue** rows displayed in **WordGen** can be selected with **Up/Down** cursors and switched between using the **Tab** key. In either column pressing '**E**' invokes the **Line Editor**. For each **Word** up to 18 Upper-case Alphabet characters with no spaces is permitted. For **Clue** up to 36 Alphanumeric Characters including spaces and punctuation marks.

Each word File can have 6 pages of 16 rows adding up to total of 96 entries. A minimum of 25 entries are required for **WordPlay** ie. the number of Random choices for a full Game. Select **Save** to store Word File on default device, and [E]dit or create a New Filename

QBITS Word Files:

WGen_Countries :
WGen _DELTA : WGen_DEMO : WGen_GAMMA
WGen_GenKnowl

QBITS Conundrum Code

```
1000 REMark QBITS_Conundrum_v3 (QBITS Conumdrum v3 2021 QPCII)

1002 OPEN_IN#9,'ram2_QBITSConfig':INPUT#9,gx\gy\dn$\dev$:CLOSE#9

1004 WHEN ERRor
1005   CLS#0:CURSOR#0,72,6:PRINT#0,'DEVICE ERROR':PAUSE 20:CLS#0:CONTINUE
1006 END WHEN
```

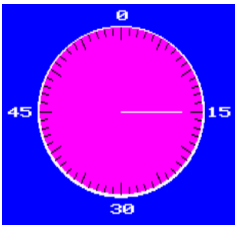

```
1008 MODE 8:cmax%=25:Init_Win:WordMenu

1010 DEFine PROCedure Init_Win
1011 DIM Word$(96,18),Clue$(96,36),WChk(96),str$(36),SDR$(5),fn$(24)
1012 OPEN#4,scr_:WINDOW#4,288,160,gx+206,gy+37:PAPER#4,1:CSIZE#4,1,0
1013 OPEN#3,scr_:WINDOW#3,144,160,gx+18,gy+37 :PAPER#3,1:CSIZE#3,1,0
1014 WINDOW#2,512,224,gx,gy     :PAPER#2,0:BORDER#2,1,3:CLS#2
1015 WINDOW#1,496,174,gx+8,gy+30 :PAPER#1,1:BORDER#1,1,3:CLS#1
1016 WINDOW#0,512, 32,gx,gy+224  :PAPER#0,0:BORDER#0,1,3:CLS#0
1017 SDR$=dev$:max_score%=0:per_score%=0:count%=0:key$=''
1018 CSIZE#2,2,1:OVER#2,1
1019 INK#2,2:FOR i=0 TO 1:CURSOR#2,164+i,8:PRINT#2,'QBITS Conundrum'
1020 INK#2,6:FOR i=0 TO 1:CURSOR#2,166+i,9:PRINT#2,'QBITS Conundrum'
1021 CSIZE#2,2,0:OVER#2,0
1022 QBold 2,5,12,1,400,16,'WordGen':QBold 2,5,12,1,-4,16,'WordPlay'
1023 QBold 2,6,12,1,38,208,'Word Play Load ← → Save WGen Clue'
1024 BLOCK#2,22,7,244,210,5:BLOCK#2,20,5,245,211,0:BLOCK#2,18,3,246,212,6
1025 AT#2,1,1:CSIZE#2,0,0:INK#2,5:CSIZE#1,2,0:CSIZE#0,1,0:INK#0,5
1026 Time%=180:Sech%=180:Sec%=180:CSrn=0:mc%=2:F=0
1027 END DEFine

1029 DEFine PROCedure QBold(ch%,col%,w%,d%,x%,y%,str$)
1030 OVER#ch%,1:INK#ch%,col%:sl%=LEN(str$)
1031 FOR a=1 TO sl%
1032   FOR b=0 TO d%:CURSOR#ch%,x%+b+a*w%,y%:PRINT#ch%,str$(a)
1033 END FOR a:OVER#ch%,0
1034 END DEFine

1036 DEFine PROCedure PlayScrn
1037 HGL 2,5,34,6,2,88:HGL 2,0,30,6,137,88
1038 BLOCK#2,40,10,370,16,0:CSrn=0:CLS#1
1039 INK#1,7:FILL#1,1:CIRCLE#1,164,50,35:FILL#1,0:CSIZE#1,2,0
1040 INK#1,3:FILL#1,1:CIRCLE#1,164,50,34:FILL#1,0:INK#1,0
1041 FOR j=3 TO 5 STEP 2
1042   FOR i=0 TO 360 STEP j*12-30
1043     x=34*SIN(RAD(i)):y=34*COS(RAD(i))
1044     x1=(34-j)*SIN(RAD(i)):y1=(34-j)*COS(RAD(i))
1045     LINE#1,x+164,y+50 TO x1+164,y1+50
1046   END FOR i
1047 END FOR j
1048 QBold 1,7,12,1,362, 14, '0':QBold 1,7,12,1,450,82,'15'
1049 QBold 1,7,12,1,356,150,'30':QBold 1,7,12,1,256,82,'45'
1050 QBold 1,7,12,1,4,  54,'Max   Points:'
1051 QBold 1,7,12,1,4,108,'Score Points:':CSIZE#1,2,1
1052 END DEFine
```


```
1054 DEFine PROCedure EditScrn
1055 HGL 2,0,34,6,2,88:HGL 2,5,30,6,137,88
1056 BLOCK#2,48,10,110,16,0:CLS#1:INK#1,5:l=1:pn%=1:CSrn=1
1057 LINE#1,2,2 TO 2,98 TO 67,98 TO 67,2 TO 2,2
1058 LINE#1,83,2 TO 83,98 TO 210,98 TO 210,2 TO 83,2
1059 END DEFine

1061 DEFine PROCedure HGL(ch%,col%,w%,d%,x%,y%)
1062 INK#ch%,col%:LINE#ch%,x%,y% TO x%+w%,y% TO x%+w%,y%+d% TO x%,y%+d% TO x%,y%
1063 END DEFine
```

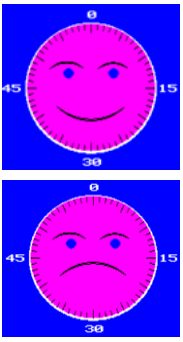

```
1065 DEFine PROCedure WordMenu
1066 F=0:ac%=0:aw%=0:cw%=0:count%=1:PlayScrn
1067 REPeat Comm_lp
1068  x%=mc%*20+15.6:y%=1:HGL 2,5,18,5.8,x%,y%
1069  k=CODE(INKEY$(-1))    :HGL 2,0,18,5.8,x%,y%:INK#2,5:CLS#0
1070  SELect ON k
1071    =192:mc%=mc%-1:IF mc%<0:mc%=6
1072    =200:mc%=mc%+1:IF mc%>6:mc%=0
1073    = 27:CSIZE#0,0,0:INK#0,7:CLS#2:STOP
1074    = 32:SELect ON mc%
1075           =0:CURSOR#2, 28,204:IF aw%=0:aw%=1:PRINT#2,'ON':ELSE aw%=0:PRINT#2,'  '
1076           =1:WordPlay
1077           =2:WordList :IF CSrn=0:mc%=1:ELSE mc%=5
1078           =3:IF CSrn=0:Time_chg :mc%=1:ELSE Word_chg:mc%=5
1079           =4:WordSave :CLS#0
1080           =5:WordGen
1081           =6:CURSOR#2,466,204:IF ac%=0:ac%=1:PRINT#2,'ON':ELSE ac%=0:PRINT#2,'  '
1082         END SELect
1083  END SELect
1084 END REPeat Comm_lp
1085 END DEFine

1087 DEFine PROCedure Time_chg
1088 CLS#0:INK#1,3:FILL#1,1:CIRCLE#1,164,50,28:FILL#1,0
1089 CURSOR#0,80,6:PRINT#0,'Set Countdown Time: ←  →or (E)xit';
1090 BLOCK#0,14,3,248,10,5:Sec%=Time%:Countdown
1091 REPeat Time_lp
1092  Countdown:k=CODE(INKEY$(-1))
1093  SELect ON k
1094   =192:Sec%=Sec%+30:IF Sec%>360:Sec%=360
1095   =200:Sec%=Sec%-30:IF Sec%< 30:Sec%= 30
1096   = 32:Time%=Sec%:CLS#0:EXIT Time_lp
1097   =69,101:PRINT#0,' Y/N':IF INKEY$(-1)=='Y':LRUN dn$:ELSE RETurn
1098  END SELect
1099 END REPeat Time_lp
1100 END DEFine
```



```
1102 DEFine PROCedure Countdown
1103 INK#1,3:LINE#1,164,50 TO 164+26*SIN(RAD(Sech%)),50+26*COS(RAD(Sech%))
1104 Sech%=Sec%:BEEP 2000,10
1105 INK#1,7:LINE#1,164,50 TO 164+26*SIN(RAD(Sech%)),50+26*COS(RAD(Sech%))
1106 END DEFine
```



```
1108 DEFine PROCedure WordPlay
1109 IF F=0 OR wc%<cmax%:mc%=2:RETurn
1110 IF CSm=1:CSIZE#1,2,0:PlayScrn:CSIZE#1,3,1
1111 CNT=DATE:Sec%=Time%:Sech%=Time%:RANDOMISE
1112 REPeat Rnd_lp
1113   n%=RND(1 TO wc%):IF WChk(n%)=0:WChk(n%)=1:WordRND:EXIT Rnd_lp
1114 END REPeat Rnd_lp
1115 Countdown:chr%=1:pos%=1:k=0:str$=FILL$(' ',wl%)
1116 REPeat Wrd_lp
1117  STRIP#1,5:CURSOR#1,16*chr%,135:PRINT#1,key$(chr%):STRIP#1,1
1118  k=CODE(INKEY$(20))
1119  SELect ON k
1120   =192:chr%=chr%-1:IF chr%<1  :chr%=1
1121   =200:chr%=chr%+1:IF chr%>wl%:chr%=wl%
1122   =65 TO 90,97 TO 122:WordChk
1123  END SELect
1124  CURSOR#1,16*pos%,135:PRINT#1,key$(pos%):pos%=chr%
1125  IF key$=Word$(n%) OR str$=Word$(n%) OR Sec%=0:WordScore:EXIT Wrd_lp
1126  IF CNT<>DATE:Sec%=Sec%-(6*(DATE-CNT)):Countdown:CNT=DATE
1127 END REPeat Wrd_lp
1128 END DEFine

1130 DEFine PROCedure WordChk
1131 IF k>96:k=k-32
1132 key$(chr%)=CHR$(k):chr%=chr%+1:IF chr%>wl%:chr%=1
1133 IF aw%=0
1134    FOR i=1 TO wl%
1135      IF key$(pos%)=Word$(n%,i)
1136        str$(i)=CHR$(k):CURSOR#1,16*i,20:PRINT#1,str$(i)
1137      END IF
1138    END FOR i
1139 END IF
1140 END DEFine

1142 DEFine PROCedure WordScore
1143 INK#1,3:FILL#1,1:CIRCLE#1,164,50,28:FILL#1,0:INK#1,1
1144 FILL#1,1:CIRCLE#1,152,58,2.5:FILL#1,0
1145 FILL#1,1:CIRCLE#1,176,58,2.5:FILL#1,0:INK#1,0
1146 ARC#1,142,60 TO 160,60,-PI/2:ARC#1,142,60 TO 160,60,-PI/2.3
1147 ARC#1,168,60 TO 184,60,-PI/2:ARC#1,168,60 TO 184,60,-PI/2.3
1148 IF Sec%<6:ARC#1,146,40 TO 182,40,-PI/2:ARC#1,146,40 TO 182,40,-PI/2.2
1149 IF Sec%>0:ARC#1,146,40 TO 182,40, PI/2:ARC#1,146,40 TO 182,40, PI/2.2
1150 INK#1,7:IF Sec%>0:per_score%=per_score%+5*wl%
1151 CURSOR#1,190,104:PRINT#1,FILL$(' ',4-LEN(per_score%))&per_score%
1152 CURSOR#1,16,20:PRINT#1,Word$(n%):count%=count%+1
1153 IF aw%=0:BLOCK#1,216,20,16,135,1
1154 IF count%>cmax%
1155    CLS#0:CURSOR#1,12,78:PRINT#1,'Game End':PAUSE
1156    max_score%=0:per_score%=0:count%=1:mc%=1:PlayScrn
1157    FOR i=1 TO 96:WChk(i)=0
1158 END IF
1159 CLS#0:CURSOR#0,140,10:PRINT#0,'Press Spacebar to continue...'
1160 END DEFine
```



```
1162 DEFine PROCedure WordRND
1163 CLS#0:BLOCK#1,290,20,16,20,1:BLOCK#1,290,20,16,135,1
1164 INK#1,3:FILL#1,1:CIRCLE#1,164,50,28:FILL#1,0:CSIZE#1,3,1:INK#1,7
1165 Sort$=Word$(n%):wl%=LEN(Sort$):cl%=LEN(Clue$(n%))
1166 REPeat Sort_lp
1167  FOR i=1 TO wl%-1
1168   r1=RND(1 TO wl%):Chr1$=Sort$(r1):r2=RND(1 TO wl%):Chr2$=Sort$(r2)
1169   Sort$(r1)=Chr2$:Sort$(r2)=Chr1$
1170  END FOR i
1171  IF Sort$<>Word$(n%):EXIT Sort_lp
1172 END REPeat Sort_lp
1173 IF aw%=0
1174   FOR blk=1 TO wl%
1175    BLOCK#1,14,20,blk*16,20,7:BLOCK#1,10,18,2+blk*16,20+1,(blk MOD 8)
1176   END FOR blk
1177 END IF
1178 IF aw%=1:CURSOR#1,16,20:PRINT#1,Sort$
1179 IF ac% =1:CURSOR#0,112,10:PRINT#0,FILL$(' ',18-cl%/2)&Clue$(n%)
1180 key$=FILL$('_',wl%):CURSOR#1,16,135:PRINT#1,key$
1181 max_score%=max_score%+5*wl%
1182 CURSOR#1,190,50:PRINT#1,FILL$(' ',4-LEN(max_score%))&max_score%:INK#2,7
1183 CURSOR#2,112,18:PRINT#2,FILL$(' ',3-LEN(count%))&count%&'/'&cmax%
1184 END DEFine
```


```
1186 REMark QBITS WordGen

1188 DEFine PROCedure WordGen
1189 IF CSrn=0:EditScrn
1190 IF F=1:pn%=1:sr%=0
1191 ch%=3:cp%=1:sr%=0:Str_Chk:Pg_Prn:BCol%=7:ICol%=0
1192 INK#4,7:STRIP#4,1:INK#3,7:STRIP#3,1:l=1:Str_Clr
1193 REPeat Edit_lp
1194  CURSOR#0,80,20:PRINT#0,'LINE[⬆⬇] : WORD<[TAB]>CLUE : [E]dit : Rtn[⬅]'
1195  BLOCK#0,2,4,422,22,5:Str_Clr:Str_Prn
1196  k=CODE(INKEY$(-1)):sl%=LEN(str$)
1197  SELect ON k
1198   =  9:Ln_Clr    :IF ch%=3:ch%=4:ELSE ch%=3:END IF :Str_Chk:Str_Prn
1199   = 10:BCol%=1 :ICol%=7:Pg_Prn:Str_Clr:INK#2,5:CLS#0:RETurn   :REMark End
1200   =69,101:Str_ED ch%,1,cp%,cs%,sr%,sm%,sx%,sy%,str$:CLS#0 :REMark Edit str$
1201   =208:Ln_Clr  :WordUP:cp%=1:Str_Chk                        :REMark ⬆Line Up
1202   =216:Ln_Clr  :WordDn:cp%=1:Str_Chk                        :REMark ⬇Line Down
1203  END SELect
1204 END REPeat Edit_lp
1205 END DEFine

1207 DEFine PROCedure Word_chg
1208 CURSOR#0,80,6:PRINT#0,'Create a (N)ew Word File or (E)xit';:PAUSE
1209 IF KEYROW(7)=64
1210   F=0:wc%=0:fn$='':CURSOR#0,270,6:CLS#0,4:CLS#3:CLS#4
1211   FOR i=1 TO 96:Word$(i)='':Clue$(i)='':PAUSE 1:CURSOR#0,280,6:PRINT#0,i
1212 END IF
1213 IF KEYROW(6)=16:PRINT#0,' Y/N':IF INKEY$(-1)=='Y':LRUN dn$
1214 END DEFine
```

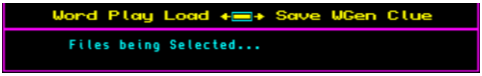
```
1216 DEFine PROCedure WordUP
1217 IF sr%= 0 AND pn%>1:pn%=pn%-1:Pg_Prn:ELSE IF sr%> 0:sr%=sr%-1:l=l-1
1218 END DEFine

1220 DEFine PROCedure WordDn
1221 IF sr%=15 AND pn%<6:pn%=pn%+1:Pg_Prn:ELSE IF sr%<15:sr%=sr%+1:l=l+1
1222 END DEFine
```

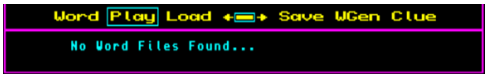
Note: The Line Editor is used to enter the Words and associated Clues when in WordGen Mode. It is also used for the creation of a New Filename or Edit or an existing one. The PROCedure **Sel_chr** controls the Character Sets applicable to Word, Clue or Filename entries.

```
1224 DEFine PROCedure Str_ED(ch%,cm%,cp%,cs%,sr%,sm%,sx%,sy%,str$)
1225 CLS#0:CURSOR#0,104,20:PRINT#0,'Add Chr Del[Ctrl ← /→] [←Cursor →] Rtn[←]'
1226 BLOCK#0,2,4,398,22,5:sl%=LEN(str$)
1227 REPeat Ed_lp
1228   Str_Clr:Str_Prn:CCol%=2:Str_Cur
1229   k=CODE(INKEY$(-1)):sl%=LEN(str$)
1230   SELect ON k
1231     =10            :CCol%=BCol%  :Str_Cur:AT#0,2,0:cls#0,4:EXIT Ed_lp
1232     =32 TO 127     :Str_Prn:k$='':Sel_chr:IF k$>'':Add_chr
1233     =194,232       :CCol%=BCol%  :Str_Cur:IF cp%>cm%:cp%=cp%-1:Del_chr
1234     =202,236       :CCol%=BCol%  :Str_Cur:Del_chr
1235     =192           :CCol%=BCol%  :Str_Cur:IF cp%>cm%:cp%=cp%-1    :REMark ← Left
1236     =200           :CCol%=BCol%  :Str_Cur:IF cp%<sl%+1:cp%=cp%+1 :REMark → Right
1237   END SELect
1238 END REPeat Ed_lp
1239 END DEFine
```


```
1241 DEFine PROCedure Pg_Prn            :REMark Print Page
1242 BCol%=1:ICol%=7:sr%=-1
1243 FOR l=pn%*16-15 TO pn%*16
1244   sr%=sr%+1:ch%=3:Str_Chk:Str_Prn:ch%=4:Str_Chk:Str_Prn
1245 END FOR l
1246 INK#2,7:CURSOR#2,370,16:PRINT#2,'Page ';pn%:l=l-15:sr%=0
1247 ch%=3:BCol%=7:ICol%=0:Str_Chk
1248 END DEFine

1250 DEFine PROCedure Ln_Clr            :REMark Clear Line
1251 Str_Clr:BCol%=1:ICol%=7:CCol%=0:Str_Prn:BCol%=7:ICol%=0:cp%=1
1252 END DEFine

1254 DEFine PROCedure Str_Prn
1255 STRIP#ch%,BCol%:INK#ch%,ICol%
1256 CURSOR#ch%,sx%,sy%+sr%*10:PRINT#ch%,str$&FILL$(' ',sm%-LEN(str$))
1257 END DEFine

1259 DEFine PROCedure Str_Cur
1260 IF cp%>=sm%:cp%=sm%:sl%=sm%
1261 BLOCK#ch%,8,1,sx%+cp%*8-8,sy%+sr%*10+9,CCol%
1262 END DEFine
```

```
1264 DEFine PROCedure Str_Chk
1265 IF ch%=0:cs%=2:cp%=1:sl%=LEN(fn$)        :cm%=5:sm%=16:str$=fn$
1266 IF ch%=3:cs%=1:cp%=1:sl%=LEN(Word$(l)):cm%=1:sm%=18:str$=Word$(l)
1267 IF ch%=4:cs%=3:cp%=1:sl%=LEN(Clue$(l)) :cm%=1:sm%=36:str$=Clue$(l)
1268 END DEFine

1270 DEFine PROCedure Str_Clr
1271 IF LEN(str$)>=sm%:str$=str$(1 TO sm%)
1272 IF ch%=0:fn$=str$:sr%=0
1273 IF ch%=3:Word$(l)=str$
1274 IF ch%=4:Clue$(l)=str$
1275 END DEFine

1277 DEFine PROCedure Sel_chr
1278              SELect ON k=65 TO 90                    :k$=CHR$(k)
1279 IF cs%=1 :SELect ON k=97 TO 122                     :k$=CHR$(k-32)
1280 IF cs%>1 :SELect ON k=48 TO 57,95,97 TO 122 :k$=CHR$(k)
1281 IF cs%=3 :SELect ON k=32 TO 47,123 TO 127    :k$=CHR$(k)
1282 END DEFine

1284 DEFine PROCedure Add_chr
1285 IF cp%=  1 AND sl%=0:str$=str$&k$
1286 IF cp%>=1 AND cp%<sl%:str$=str$(1 TO cp%-1)&k$&str$(cp% TO sl%)
1287 IF cp%>=1 AND cp%=sl%:str$=str$(1 TO cp%-1)&k$&str$(cp%)
1288 IF cp%>  1 AND cp%>sl%:str$=str$&k$
1289 IF cp%=sm%:str$(cp%)=k$
1290 IF sl% <sm%:sl%=sl%+1   :ELSE sl%=sm%
1291 IF cp%<sm%:cp%=cp%+1:ELSE cp%=sm%
1292 END DEFine

1294 DEFine PROCedure Del_chr
1295 IF cp%=sl%:str$=str$(1 TO sl%-1):sl%=sl%-1
1296 IF cp%>=1 AND cp%<sl%:str$=str$(1 TO cp%-1)&str$(cp%+1 TO sl%):sl%=sl%-1
1297 IF cp%=sm%:str$=str$(1 TO sm%-1):cp%=cp%-1:sl%=sl%-1
1298 IF cp%>1 AND sl%=0:str$=""
1299 END DEFine

1301 REMark QBITS Conundrum Word Files

1303 DEFine PROCedure WordList
1304 DIM file$(20,36),df$(36)
1305 CLS#0:QBold 0,5,9,1,60,6,'Files being Selected...'
1306 f%=1:ft%=0:fm%=20:DELETE SDR$&'FList'
1307 OPEN_NEW#9,SDR$&'FList':DIR#9,SDR$:CLOSE#6
1308 OPEN_IN#9 ,SDR$&'FList':INPUT#9,Volumn$,Sector$
1309 REPeat DIR_lp
1310  IF EOF(#9) OR f%>fm%:ft%=f%-1:CLOSE#9:EXIT DIR_lp
1311  INPUT#9,df$:IF 'WGen_' INSTR df$>0:file$(f%)=df$:f%=f%+1:PAUSE 2
1312 END REPeat DIR_lp
1313 IF ft%<1
1314    CLS#0:QBold 0,5,9,1,60,6,'No Word Files Found...'
1315    F=0:mc%=4:PAUSE 50:RETurn
1316 END IF
1317 PAUSE 20:WordFile:mc%=5
1318 END DEFine
```





```
1320 DEFine PROCedure WordFile
1321 CLS#0:QBold 0,5,9,1,60,6,'Select File < ↑↓ ← >:':BLOCK#0,2,4,220,8,5:f%=1
1322 REPeat File_lp
1323  CURSOR#0,260,6:PRINT#0,f%;'  ';file$(f%,5+('WGen_' INSTR file$(f%)) TO)
1324  CLS#0,4:k=CODE(INKEY$(-1))
1325  SELect ON k
1326   =208:f%=f%-1:IF f%<1:f%=ft%
1327   =216:f%=f%+1:IF f%>ft%:f%=1
1328   = 10:fn$=file$(f%):WLoad :EXIT File_lp
1329  END SELect
1330 END REPeat File_lp
1331 END DEFine

1333 DEFine PROCedure WLoad
1334 FOR i=1 TO 96:Word$(i)='':Clue$(i)=''
1335 CLS#0:QBold 0,5,9,1,100,6,'Loading Word File...'
1336 OPEN_IN#9,SDR$&fn$:wc%=1
1337 REPeat Ld_lp
1338   INPUT#9,Word$(wc%),Clue$(wc%):CURSOR#0,292,6:PRINT#0,wc%:CLS#0,4
1339   IF EOF(#9) OR wc%=96:CLOSE#9:EXIT Ld_lp:ELSE wc%=wc%+1:PAUSE 1
1340 END REPeat Ld_lp
1341 PAUSE 20:CLS#0:F=1:l=1:IF CSrn=1:pn%=1:Pg_Prn
1342 END DEFine

1344 DEFine PROCedure WordSave
1345 sm%=16:IF LEN(fn$)>sm% OR fn$='':fn$='WGen_'
1346 CLS#0:CURSOR#0,80,6
1347 PRINT#0,'Save ';SDR$&fn$;FILL$(' ',sm%-LEN(fn$));'(S)ave or (E)dit'
1348 k=CODE(INKEY$(-1)):CURSOR#0,286,6:CLS#0,4
1349 IF k=69 OR k=101
1350   BCol%=0:ICol%=7:Str_ED 0,5,6,2,0,16,160,6,fn$
1351   IF LEN(fn$)<6 OR k=32:mc%=2:RETurn :ELSE WSave
1352 END IF
1353 IF k=83 OR k=115
1354   IF LEN(fn$)<6:BCol%=0:ICol%=7:Str_ED 0,5,6,2,0,16,160,6,fn$
1355   IF LEN(fn$)<6 OR k=32:mc%=2:RETurn :ELSE WSave
1356 END IF
1357 END DEFine

1359 DEFine PROCedure WSave
1360 CURSOR#0,300,6:PRINT#0,' Save Y/N ';:PAUSE
1361 IF KEYROW(5)<>64:CLS#0:RETurn
1362 DELETE SDR$&fn$:OPEN_NEW#9,SDR$&fn$:fm%=96
1363 FOR n=1 TO 96
1364   IF Word$(n)<>'' :PRINT#9,Word$(n)\Clue$(n):fm%=fm%-1
1365   CURSOR#0,380,6:PRINT#0,'Chk:';96-fm%:PAUSE 1
1366 END FOR n
1367 CLOSE#9:CLS#0:mc%=1:F=1:IF fm%=96:DELETE SDR$&fn$:F=0
1368 END DEFine
```

Dart Introduction

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

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

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

The Modern Dart Board

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

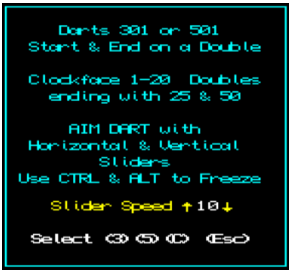
At first view the numbering seems quite random, but is in fact a meticulously picked order, the design of which reduces the incidents of a 'lucky shot' and diminishes the element of chance, so that only skill and accuracy will attain the best scores.

Computer Darts
Compared with other sports few computer-based Darts Games have been released. The first notable one being Metronics180 released in 1986 for the commadore 64, ZX Spectrum, Amstrad and Atari 8-bit. Indoor Sports released a game set in 1987 which included Darts and in 1991 Seta Corporation released Magic Darts!

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

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

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



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

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

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

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

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

QBITS Dartboard

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

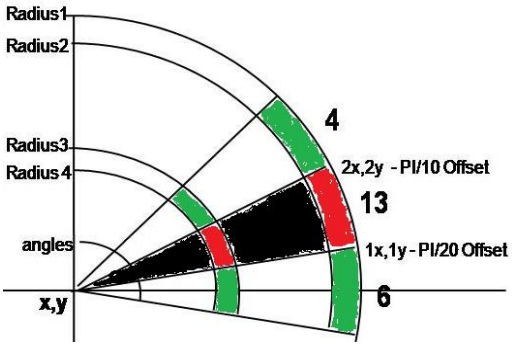
The centre of the circle gives the primary coordinates x, y, then each side of a segment requires a straight LINE drawn from x, y TO 1x,1y and another from x, y TO 2x, 2y between them is drawn an ARC 1x,1y TO 2x,2y its radial angle applicable to the portion of the dartboard's circumference. That's the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

QBITS Dartboard Segments

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

Using the INK & FILL commands a coloured segment is drawn, then by reducing the radius and change of colour this creates the Double, Treble and single sections of each segment.

By adding multiples of **π/10** to the angle, we can then process the next segment and so on around the board. Last but not least we need a couple of FILL'd CIRCLES for the 25 and 50 bullseye at the centre.



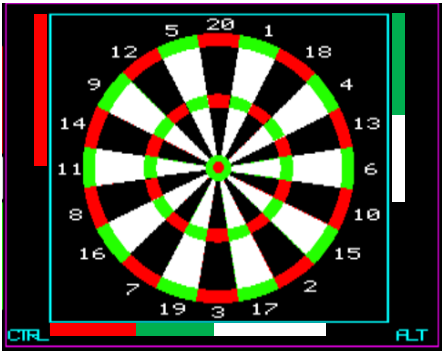
QBITS Dartboard Numbering

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

QBITS Dart Throws

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

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



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

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

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

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

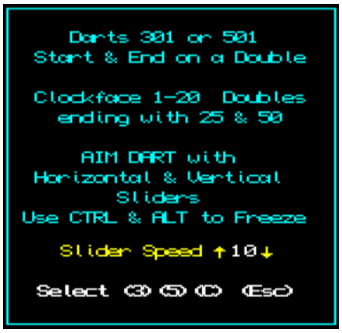
As for the **Double** and **Treble** or **Centre** circles these can be checked against the radius values set up for the dartboard.

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

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

QBITS Darts Code

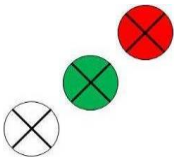
```
1000 REMark QBITS_Darts_v3 (QBITS Darts v3 2021 QPCII)

1002 OPEN_IN#9,'ram2_QBITSConfig':INPUT#9,gx\gy\dn$:CLOSE#9
1003 MODE 8:Init_Scr:sp=10:Darts_Intro:QBITS_Darts          sp Slider Speed 5-10-15 etc

1005 DEFine PROCedure Darts_Intro
1006 DIM str$(10,35),dxy(6):ch=1:CLS#ch:CSIZE#ch,2,0:INK#ch,7
1007 str$(1)="    Darts 301 or 501"
1008 str$(2)=" Start & End on a Double"
1009 str$(4)=" Clockface 1-20  Doubles"
1010 str$(5)="   ending with 25 & 50"
1011 str$(7)="    AIM DART with"
1012 str$(8)=" Horizontal & Vertical"
1013 str$(9)="       Sliders"
1014 str$(10)="Use CTRL & ALT to Freeze"
1015 FOR lp=1 TO 10:QBold 1,5,0,10,0,lp*12,str$(lp)
1016 QBold 1,6,0,10,32,140,"Slider Speed ↑  ↓"
1017 QBold 1,7,1,10,12,164,"Select (3)(5)(C) (E)xit":CURSOR#2,0,0
1018 REPeat key
1019   CURSOR#1,186,140:PRINT#1,FILL$(' ',2-LEN(sp))&sp:k=CODE(INKEY$(-1)):pcol=0
1020   SELect ON k
1021     =208:sp=sp+1   :IF sp>15   :sp=15
1022     =216:sp=sp -1   :IF sp<  5   :sp= 5
1023     =51:score1=301:score=301:score2=301:EXIT key
1024     =53:score1=501:score=501:score2=501:EXIT key
1025     =67, 99:pcol=7:EXIT key
1026     =69,101:LRUN dn$
1027     =27:MODE 4:CSIZE#2,0,0:INK#2,7:CSIZE#0,0,0:INK#0,7:PRINT #0,'Bye':STOP
1028   END SELect
1029 END REPeat key
1030 CLS#1:dartbd:bdcnt:bdnums
1031 IF pcol=7
1032    Clockface
1033 ELSE
1034    ch=5:CSIZE#ch,2,1:CURSOR#ch,6,2:PRINT#ch,'RED Team'  :Arrow 2,2,38,71
1035    ch=4:CSIZE#ch,2,1:CURSOR#ch,6,2:PRINT#ch,'GREEN Team':Arrow 2,4,38,27
1036    pcol=0:teamscr:pcol=2
1037 END IF
1038 ch=1:INK#2,7
1039 END DEFine

1041 DEFine PROCedure Clockface
1042 ch=3:CSIZE#ch,2,0:INK#ch,2:FOR cn=1 TO 20:clk(cn)=cn
1043 n$=25:QBold ch,2,1,9,-9,176,n$:n$=50  :QBold ch,2,1,9,317,176,n$
1044 FOR cn=1 TO 10:n$=cn+10               :QBold ch,2,1,9,317,cn*16,n$
1045 FOR cn=1 TO  9:n$=cn                  :QBold ch,2,1,9,0  ,cn*16,n$
1046 n$=10                                 :QBold ch,2,1,9,-9,160,n$:cn=1
1047 CSIZE#6,2,1:CURSOR#6,2,3:PRINT#6,'CLOCK':Arrow 2,7,38,49
1048 END DEFine
```



```
1050 DEFine PROCedure Game_End
1051 BEEP 20000,1,10,300,30:CSIZE#1,2,1:PAUSE 50
1052 BLOCK#3,32,194,1,2,0  :BLOCK#3,36,194,314,2,0
1053 BLOCK#3,280,8,34,196,0:CLS#6:CLS#4:CLS#5
1054 ch=1:FOR up=1 TO 50:SCROLL#ch,-4:PAUSE 1
1055 IF pcol=2:shots=shot1:mes$='Winning Team':win$='  REDS  '
1056 IF pcol=4:shots=shot2:mes$='Winning Team':win$=' GREENS '
1057 IF pcol=7:shots=shot3:mes$=' Clock-Face ':win$='Complete'
1058 QBold 1,pcol,1,12,54,60,mes$:QBold 1,pcol,1,14,68,90,win$
1059 QBold 1,7,0,12,32,130,'Darts Thrown: '&shots:PAUSE:Darts_Intro
1060 END DEFine
```


```
1062 DEFine PROCedure QBITS_Darts
1063 dp1=0:shot1=0:dp2=0:shot2=0:dp3=0:shot3=0
1064 REPeat Darts
1065  FOR p=1 TO 6 STEP 2
1066    sliders:dxy(p)=dx:dxy(p+1)=dy
1067    IF pcol=2
1068      dp1=num*dm:shot1=shot1+1
1069      IF score1=score AND dm<>2:dp1=0
1070      IF score1-dp1=0 AND dm=2:Game_End:RETurn
1071      IF score1-dp1<=1 OR score1<dp1:dp1=0:EXIT p
1072      score1=score1-dp1:teamscr
1073    END IF
1074    IF pcol=4
1075      dp2=num*dm:shot2=shot2+1
1076      IF score2=score AND dm<>2:dp2=0
1077      IF score2-dp2=0 AND dm=2:Game_End:RETurn
1078      IF score2-dp2<=1 OR score2<dp2:dp2=0:EXIT p
1079      score2=score2-dp2:teamscr
1080    END IF
1081    IF pcol=7
1082      IF cn<21 AND dm=2 AND num=clk(cn)
1083        IF cn<10:n$=cn:QBold 3,7,1,9,  0,cn*16,n$
1084        IF cn=10:n$=cn:QBold 3,7,1,9, -9,cn*16,n$
1085        IF cn>10:n$=cn:QBold 3,7,1,9,317,(cn-10)*16,n$
1086        cn=cn+1:PAUSE 20:BEEP 2000,5,10
1087      END IF
1088      IF cn=21 AND num=25:cn=cn+1:n$=25:QBold 3,7,1,9, -9,176,n$
1089      IF cn=22 AND num=50:cn=cn+1:n$=50:QBold 3,7,1,9,317,176,n$
1090      shot3=shot3+1:IF cn=23:Game_End:RETurn
1091    END IF
1092  END FOR p
1093  ch=1:PAUSE 20
1094  FOR n=1 TO 6 STEP 2
1095    dx=dxy(n):dy=dxy(n+1):dc=0:dart
1096  END FOR n
1097  dartbd:bdcnt:bdnums:dp1=0:dp2=0:dp3=0:IF pcol<>7:pcol=6-pcol:ELSE pcol=7
1098 END REPeat Darts
1099 END DEFine
```

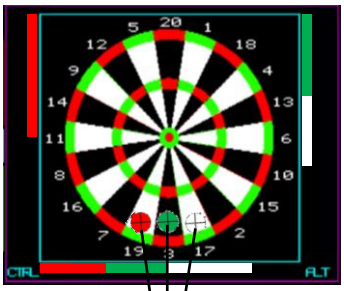
```
1101 DEFine PROCedure dartbd
1102 x=54:y=50:an=PI/20:dx=0:dy=0
1103 FOR f=1 TO 10
1104  c1=2:c2=0:anseg:bdseg
1105  c1=4:c2=7:anseg:bdseg
1106 END FOR f
1107 END DEFine

1109 DEFine PROCedure anseg
1110 x1=COS(an):y1=SIN(an)
1111 an=an+PI/10
1112 x2=COS(an):y2=SIN(an)
1113 END DEFine

1115 DEFine PROCedure bdseg
1116 r=44:c=c1:dwseg
1117 r=40:c=c2:dwseg
1118 r=24:c=c1:dwseg
1119 r=20:c=c2:dwseg
1120 END DEFine

1122 DEFine PROCedure dwseg
1123 ch=1:FILL#ch,1:INK#ch,c
1124 ARC#ch,x+x1*r,y+y1*r TO x+x2*r,y+y2*r,PI/10
1125 LINE#ch TO x,y TO x+x1*r,y+y1*r:FILL#ch,0
1126 END DEFine

1128 DEFine PROCedure bdcnt
1129 INK#ch,4:FILL#ch,1:CIRCLE#ch,x,y,4  :FILL#ch,0
1130 INK#ch,2:FILL#ch,1:CIRCLE#ch,x,y,1.7:FILL#ch,0
1131 END DEFine

1133 DEFine PROCedure bdnums
1134 RESTORE 1139:ch=1:OVER#ch,1:CSIZE#ch,2,0:INK#ch,7
1135 FOR n=1 TO 20
1136    READ num,nx,ny:CURSOR#ch,nx,ny:PRINT#ch,num
1137 END FOR n
1138 OVER#ch,0                          :REMark Board nums,x,y coordinates
1139 DATA 1,172,5,18,206,18,4,236,38,13,246,62,6,256,90
1140 DATA 10,246,118,15,230,142,2,206,162,17,162,176,3,130,178
1141 DATA 19,86,176,7,60,164,16,20,142,8,13,118,11,2,90
1142 DATA 14,4,62,9,28,38,12,46,18,5,92,5,20,126,1
1143 END DEFine

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



QBITS Darts - Sliders

This consist of two **REP**eat loops, one for the Horizontal [dx] and one for the Vertical [dy] positioning to calculate the Dart throw as an angle [da] and radial distance [dr].

```
1153 DEFine PROCedure Sliders
1154 ch=2:BLOCK#ch,8,190,174,11,0:BLOCK#ch,8,190,466,11,0
1155 BLOCK#ch,278,6,185,202,0
1156 REPeat lp_x
1157  FOR h=0 TO 278 STEP 4
1158   BLOCK#ch,h,6,185,202,pcol:PAUSE sp/10
1159   IF KEYROW(7)=2:EXIT lp_x
1160  END FOR h
1161  FOR h=278 TO 0 STEP -4
1162   BLOCK#ch,278-h,6,h+185,202,0:PAUSE sp/10
1163   IF KEYROW(7)=2:EXIT lp_x
1164  END FOR h
1165 END REPeat lp_x
1166 dx=(h*.383)
1167 IF pcol=2:xp=174:ELSE xp=466
1168 REPeat lp_y
1169  FOR v=4 TO 190 STEP 2
1170   BLOCK#ch,8,v,xp,11,pcol:PAUSE sp/10
1171   IF KEYROW(7)=4:EXIT lp_y
1172  END FOR v
1173  FOR v=180 TO 4 STEP -2
1174   BLOCK#ch,8,190-v,xp,11+v,0:PAUSE sp/10
1175   IF KEYROW(7)=4:EXIT lp_y
1176  END FOR v
1177 END REPeat lp_y
1178 dy=100-(v*.521)
1179 BLOCK#ch,278,6,185,202,0:BLOCK#ch,8,190,xp,11,0
1180 dr=SQRT((dy-50)^2+(dx-54)^2)
1181 da=ACOS((dx-54)/dr)
1182 dc=pcol:dart:dnum
1183 END DEFine

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



Sliders used to aim Dart

dx darts x coordinate

Darts Positions

dy darts y coordinate

dr darts radius

da darts angle

dc darts colour

calculates Darts positions in terms of segment number 1 to 20 and darts multiplier

single

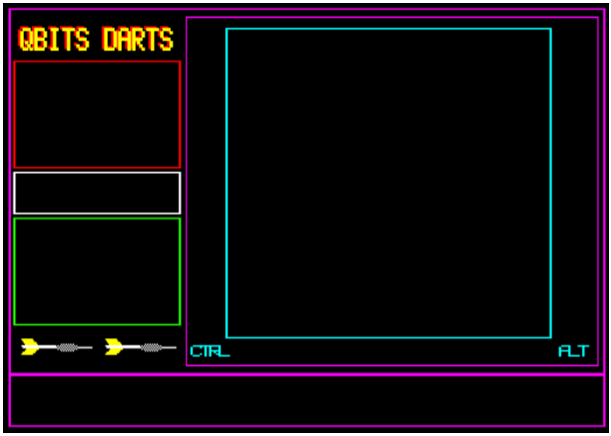
double

treble

or 25

or 50

```
1200 DEFine PROCedure teamscr
1201 ch=5:INK#ch,2:CURSOR#ch,6,24:PRINT#ch,score1;"  "
1202 IF pcol=2:CURSOR#ch,p*18,44 :PRINT#ch,FILL$(' ',2-LEN(dp1))&dp1:CLS#ch,4
1203 ch=4:INK#ch,4:CURSOR#ch,6,24:PRINT#ch,score2;"  "
1204 IF pcol=4:CURSOR#ch,p*18,44 :PRINT#ch,FILL$(' ',2-LEN(dp2))&dp2:CLS#ch,4
1205 END DEFine

1207 DEFine PROCedure QBold(ch,col,d,w,x,y,str$)
1208 OVER#ch,1:INK#ch,col:sl=LEN(str$)
1209 FOR a=1 TO sl
1210     FOR b=0 TO d:CURSOR#ch,b+x+a*w,y:PRINT#ch,str$(a)
1211 END FOR a:OVER#ch,0
1212 END DEFine

1214 DEFine PROCedure Arrow(ch,col,x,y)
1215 INK#ch,col:FILL#ch,1:LINE#ch,x-6,y TO x-8,y+2 TO x-11,y+2
1216 LINE#ch TO x-9,y TO x-11,y-2 TO x-8,y-2 TO x-6,y:FILL#ch,0
1217 INK#ch,7  :FILL#ch,1:LINE#ch,x-11,y+.2 TO x+9,y TO x-11,y-.2:FILL#ch,0
1218 INK#ch,248:FILL#ch,1:CIRCLE#ch,x+2,y,3,.3,PI/2:FILL#ch,0
1219 END DEFine

1221 DEFine PROCedure Init_Scr
1222 WINDOW#2,512,224,gx,gy       :PAPER#2,0:CLS#2:BORDER#2,1,3 :SCALE#2,100,0,0
1223 WINDOW#0,512,32,gx,gy+224  :PAPER#0,0:CLS#0:BORDER#0,1,3
1224 ch=6:OPEN#ch,scr_:WINDOW#ch,144,26,gx+4,gy+100:BORDER#ch,1,7:INK#ch,7
1225 ch=5:OPEN#ch,scr_:WINDOW#ch,144,66,gx+4,gy+ 32:BORDER#ch,1,2:INK#ch,2
1226 ch=4:OPEN#ch,scr_:WINDOW#ch,144,66,gx+4,gy+128:BORDER#ch,1,4:INK#ch,4
1227 ch=3:OPEN#ch,scr_:WINDOW#ch,358,210,gx+152,gy+6
1228 WINDOW#1,280,190,gx+186,gy+12:CLS#1:BORDER#1,1,5:SCALE#1,100,0,0
1229 CSIZE#2,2,1:OVER#2,1:Arrow 2,6,14,7:Arrow 2,6,38,7
1230 INK#2,2:FOR i=0 TO 1:CURSOR#2,4+i,8:PRINT#2,'QBITS DARTS'
1231 INK#2,6:FOR i=0 TO 1:CURSOR#2,6+i,9:PRINT#2,'QBITS DARTS'
1232 OVER#2,0:INK#2,3:LINE#2,50,2 TO 50,98 TO 168,98 TO 168,2 TO 50,2
1233 CSIZE#2,2,0:QBold 2,5,0,8,144,204,'CTRL':QBold 2,5,0,8,460,204,'ALT'
1234 END DEFine
```



Golf Introduction

While the game's ancient origins are unclear, some believe it began with the Roman game of Pagancia which spread across western Europe during the 1st century BC. Others cite Chiwan played in the Ming Dynasty (1368-1644) and introduced to Europe during the Middle Ages. Another contender is Apocryphally, where the Dutch played a game with a stick and leather ball. The winner was whoever hit the ball with the fewest strokes into a target several hundred yards away. However, it is generally accepted that the modern game of Golf was developed in Scotland from the Middle Ages onwards.

The game gained international popularity in the late 19th century when it spread across the rest of the UK and commonwealth countries, which at the time were part of the British Empire and with Scottish emigrants to the United States.

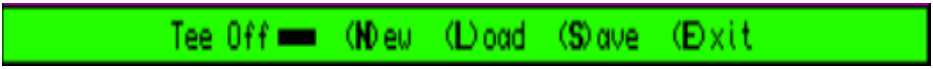
Computer Golf Games

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

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

QBITS Golf Objectives
Prompted by other emerging Golf games of the 1980's the aim was to create aspects of an 18 Hole Golf course with added graphics to depict a clubs Drive Power and Angle of direction. The Fairways created with varying lengths and difficulty, bordered with Trees and Rough ground and ending with a Green with Bunkers. A Lake added to the more difficult Fairways and a Wind Speed / Direction indicator used in the calculations of the ball position.

QBITS Golf Welcome Screen & Menu
An opening screen with the Background of a Green and Golf cart images introduces the QBITS Golf Game with options for **NEW** game or **LOAD** a previously Saved one or simply **Exit!**



After selecting (**N**)ew or (**L**)oad use the Spacebar to **Tee Off**. This will present a new Fairway. (**S**)ave is inactive until at least one hole has been played. Use (**E**)xit to leave the Game between Fairways. Save & Load allows the completion of the 18 holes course at a later time.

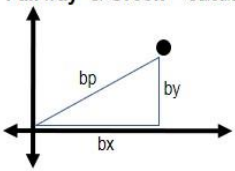
QBITS Golf Wind Speed/Direction
The Wind Speed/Direction indicator is based on a simple eight-point compass design. The current direction shown in Red the others in Cyan with the Speed printed in the centre.

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

Distance is calculated as a ratio between Power and Fairway length with angle to evaluate ball position bx by to which the Wind Speed and Direction wdx wdy values are added.

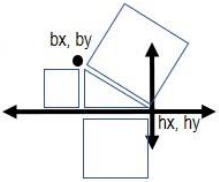
The ball distance from hole is then calculated and displayed.

Fairway & Green – Calculation of Ball Position



Wind speed/Direction = wdx,wdy

```
1034  IF club=40:wx=0:wy=0:ELSE wx=wdx:wy=wdy
1035  bp=INT((79-ny)*1.3)*(196/Gf(h,2)*cmax/100)
1036  bx=bx+bp*COS(RAD(ang))+wx/Gf(h,2)*bp
1037  by=by+bp*SIN(RAD(ang))+wy/Gf(h,2)*bp
```



Calculation of distance to hole

```
1038  lgth=INT(Gf(h,2)/hx*INT(SQRT((hx-bx*2)^2 +(hy-by*2)^2)))
1039  CLS#5:CURSOR#5,20,2:PRINT#5,'Distance to Hole ';lgth;'yds '
```

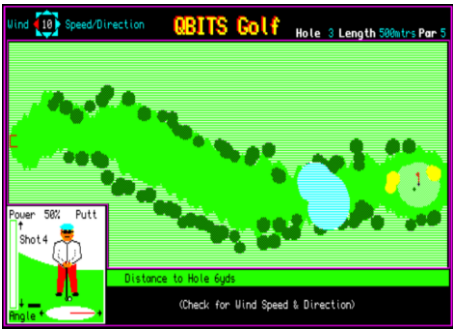
QBITS Golf Hazards

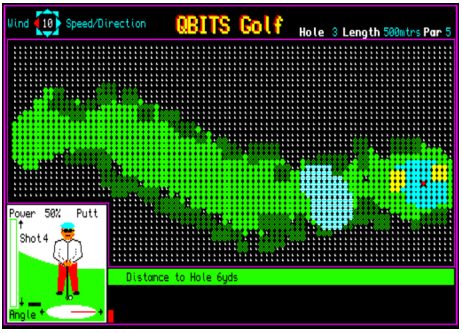
Club Drive for the Fairway is set at 100%, when on the Green a Club Putt is only 50% and the upper Power limit is reduced. Hitting a Boundary incurs a Penalty Shot, landing in the Rough reduces Power to 50%, landing in a Bunker reduces Power to 25%. If the ball hits a Tree the ball is randomly bounced back until clear. Landing in Water returns the ball to the Tee. The maximum number of shots for each hole is limited to 9.

QBITS Golf Fairway

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

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





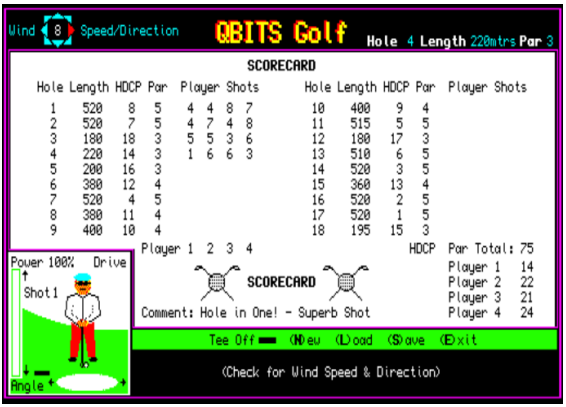
Used in the development phase **GTest** shows a grid(84,40) configured with colour patterns identifying the different areas, Fairway, Trees, Bunkers, Lake, Green and Hole position. The array is used by procedure **CHaz [Club Hazards]** to process the ball location after each shot.

```
1800 DEFine PROCedure GTest
1801 BLOCK#3,500,129,0,0,0:BLOCK#3,386,52,114,128,0
1802 FOR b=1 TO 40
1803  FOR a=1 TO 84
1804     IF b<13 AND a<20:NEXT a
1805     IF grid(a,b)=   4:INK#3,   4:FILL#3,1:CIRCLE#3,a*2,b*2,.7:FILL#3,0      Fairway
1806     IF grid(a,b)=223:INK#3,223:FILL#3,1:CIRCLE#3,a*2,b*2,.7:FILL#3,0        Green
1807     IF grid(a,b)=224:INK#3,224:FILL#3,1:CIRCLE#3,a*2,b*2,.7:FILL#3,0        Trees
1808     IF grid(a,b)=   6:INK#3,   6:FILL#3,1:CIRCLE#3,a*2,b*2,.7:FILL#3,0      Bunker
1809     IF grid(a,b)=  85:INK#3,  85:FILL#3,1:CIRCLE#3,a*2,b*2,.7:FILL#3,0      Lake
1810     IF grid(a,b)=   0:INK#3,   7:FILL#3,0:CIRCLE#3,a*2,b*2,.7:FILL#3,0      Rough
1811     IF grid(a,b)=255:INK#3,   2:FILL#3,1:CIRCLE#3,a*2,b*2,.4:FILL#3,0      Hole
1812   END FOR a
1813 END FOR b
1814 END DEFine
```

QBITS Golf SCORECARD

At the end of each hole played a SCORECARD is shown with the Player results together with three other simulated opponents. To continue - **Tee/Off** with Spacebar.

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



Score Ratings

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

Upon Course completion **SCORECARD** calculates and displays your overall **HandiCap**.

QBITS Golf_Procedures

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

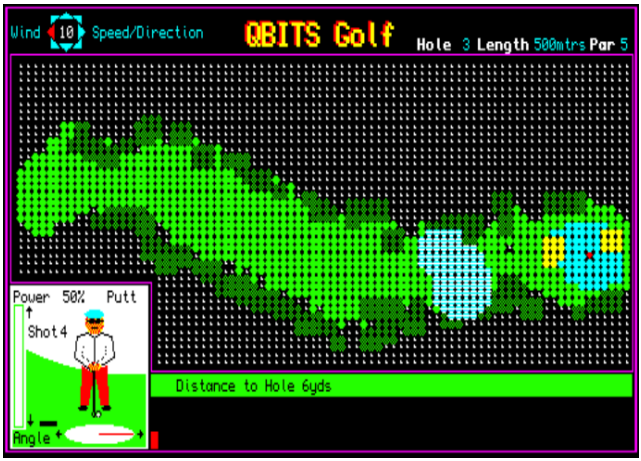
GSave GLoad GExit Confirm Y/N **GTest** Displays Fairway grid(84,40) configuration

QBITS Golf Code

```
1000 REMark QBITS_Golf_v3 (QBITSGolf v3 2021 QPCII)

1002 OPEN_IN#9,'ram2_QBITSConfig'     :INPUT#9,gx\gy\dn$\dev$:CLOSE#9
1003 DevFName$=dev$&"QBGolf_v3Dat":REMark user device_filename
1004 REMark Gf-Fairway Gn-Green Gs-shots Gp-Golfer Wn-Wind
1005 DIM Gf(18,7),HDCap(18),Gn(18,6),Gs(18,4),Gp(18,6),Wd(8,8),pw(18)

1007 WHEN ERRor
1008       eck=1:CONTINUE
1009 END WHEN

1011 MODE 4:Init_win:PAUSE 30:h=0:QBITS_Golf

1013 DEFine PROCedure Golf_Game
1014 REPeat hole
1015   eck=0:Golf_Sel:h=h+1:GFairway h:CWind:CLS#5
1016   bx=1:by=23:sht=0:CPower:Scorecard
1017 END REPeat hole
1018 END DEFine

1020 DEFine PROCedure CPower
1021 cmax=100:club=13:club$='Drive':ang=0:lgth=Gf(h,2)
1022 REPeat h_lp
1023 IF sht>8 OR grid(bx,by)=255 OR lgth<1:EXIT h_lp
1024 INK#4,0:CURSOR#4,36,2:PRINT#4,FILL$(' ',3-LEN(cmax));cmax;'%'
1025 ny=79  :CURSOR#4,78,2:PRINT#4,club$:CURSOR#4,40,22:PRINT#4,sht+1
1026  REPeat T_lp
1027   INK#4,2:LINE#4,57,8.5 TO 57+22*COS(RAD(ang)),8.5+6*SIN(RAD(ang))
1028   k=CODE(INKEY$(-1))
1029   INK#4,7:LINE#4,57,8.5 TO 57+22*COS(RAD(ang)),8.5+6*SIN(RAD(ang))
1030   SELect ON k
1031    =192:ang=ang+7.5:IF ang>=360:ang=0
1032    =200:ang=ang -7.5:IF ang<=0:ang=360
1033    =208:BLOCK#4,4,1,5,ny,2:ny=ny-1:IF ny<club:ny=club
1034    =216:BLOCK#4,4,1,5,ny,7:ny=ny+1:IF ny>79:ny=79
1035    = 32:BLOCK#4,4,68,5,13,7:sht=sht+1:EXIT T_lp
1036   END SELect
1037  END REPeat T_lp
1038   IF club=40:wx=0:wy=0:ELSE wx=wdx:wy=wdy
1039   bp=INT((79-ny)*1.3)*(196/Gf(h,2)*cmax/100)
1040   bx=bx+bp*COS(RAD(ang))+wx/Gf(h,2)*bp
1041   by=by+bp*SIN(RAD(ang))+wy/Gf(h,2)*bp
1042   lgth=INT(Gf(h,2)/px*INT(SQRT((px-bx*2)^2 +(py-by*2)^2)))
1043   CLS#5:CURSOR#5,20,2:PRINT#5,'Distance to Hole ';lgth;'yds  '
1044   CShoot bx,by
1045 END REPeat h_lp
1046 IF sht>9:Gs(h,1)=9:ELSE Gs(h,1)=sht
1047 END DEFine
```

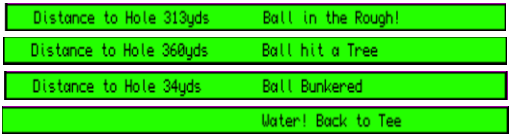
```
1049 DEFine PROCedure CShoot(bx,by)
1050 bcol=grid(x0,y0):IF bcol=0:bcol=95
1051 INK#3,bcol:FILL#3,1:CIRCLE#3,x0*2,y0*2,.4:FILL#3,0
1052 IF club=40:Flag px,py
1053 CTest bx,by:x0=bx:y0=by
1054 FOR i=1 TO 4
1055   INK#3,bcol:FILL#3,1  :CIRCLE#3,bx*2,by*2,.9:FILL#3,0
1056   INK#3,0     :PAUSE 5 :CIRCLE#3,bx*2,by*2,.4:PAUSE 5
1057 END FOR i
1058 END DEFine
```

Array cell codes shown by **GTest**



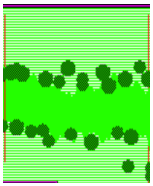
Club Hazards checked by **CHaz**

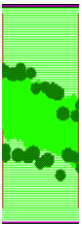
```
1060 DEFine PROCedure CHaz(bx,by)
1061 xmin=1:xmid=20:xmax=84:ymin=1:ymid=13:ymax=40:bck=0:cmax=100
1062 IF bx>xmax:bx=xmax:bck=1
1063 IF bx<xmin:bx=xmin:bck=1
1064 IF by>ymax:by=ymax:bck=1
1065 IF by<ymin:by=ymin:bck=1
1066 IF bx<xmid AND by<ymid:by=ymid:bck=1
1067 IF bck=1:BPlay 'Out of Bounds / Penalty Shot':sht=sht+1:RETurn
1068 bcol=grid(bx,by)
1069 IF bcol=223:club$='Putt ':club=40:cmax=50      :RETurn
1070 IF bcol=  0:BPlay 'Ball in the Rough!'          :cmax=50:bcol=95
1071 IF bcol=  6:BPlay 'Ball Bunkered'               :cmax=25
1072 IF bcol= 85:CLS#5:BPlay 'Water! Back to Tee':bx=1:by=23:bcol=4
1073 IF bcol=224
1074   BPlay 'Ball hit a Tree':PAUSE 30:BPlay ''
1075   x0=bx:y0=by:bx=bx-1:by=by+RND(-1 TO 1):PAUSE 20:CShoot bx,by
1076 END IF
1077 END DEFine
```

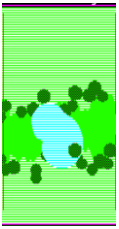



```
1079 DEFine PROCedure BPlay(str$)
1080 CURSOR#5,200,2:PRINT#5,str$:CLS#5,4
1081 END DEFine
```

```
1083 DEFine PROCedure GFairway(h)
1084 DIM grid(84,40):RANDOMISE:par=Gf(h,1):dist=Gf(h,2):INK#2,5
1085 CURSOR#2,362,16:PRINT#2,FILL$(' ',2-LEN(h))&h
1086 CURSOR#2,426,16:PRINT#2,dist;'mtrs':CURSOR#2,496,16:PRINT#2,par
1087 BLOCK#3,500,129,0,0,95:BLOCK#3,386,52,114,128,95
1088 REMark Fairway Tee                                                    Section Tee
1089 RESTORE 1086:FOR fx=1 TO 10:READ fy,fd:Haz 9,4,fx,fy,fd
1090 DATA 23,3,23,4,23,5,23,5,24,6,25,5,26,6,26,6,26,5,26,5
1091 INK#3,2:LINE#3,3,44 TO .5,44 TO .5,48 TO 3,48
1092 fy=26:x0=1:y0=23:grid(x0,y0)=4:CIRCLE#3,x0*2,y0*2,.4
1093 REMark Fairway Sect1
1094 FOR fx=fx+1 TO Gf(h,5)
1095   fy=fy+(-par+RND(1 TO 3))/10:fd=9-par+RND(0 TO 1):fr=fd+1
1096   IF fy+fd>38:fy=36-fd
1097   IF fy-fd<18:fy=20+fd
1098   Haz 9,4,fx,fy,fd
1099   Haz RND(0 TO 3),224,fx-RND(2 TO 4),fy+fd+RND(0 TO 2),fd
1100   Haz RND(0 TO 4),224,fx-RND(2 TO 3),fy-fd+RND(0 TO 1),fd
1101 END FOR fx
1102 REMark Fairway Sect2
1103 FOR fx=fx+1 TO Gf(h,6)
1104   fy=fy+(-par+RND(-1 TO 2))/10:fd=9+RND(2 TO 3)-par:fr=fd+1
1105   IF fy+fd>38:fy=36-fd
1106   IF fy-fd< 5:fy= 7+fd
1107   Haz 9,4,fx,fy,fd
1108   Haz RND(0 TO 2),224,fx-RND(2 TO 4),fy+fd+RND( 1 TO 2),fd
1109   Haz RND(0 TO 3),224,fx-RND(3 TO 5),fy-fd+RND(-1 TO 0),fd
1110 END FOR fx
1111 REMark Fairway Sect3
1112 FOR fx=fx+1 TO 72
1113   fy=fy+(-par+RND(4 TO 8))/10:fd=9-par+RND(0 TO 2):fr=fd+1
1114   IF fy+fd>38:fy=36-fd
1115   IF fy-fd< 5:fy= 7+fd
1116   Haz 9,4,fx,fy,fd
1117   Haz RND(0 TO 3),224,fx-RND(2 TO 5),fy+fd+RND( 0 TO 1),fd
1118   Haz RND(0 TO 4),224,fx-RND(3 TO 5),fy-fd+RND(-2 TO 0),fd
1119 END FOR fx
1120   IF par>3 AND RND(1 TO 3)=3:Lake 3,fx-10,fy+RND(-2 TO +2)
1121 REMark Fairway Green
1122 y2=fy+Gn(h,1):RESTORE 1119:FOR fx=73 TO 84:READ fd:Haz 9,4,fx,fy,fd     Section Green
1123 DATA 5,5,6,6,6,7,7,6,6,6,5,4
1124 GDraw 3,223,158,fy*2,8,1:FOR fx=76 TO 83:READ fd:GMark 223,fx,fy-fd,fx,fy+fd
1125 DATA 3,3,4,4,4,4,3,3
1126 GDraw 3,223,154,y2*2,5,1:FOR fx=74 TO 77:READ fd:GMark 223,fx,y2-fd,fx,y2+fd
1127 DATA 2,2,3,3
1128 grid(79,fy)=255:px=79*2:py=fy*2:Flag px,py                            Flag
1129 FOR fx=64 TO 82
1130   Haz RND(0 TO 5),224,fx,fy+6+RND( 0 TO 1)                            Trees
1131   Haz RND(0 TO 4),224,fx,fy-5+RND(-2 TO 0)                            Trees
1132 END FOR fx
1133 Haz 1,6,73,fy+Gn(h,2),1:Haz 2,6,74,fy+Gn(h,3),1:Haz 1,6,73,fy+Gn(h,4),1   Bunker
1134 Haz 1,6,81,fy+Gn(h,5),0:Haz 2,6,82,fy+Gn(h,6),1                       Bunker
1135 END DEFine
```

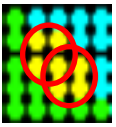




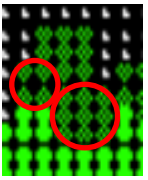


```
1137 DEFine PROCedure Haz(gh,col,hx,hy,hd)
1138 IF gh=1:GMark col,hx,hy-1,hx+1,hy      :GDraw 3,col,hx*2+1,hy*2-1,2,1
1139 IF gh=2:GMark col,hx-1,hy-1,hx+1,hy+1:GDraw 3,col,hx*2  ,hy*2  ,2.5,1
1140 IF gh=9:GMark col,hx,hy-hd,hx,hy+hd  :GDraw 3,col,hx*2  ,hy*2  ,hd*2,.2
1141 :END DEFine
```

Bunker



Trees



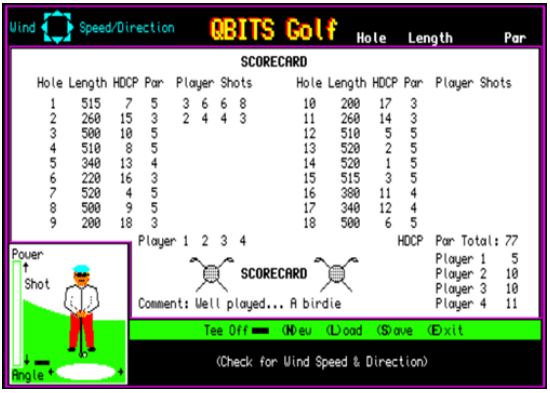
```
1143 DEFine PROCedure Lake(ch,x,y)
1144 INK#3,85:FILL#3,1:ARC#3,x*2,y*2 TO x*2-12,y*2-6,PI TO x*2-8,y*2-10,PI/2
1145 ARC#3 TO x*2+6,y*2-16,PI TO x*2,y*2,PI /2:FILL#ch,0
1146 RESTORE 1143:x=x-6:y=y+1:FOR i=0 TO 9:READ y1,y2:GMark 85,x+i,y+y1,x+i,y+y2
1147 DATA -4,-1,-5,0,-9,0,-10,0,-11,0,-11,0,-11,-2,-11,-2,-10,-3,-9,-5
1148 END DEFine
```

Lake

```
1150 DEFine PROCedure GDraw(ch,col,fx,fy,fr,fe)          Draw on Screen
1151 INK#ch,col:FILL#ch,1:CIRCLE#ch,fx,fy,fr,fe,PI:FILL#ch,0
1152 END DEFine

1154 DEFine PROCedure GMark(col,mw,md,fx,fy)          Save to Grid Array
1155 FOR my=md TO fy:FOR mx=mw TO fx:grid(mx,my)=col:END FOR mx:END FOR my
1156 END DEFine

1158 DEFine PROCedure Flag(px,py)                     Hole & Flag
1159 INK#3,2:LINE#3,px,py TO px,py+4 TO px-1,py+3.5 TO px,py+3
1160 INK#3,0:FILL#3,1:CIRCLE#3,px,py,.4:FILL#3,0
1161 END DEFine
```

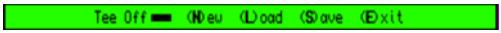
```
1163 DEFine PROCedure CWind
1164 wdx=0:wdy=0:wnd=Gf(h,3):wns=Gf(h,4):IF wns<10:cp=44:ELSE cp=41
1165 INK#2,7:CURSOR#2,41,9:PRINT#2,'  ':CURSOR#2,cp,9:PRINT#2,wns
1166 IF h>1
1167   a=Gf(h-1,3):INK#2,5:FILL#2,1:LINE#2,Wd(a,1),Wd(a,2) TO Wd(a,3),Wd(a,4)
1168   LINE#2 TO Wd(a,5),Wd(a,6) TO Wd(a,7),Wd(a,8):FILL#2,0
1169 END IF
1170   a=Gf(h,3)  :INK#2,2:FILL#2,1:LINE#2,Wd(a,1),Wd(a,2) TO Wd(a,3),Wd(a,4)
1171   LINE#2 TO Wd(a,5),Wd(a,6) TO Wd(a,7),Wd(a,8):FILL#2,0
1172 SELect ON wnd
1173   ON wnd=1:wdx=0         :wdy=wns      :REMark north
1174   ON wnd=2:wdx=wns/2     :wdy=wns/2    :REMark n/e
1175   ON wnd=3:wdx=wns       :wdy=0        :REMark east
1176   ON wnd=4:wdx=wns/2     :wdy=-wns/2   :REMark s/e
1177   ON wnd=5:wdx=0         :wdy=-wns     :REMark south
1178   ON wnd=6:wdx=-wns/2    :wdy=-wns/2   :REMark s/w
1179   ON wnd=7:wdx=-wns      :wdy=0        :REMark west
1180   ON wnd=8:wdx=-wns/2    :wdy=wns/2    :REMark n/w
1181 END SELect
1182 END DEFine
```

```
1184 DEFine PROCedure QBold(ch,col,cs,cx,cy,str$)
1185 INK#ch,col:OVER#ch,1
1186 FOR a=1 TO LEN(str$)
1187   FOR b=0 TO cs:CURSOR#ch,cx+b+a*(6+cs),cy:PRINT#ch,str$(a)
1188 END FOR a:OVER#ch,0
1189 END DEFine
```

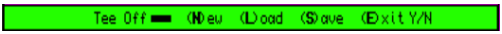



```
1191 DEFine PROCedure Scorecard
1192 BLOCK#3,500,129,0,0,7:BLOCK#3,386,52,114,128,7 :STRIP#3,7
1193 QBold 3,0,1,210,4,'SCORECARD':QBold 3,0,1,210,146,'SCORECARD'
1194 CURSOR#3,  24,18:PRINT#3,"Hole Length HDCP Par  Player Shots"
1195 CURSOR#3,270,18:PRINT#3,"Hole Length HDCP Par  Player Shots"
1196 cpar=0:FOR hp=1 TO 18:cpar=cpar+Gf(hp,1)
1197 CURSOR#3,366,124:PRINT#3,'HDCP  Par Total: ';cpar
1198 CURSOR#3,120,124:PRINT#3,'Player 1  2  3  4'
1199 FOR c=1 TO 18
1200  IF c<10:CURSOR#3,36,22+c*10:ELSE CURSOR#3,276,22+(c-9)*10
1201  PRINT#3,c;"   ";Gf(c,2);FILL$(' ',5-LEN(Gf(c,7)));Gf(c,7);"  "; Gf(c,1)
1202 END FOR c
1203 Sc=0:FOR i=1 TO 4:pw(i)=0
1204 FOR hs=1 TO h
1205  FOR i=1 TO 4
1206    IF hs<10:CURSOR#3,144+i*18,22+hs*10:ELSE CURSOR#3,390+i*18,22+(hs-9)*10
1207    PRINT#3,Gs(hs,i):pw(i)=pw(i)+Gs(hs,i):n$=pw(i)
1208    CURSOR#3,402,126+i*10:PRINT#3,'Player ';i;FILL$(' ',5-LEN(n$))&n$
1209    IF hs=18:CURSOR#3,366,126+i*10:n$=n$-cpar:PRINT#3,FILL$(' ',3-LEN(n$))&n$
1210  END FOR i
1211  Sc=Sc+Gs(hs,1)-Gf(hs,1):sht=Gs(h,1):par=Gf(h,1)
1212 END FOR hs
1213 GClub 64,13:GClub 104,13:CURSOR#3,120,166:PRINT#3,'Comment: ';
1214 IF sht=1       :PRINT#3,"Hole in One! - Superb Shot"    :RETurn
1215 IF sht=par-3  :PRINT#3,"An Albatross... Incredible"
1216 IF sht=par-2  :PRINT#3,"Fantasitc shot... An Eagle"
1217 IF sht=par-1  :PRINT#3,"Well played... A birdie"
1218 IF sht=par    :PRINT#3,"A Par - Not Bad"
1219 IF sht=par+1  :PRINT#3,"A Single Bogey"
1220 IF sht=par+2  :PRINT#3,"A Double Bogey"
1221 IF sht>8      :PRINT#3,"You are out of Shots"            :RETurn
1222 IF sht>par+2  :PRINT#3,"Not so good on this Hole"
1223 END DEFine
```

```
1225 DEFine PROCedure Golf_Sel
1226 REPeat lp
1227   Commds:k=CODE(INKEY$(-1))
1228   IF k=32 AND chk=1 AND h<18:EXIT lp
1229   IF k=69 OR k=101 :GExit:BLOCK#5,20,10,230,2,4
1230   IF k=78 OR k=110 :chk=1:Init_Holes:h=0:EXIT lp
1231   IF k=83 OR k=115 :IF chk=1:GSave
1232   IF k=76 OR k=108 :chk=1   :GLoad:IF eck=0:chk=1:Scorecard
1233 END REPeat lp
1234 END DEFine
```




```
1236 DEFine PROCedure Commds
1237 CLS#5:BLOCK#5,16,4,116,5,0:OVER#5,1
1238 CURSOR#5, 70,2:PRINT#5,'Tee Off    (N)ew  (L)oad  (S)ave  (E)xit'
1239 CURSOR#5,149,2:PRINT#5,'N     L      S      E ' :OVER#5,0
1240 END DEFine
```




```
1242 DEFine PROCedure GExit
1243 CURSOR#5,320,2:PRINT#5,'Y/N':PAUSE:IF KEYROW(5)=64:LRUN dn$
1244 END DEFine
```

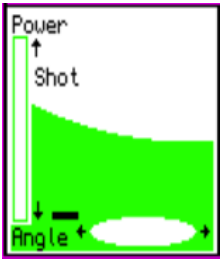
Note: Load & Save use retrieved Dev$ & 'QBGolf_v3Dat' for DevFName$.


```
1246 DEFine PROCedure GSave
1247 CLS#5:CURSOR#5,76,2:PRINT#5,"Please wait - Saving..."
1248 DELETE DevFName$:OPEN_NEW#9,DevFName$:PRINT#9,h
1249 FOR a=1 TO 18
1250   FOR b=1 TO 7:PRINT#9,Gf(a,b):END FOR b
1251   FOR b=1 TO 6:PRINT#9,Gn(a,b):END FOR b
1252   FOR b=1 TO 4:PRINT#9,Gs(a,b):END FOR b
1253   CURSOR#5,196+a*6,2:PRINT#5,'.':PAUSE 1
1254 END FOR a:CLOSE#9
1255 END DEFine
```


```
1257 DEFine PROCedure GLoad
1258 OPEN_IN#9,DevFName$:INPUT#9,h:IF eck=1:RETurn
1259 CLS#5:CURSOR#5,76,2:PRINT#5,"Please wait - Loading..."
1260 FOR a=1 TO 18
1261   FOR b=1 TO 7:INPUT#9,Gf(a,b) :END FOR b
1262   FOR b=1 TO 6:INPUT#9,Gn(a,b):END FOR b
1263   FOR b=1 TO 4:INPUT#9,Gs(a,b):END FOR b
1264   CURSOR#5,196+a*6,2:PRINT#5,'.':PAUSE 1
1265 END FOR a
1266 CLOSE#9:Init_wind:BLOCK#4,10,10,38,22,7:BLOCK#4,76,10,36,2,7
1267 BLOCK#2,14,10,362,16,0:BLOCK#2,42,10,426,16,0:BLOCK#2,8,10,496,16,0
1268 END DEFine
```

```
1270 DEFine PROCedure Init_win
1271 OPEN#6,scr_:WINDOW#6,512,256,gx,gy:BORDER#6,1,3:PAPER#6,0:CLS#6
1272 OPEN#5,scr_:WINDOW#5,390,13,gx+118,gy+211:PAPER#5,4:INK#5,0:CLS#5
1273 OPEN#4,scr_:WINDOW#4,112,94,gx+4,gy+160    :PAPER#4,7:CLS#4:SCALE#4,100,0,0
1274 OPEN#3,scr_:WINDOW#3,500,180,gx+6,gy+29    :SCALE#3,82,0,0
1275 WINDOW#2,508,212,gx+2,gy+1:PAPER#2,0:SCALE#2,140,0,0
1276 WINDOW#1,246,166,gx+250,gy+36:
1277 WINDOW#0,390,30,gx+118,gy+224:PAPER#1,0:CSIZE#0,0,0
1278 CSIZE#2,2,1:OVER#2,1
1279 INK#2,2:FOR i=0 TO 1:CURSOR#2,190+i,5:PRINT#2,'QBITS Golf'
1280 INK#2,6:FOR i=0 TO 1:CURSOR#2,192+i,6:PRINT#2,'QBITS Golf'
1281 CSIZE#2,0,0:OVER#2,0
1282 INK#2,3:LINE#2,1,36 TO 1,123 TO 248,123 TO 248,2 TO 57,2 TO 57,36 TO 1,36
1283 QBold 2,7,1,324,16,'Hole   Length      Par'
1284 CURSOR#0,80,10:PRINT#0,"(Check for Wind Speed & Direction)"
1285 Init_Tee:Init_Golfer 54,74:Init_wind:QBGolf_Welcome:chk=0:Commds
1286 END DEFine
```

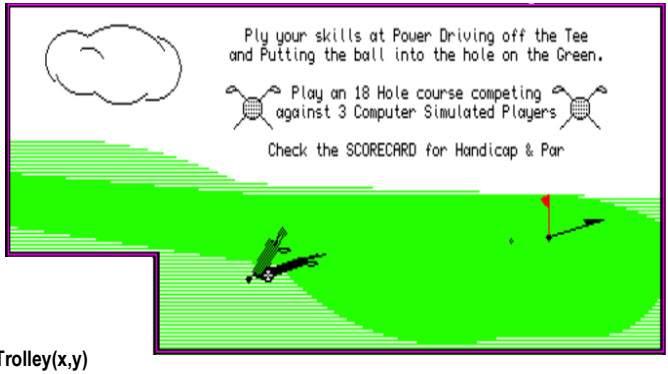
```
1288 DEFine PROCedure Init_Tee
1289 FILL#4,1:INK#4,4:CSIZE#4,0,0
1290 ARC#4,10,60 TO 86.5,45,PI/6:LINE#4 TO 86.5,1  TO 10,1 TO 10,60
1291 FILL#4,0:INK#4,0:CURSOR#4,14,22:PRINT#4,'Shot':STRIP#4,4
1292 CURSOR#4, 2,83:PRINT#4,'Angle':CURSOR#4,36,80:PRINT#4,'← →
1293 CURSOR#4,13,72:PRINT#4,'↓ ':STRIP#4,7:BLOCK#4,14,3,24,78,0
1294 CURSOR#4,12,10:PRINT#4,'↑ ':CURSOR#4,2,2:PRINT#4,'Power'
1295 INK#4,7:FILL#4,1:CIRCLE#4,57,8,22,.25,PI/2:FILL#4,0          :REMark Angle
1296 INK#4,4:LINE#4,2,13 TO 2,88 TO 8,88 TO 8,13 TO 2,13          :REMark Power
1297 END DEFine
```




```
1299 DEFine PROCedure Init_Golfer(x,y)
1300 INK#4,226:FILL#4,1:CIRCLE#4,x,y+3,7,.8,PI:FILL#4,0           :REMark Head
1301 INK#4,0:LINE#4,x-5,y+4 TO x+6,y+4:LINE#4,x-2,y TO x+2,y      :REMark Mouth
1302 CIRCLE#4,x-2,y+4,2,.6,PI/2:CIRCLE#4,x+2,y+4,2,.6,PI/2        :REMark Specs
1303 INK#4,5:FILL#4,1:CIRCLE#4,x,y+8.5,6,.4,PI/2:FILL#4,1         :REMark Cap
1304 FILL#4,1:INK#4,2                                             :REMark Left Leg
1305 LINE#4,x,y-29 TO x-3,y-40 TO x-4,y-47 TO x-9,y-47 TO x-8,y-23 TO x,y-29
1306 FILL#4,0:FILL#4,1                                            :REMark Right leg
1307 LINE#4,x,y-27 TO x+3,y-40 TO x+4,y-47 TO x+9,y-47 TO x+8,y-23 TO x,y-27
1308 FILL#4,0:INK#4,0
1309 FILL#4,1:CIRCLE#4,x-7,y-49,3,.4,PI/2:FILL#4,0                :REMark Left Shoe
1310 FILL#4,1:CIRCLE#4,x+8,y-49,3,.4,PI/2:FILL#4,0                :REMark Right Sh
1311 REMark Left / Right Body
1312 LINE#4,x,y-6 TO x-4,y-3 TO x-11,y-5 TO x-13,y-18 TO x,y-28 TO x,y-6
1313 LINE#4,x,y-6 TO x+4,y-3 TO x+11,y-5 TO x+13,y-18 TO x,y-26 TO x,y-6
1314 LINE#4,x,y-22 TO x,y-55:ARC#4 TO x-1,y-51,-PI                :REMark Club
1315 LINE#4 TO x-1,y-22 TO x,y-22:FILL#4,0:CIRCLE#4,x+2,y-53,1.8
1316 INK#4,7:FILL#4,1:CIRCLE#4,x+2,y-53,1:FILL#1,0                :REMark Golf Ball
1317 INK#4,226:FILL#4,1:CIRCLE#4,x,y-24,4,.6,PI/2:FILL#4,0        :REMark Hands
1318 INK#4,248:LINE#4,x-7,y-11 TO x-8,y-17 TO x,y-22  TO x+7,y-17 TO x+7,y-11
1319 END DEFine
```


```
1321 DEFine PROCedure Init_wind
1322 RESTORE 1327:x=23:y=131.5
1323 FOR a=1 TO 8
1324   FOR b=1 TO 8 STEP 2:READ dat1,dat2:Wd(a,b)=x+dat1:Wd(a,b+1)=y+dat2
1325 END FOR a
1326 INK#2,5:CURSOR#2,2,9:PRINT#2,'Wind       Speed/Direction':INK#2,5
1327 FOR a=1 TO 8
1328   FILL#2,1:LINE#2,Wd(a,1),Wd(a,2) TO Wd(a,3),Wd(a,4)
1329   LINE#2 TO Wd(a,5),Wd(a,6) TO Wd(a,1),Wd(a,2):FILL#2,0
1330 END FOR a
1331 DATA 0,7.5,-2.5,5,2.5,5,0,7.5,6,6,6,3,3,6,6,6          :REMark North N/E
1332 DATA 7.5,0,5,2.5,5,-2.5,7.5,0,6,-6,6,-3,3,-6,6,-6     :REMark East  S/E
1333 DATA 0,-7.5,2.5,-5,-2.5,-5,0,-7.5,-6,-6,-3,-6,-6,-3,-6,-6   :REMark South S/W
1334 DATA -7.5,0,-5,-2.5,-5,2.5,-7.5,0,-6,6,-3,6,-6,3,-6,6       :REMark West  N/W
1335 END DEFine


1337 DEFine PROCedure Init_Holes
1338 RANDOMISE:CLS#5:CURSOR#5,48,2:PRINT#5,"Please wait - Initialising..."
1339 FOR h=1 TO 18
1340   Gf(h,1)=RND(3 TO 5):par=Gf(h,1)                 :REMark Par
1341   Gf(h,2)=140*(par-2)+20*(par+RND(-2 TO 3))       :REMark Distance
1342   IF Gf(h,2)<180:Gf(h,2)=180+RND(1 TO 3)*5        :REMark Check length
1343   IF Gf(h,2)>525:Gf(h,2)=525-RND(1 TO 3)*5        :REMark Check length
1344   Gf(h,3)=RND(1 TO  8)                            :REMark Wind Direction
1345   Gf(h,4)=RND(1 TO 15)                            :REMark Wind Strength
1346   Gf(h,5)=RND(20 TO 40)                           :REMark fxe Sect 1
1347   Gf(h,6)=RND(48 TO 52)                           :REMark fxe Sect 2
1348   Gf(h,7)=Gf(h,2)+h:HDCap(h)=Gf(h,7)              :REMark HDCP
1349   Gn(h,1)=RND(-1 TO 1)                            :REMark Green Angle
1350   Gn(h,2)=RND(-1 TO 0)                            :REMark Bunker 1
1351   Gn(h,3)=RND( 0 TO 1)                            :REMark Bunker 1
1352   Gn(h,4)=RND(-2 TO 0)                            :REMark Bunker 1
1353   Gn(h,5)=RND( 3 TO 4)                            :REMark Bunker 2
1354   Gn(h,6)=RND( 2 TO 3)                            :REMark Bunker 2
1355   REMark Player Score Gs(h,1) + Gs(h,2 to 4) Computer Players
1356   Gs(h,1)=0:FOR p=2 TO 4:Gs(h,p)=Gf(h,1)+RND(-1 TO 3):END FOR p
1357   CURSOR#5,210+h*6,2:PRINT#5,".":PAUSE 1
1358 END FOR h
1359 FOR h=17 TO 1 STEP -1
1360   FOR i=1 TO h
1361     num1=HDCap(i):num2=HDCap(i+1)                 :REMark Sort HDCP
1362     IF num2>num1:HDCap(i+1)=num1:HDCap(i)=num2
1363   END FOR i
1364 END FOR h
1365 FOR h=1 TO 18
1366   FOR i=1 TO 18:IF Gf(h,7)=HDCap(i):Gf(h,7)=i:END IF
1367 END FOR h
1368 END DEFine
```

```
1370 DEFine PROCedure QBGolf_Welcome
1371 BLOCK#3,500,100,0,0,7:INK#3,0:STRIP#3,7
1372 ARC#3,40,72 TO 20,75,PI/2 TO 12,64,PI:ARC#3,24,70 TO 18,72,PI/2
1373 ARC#3,12,66 TO 22,62,PI/2 TO 36,62,PI/2:ARC#3,35,60 TO 38,75,PI
1374 CURSOR#3,184,12:PRINT#3,"Ply your Skill at Power Driving Off the Tee"
1375 CURSOR#3,170,24:PRINT#3,"and Putting the ball into the hole on the Green"
1376 CURSOR#3,218,40:PRINT#3,"Play an 18 Hole course competing"
1377 CURSOR#3,206,50:PRINT#3,"against 3 Computer Simulated Players"
1378 CURSOR#3,212,70:PRINT#3,"Check SCORECARD for Handicap & Par"
1379 INK#3,95:FILL#3,1:LINE#3,170,.5 TO 170,36 TO 60,40:ARC#3 TO 0,50,PI/6
1380 LINE#3 TO 0,24 TO 39,24 TO 39,.5 TO 170,.5:FILL#3,0
1381 INK#3,4:FILL#3,1:LINE#3,0,42 TO 115,33:ARC#3 TO 170,30,-PI/4 TO 140,8,-PI/1.2
1382 ARC#3 TO 60,35,-PI/2:LINE#3 TO 115,14 TO 0,30 TO 0,42:FILL#3,0
1383 INK#3,2:FILL#3,1:LINE#3,140,27 TO 140,37 TO 138,36 TO 140,34 TO 140,27
1384 INK#3,0:CIRCLE#3,140,27,.6:FILL#3,0
1385 FILL#3,1:LINE#3,140,27 TO 154,31 TO 149,31 TO 150,30:FILL#3,0
1386 CIRCLE#3,130,26,.4:Golf_Trolley 70,25:GClub 63,58:GClub 148,58
1387 END DEFine
```




```
1389 DEFine PROCedure Golf_Trolley(x,y)
1390 INK#3,0:LINE#3,x-6,y-6 TO x-8,y-8:CIRCLE#3,x+8,y-4.5,2,.2,PI/2
1391 FILL#3,1:LINE#3,x-7,y-7 TO x-4,y-8 TO x+8,y-4 TO x+5,y-3 TO x-7,y-7:FILL#3,0
1392 LINE#3,x+4,y-4 TO x+12,y-2:LINE#3,x+3,y-4 TO x+10,y-2
1393 CIRCLE#3,x+1,y-1,2,.2,PI/2 :LINE#3,x,y+.4 TO x-3,y+1.4
1394 INK#3,224:FILL#3,1:LINE#3,x,y TO x-3,y+1 TO x-7,y-6 TO x-4,y-7 TO x,y:FILL#3,0
1395 INK#3,  0:FILL#3,1:CIRCLE#3,x-8,y-8,.6 :FILL#3,0
1396 INK#3,  0:FILL#3,1:CIRCLE#3,x-3,y-7,1.6:FILL#3,0
1397 INK#3,  7:FILL#3,1:CIRCLE#3,x-3,y-7,.9 :FILL#3,0
1398 INK#3,  0:LINE#3,x-4,y-6 TO x-2,y-8:LINE#3,x-4,y-8 TO x-2,y-6
1399 LINE#3,x-1.5,y TO x-.5,y+3 :CIRCLE#3,x,y+3,.4
1400 LINE#3,x-1,y TO x,y+4       :CIRCLE#3,x+.5,y+4,.4
1401 LINE#3,x-.5,y TO x+1,y+3   :CIRCLE#3,x+1.2,y+3,.4
1402 END DEFine
```

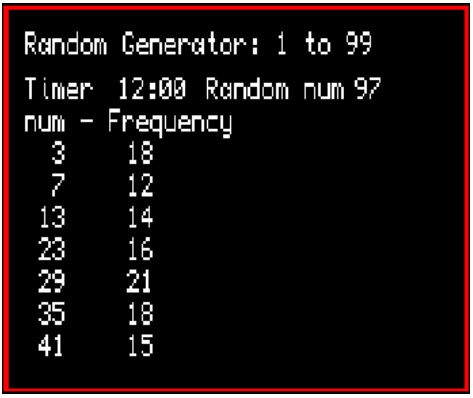
```
1404 DEFine PROCedure GClub(x,y)
1405 LINE#3,x-5,y-5 TO x+5,y+5:ARC#3 TO x+7,y+4,-PI/2:LINE#3 TO x+4,y+4.5
1406 LINE#3,x+5,y-5 TO x-5,y+5:ARC#3 TO x-7,y+4, PI/2:LINE#3 TO x-4,y+4.5
1407 INK#3,63:FILL#3,1:CIRCLE#3,x,y,2:FILL#3,0:INK#3,0:CIRCLE#3,x,y,2.5
1408 END DEFine
```

QBITS Random Numbers`

Obstacles, misdirection, misfortune, unpredictable events or simply sources of danger are the Hazards encountered in Games. Hazards are triggered by Timeouts and/or Boundaries but mostly rely on the use of Random numbers in choosing when and where to Initiate.

This short program checks Random numbers against their frequency of occurrence. To begin the range within which the Random numbers are to be Generated (1 to RGen). Then Specific numbers to check within a time period (Timer) and an imposed Frame Delay (FDelay) waiting between any keyboard inputs.

```
100 REMark RanGen
101 gx=20:gy=30:WINDOW#2,512,256,gx,gy:CLS#2
102 FDelay=20:RGen=99:RanGen

104 DEFine PROCedure RanGen
105 ch=1:WINDOW#ch,200,120,gx,gy:BORDER#ch,1,2:INK#ch,7
106 CURSOR#ch,0, 6:PRINT#ch,' Random Generator: 1 to ';RGen;
107 CURSOR#ch,0,20:PRINT#ch,' Timer       Random  num';
108 clkold=DATE:a=0:b=0:c=0:d=0:e=0:f=0:g=0
109 PRINT#ch,\' num - Frequency'\'   3'\'   7'\'  13'\'  23'\'  29'\'  35'\'  41'
110 REPeat lp
111   clk=(DATE)-clkold:clock$=DATE$(clk*60)
112   CURSOR#ch, 46,20:PRINT#ch,clock$(13 TO 17)                    Timer 00:01 shown duration
113   k=CODE(INKEY$(FDelay))
114   IF k=32:EXIT lp
115   num=RND(1 TO RGen):CURSOR#ch,148,20:PRINT#ch,num;' '     Displays Current Random Number
116   SELect ON num
117     = 3:a=a+1:CURSOR#ch,50,40:PRINT#ch,a;' '
118     = 7:b=b+1:CURSOR#ch,50,50:PRINT#ch,b;' '
119     =13:c=c+1:CURSOR#ch,50,60:PRINT#ch,c;' '
120     =21:d=d+1:CURSOR#ch,50,70:PRINT#ch,d;' '
121     =29:e=e+1:CURSOR#ch,50,80:PRINT#ch,e;' '
122     =35:f=f+1:CURSOR#ch,50,90:PRINT#ch,f;' '
123     =41:g=g+1:CURSOR#ch,50,100:PRINT#ch,g;' '
124   END SELect
125 END REPeat lp
126 WINDOW 492,200,10,0
127 END DEFine
```



Note: The Random Generator in this example ran for 12 minutes. With seven numbers the average frequency of occurrence is 16 times within the 720 seconds ie. once every 45 seconds.

Warehouse Introduction

Amongst old documentation of QL Programs, Sketches and Notes, I came across a two-sided sheet of graph paper headed ASRS. In the mid-eighties the company I work for at the time invested in a new innovation, a computer controlled Automated Storage and Retrieval System (ASRS) This was quite a Project and I guess a starting point for writing the QBITS Game.

QBITS WareHouse Game

Basically, the aim is to fulfil **Invoices** of requested goods from Warehouse **Stocks** and achieve a **Profit.** To accomplish this Credits **(Sales)** will need to overtake the current value of Warehouse Stocks **(Asset)**. The Game can be set to run between 10 and 40 Invoices. The Store holds 128 bins and 24 different types of goods. Stock held for each item is between 0 and 8, the available stock is displayed on the Store Computer in three rows A/B/C and eight columns 1-8. Invoices and Deliveries are identified by screen displayed Printer Outputs. Lorries vacate their Bay's at the set Departure times irrespective of their Load/Unloaded status.

A Game requires actions that randomly upset the flow, the Hazards chosen here are for a Store **Computer glitch** which is just annoying, losses due to **Missing Stock** which reduces the **Asset** and various others such as **Tax Revenues**, **Energy Bills,** and **Stolen Goods**, each of which deducts credits from the accrued **Sales**.

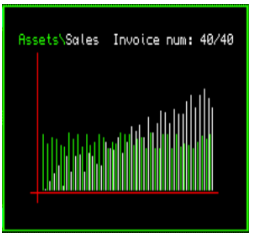
At the start of a **New** Game select the number of **Invoices,** these are chosen in multiples of 5 from 10 to 40 and then a **Skill** level (Forman Storeman Trainee).

As an Invoice is completed, the count is incremented to the next. Upon each Lorry Departure or Hazzard, **Assets** and **Sales** are checked and the corresponding results are displayed.

QBITS Warehouse Sales & Assets

As **Sales** are fulfilled the Store Stocks (**Assets**) are reduced. To replenish diminishing stock, the (**S**)tock Request brings in new Goods Deliveries to provide a means of restocking. It uses the Store Computer display to highlight the Stock item for selection.

You cannot make a Stock Request if you have less than 12 Sales credits, or a delivery is already being processed.

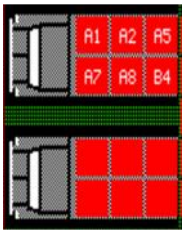


Lorry Departures with loaded Invoices generate **Sales.** Each unit of Stock (**Assets**) is valued as 1 credit. If an Invoice is fulfilled and Lorry loaded as shown on the Printout each unit is worth 2 credits. If incorrectly loaded each item counts as only one credit.

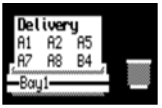
The **SCORE** is shown as **Sales/Assets**, press (**M**) for Menu and the Wearhouse screen area shows an **Audit Report** a Chart displaying the status of Profitability from completed transactions.

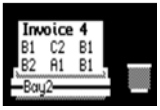
Truck movement, Storage Level, Pick and Drop, variables keep track of Stock following a Delivery and/or fulfilling an Invoice (**Sales**). These are processed to update the Store Computer Stocks (**Assets**) and the Score **(Sales)**.

QBITS Warehouse Printers and Lorries

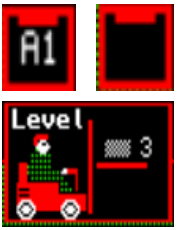


The Printers are presented in line with their respective loading Bay. Printouts have to cater for **Invoices** and **Deliveries** (Stock Requests) and as such, the top of each show whether they are an Invoice or a Delivery. The six spaces of an empty Lorry are to be filled as shown by the Invoice. For a Delivery, the six spaces display the Stock items Requested. As a lorry departs, the displayed printout is screwed up into a ball then further reduced to finally drop into the adjacent wastepaper basket.





Bay1 & **Bay2** =0 when empty, =1 for an Invoice or =2 for a Delivery. For deliveries, **Del=0/1/2** and identifies if a Delivery is pending and which Loading Bay it will be delivered to. **Din=0** or **=1** active, when a Delivery comes in.



For Pickup Truck direction and movement along Track ways to Access Stocks use the Cursor keys. Also, direction of Pick or Drop by cycling around with Spacebar. Warehouse location, levels and loading bays are identified as part of an array **Stock(r, c, l),** row (**r**), column (**c**) level (**l**). Four levels of storage added more gaming difficulty and introduced the graphics of Sam seated on his Pick-Up Truck to indicate which level is being accessed (**lev**).

QBITS Warehouse Stock Movement
The basic graphics for the Warehouse display began with Storage bays and Truck gangways. This is an Array of 7 rows with 15 columns. Truck movement was restricted to cells holding a 0. Cells containing Stock would be Set 1 to 24 (a1-8 b1-8 c1-8) or for the storage bays if empty set to 32. All other cells set to 255 are not accessible. As for the six cells for a Lorry if empty they would be set as 0 for Truck movement or if the bay was empty set to 255.

QBITS WareHouse Timing
Key to all is the Warehouse Clock against which the **Arrival** and **Departure** of Lorries are checked. The clock start time is created using the QL Date (**ClkOld**=DATE). The clock (**Clk**) updates are by reading the QL clock and deducting the **ClkOld** to give a time span. Divided by a clock multiple (**cm**) in this case 60 to change seconds into minutes on the Warehouse Clock.

The two loading bay's Departure Times are held by (**Dtime1** & **Dtime**2)

QBITS WareHouse Skills
A higher skill implies a more planned approach and economic decision making. The selection of **Forman, Stockman, Trainee**, therefore reflects this by altering the duration of time set for each Lorry departure.

Forman	Departure Time + 75min (75 second to complete)
Storeman	Departure Time + 120min (120 second to complete)
Trainee	Departure Time + 225min (225 second to complete)

QBITS Warehouse Channels/Windows
For the movement of Lorries and Printer Outputs the graphic displays presented a number of choices. Open a large number of new channels (windows) or simply resize a window according to the action taking place. For some I chose the latter.

Channel / Window defined by - w-*width*, d-*depth*, x, y-*coordinates*

Intro and Menu	WINDOW#1 Clear for Opening Introduction and resize for options Menu.
Store Layout	ch=3:OPEN #ch,scr_356x194a16x30:PAPER #ch,32:CLS #ch:BORDER #ch1,2
Game Title	ch=3:WINDOW#ch,216,26,13,x10 :PAPER #ch,24:CLS #ch:BORDER #ch1,2
Pick-Up Truck	ch=4:OPEN #ch,scr_80x40a32,12 :PAPER #ch, 0:CLS #ch:BORDER #ch1,2
	ch=4:WINDOW #ch,20,30,70,13 (Reduce Window area for graphics to show Storage Bin Level)
Print/Lorries	ch=5:OPEN #ch,scr_ (Open Window #5 reconfigure for Printer & Lorry Bays)
Bay1 Print	ch=5:WINDOW #ch,116,24,380,40
Bay2 Print	ch=5:WINDOW #ch,116,24,380,195
Bay1 Lorry	ch=5:WINDOW #ch,120,40,18,75
Bay2 Lorry	ch=5:WINDOW #ch,120,40,18,135
Store Score	ch=6:OPEN #ch,scr_80x22a32,198 :PAPER #ch,0:CLS #ch:BORDER #ch1,2
QL2K Display	ch=7:OPEN #ch,scr_120x806a380,80:PAPER #ch,2:CLS #ch
Stock Request	ch=8:OPEN #ch,scr_216x22a136,198:PAPER #ch,2:CLS #ch:BORDER #ch1,2
Keyboard Input	ch=9:OPEN #ch,con_20x10a10x10 [Consul for k$=INKEY$:k=COD$(k)]

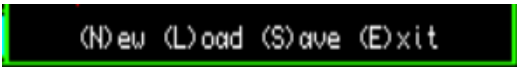
Note: Use of BLOCK, BORDER, CURSOR, INK, PRINT, OVER, STRIP commands etc. the - x, y coordinates being specific to a Channel / Window.

QBITS Warehouse Procedures

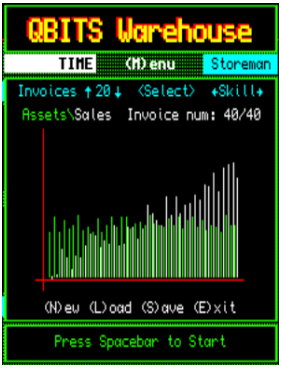
QBWH_Intro	Main instruction on Game
QBWH_Menu	(1)New (2)Load (3)Save (4)Exit options Menu
New_Game	Set up Select New Skill Foreman Storeman Trainee and number of Invoices
Save_Audit	SAVE Game in progress for re- LOAD at later date
Load_Audit	LOAD previously saved Game
Sel_Path	Selects drive & Audit file
QBWH_Game	The main repeat loop for Game play until Inv=Invmax (Completed Total Orders).
Init_clk	Displays a digital counter hour: minutes
QBWH_Init	Graphics – Stock bins, Lorry bays, Clocks, Title, Truck, Score, Stock Request, Stock Display, Printers & Baskets.
Init_stock	Loads Data Array of Warehouse stock
Init_Layout	Displays Stock in Bin locations
PickUP	Graphics for Truck
Bin_lev	Use L level-key to change Bin Level in store
Bin_id	Convert Bin Stock number (1 to 24) to string (a1- c 8).
Bin_cls	Clear Bin
Bin_prn	Print Bin Stock number
Stock_aud	Checks Stock held in Store
Stock_prn	Print Stock numbers in QL2K Stock Display Screen.
Truck_pos(n)	Use Arrow-keys and Spacebar to move and show direction of Pick-Up truck.
Truck_Pick	Pick selected Bin Stock
Truck_Drop	Drop selected Bin Stock
Arrival(bay)	Bay 1 or 2: Generates Random Order or loads Delivery from Stock Request.
Prn_in	Prints Order or Delivery details to bay Printer
Lorry_in	PAN's Lorry Graphic into appropriate bay - Delivery stock shown loaded.
Depart(bay)	Bay 1 or 2: Graphics for Lorry Departure + Updates - Score / Assets.
Prn_out	Throws away paper to Basket.
Lorry_out	PAN's Graphics for Lorry Departure
Stock_Request	Stock requested for next Delivery (maintain Stocks in Warehouse).
Stock_loss	Randomly deletes Warehouse Stock
Stock_er	Computer crash – Warehouse QL2K screen shows all stocks at zero.
Asset_Tax	**?** paid. Random amount taken from sales credits
Energy_Bill	**?** paid. Random amount taken from sales credits
Stolen_Stock	**?** lost. Random amount taken from sales credits
ASReport	Generates Asset /Sales Graph of Players progress
AS_Demo1	Asset/Sales graph for QBWH_Intro page
As_Demo2	Asset/Sales graph for QBWH_Menu page

QBITS Warehouse Code

```
1000 REMark QBITS_WH21_v3 [QBITS WhareHouse v3 2021 QPCII]

1002 OPEN_IN#9,'ram2_QBITSConfig':INPUT#9,gx\gy\dn$\dev$\dn%\dm%
1003 DIM drv$(dm%,5):FOR d=0 TO dm%:INPUT#9,drv$(d):END FOR d:CLOSE#9

1005 REMark Store Variables
1006 ClkOld=DATE:Tim=3600          :REMark ClkOld=Start Time Tim=1hr Increment
1007 Clk=0:cm=60:Skill=6           :REMark Clk=Clock cm=30/60/90 Skill 3/6/9 Timings
1008 Asset=0:Sales=0               :REMark Asset=Stock Sales=Orders fulfilled
1009 Invmax=10:Inv=1:Ino=1         :REMark Invmax=Max Orders Inv=Current Ino Chk Sales
1010 stk=32:stk$='  '              :REMark Stocks (1-24;32=Empty)Stk$='a1 to c8'
1011 r=4:c=5:l=1:lev=1             :REMark Warehouse row,column,level & Bin Level
1012 p=192:box=0:box$='  '         :REMark Truck Moves/Pick/Drop Status
1013 Dtime1=0:Dtime2=0             :REMark Scheduled Lorry Departure Times Bay 1&2
1014 bay1=0:bay2=0                 :REMark Bay 0=empty 1=Order 2=Delivery
1015 Del=0:Din=0:Gs=0              :REMark Del Din Delivery 0/1/2  Gs 0/1/2/3 Game Satus
1016 SD$='QBWHAudit_' :fnum=0      :REMark Audit Data files 0 - 9

1018 REMark Arrays
1019 DIM Aud(24),InB(2,6),Stock(7,14,4),SReq(6),ASRep(40,2),Mes$(10,60)
1020 DIM Sk$(3,8):RESTORE 1021:FOR sk=1 TO 3:READ Sk$(sk)
1021 DATA 'Foreman ','Storeman','Trainee '

1023 WHEN ERRor
1024    eck=1:CONTINUE
1025 END WHEN

1027 MODE 4:QBWH_Intro:QBWH_Init:QBWH_Menu

1029 DEFine PROCedure QBWH_Menu
1030 ch=1:WINDOW#ch,220,144,gx+138,gy+49:CSIZE#1,0,0:INK#ch,0
1031 FOR i=1 TO 20:FILL#ch,1:CIRCLE#ch,60,50,i*3:FILL#ch,0:PAUSE 1
1032 BORDER#ch,1,4:PAPER#ch,0:CLS#ch
1033 IF demo=2:AS_Demo2:demo=0:ELSE ASReport
1034 IF Gs=1:ch=8:CLS#ch:INK#ch,4:CURSOR#ch,50,4:PRINT#ch,'Spacebar to Return'
1035 IF Gs=4:ch=8:CLS#ch:INK#ch,4:CURSOR#ch,84,4:PRINT#ch,'GAME END'
1036 ch=1:INK#ch,7:CURSOR#ch,28,130:PRINT#ch,'(N)ew (L)oad (S)ave (E)xit'
1037 REPeat lp
1038   k=CODE(INKEY$(-1))
1039   SELect ON k
1040     =32 :IF Gs=1:CLS#8  :Init_Layout:QBWH_Game     :REMark continue
1041     =78,110:New_Game :QBWH_Game                    :REMark (N)ew Game
1042     =76,108:Load_Audit :IF Gs=2:QBWH_Game          :REMark (L)oad Game
1043     =83,115:IF Ino>1       :Save_Audit             :REMark (S)ave Game
1044     =69,101:CLS#8          :GExit   :CLS#8         :REMark (E)xit Game
1045   END SELect
1046 END REPeat lp
1047 END DEFine

1049 DEFine PROCedure GExit
1050 CURSOR#8,84,4:PRINT#8,'Exit Y/N':PAUSE:IF KEYROW(5)=64:LRUN dn$
1051 END DEFine
```



```
1053 DEFine PROCedure New_Game
1054 IF bay1>0:Depart 1:bay1=0
1055 IF bay2>0:Depart 2:bay2=0
1056 ch=1:sk=2:iv=3:CLS#8:INK#8,4:CURSOR#8,38,4:PRINT#8,'Press Spacebar to Start'
1057 INK#ch,5:CURSOR#ch,8,2:PRINT#ch,'Invoices ↑ ↓ <Select> ←Skill→ '
1058 REPeat Choice
1059   ch=3:BLOCK#ch,60,11,276,7,5:QBold 0,0,276,8,Sk$(sk)
1060   ch=1:CURSOR#ch,71,2:PRINT#ch,FILL$(' ',2-LEN(5+iv*5))&5+iv*5
1061   k=CODE(INKEY$(#ch,20))
1062   SELect ON k
1063     =192:sk=sk-1:IF sk<1:sk=3
1064     =200:sk=sk+1:IF sk>3:sk=1
1065     =208:iv=iv+1:BLOCK#1,12,10,71,2,0:IF iv>7:iv=7
1066     =216:iv=iv-1:BLOCK#1,12,10,71,2,0:IF iv<1:iv=1
1067     =69,101:LRUN dn$
1068     = 32:Skill=(sk*3):Invmax=5+(iv*5):CLS#8:EXIT Choice
1069   END SELect
1070 END REPeat Choice
1071 Inv=1:Ino=1:Sales=0:FOR i=1 TO 40:ASRep(i,1)=0:ASRep(i,2)=0
1072 Init_Stock:Init_Layout:Gs=1
1073 END DEFine
```

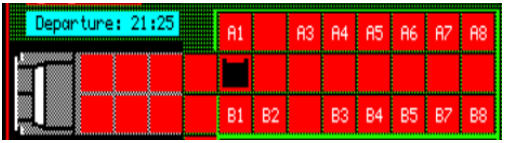



```
1075 DEFine PROCedure QBWH_Game
1076 Bin_lev:Stock_aud:Score:Gs=1:r=4:c=5
1077 REPeat Game_lp
1078   Store_Clk:Truck_Pos p
1079   IF Inv>7 AND Sales>24 AND Del=0
1080     num=RND(1 TO 99)
1081     IF num=  3:Store_err
1082     IF num=  7:Stock_loss
1083     IF num=13:Asset_Tax
1084     IF num=21:Energy_Bill
1085     IF num=29:Stolen_Stock
1086   END IF
1087   IF Inv>Invmax:Gs=4:QBWH_Menu
1088   IF bay1=0 AND Inv<=Invmax AND RND(1 TO 24)=7:bay1=1:Arrival(1)
1089   IF bay1>0 AND Clk>=Dtime1:Depart 1:bay1=0
1090   IF bay2=0 AND Inv<=Invmax AND RND(1 TO 24)=7:bay2=1:Arrival(2)
1091   IF bay2>0 AND Clk>=Dtime2:Depart 2:bay2=0
1092   k=CODE(INKEY$(#1,20))
1093   SELect ON k
1094     =  32:p=p+8:Bin_cls:IF p>216:p=192
1095     =192:p=192:Bin_cls:IF Stock(r,c-1, 1)=0:c=c-1:IF c<6:lev=1:Bin_lev
1096     =200:p=200:Bin_cls:IF Stock(r,c+1,1)=0:c=c+1
1097     =208:p=208:Bin_cls:IF Stock(r-1,c, 1)=0:r=r-1
1098     =216:p=216:Bin_cls:IF Stock(r+1,c,1)=0:r=r+1
1099     =68,100:IF box<>0 :Truck_Drop          :REMark (D)rop
1100     =80,112:IF box= 0  :Truck_Pick          :REMark (P)ick
1101     =76,108:lev=lev+1  :Bin_lev             :REMark (L)evel
1102     =83,115            :Stock_Request       :REMark (S)tock
1103     =77,109:Gs=1       :QBWH_Menu           :REMark (M)enu
1104     =232               :Test_Mode           :REMark  F1
1105   END SELect
1106 END REPeat Game_lp
1107 END DEFine
```

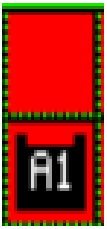
```
1109 DEFine PROCedure Store_Clk
1110 Clk=Tim+25200+((DATE-ClkOld)*cm):clock$=DATE$(Clk)
1111 ch=3:BLOCK#ch,30,10,124,8,7:STRIP#ch,7:INK#ch,0
1112 CURSOR#ch,126,8:PRINT#ch,clock$(13 TO 17)
1113 END DEFine

1115 DEFine PROCedure Bin_lev
1116 LOCal r,c:IF lev>4:lev=1
1117 ch=4:CLS#ch:INK#ch,7:BLOCK#ch,24,2,0,44-(8*lev),2
1118 IF box>=1 AND box<=24 :BLOCK#ch,12,6,4,36-(8*lev),248
1119 CURSOR#ch,20,34-(8*lev):PRINT#ch,lev
1120 FOR r=1 TO 7 STEP 2
1121   FOR c=6 TO 13:stk=Stock(r,c,lev):Bin_id:Bin_cls:Bin_prn
1122 END FOR r
1123 END DEFine

1125 DEFine PROCedure Bin_id
1126 stk$='  '
1127 SELect ON stk
1128  =  1 TO   8:stk$='A'&stk
1129  =  9 TO 16:stk$='B'&(stk-8)
1130  =17 TO 24:stk$='C'&(stk-16)
1131 END SELect
1132 END DEFine

1134 DEFine PROCedure Bin_cls
1135 ch=3:BLOCK#ch,22,18,2+c*24,5+r*20,2: IF c<=5 AND lev>1:lev=1:Bin_lev
1136 END DEFine

1138 DEFine PROCedure Bin_prn
1139 ch=3:INK#ch,7:PAPER#ch,2:CURSOR#ch,7+c*24,10+r*20:PRINT#ch,stk$:stk$='  '
1140 END DEFine

1142 DEFine PROCedure Stock_aud
1143 LOCal r,c,l:Asset=0
1144 FOR i=1 TO 24:Aud(i)=0
1145 FOR l=1 TO 4
1146  FOR r=1 TO 7 STEP 2
1147   FOR c=6 TO 13
1148    IF Stock(r,c,l)>=1 AND Stock(r,c,l)<=24
1149       Aud(Stock(r,c,l))=Aud(Stock(r,c,l))+1
1150       IF Aud(Stock(r,c,l))>8:Stock(r,c,l)=32:ELSE Asset=Asset+1
1151    END IF
1152   END FOR c
1153  END FOR r
1154 END FOR l
1155 FOR stk=1 TO 24:Stock_prn stk
1156 END DEFine

1158 DEFine PROCedure Score
1159 ch=6:INK#ch,7
1160 CURSOR#ch,54, 1:PRINT#ch,FILL$(' ',3-LEN(Sales))&Sales:INK#ch,4
1161 CURSOR#ch,54,12:PRINT#ch,FILL$(' ',3-LEN(Asset))&Asset
1162 END DEFine
```

```
1164 DEFine PROCedure Stock_prn(stk)
1165 ch=7:INK#ch,2
1166 SELect ON stk
1167  = 1 TO   8:CURSOR#ch,(stk*10) +10,12:PRINT#ch,Aud(stk)
1168  = 9 TO  16:CURSOR#ch,(stk*10) - 70,22:PRINT#ch,Aud(stk)
1169  =17 TO 24:CURSOR#ch,(stk*10)-150,32:PRINT#ch,Aud(stk)
1170 END SELect
1171 END DEFine
```

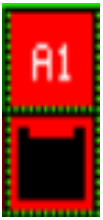

```
1173 DEFine PROCedure Truck_Pos(p)
1174 ch=3:BLOCK#ch,18,14,4+c*24,7+r*20,0
1175 SELect ON p
1176  =192:BLOCK#ch,4,10,4+c*24,9+r*20,2
1177  =200:BLOCK#ch,4,10,18+c*24,9+r*20,2
1178  =208:BLOCK#ch,12,4,7+c*24,6+r*20,2
1179  =216:BLOCK#ch,12,4,7+c*24,18+r*20,2
1180 END SELect
1181 INK#ch,7:STRIP#ch,0:CURSOR #ch,7+c*24,9+r*20:PRINT#ch,box$
1182 END DEFine
```




```
1184 DEFine PROCedure Truck_Pick
1185 rt=r:ct=c:l=lev                    :REMark rt, ct, l temp hold of r, c, lev
1186 IF p=208:r=r-1                     :REMark Pick Up
1187 IF p=216:r=r+1                     :REMark Pick Down
1188 IF c>5 AND c<14 AND Stock(r,c,l)>=1 AND Stock(r,c,l)<=24
1189    Aud(Stock(r,c,l))=Aud(Stock(r,c,l))-1
1190    stk=Stock(r,c,l):Stock(r,c,l)=32:Stock_prn stk:pick=1
1191    Bin_cls:Bin_prn:Bin_id:box=stk:box$=stk$
1192 END IF
1193 IF p=192:c=c-1                     :REMark Pick Left
1194 IF p=200:c=c+1                     :REMark Pick Right
1195 IF c<5 AND Stock(r,c,1)<25
1196   stk=Stock(r,c,1):Stock(r,c,1)=0:Bin_cls:Bin_prn:Bin_id:box=stk:box$=stk$
1197 END IF
1198 r=rt:c=ct:Bin_prn:IF box<>0:Bin_lev
1199 END DEFine
```



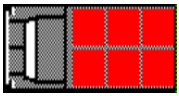

```
1201 DEFine PROCedure Truck_Drop
1202 rt=r:ct=c:l=lev
1203 IF p=208:r=r-1                     :REMark Drop Up
1204 IF p=216:r=r+1                     :REMark Drop Down
1205 IF c>5 AND c<14 AND Stock(r,c,l)=32
1206    Stock(r,c,l)=box
1207    Aud(Stock(r,c,l))=Aud(Stock(r,c,l))+1:Stock_prn box
1208    stk$=box$:Bin_prn:box=0:box$='  ':pick=0
1209 END IF
1210 IF p=192:c=c-1                     :REMark Left Drop
1211 IF p=200:c=c+1                     :REMark Right Drop
1212 IF c<5 AND Stock(r,c,1)=0
1213   Stock(r,c,1)=box:box=0:stk$=box$:Bin_prn:box$='  '
1214 END IF
1215 r=rt:c=ct:IF box=0:ch=4:BLOCK#ch,12,6,4,36-(8*lev),0
1216 END DEFine
```



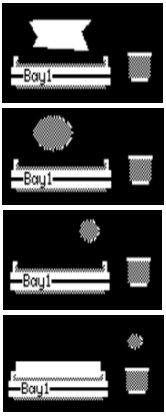
```
1218 DEFine PROCedure Arrival(bay)
1219 IF bay=1:py=16:ly=71:rl=2:ELSE py=171:ly=131:rl=5
1220 ar=r:ac=c:Prn_in:Lorry_in:n=0:ch=5:WINDOW#ch,352,192,gx+18,gy+31
1221 FOR r=rl TO rl+1
1222  FOR c=2 TO 4
1223   Stock(r,c,1)=0:Bin_cls
1224   IF Del=bay
1225    n=n+1:stk=SReq(n):Stock(r,c,l)=stk:Bin_id:Bin_prn
1226   END IF
1227  END FOR c
1228 END FOR r
1229 r=ar:c=ac:IF Del=bay:Din=1
1230 IF bay=1 AND bay2=1:dlay=300*Skill:ELSE dlay=0
1231 IF bay=1:ty=25 :Dtime1=dlay+Clk+(1200*Skill):Dept$=DATE$(Dtime1)
1232 IF bay=2 AND bay1=1:dlay=300*Skill:ELSE dlay=0
1233 IF bay=2:ty=152:Dtime2=dlay+Clk+(1200*Skill):Dept$=DATE$(Dtime2)
1234 STRIP#ch,5:INK#ch,0:CURSOR#ch,84,ty:PRINT#ch,Dept$(13 TO 17)
1235 END DEFine
```

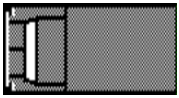
```
1237 DEFine PROCedure Prn_in
1238 ch=5:WINDOW#ch,70,30,gx+386,gy+py:PAPER#ch,0:CLS#ch
1239 n=0:PAPER#ch,7:INK#ch,0:OVER#ch,1:SCROLL#ch,-10
1240 IF Del=bay
1241    String$='Delivery':CLS#8      :REMark clear Stock Request Window
1242 ELSE
1243    String$='Invoice '&Inv:Inv=Inv+1
1244 END IF
1245 FOR i=0 TO 1:CURSOR#ch,2+i,20:PRINT#ch,String$
1246 FOR d=1 TO 2
1247 PAUSE 2:SCROLL #ch,-10
1248  FOR a=1 TO 3
1249   n=n+1:IF Del=bay:stk=SReq(n):ELSE stk=RND(1 TO 24):InB(bay,n)=stk
1250   Bin_id:CURSOR#ch,-22+a*24,20:PRINT#ch,stk$:stk$='  '
1251  END FOR a
1252 END FOR d
1253 END DEFine
```


```
1255 DEFine PROCedure Lorry_in
1256 ch=5:WINDOW#ch,120,40,gx+18,gy+ly:PAPER#ch,248
1257 FOR i=1 TO 19:PAN#ch,4:PAUSE 2
1258 BLOCK#ch,2,40,0,0,0
1259 FOR i=1 TO 5
1260  PAN#ch,4:BLOCK#ch,4,40,0,0,248:BLOCK#ch,4,2,0,4,0:BLOCK#ch,4,2,0,36,0
1261 END FOR i
1262 BLOCK#ch,2,32,0,4,0
1263 FOR i=1 TO 2
1264  PAN#ch,4:BLOCK#ch,4,2,0,4+i,0:BLOCK#ch,4,29-i,0,6+i,7:BLOCK#ch,4,2,0,36-i,0
1265 END FOR i
1266 BLOCK#ch,2,28,0,6,0
1267 FOR i=1 TO 3
1268  PAN#ch,4:BLOCK #ch,4,2,0,6,0:BLOCK#ch,4,2,0,19,0:BLOCK#ch,4,2,0,32,0
1269 END FOR i
1270 BLOCK#ch,6,4,0,0,7 :BLOCK#ch,4,4,0,0,0 :BLOCK#ch,6,4,0,36,7
1271 BLOCK#ch,4,4,0,36,0:BLOCK#ch,2,36,0,2,7:PAPER#ch,0:PAN#ch,4
1272 END DEFine
```



```
1274 DEFine PROCedure Depart(bay)
1275 IF bay=1:lr=2:py=16:ly=71:ty=21:ELSE lr=5:py=171:ly=131:ty=148
1276 dr=r:dc=c:n=0
1277 FOR r=lr TO lr+1
1278  FOR c=2 TO 4
1279   IF Del=bay
1280      IF Stock(r,c,1)=0:Sales=Sales-1:END IF
1281   ELSE
1282      n=n+1:IF Stock(r,c,1)=InB(bay,n):Sales=Sales+2:END IF
1283      IF Stock(r,c,1)>=1 AND Stock(r,c,1)<=24:Sales=Sales+1:END IF
1284   END IF
1285   Stock(r,c,1)=255
1286  END FOR c
1287 END FOR r
1288 IF Del=bay AND Din=1
1289     Del=0:Din=0
1290 ELSE
1291    Stock_aud:Score:ASRep(Ino,1)=Asset:ASRep(Ino,2)=Sales:Ino=Ino+1
1292 END IF
1293 ch=5:WINDOW#ch,352,192,gx+18,gy+31:BLOCK#ch,30,10,84,ty,5   :REMark Clr DTime
1294 r=dr:c=dc:Lorry_out:Prn_out:Stock_aud:Score
1295 END DEFine

1297 DEFine PROCedure Prn_out
1298 ch=5:WINDOW#ch,70,30,gx+386,gy+py:PAPER#ch,0:CLS#ch:INK#ch,7:FILL#ch,1
1299 LINE#ch,10,90 TO 170,90 TO 150,60 TO 170,15 TO 10,20 TO 25,50 TO 10,90
1300 PAUSE 8:CLS#ch:FILL#ch,1
1301 LINE#ch,30,80 TO 150,80 TO 130,60 TO 145,25 TO 30,30 TO 40,50 TO 20,80
1302 PAUSE 5:CLS#ch:INK#ch,248
1303 FILL#ch,1:CIRCLE#ch,70,60,40,.8,PI/2:PAUSE 5:CLS#ch
1304 FILL#ch,1:CIRCLE#ch,150,70,20:PAUSE 10:CLS#ch:BLOCK#ch,70,10,0,20,7
1305 ch=5:WINDOW#ch,20,30,gx+476,gy+py-6:PAPER#ch,0:CLS#ch:INK#ch,248
1306 FILL#ch,1:CIRCLE#ch,20,70,15:PAUSE 5:FOR i=1 TO 6:SCROLL#ch,5:PAUSE 3
1307 END DEFine

1309 DEFine PROCedure Lorry_out
1310 ch=5:WINDOW#ch,120,40,gx+18,gy+ly:PAPER#ch,0:BLOCK#ch,72,40,48,0,248
1311 FOR i=1 TO 30:PAN#ch,-4:PAUSE 1
1312 IF c<5
1313   IF bay=1 AND r<4:c=5:r=4:Bin_cls:Truck_Pos 192
1314   IF bay=2 AND r>4:c=5:r=4:Bin_cls:Truck_Pos 192
1315 END IF
1316 END DEFine

1318 DEFine PROCedure Stock_loss
1319 IF Asset<36:RETurn :ELSE ch=8:CLS#ch:INK#ch,7:OVER#ch,1
1320 CURSOR#ch,48,5:PRINT#ch,'Attention Stock LOSS!':att=RND(1 TO 7)
1321 IF att=1:FOR i=8 TO RND(9 TO 11):Stock(1,i,1)=32:Stock(7,i,1)=32
1322 IF att=3:FOR i=1 TO RND(2 TO  3):Stock(3,i,3)=32:Stock(5,i,3)=32
1323 IF att=5:FOR i=7 TO RND(8 TO 10):Stock(1,i,1)=32:Stock(7,i,1)=32
1324 IF att=7:FOR i=4 TO RND(5 TO  7):Stock(3,i,3)=32:Stock(5,i,3)=32
1325 Stock_aud:Score:FOR stk=1 TO 24:Stock_prn stk
1326 END DEFine
```





```
1328 DEFine PROCedure Store_err
1329 ch=8:CLS#ch:INK#ch,7:CURSOR#ch,36,5:PRINT#ch,'Store Computer Error!';
1330 FOR stk=1 TO 24:Aud(stk)=0:Stock_prn(stk)
1331 FOR t=1 TO 5:PAUSE 30:Store_Clk:ch=8:PRINT#ch,'!';
1332 CLS#ch:Stock_aud:FOR stk=1 TO 24:Stock_prn stk
1333 END DEFine
```


```
1335 DEFine PROCedure Asset_Tax
1336 IF Sales<42:RETurn
1337 ch=8:CLS#ch:INK#ch,7:Tr=INT(Sales/10):Sales=Sales-Tr
1338 CURSOR#ch,20,5:PRINT#ch,'Revenue Tax ';Tr;' credits paid':Score
1339 END DEFine
```


```
1341 DEFine PROCedure Energy_Bill
1342 IF Sales<36:RETurn
1343 ch=8:CLS#ch:INK#ch,7:Ec=INT(Asset/20):Sales=Sales-Ec
1344 CURSOR#ch,20,5:PRINT#ch,'Energy Bill ';Ec;' credits paid':Score
1345 END DEFine
```


```
1347 DEFine PROCedure Stolen_Stock
1348 IF Sales<24:RETurn
1349 ch=8:CLS#ch:INK#ch,7:Ls=RND(2 TO 6):Sales=Sales-Ls
1350 CURSOR#ch,20,5:PRINT#ch,'Lorry Theft ';Ls;' credits lost':Score
1351 END DEFine
```


```
1353 DEFine PROCedure Stock_Request
1354 IF Del>0 OR Sales<12:RETurn
1355 ch=8:CLS#ch:x=1:y=0:n=1:s=0
1356 INK#ch,7:CURSOR#ch, 4, 0:PRINT#ch,'Stock Request'
1357 INK #ch,4:CURSOR #ch,36,10:PRINT #ch,'Select 6 Items ← → ↑↓ '
1358 BLOCK#ch,20,4,140,14,4  :REMark SpaceBar
1359 Stock_aud:FOR stk=1 TO 24:Stock_prn stk
1360 REPeat Stock_lp
1361  Store_Clk:IF n>6:Del=RND(1 TO 2):RETurn
1362  ch=7:STRIP#ch,4:INK#ch,0:Stock_prn x+8*y
1363  k$=INKEY$(#1,20):k=CODE(k$)
1364  ch=7:STRIP#ch,0:INK#ch,2:Stock_prn x+8*y
1365  SELect ON k
1366    =192:IF x>1:x=x -1:ELSE x=1
1367    =200:IF x<8:x=x+1:ELSE x=8
1368    =208:IF y>0:y=y -1:ELSE y=0
1369    =216:IF y<2:y=y+1:ELSE y=2
1370    = 32:IF n<=6
1371          ch=8:INK#ch,7:stk=x+y*8:SReq(n)=stk:Bin_id
1372          CURSOR#ch,80+(n*18),1:PRINT#ch,stk$:n=n+1:str$=' '
1373         END IF
1374  END SELect
1375 END REPeat Stock_lp
1376 END DEFine
```

```
1378 DEFine PROCedure SelPath
1379 INK#ch,5:CURSOR#ch,8,6:PRINT#ch,'Select:⬆⬇    ;SD$;'⬅➡    ⬅'
1380 BLOCK#ch,12,3,186,10,5:BLOCK#ch,2,4,206,8,5:OVER#ch,0:INK#ch,7
1381 REPeat Path_lp
1382  CURSOR#ch,68,6:PRINT#ch,drv$(dn%):CURSOR#ch,156,6:PRINT#ch,af
1383  k=CODE(INKEY$(-1))
1384  SELect ON k
1385   =216:dn%=dn%  -1:IF dn%<0:dn%=dm%
1386   =208:dn%=dn% +1:IF dn%>dm%:dn%=0
1387   =200:fnum=fnum+1:IF fnum>9:fnum=9
1388   =192:fnum=fnum -1:IF fnum<0:fnum=0
1389   =  10:pck=1:EXIT Path_lp
1390   =  32:pck=0:EXIT Path_lp
1391  END SELect
1392 END REPeat Path_lp
1393 END DEFine
```

Note: BLOCK Spacebar - Enter tail


```
1395 DEFine PROCedure FCheck
1396 CLS#ch:CURSOR#ch,56,6:PRINT#ch,'Searching...'
1397 PAUSE 20:DELETE drv$(dn%)&'FList'
1398 OPEN_NEW#99,drv$(dn%)&'FList':DIR#99,drv$(dn%):CLOSE#99
1399 OPEN_IN#99,drv$(dn%)&'FList'
1400 REPeat Dir_lp
1401   IF EOF(#99):CLOSE#99:ck=0:CLS#ch:pck=0:EXIT Dir_lp
1402   INPUT#99,Fchk$:IF Fchk$==SD$&fnum:CLOSE#99:pck=1:EXIT Dir_lp
1403 END REPeat Dir_lp
1404 END DEFine
```

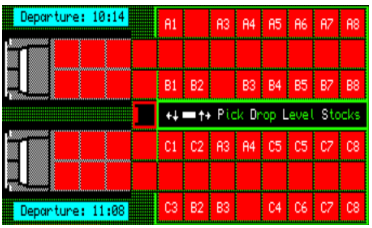

```
1406 DEFine PROCedure Load_Audit
1407 ch=8:CLS#ch:SelPath:IF pck=0:CLS#ch:RETurn :ELSE FCheck
1408 IF pck=0 OR eck=1
1409    CLS#ch:eck=0:CURSOR#ch,56,6:PRINT#ch,'File Not Found...'
1410    PAUSE 50:CLS#ch:RETurn
1411 END IF
1412 IF bay1>0:Depart(1):bay1=0
1413 IF bay2>0:Depart(2):bay2=0
1414 CLS#ch:CURSOR#ch,56,6:PRINT#ch,'Loading...';
1415 OPEN_IN#99,drv$(dn%)&SD$&fnum
1416 FOR l=1 TO 4
1417  FOR r=1 TO 7
1418   FOR c=1 TO 14:INPUT#99,Stock(r,c,l)
1419   CURSOR#ch,104+r*6,6:PRINT#ch,'.';:PAUSE 2
1420  END FOR r
1421 END FOR l
1422 FOR n=1 TO 40:INPUT#99,ASRep(n,1)\ASRep(n,2)
1423 INPUT#99,Inv\Ino\Invmax\Sales\Sk$(sk):CLOSE#99:CLS#ch
1424 Init_Layout:Gs=2:BLOCK#3,60,11,276,7,5:QBold 0,0,276,8,Sk$(sk)
1425 END DEFine
```

```
1427 DEFine PROCedure Save_Audit
1428 ch=8:CLS#ch::SelPath:IF pck=0:CLS#ch:RETurn :ELSE FCheck
1429 IF eck=1
1430   eck=0:CLS#ch:CURSOR#ch,56,6:PRINT#ch,'DEVICE ERROR'
1431   PAUSE 50:CLS#ch:RETurn
1432 END IF
1433 IF pck=1
1434   CLS#ch:CURSOR#ch,56,6:PRINT#ch,'Overwrite y/n':PAUSE
1435   IF KEYROW(5)<>64:CLS#ch:RETurn
1436 END IF
1437 FOR r=2 TO 3:FOR c=2 TO 4:Stock(r,c,1)=255:END FOR c:END FOR r
1438 FOR r=5 TO 6:FOR c=2 TO 4:Stock(r,c,1)=255:END FOR c:END FOR r
1439 DELETE drv$(dn%)&SD$&af:OPEN_NEW#99,drv$(dn%)&SD$&fnum
1440 CLS#ch:CURSOR#ch,56,6:PRINT#ch,'Saving...';:CLS#ch,2
1441 FOR l=1 TO 4
1442  FOR r=1 TO 7
1443   FOR c=1 TO 14:PRINT#99,Stock(r,c,l)
1444   CURSOR#ch,104+r*6,6:PRINT#ch,'.';:PAUSE 2
1445  END FOR r
1446 END FOR l
1447 FOR n=1 TO 40:PRINT#99,ASRep(n,1)\ASRep(n,2)
1448 Inv=Ino:PRINT#99,Inv\Ino\Invmax\Sales\Sk$(sk):CLOSE#99:CLS#8
1449 END DEFine
```


```
451 DEFine PROCedure ASReport
1452 INK#ch,2:LINE#ch,12,16 TO 100,16:LINE#ch,15,12 TO 15,80
1453 INK#ch,4:CURSOR#ch,12,16:PRINT#ch,'Assets\';
1454 INK#ch,7:PRINT#ch,'Sales  Invoice num: ';Ino-1;'/';Invmax
1455 x=16:y=18:z=2:IF Asset>120 OR Sales>120:z=4
1456 FOR n=2 TO 80 STEP 2
1457   INK#ch,4:LINE#ch,x+n,y TO x+n,y+ASRep(n/2,1)/z
1458   INK#ch,7:LINE#ch,x+n+1,y TO x+n+1,y+ASRep(n/2,2)/z
1459 END FOR n
1460 END DEFine
```


```
1462 DEFine PROCedure Init_Stock
1463 RESTORE 1472
1464 FOR r=1 TO 7
1465  FOR c=1 TO 14
1466   READ Stock(r,c,1)
1467   IF c>=6 AND c<=13
1468    n=RND(1 TO 32):IF n>24:n=32
1469    Stock(r,c,3)=n:Stock(r,c,2)=32:Stock(r,c,4)=32
1470   END IF
1471  END FOR c
1472 END FOR r
1473 DATA 225,225,225,225,255,1,32,3,4,5,6,7,8,255
1474 DATA 225,225,225,225,0,0,0,0,0,0,0,0,0,255
1475 DATA 225,225,225,225,0,9,10,32,11,12,13,15,16,255
1476 DATA 225,225,225,225,0,255,255,255,255,255,255,255,255,255
1477 DATA 225,225,225,225,0,17,18,3,4,21,21,23,24,255
1478 DATA 225,225,225,225,0,0,0,0,0,0,0,0,0,255
1479 DATA 225,225,225,225,255,19,10,11,32,20,22,23,24,255
1480 END DEFine
```

Note Data lines represent **the Store** Layout, Storage Rows, PickUp Tracks and Lorry Bays.

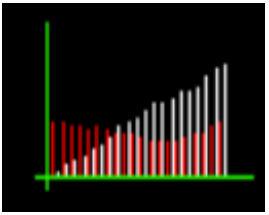


```
1482 DEFine PROCedure QBWH_Init
1483 CLS#2:F1=0:stk=0:r=4:c=5:l=1:p=192:bay1=0:bay2=0
1484 REMark ** Store Layout **
1485 ch=3:OPEN#ch,scr_:WINDOW#ch,356,194,gx+16,gy+26
1486 BORDER#ch,1,2:PAPER#ch,32:CLS#ch
1487 BLOCK#ch,120,50,0,38,0:BLOCK#ch,120,50,0,98,0  :REMark Drive Bay 1&2
1488 REMark ** QBITS Title **
1489 ch=3:WINDOW#ch,220,41,gx+138,gy+6:BORDER#ch,1,4:PAPER#ch,0:CLS#ch
1490 OVER#ch,1:CSIZE#ch,2,1
1491 INK#ch,2:FOR i=0 TO 3:CURSOR#ch,14+i,4:PRINT#ch,'QBITS Warehouse'
1492 INK#ch,6:FOR i=0 TO 1:CURSOR#ch,16+i,5:PRINT#ch,'QBITS Warehouse'
1493 OVER#ch,0:BLOCK#ch,212,1,2,25,4
1494 REMark ** Headings **
1495 ch=3:WINDOW#ch,352,192,gx+18,gy+27:CSIZE#ch,0,0
1496 BLOCK#ch, 74,11,124   , 7,7:QBold 0,1,156,8,'TIME'
1497 BLOCK#ch, 60,11,276,    7,5:QBold 7,1,208,8,'(M)enu'
1498 BLOCK#ch,106,12, 12,  24,5:QBold 0,0,12, 25,'Departure:'
1499 BLOCK#ch,106,12, 12,151,5:QBold 0,0,12,152,'Departure:'
1500 REMark ** PickUp **
1501 ch=4:OPEN#ch,scr_:WINDOW#ch,80,42,gx+30,gy+6:BORDER#ch,1,2:PAPER#ch,0
1502 CLS#ch:Init_PickUp:QBold 7,1,-6,0,'Level':WINDOW#ch,30,40,gx+74,gy+7
1503 REMark ** Printers Bay 1 & 2 **
1504 ch=5:OPEN#ch,scr_
1505 FOR b=1 TO 2
1506   IF b=1:y=36:ELSE y=191
1507   WINDOW#ch,82,24,gx+380,gy+y                    :REMark Printer
1508   BLOCK#ch,78,16,2,6,7:BLOCK#ch,74,16,4, 8,248:BLOCK#ch,82,10,0,12,7
1509   BLOCK#ch,82,2,0,16,0:BLOCK#ch,70,10,6, 0,  7
1510   STRIP#ch,7:INK#ch,0  :CURSOR#ch,10,12:PRINT#ch,'Bay'&b
1511   WINDOW#ch,20,16,gx+476,gy+y+4                  :REMark Waste basket
1512  BLOCK#ch,20,2,0,0,7 :BLOCK#ch,14, 2,3,14,  7:BLOCK#ch,18,12,1,2,248
1513 END FOR b
1514 REMark ** Score **
1515 ch=6:OPEN#ch,scr_:WINDOW#ch,80,24,gx+30,gy+193:BORDER#ch,1,2:PAPER#ch,0
1516 CLS#ch:QBold 7,1,0,1,'Sales':QBold 4,1,0,12,'Assets'
1517 REMark ** QL2K PC **
1518 ch=7:OPEN#ch,scr_:WINDOW#ch,120,80,gx+380,gy+76:PAPER#ch,0:CLS#ch
1519 BLOCK#ch,116,62,2,0,7 :BLOCK#ch,114,60,3,1,248:BLOCK#ch,110,60,5,1,7
1520 BLOCK#ch,108,58,6,2,0 :BLOCK#ch,120,14,0,64,7  :BLOCK#ch,76,1,36,70,0
1521 BLOCK#ch,14,2,80,71,0 :BLOCK#ch,14,2,98,71,0    :BLOCK#ch,116,2,2,78,248
1522 QBold 0,1,0,66,'QPC2':WINDOW#ch,100,56,gx+388,gy+80:PAPER#ch,0:CLS#ch
1523 QBold 7,1,20,0,'STOCKS':INK#ch,4:CURSOR#ch,0,12:PRINT#ch,'A'\'B'\'C'
1524 FOR i=1 TO 8:CURSOR#ch,10+(i*10),44:PRINT#ch,i
1525 REMark ** Requests & Messages **
1526 ch=8:OPEN#ch,scr_:WINDOW#ch,220,22,gx+138,gy+195:BORDER#ch,1,4:PAPER#ch,0
1527 END DEFine

1529 DEFine PROCedure QBold(col,cs,cx,cy,str$)
1530 INK#ch,col:OVER#ch,1
1531 FOR a=1 TO LEN(str$)
1532   FOR b=0 TO cs:CURSOR#ch,cx+b+a*(cs+6),cy:PRINT#ch,str$(a)
1533 END FOR a:OVER#ch,0
1534 END DEFine
```

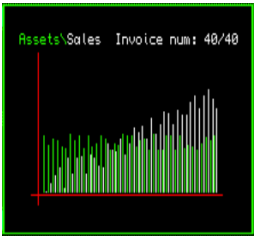
```
1536 DEFine PROCedure Init_Layout
1537 ch=3:WINDOW#ch,352,192,gx+18,gy+27:BLOCK#ch,222,148,120,20,32
1538 BLOCK#ch,196,1,142, 23,4:BLOCK#ch,2,143,338,23,4:BLOCK#ch,1,60,144, 65,4
1539 BLOCK#ch,196,1,142,165,4:BLOCK#ch,2, 20,142,23,4:BLOCK#ch,2,20,142,145,4
1540 BLOCK#ch,196,20,144,84,4 :BLOCK#ch,192,16,146,86,0:INK#ch,7:OVER#ch,1
1541 CURSOR#ch,152,89:PRINT#ch,'⬅⬆  ⬇➡':BLOCK#ch,12,4,168,92,7
1542 CURSOR#ch,200,89:PRINT#ch,'P  D  L  S':INK#ch,4
1543 CURSOR#ch,200,89:PRINT#ch,' ick  rop  evel  tocks':OVER#ch,0
1544 FOR d=1 TO 7
1545   FOR a=1 TO 14:IF Stock(d,a,1)<200:BLOCK#ch,22,18,2+a*24,5+d*20,2
1546 END FOR d
1547 END DEFine
```

```
1549 DEFine PROCedure Init_PickUp
1550 INK#ch,32:FILL#ch,1:CIRCLE#ch,46,32,5,3,-.8:FILL#ch,0          :REMark Legs
1551 INK#ch,32:FILL#ch,1:CIRCLE#ch,28,44,18,.6,PI:FILL#ch,0        :REMark Body
1552 INK#ch, 7:FILL#ch,1:CIRCLE#ch,30,65,5:FILL#ch,0               :REMark Head
1553 BLOCK#ch,2,1,16,13,0:BLOCK#ch,2,3,12,11,2:BLOCK#ch,6,2,13,10,2:REMark Hair
1554 INK#ch, 7:FILL#ch,1:CIRCLE#ch,50,42,4,.5,PI/2:FILL#ch,0       :REMark Hand
1555 BLOCK#ch,1,8,32,22,2  :BLOCK#ch,4,2,28,22,2: FILL#ch,1:INK#ch,2
1556 LINE#ch,8,8 TO 68,8 TO 68,20 TO 44,20 TO 44,30 TO 16,30       :REMark PickUp
1557 LINE#ch TO 16,50 TO 9,50 TO 9,30 TO 8,30 TO 8,8:FILL#ch,0
1558 INK#ch,0:LINE#ch,9,21 TO 18,21:LINE#ch,9,26 TO 18,26          :REMark Vents
1559 INK#ch,2:LINE#ch,72,4 TO 72,90:LINE#ch,74,4 TO 74,90          :REMark Lift bar
1560 INK#ch,7:FILL#ch,1:CIRCLE#ch,16,7,8:FILL#ch,0                 :REMark Left Wheel
1561 INK#ch,7:FILL#ch,1:CIRCLE#ch,55,7,8:FILL#ch,0                 :REMark Right Wheel
1562 INK#ch,0:CIRCLE#ch,16,7,5:CIRCLE#ch,55,7,5                    :REMark Wheel Hubs
1563 END DEFine
```

```
1565 DEFine PROCedure AS_Demo1
1566 INK#ch,4:LINE#ch,10,20 TO 110,20:LINE#ch,15,15 TO 15,90
1567 a=50:s=0:x=16:y=22:demo=2
1568 FOR n=2 TO 80 STEP 4
1569    o=INT(RND(1 TO 24)/RND(4 TO 8)):d=RND(5 TO 6)
1570    IF a>RND(8 TO 24):s=s+o*3:a=a-o
1571    IF s>28 AND RND(1 TO 24)=7:a=a+d:s=s-d
1572    IF n>40 AND RND(1 TO 3)=3:a=a+6
1573    INK#ch,2:LINE#ch,x+n,y TO x+n,y+a/2
1574    INK#ch,7:LINE#ch,x+n+2,y TO x+n+2,y+s/2
1575 END FOR n
1576 END DEFine
```



```
1578 DEFine PROCedure AS_Demo2
1579 FOR n=1 TO 40
1580  ASRep(n,1)=(50+RND(-12 TO 2)):ASRep(n,2)=(n*2+RND(-6 TO 18))
1581 END FOR n
1582 lno=41:lnvmax=40:ASReport
1583 lno=1:lnvmax=20:FOR n=1 TO 40:ASRep(n,1)=0:ASRep(n,2)=0
1584 END DEFine
```

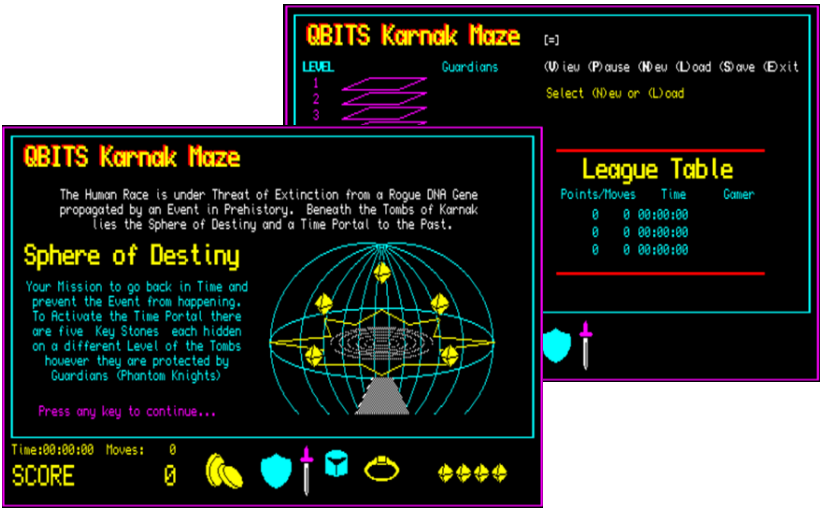



```
1586 DEFine PROCedure QBWH_Intro
1587 WINDOW#2,512,224,gx,gy    :PAPER#2,0:BORDER#2,1,3:CLS#2
1588 WINDOW#0,512, 32,gx,gy+224:PAPER#0,0:BORDER#0,1,3:CLS#0
1589 CSIZE#2,2,1:OVER#2,1:str$='Welcome to QBITS WareHouse'
1590 INK#2,2:FOR i=0 TO 1 :CURSOR#2,94+i,17:PRINT#2,str$
1591 INK#2,6:FOR i=0 TO 1 :CURSOR#2,96+i,18:PRINT#2,str$
1592 CSIZE#2,0,0:OVER#2,0 :INK#2,5:RESTORE 1595
1593 :
1594 FOR i=1 TO 9:READ x,y,str$:CURSOR#2,x,y:PRINT#2,str$
1595 DATA 86,44,'Check Bay Printouts for Invoice Requests and Deliveries'
1596 DATA 80,56,'As Lorries arrive you Load (Invoice) or Unload (Delivery)'
1597 DATA 112,68,'Use      Cursors and    Spacebar to move PickUP'
1598 DATA 162,80,'For Pick or Drop use   &   keys'
1599 DATA 100,92,'Use   for other Levels and   for New Stock Requests'
1600 DATA 82,104,'Lorries Depart on Time irrespective of Load/Unload Status'
1601 DATA 84,116,'As Invoices are fulfilled Sales Increase. Check PC Stock'
1602 DATA 84,128,'as this shows the Assets currently held in the Warehouse'
1603 DATA 106,146,'To Access Menu ( ew  oad  ave  xit) press   key'
1604 :
1605 INK#2,7:FOR i=1 TO 9:READ x,y,str$:CURSOR#2,x+i,y:PRINT#2,str$
1606 DATA 288,80,'P',310,80,'D',122,92,'L',260,92,'S'
1607 DATA 196,146,'N',220,146,'L',248,146,'S',278,146,'E',350,146,'M'
1608 CURSOR#2,160,176:PRINT#2,'press any key to continue...'
1609 INK#2,7:CURSOR#2,136,68:PRINT#2,'←↑↓→':BLOCK#2,14,3,240,72,7
1610 WINDOW#1,80,40,gx+72,gy+172:ch=1:Init_PickUp:BLOCK#2,24,2,112,206,2
1611 WINDOW#1,100,60,gx+360,gy+160   :AS_Demo1   :PAUSE
1612 END DEFine
```

QBITS WareHouse Test Mode

Test Mode was added to check various aspects of the Graphics and Procedures.

```
1800 DEFine PROCedure Test_Mode
1801 F1=1:ATemp=Asset:STemp=Sales:Sales=50
1802 REPeat Test
1803  ch=8:CLS#ch:PRINT#ch,'Test Mode':Score:tk=CODE(INKEY$(-1))
1804  SELect ON tk
1805    =49:bay1=1:Arrival 1                  :REMark (1)Bay1_in Print & Lorry
1806    =50:Depart 1:bay1=0                   :REMark (2)Bay1_out Print & Lorry
1807    =51:bay2=1:Arrival 2                  :REMark (3)Bay2_in Print & Lorry
1808    =52:Depart 2:bay2=0                   :REMark (4)Bay2_out Print & Lorry
1809    =53:Store_err                         :REMark (5)Compute Crash
1810    =54:Stock_loss  :PAUSE                :REMark (6)Assets Lost
1811    =55:Asset_Tax   :PAUSE                :REMark (7)Loss of Sales credits
1812    =56:Energy_Bill :PAUSE                :REMark (8)Loss of Sales credits
1813    =57:Stolen_Stock:PAUSE                :REMark (9)loss of Sales credits
1814    =97:Sales=Sales+6:Stock_aud           :REMark (a)Increase Sales
1815    =65:Sales=Sales-6:Stock_aud           :REMark (A)Decrease Sales
1816    =67:Tim=Tim+3600 :Store_Clk           :REMark (C)Clock 1hr increments
1817:    =71:Gs=4:QBWH_Menu                   :REMark (G)Game End
1818    =83:Stock_Request                     :REMark (S s) Note:Sales +12
1819    =232:F1=0:CLS#8:Asset=ATemp:Sales=STemp:Score:EXIT Test
1820  END SELect
1821 END REPeat Test
1822 END DEFine
```



Karnak Maze Introduction

Maze algorithms in today's Gaming World are organized in various Classifications involving Dimensions, Topology, Tessellation, Routing Texture, Focus and more. For Game design they will typically use a combination of these classifications. A two-dimensional Maze uses simple compass directions. Three-dimensional Mazes use stairways and bridges to connect across one area of a Maze to another. Fourth dimensional Mazes, use Portals to Transport between Past and Future areas of the Maze.

Maze Creating Algorithms

Maze generation is usually a predetermined arrangement of cells, commonly based around a rectangular grid, but other arrangements are possible. A Maze Generation Algorithm is to fulfil the challenge of finding a route between any two particular cells. They are usually in the form of a Random Spanning Tree, which simply means a tree structure with branches that form the minimum number of undirected links between all cells.

The **Recursive Backtracking** Algorithm creates a pathway by Randomly selecting an unvisited cell adjacent to one of the sides of the present one. The path is then made by knocking down the wall between. Moving into the new cell, the random selection continues until there are no surrounding unvisited cells to select. The visited cells of the pathway are then backtracked until one is found to still have an unvisited adjacent cell or cells. The passage way is carved forward again and so on until there are no unvisited cells.

There are a solid handful of Algorithms similar to the above, two others were chosen, one called **Prim's** which randomly selects a visited cell from the generated list, then continues forward similarly carving a pathway through unvisited cells. A third sometimes referred to as **Hunt and Kill** purposely moves through the grid selecting a cell and if a Randomly chosen side has an unvisited cell will carve a path between the two.

QBITS Karnak Maze
The aspiration of this Prog was to create a theme and explore the possibilities of a Maze Game. The aim is to seek out the **Treasures** of the **Maze** while defending against and defeating the **Guardians** encountered along the way. To reach each new level, you have to find a **Key Stone**. The last of these give access to the **Sphere of Destiny** where the collected **Key Stones** have to be arranged to open a **Time Portal** to an event in prehistory. This is to prevent the creation of a **Rogue DNA Gene** in the distant past that in today's world is threatening the extinction of the human race.

QBITS Maze - Treasures
Moving around the Maze on each Level will uncover a number of Treasures; **Coins** of Karnak, **Mask** of Wisdom, **Ring** of Power and one of the **Key Stones**, collection of each adds Points to the **Score.** The **Mask** and **Ring** are assets that also aid in defending against or eliminating one or more of the **Guardians**. The **Key Stone** is required to activate the **Portal** between **Levels** and then access to the **Sphere of Destiny**.

QBITS Maze - Guardians
Encountering a **Guardian** a Player has four possible options, **[1] Shield** which avoids the confrontation by Teleporting to another part of the current Maze level. **[2] Sword**, here you attack and are required to throw a six to defeat the **Guardian**. If acquired **[3] Mask** will banish Level **Guardians** for 120 moves and **[4] Ring** will delete all of the current **Level Guardians**. Each of these choices will incur a loss of Points.

QBITS Maze - Levels
You can accept a **Key Stone** or return to it later. Accepting activates the **Portal** and makes the jump to the next **Maze Level**. The final jump to the **Sphere of Destiny** can only be made if all remaining **Guardians** have been defeated.

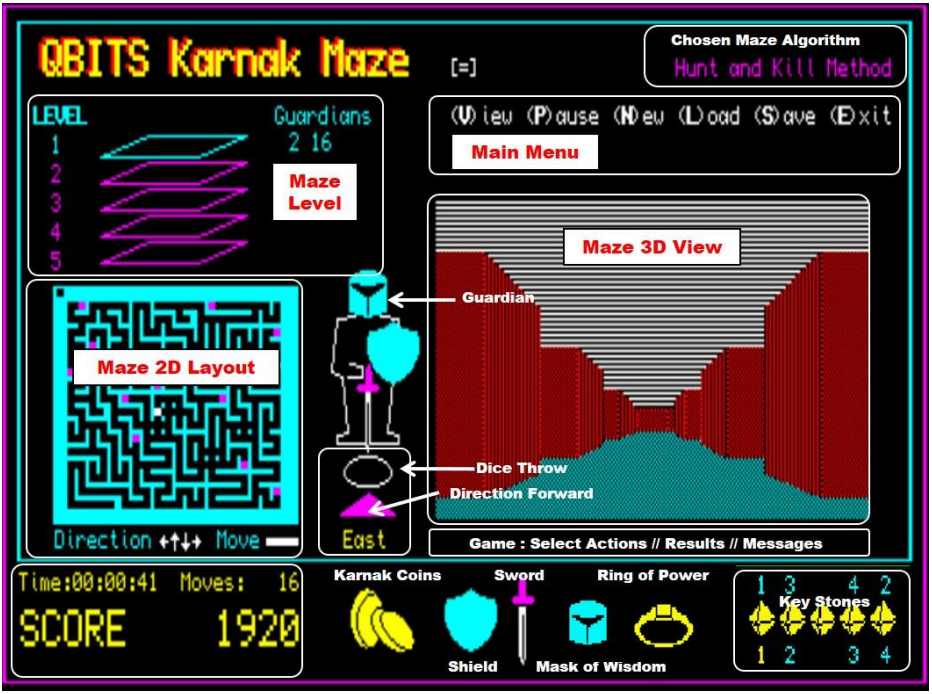
QBITS Maze - Sphere of Destiny
Upon reaching the **Sphere of Destiny** the five acquired **Key Stones** from the Maze Levels have to be aligned to their correct position to those within the Sphere. The fifth Key Stone position is given, the other four must be **Matched**. Twenty-four different combinations are possible. If successful the **Time Porta**l is open and humanity saved from extinction. If not then sorry, maybe you will have better luck next time.

QBITS Maze - Strategy
Apart from the **Key Stone** all Treasures are acquired upon entering the Grid cell containing them and any points added to the Score. The **Key Stone** location offers a choice of taking the **Key Stone** or leaving it for a later pickup. This allows gathering more treasure - increasing the Score, and eliminating further Guardians on the present Level. The final stage is matching the **Key Stones** to their correct positions in the **Sphere of Destiny** and may take a number of tries. This can deplete your **Score** dramatically, leading to a failed attempt.

The strategy is therefore a balance between gaining the highest **Point count** as possible with minimum **Moves**, while still managing to defeat all the **Guardians**. Thereby, preparing for multiple tries when attempting to activate the **Time Portal** and save **Humanity**.

QBITS Karnak Maze - a Walk Through
The opening screen displays a Warning and Mission Statement, plus Graphics representing the **Sphere of Destiny.** This is the where the final part of the mission is played out. Pressing any key reveals the Game board and the League Table. At this point you may Start a **New** Game or **Load** a previously saved one.

See below Title '**QBITS Karnak Maze**' the **Maze Levels** the active level shown in a different colour and the number of **Defending Guardians** to defeat. To the right of these the **Main Menu** with two additions (**V**)iew (**P**)ause to the usual (**N**)ew (**L**)oad (**S**)ave (**E**)xit.



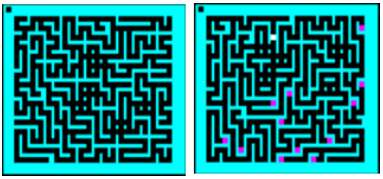
Centre of the screen is a **Guardian** Knight standing over an **Ellipse** within which a random number is shown when a Player uses the **Sword** option to attack. The **Triangle** below and printed **Compass Direction** indicate the forward direction through the Maze.

Below the **Maze 2D** layout **Direction** is changed as shown by use of the **CURSOR keys**, this will action a change in the **3D** Maze view and the forward-facing direction of **North**, **East**, **West** or **South.** To **Move** forward press the **Spacebar**.

Bottom left shows the **Score Board**, **Game Time**, number of **Moves** and total **Points** collected. For the rest of the lower screen area Game icons are displayed. In the middle; **Coin**s of Karnak, **Shield**, **Sword**, and as acquired the **Mask** of Wisdom and **Ring** of Power. On the far right the five **Key Stones** are displayed as each is collected from the **levels** of the **Maze**.

QBITS Maze - (V)iew

This is an **ON/OFF** switch that displays the location of **Maze Treasures** and the **Current Position** of the player within the **Maze 2D** layout. The number of **Points** taken for **each move** depends on the **Level** and if **View** is switched **ON** or **OFF.**



QBITS Maze - (P)ause

The **Game Timer** is halted and time stored (**GTS**). Message displayed '**Press any key to continue…**', pressing of which will restart the Game Timer and continuation of the game.

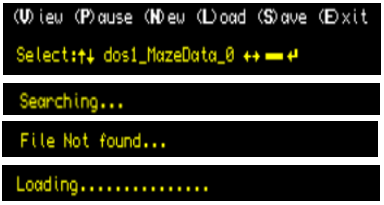
QBITS Maze - (N)ew Game

For a **(N)ew** Game Select from one of three [1] [2] [3] presented Maze Algorithms. Top right displays the one chosen. To abort press <**Spacebar**> <**Enter**> sets things in motion creating a **2D Maze** Layout in the lower left-hand part of the screen (Window#3).

The first **Maze 2D** Level of a **New Game** is drawn with a **Pause delay** to show its construction. For other levels depending on the speed of QL environment there is no Pause delay and the Maze may appear almost instantaneously. A **3D** view of the present location within in the **Maze** (Window#1) is shown to the right. The Maze Level is highlighted and the number of **Guardian**s deployed out of a total of sixteen. The 'Press any key to continue…' prompt is displayed, any key depressed will start the **Game**.

QBITS Maze - (L)oad

Select **Device** and **MazeData** File <**Spacebar**> will abort or continue with <**Enter**>. Activation and a search is made returning a '**File NOT found**' or continues with '**Loading…**'.

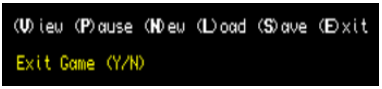


QBITS Maze - (S)ave

Select **Device** and **MazeData** File, press <**Spacebar**> to abort or continue with <**Enter**>. Activation and the search will return **DEVICE ERROR** if not found. If File exists the **Overwrite y/n** prompt is displayed. No aborts the action. If not previously existing or reply is 'Yes' the **MazeData** File is Saved to selected Device.

QBITS Maze - (E)xit

Prompted with 'Y/N' Any key other than 'Y' or 'y' will return to the Game. A Yes will **LRUN** the **QBITSProgs** Menu.



QBITS Maze - SCORE

The **SCORE** shows the **Game Duration** in hours, minutes and seconds a count of the **Moves** taken and **Points** Accrued. The Timer uses QL Super/SBASIC Commands DATE to set the **Game Clock** (**Gclk**) at start and DATE$ to create an hh:mm:ss display.

ie. clk$**=DATE$(DATE-Gclk**+**GTS**) : PRINT **clk$**(13 to 20)

GTS holds the current **Game Time Seconds** for a Game /Pause/Save/Load.

The **Moves** and **Points** are printed using the FILL$ command and with spaces so the counters grow right to left as the number increases.

ie. PRINT 'SCORE ',FILL$(' ',6-LEN(snum))&snum

QBITS Maze – Points Table

When **Points** are **Gained** or **Lost**, these changes are ascribed to variable snum.

Maze Moves Calculator **View OFF** **View ON**

On each move **S**core Points are **L**ost: **sl**=**lev** (ie.1 to 5) or **sl**=**lev***5 (ie, 5 to 25)

QBITS Maze Treasure Gains

As you move around the Maze, check the dead-end passageways as they may contain a hidden doorway to a **Treasure**. Apart from adding valuable **Points** to the Score they may be helpful when dealing with the **Guardians**.

Coins	- Random Selections (100 to 300)	snum=snum+50+50*RND(2 to 6)
Mask	- Increases with level (100 to 300)	snum=snum+50+50*lev
Ring	- Increases with level (1000 to 3000)	snum=snum+500+500*lev
KeyStone	- In acquiring Activates the Portal	snum=snum+2000

QBITS Maze Guardian Encounters

Confronted with a **Maze Guardian** you have between two to four choices the **Shield**, **Sword** and if acquired the **Mask** and/or **Ring.** The first two are given at the beginning of each level of Game the other two have to be found and acquired on each of the five Levels.

[1]Shield	- Portal Jump	snum=snum-50
[2]Sword	- Each dice thrown if not a 6	snum=snum-50
	(if dice throw is a 6 Delete Guardian)	gmax=gmax-1:glev=glev-1
[3]Mask	- Banish Guardian for 120 moves	snum-snum-50-50*lev
[4]Ring	- In Deleting Level Guardians	gmax=gmax-glev:glev=0
		snum=snum-500-500*lev

QBITS Maze Sphere of Destiny

For each failed try to Match the Key Stones snum=snum-500

QBITS Maze - Vector Graphics

As computer games developed from the early nineteen-eights it was the graphical displays that most impressed and intrigued, releasing in some cases a rewarding talent of expression. Bitmap images have their place, but may require a considerable amount of DATA. Vector Graphics on the other hand offer so much more and, in many cases, minimal codding. Vectors graphics use straight or curved lines drawn between coordinated points; this makes them easily scalable.

When drawing vector graphics with QL S/SuperSBASIC you are using the Graphics coordinate system as opposed to Pixel coordinates. This has a couple of idiosyncrasies; the simple one is related to CURSOR coordinates. When attached to a Graphic drawing four coordinates are used, the first pair interpreted as Graphical with the second as a Pixel offset.

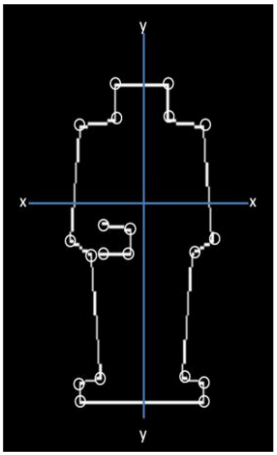
CURSOR gx, gy, px, py

The second relates to a drawn object as shown below and the use of FILL. When an object is filled with a solid colour QL S/SuperBASIC FILLs between min and max line coordinates so it looks a little different to what you might expect. To overcome this some shapes will need to be drawn as multiple objects. For example, when drawing the Guardians Helmet Visor as a single shape (a), it FILLs as shown (b). What is needed is to be drawn as two joined triangles.

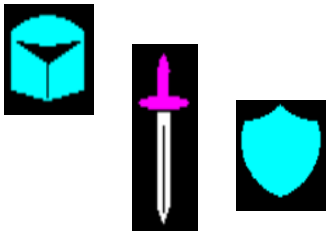
.

QBITS Maze - Guardian

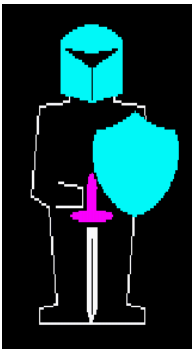
Vector Graphics is a bit like joining the dots. After setting the scale and location of the x, y zero coordinates; you need to work out the offsets to each position that describes the object. To create the Maze Guardian, I based this on an image of a Knight taken from an old church brass rubbing. The body with the head masked by a helmet, holding a Shield over the left arm and the Sword held with the blade tip down at the feet.



For the Maze Guardian this requires first drawing the outline of a body image with an added sword arm.



Taking the maze Treasures of the Mask, Sword and Shield and combining them with the Guardian body outline they create the Phantom Knight as a Tomb Defender.



QBITS Karnak Maze Code

```
1000 REMark QBITS_Maze_v4 [QBITS Maze Game v4 2021 QPCII]

1002 OPEN_IN#9,'ram2_QBITSConfig':INPUT#9,gx\gy\dn$\dev$\dn%\dm%
1003 DIM drv$(15,5):FOR d=0 TO 15:INPUT#9,drv$(d):END FOR d:CLOSE#9

1005 WHEN ERRor
1006  eck=1:CONTINUE
1007 END WHEN

1009 REMark Arrays
1010 SD$='MazeData_':LLoad
1011 DIM dir$(4,5)  :RESTORE 1011:FOR c=1 TO 4:READ dir$(c)
1012 DATA 'West ','East ','North', 'South'
1013 DIM SKey(5,3):RESTORE 1013:FOR i=1 TO 5:READ SKey(i,1):READ SKey(i,2)
1014 DATA -34,32, -40,3, 40,3, 34,32, 0,50
1015 DIM grid(21,17),cell(20*16):w=20:h=16
1016 DIM Mkey(5),Tres(12,3),name$(3,10),Grad(3,3)

1018 REMark Variables
1019 w=20:h=16:x=0:y=0:cx=0:cy=0:px=0:py=0  :REMark Various Coordinates
1020 lev=1:glev=2:gmax=16:gdel=120          :REMark Maze Level/Guardians
1021 Gclk=DATE:GTS=0:sm=0:snum=0:sl=1       :REMark Score/Time/Moves/Points
1022 bc=0:sc=0:tc=0:scol=0:col=0            :REMark Various Colours
1023 gst=0:fd=3                             :REMark Maze Moves/Direction
1024 eck=0:pck=0:mck=0:TM=0                 :REMark Checks:ERR/FilePath/MazeGen/Test Mode

1026 MODE 4:f=0:m=1:Init_Win:Init_Maze:LScore:Mes1:QBITS_Maze
```

Note: Game Checks

Often referred to as program cheats, hidden commands are a Programmers way to check actions and possibly to expose weaknesses in the code. For QBITS Maze **F1-F5** provides game controls to test the Game in various ways. Press the Equal key to action any of the **F1- F5** keys. Top mid screen the symbol [=] is given as a visual reminder.

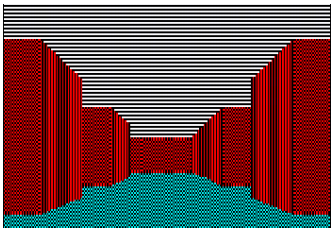
F1 Activates **Guardian** with **[1]Shield [2]Sword [3]Mask [4]Ring** – Full Action Choices.
F2 Activates the **Portal** (**Y/N**) forces a jump to the next level.
F3 adds 50 to SCORE
F4 subtracts 50 from SCORE
F5 For **Maze Levels** this flashes the **Key Stone** Location in the **2D Maze**

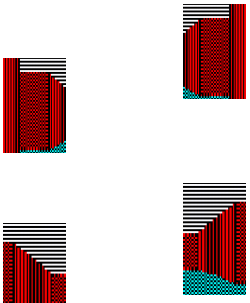
For the **Sphere of Destiny,** it identifies the order of the **Key Stones**.

```
1028 DEFine PROCedure QBITS_Maze
1029 REPeat Maze_lp
1030  IF gst=1:Score:ELSE Gclk=DATE:Score
1031  IF gdel=0:MGuard
1032  k=CODE(INKEY$(20))
1033  SELect ON k
1034    =  61:IF TM=0:TM=1:ELSE TM=0          :REMark [=]  Test Mode
1035    =232:IF TM=1:km=1:kr=1:MGuard         :REMark [F1] Guardians
1036    =236:IF TM=1:PortChk:GView            :REMark [F2] Open Portal
1037    =240:IF TM=1:snum=snum+50:Score       :REMark [F3] Score +
1038    =244:IF TM=1:snum=snum-50:Score       :REMark [F4] Score -
1039    =248:IF TM=1:tc=0:GView:tc=3:GView    :REMark [F5] Key Stone
1040    = 86,118:GView                        :REMark [V]iew ON/OFF
1041    = 80,112:GPause:Gclk=DATE             :REMark [P]ause
1042    = 78,110:GNew :mck=1:GPause           :REMark [N]ew (Maze Check)
1043    = 76,108:MLoad:mck=1:GPause           :REMark [L]oad (Maze check)
1044    = 83,115:IF mck=1:MSave               :REMark [S]ave
1045    = 69,101:GExit                        :REMark [E]xit
1046    =192:IF gst=1:fd=1:MazView            :REMark Left  West
1047    =200:IF gst=1:fd=2:MazView            :REMark Right East
1048    =208:IF gst=1:fd=3:MazView            :REMark Up    North
1049    =216:IF gst=1:fd=4:MazView            :REMark Down  South
1050    ON k=32                               :REMark SpaceBar Forward
1051      IF snum<   5:Mes2:GO TO 1067
1052      IF snum<  50+  50*lev:BLOCK#4,30,30,300,216,0:km=0
1053      IF snum<500+500*lev:BLOCK#4,50,30,332,216,0:kr=0
1054      IF fvn=1
1055        INK#2,5:CURSOR#2,236,190:PRINT#2,'Solid Wall!':CLS#2,4
1056        BEEP 1000,1,140,190,0,0,0:PAUSE 20
1057      ELSE
1058        IF fd=1 : px=px-1                 :REMark One Cell West
1059        IF fd=2 : px=px+1                 :REMark One Cell East
1060        IF fd=3 : py=py-1                 :REMark One cell North
1061        IF fd=4 : py=py+1                 :REMark One cell South
1062        BLOCK#3,4,3,2+cx*6,1+cy*5,0  :cx=px:cy=py
1063        BLOCK#3,4,3,2+cx*6,1+cy*5,bc      :REMark 2D Maze cell position
1064        BEEP 2000,20,40,190,0,0,0:gst=1
1065        ofd=fd:gdel=gdel-1:snum=snum-sl:sm=sm+1:MazView
1066        Loot=grid(px,py)    :SELect ON Loot=1,2,4,8:TresChk
1067      END IF
1068  END SELect
1069  END REPeat Maze_lp
1070 END DEFine

1072 DEFine PROCedure Score
1073 INK#4,6:clk$=DATE$(DATE-Gclk+GTS):CURSOR#4,6,212
1074 PRINT#4,'Time:';clk$(13 TO 20);'  Moves: ';FILL$(' ',4-LEN(sm))&sm
1075 PRINT#5,'SCORE  ';FILL$(' ',6-LEN(snum))&snum
1076 END DEFine
```

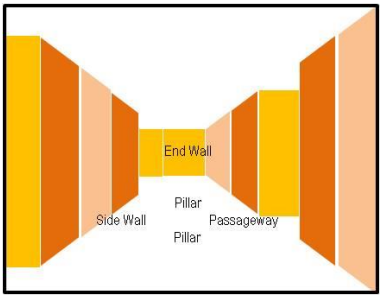
Note: Showing Max possible Moves & Score Points

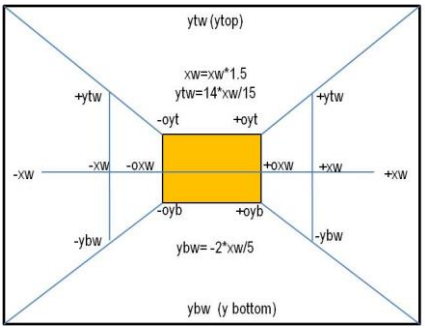
```
1078 DEFine PROCedure MazView                          Note: Calculates and Displays the
1079 fvn=0:INK#2,6:CURSOR#2,178,190:PRINT#2,dir$(fd):CLS#2,4   3D  View of Passageway.
1080 FOR n=1 TO 5
1081  fv(n)=0                                  :REMark fv forward view
 1082  IF fd=1
1083    cw=grid(px-n+1,py):IF Walls(8,cw)=1:fv(n)=1
1084    IF Walls(4,cw)=1:fv(n)=fv(n)+2         :REMark cw cell walls
1085    IF Walls(1,cw)=0:fvn=n:EXIT n          :REMark fvn forward view num cell
1086  END IF
1087  IF fd=2
1088    cw=grid(px+n-1,py):IF Walls(8,cw)=1:fv(n)=2
1089    IF Walls(4,cw)=1:fv(n)=fv(n)+1
1090    IF Walls(2,cw)=0:fvn=n:EXIT n
1091  END IF
1092  IF fd=3
1093    cw=grid(px,py-n+1):IF Walls(1,cw)=1:fv(n)=1
1094    IF Walls(2,cw)=1:fv(n)=fv(n)+2
1095    IF Walls(4,cw)=0:fvn=n:EXIT n
1096  END IF
1097  IF fd=4
1098    cw=grid(px,py+n-1):IF Walls(1,cw)=1:fv(n)=2
1099    IF Walls(2,cw)=1:fv(n)=fv(n)+1
1100    IF Walls(8,cw)=0:fvn=n:EXIT n
1101  END IF
1102 END FOR n
1103 vn=fvn:IF fvn=0 :fvn=6:vn=5              :REMark fvn vn forward view num cells
1104 xw=58*(2/3)^((vn-1)*2):ytw=14*xw/15:ybw=-2*xw/5
1105 BLOCK 240,82,0,0,7,0,1:BLOCK 240,38,0,82,0,5,3   :REMark Roof & Floor
1106 IF fvn=6
1107   INK 0,2,1
1108   FILL 1:LINE -xw,ytw TO xw,ytw TO xw,ybw TO -xw,ybw TO -xw,ytw:FILL 0
1109   GO TO 1113
1110 END IF
1111 INK 0,2,3
1112 FILL 1:LINE -xw,ytw TO xw,ytw TO xw,ybw TO -xw,ybw TO -xw,ytw:FILL 0
1113 REPeat sidewalls
1114  Wallcalc:cdv=fv(vn):INK 0,2,3
1115  IF cdv=2 OR cdv=3
1116     FILL 1:LINE oxw,oyt TO oxw,oyb TO xw,oyb TO xw,oyt TO oxw,oyt:FILL 0
1117  END IF
1118  IF cdv=1 OR cdv=3
1119    FILL 1:LINE -oxw,oyt TO -oxw,oyb TO -xw,oyb TO -xw,oyt TO -oxw,oyt:FILL 0
1120  END IF
1121  INK 0,2,2
1122  IF cdv=0 OR cdv=1
1123    FILL 1:LINE oxw,oyt TO oxw,oyb TO xw,ybw TO xw,ytw TO oxw,oyt:FILL 0
1124  END IF
1125  IF cdv=0 OR cdv=2
1126    FILL 1:LINE -oxw,oyt TO -oxw,oyb TO -xw,ybw TO -xw,ytw TO -oxw,oyt:FILL 0
1127  END IF
1128  Wallcalc
1129  FILL 1:LINE oxw,oyt TO oxw,oyb TO xw,ybw TO xw,ytw TO oxw,oyt:FILL 0
1130  FILL 1:LINE -oxw,oyt TO -oxw,oyb TO -xw,ybw TO -xw,ytw TO -oxw,oyt:FILL 0
1131  vn=vn-1:IF vn=0 : EXIT sidewalls
1132 END REPeat sidewalls
1133 END DEFine
```





Pillars

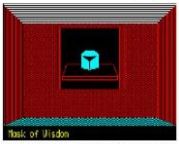




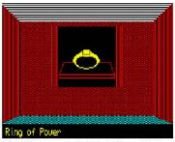
Note: The Maze Passageways consist of an End Wall, Gaps with Horizontal Walls for Side Entrances, or Blocked by a Perspective Side Wall in between Pillars.

```
1135 DEFine FuNction Walls(side,wall)
1136 ans=0
1137 IF side=1:SELect ON wall=1,3,5,7, 9,11,13,15  :ans=1
1138 IF side=2:SELect ON wall=2,3,6,7,10,11,14,15 :ans=1
1139 IF side=4:SELect ON wall=4,5,6,7,12,13,14,15 :ans=1
1140 IF side=8 AND wall>7 :ans=1
1141 RETurn ans                    :REMark identify fd (forward direction)
1142 END DEFine

1144 DEFine PROCedure Wallcalc
1145 oxw=xw:xw=xw*1.5:oyt=ytw:oyb=ybw:ytw=14*xw/15:ybw=-2*xw/5
1146 END DEFine

1148 DEFine PROCedure TresChk
1149 FOR i=1 TO 12
1150  IF Tres(i,1)=px AND Tres(i,2)=py
1151    tn=Tres(i,3):IF tn=0:EXIT i
1152    BLOCK 100,60,70,24,0,2,2:FOR j=1 TO 8:BLOCK j*10,60,120-j*5,24,0:PAUSE 5
1153    ch=1:INK 2:x=0:y=20:INK#2,6:CURSOR#2,236,190
1154    LINE x-24,y-10 TO x-20,y TO x+20,y TO x+24,y-10 TO x-24,y-10
1155    LINE x-24,y-10 TO x-24,y-12 TO x+24,y-12 TO x+24,y-10
1156    IF tn>1 AND tn<7
1157      Coin 1,-4,26:PRINT#2,'Coin of Karnak':snum=snum+50*tn
1158    END IF
1159    IF tn=7
1160      Mask 4,220,20:Mask 1,0,20:km=1
1161      PRINT#2,'Mask of Wisdom':snum=snum+50+50*lev
1162    END IF
1163    IF tn=8
1164      Ring 4,250,22:Ring 1,0,22:kr=1
1165      PRINT#2,'Ring of Power ':snum=snum+500+500*lev
1166    END IF
1167    IF tn=9:KStone 1,0,20:PortChk:EXIT i
1168    Tres(i,3)=0
1169  END IF
1170 END FOR i
1171 END DEFine
```

Coins of Karnak: Gain 50+50xRND(2 to 6) Points



Mask of Wisdom - Gain 50+50xLevel Points

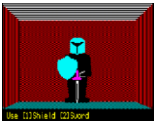


Ring of Power - Gain 500+500xLevel Points

Key Stone - Gain 500+500xLevel Points

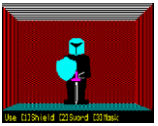
```
1173 DEFine PROCedure PortChk
1174 CURSOR#2,236,190:PRINT#2,'Key Stone - Portal to Next Level (Y/N)'
1175 IF INKEY$(#2,-1)=='Y'
1176   IF lev=5
1177     IF gmax>0:CURSOR#2,236,190:PRINT#2,'Defeat All Guardians':CLS#2,4:RETurn
1178     CURSOR#2,236,190:CLS#2,4:CLS:KeyStone
1179   ELSE
1180     lev=lev+1:glev=lev+1:IF glev>gmax OR lev=5:glev=gmax
1181     CLS:MPort:PAUSE 20:col=5:MazLev:MazNew:MazHall:MazTres:MazView
1182     snum=snum+2000:tc=0:bc=0:km=0:kr=0:gdel=120/RND(2 TO 4)
1183     px=RND(2 TO 19):py=RND(2 TO 15):CURSOR#2,220,190:CLS#2,4
1184   END IF
1185 ELSE
1186   tc=1:BLOCK#2,240,10,230,190,0
1187 END IF
1188 END DEFine

1190 DEFine PROCedure MGuard
1191 IF glev=0:RETurn :ELSE Guard 1:INK 7:MPort
1192 PAUSE 20:MazView:Guard 1:INK#2,6:gdel=120/RND(2 TO 4) .
1193 REPeat G_lp
1194   IF snum<  50:EXIT G_lp
1195   CURSOR#2,236,190:PRINT#2,'Use [1]Shield [2]Sword':CLS#2,4
1196   IF km=1 AND snum>50+  50*lev:CURSOR#2,374,190:PRINT#2,'[3]Mask'
1197   IF kr  =1 AND snum>500+500*lev:CURSOR#2,422,190:PRINT#2,'[4]Ring'
1198   k=CODE(INKEY$(-1))
1199   IF k=49:snum=snum-50:px=RND(3 TO 17):py=RND(3 TO 14):CLS:EXIT G_lp
1200   IF k=50
1201     INK#4,5:FOR i=1 TO 6   :CURSOR#4,197,170:PRINT#4,i:PAUSE 20
1202     a=RND(1 TO 6):INK#4,7:CURSOR#4,197,170:PRINT#4,a:PAUSE 20
1203     IF a=6 :INK#2,4:gmax=gmax-1:glev=glev-1:EXIT G_lp
1204     IF a<>6:INK#2,2:snum=snum-50:Score
1205   INK#2,6:CURSOR#2,236,190:PRINT#2,'Try Again':CLS#2,4:PAUSE 30
1206   END IF
1207   IF k=51 AND snum>50+50*lev:snum=snum-50-50*lev:gdel=120:EXIT G_lp
1208   IF k=52 AND snum>500+500*lev
1209     snum=snum-500-500*lev:gmax=gmax-glev:glev=0:EXIT G_lp
1210   END IF
1211 END REPeat G_lp
1212 GView:Score:CURSOR#2,236,190:CLS#2,4:INK 7:MPort:PAUSE 20:MazView
1213 INK#2,5:CURSOR#2,148,30+10*lev:PRINT#2,glev;'  ';gmax;'  '
1214 END DEFine
```

Activated after Random Number of Moves
[120 / RND (2 to 4) {i.e. 30 40 60 moves}]



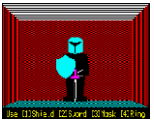
[1] Coin Portal to another Grid Cell

[2] Sword Challenge Roll Dice



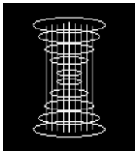
[3] Mask No Guardians 120 Moves

[4] Ring Delete all Guardian on level



```
1216 REMark Vector Graphics

1218 DEFine PROCedure Guard(ch)
1219 IF ch=1:x=   0:y=  - 4 :INK#ch,0:FILL#ch,1
1220 IF ch=4:x=140:y=104 :INK#ch,7:FILL#ch,0
1221 LINE#ch,x-5,y+32 TO x-5,y+26 TO x-12,y+25 TO x-14,y+6 TO x-10,y+4
1222 LINE#ch TO x-8,y-16 TO x-12,y-17 TO x-12,y-20 TO x+12,y-20 TO x+12,y-17
1223 LINE#ch TO x+8,y-16 TO x+10,y+4 TO x+14,y+6 TO x+12,y+25 TO x+5,y+26
1224 LINE#ch TO x+5,y+32 TO x-5,y+32:FILL#ch,0
1225 INK#ch,7:LINE#ch,x-8,y+4 TO x-2,y+4 TO x-2,y+8 TO x-8,y+9
1226 IF ch=1:Shield 1, 10, 20:Sword 1, 0,-24:Mask 1,  0, 26
1227 IF ch=4:Shield 4,150,128:Sword 4,140,82:Mask 4,140,134
1228 END DEFine
```

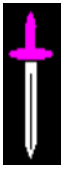
```
1230 DEFine PROCedure MPort
1231 BEEP 2000,20,40,190,0,0,0:ch=1:x=0:y=20:INK#ch,7
1232 FOR i=0 TO 16 STEP 4
1233   CIRCLE#ch,0,-20+i*2,25-i,.2,PI/2:CIRCLE#ch,0,50-i*2,25-i,.2,PI/2
1234   LINE#ch,-i,-22+i/8 TO -i,52-i/8:LINE#ch,+i,-22+i/8 TO i,52-i/8
1235 END FOR i
1236 END DEFine
```



```
1238 DEFine PROCedure Coin(ch,x,y)
1239 INK#ch,6:FILL#ch,1:CIRCLE#ch,x,y,10,.6,PI:FILL#ch,0
1240 INK#ch,0:CIRCLE#ch,x+3,y-1,10,.7,PI
1241 INK#ch,6:FILL#ch,1:CIRCLE#ch,x+10,y-4,10,.6,PI/4:FILL#ch,0
1242 INK#ch,0:CIRCLE#ch,x+10,y-4,10,.6,PI/4
1243 INK#ch,0:CIRCLE#ch,x+12,y-4,9,.5,PI/4
1244 END DEFine
```

```
1246 DEFine PROCedure Shield(ch,x,y)
1247 FILL#ch,1:INK#ch,5:ARC#ch,x,y TO x-9,y-4, -PI/4
1248 ARC#ch,x-9,y-4 TO x,y-22, PI/2:ARC#ch,x,y-22 TO x+9,y-4, PI/2
1249 ARC#ch,x+9,y-4 TO x,y, -PI/4:FILL#ch,0
1250 END DEFine
```

```
1252 DEFine PROCedure Sword(ch,x,y)
1253 FILL#ch,1:INK#ch,7
1254 LINE#ch,x,y TO x-1,y+3 TO x-1,y+20 TO x+1,y+20 TO x+1,y+3 TO x,y
1255 FILL#ch,0:INK#ch,0:LINE#ch,x,y+2 TO x,y+18:INK#ch,3
1256 FILL#ch,1:CIRCLE#ch,x,y+22,5,.2,PI/2:FILL#ch,0
1257 FILL#ch,1:CIRCLE#ch,x,y+26,5,.2,PI:FILL#ch,0:CIRCLE#ch,x,y+28,1
1258 END DEFine
```

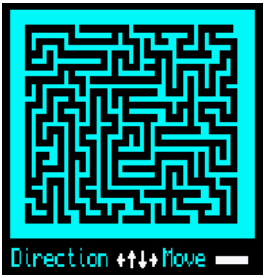


```
1260 DEFine PROCedure Mask(ch,x,y)
1261 INK#ch,5:FILL#ch,1:ARC#ch,x+7,y+9 TO x-7,y+9,PI/2
1262 LINE#ch,x-7,y+9 TO x-7,y-2 TO x,y-4 TO x+7,y-2 TO x+7,y+9:FILL#ch,0
1263 INK#ch,0:FILL#ch,1:LINE#ch,x+6,y+7 TO x,y+6 TO x,y+3 TO x+6,y+7:FILL#ch,0
1264 FILL#ch,1:LINE#ch,x-6,y+7 TO x,y+6 TO x,y+3 TO x-6,y+7:FILL#ch,0
1265 LINE#ch,x,y+4 TO x,y-4
1266 END DEFine
```

```
1268 DEFine PROCedure Ring(ch,x,y)
1269 INK#ch,6:FILL#ch,1:CIRCLE#ch,x,y,11,.6,PI/2 :FILL#ch,0
1270 INK#ch,0:FILL#ch,1:CIRCLE#ch,x,y-1,9,.5,PI/2:FILL#ch,0
1271 INK#ch,6:FILL#ch,1:CIRCLE#ch,x,y+6,5,.5,PI/2:FILL#ch,0
1272 INK#ch,0:LINE#ch,x-3,y+9 TO x+3,y+9 TO x+3,y+5 TO x-3,y+5 TO x-3,y+9
1273 END DEFine
```

```
1275 DEFine PROCedure KStone(ch,x,y)
1276 BEEP 2000,20,40,190,0,0,0:INK#ch,6:FILL#ch,1
1277 LINE#ch,x,y+6 TO x-6,y TO x,y-6 TO x+6,y TO x,y+6:FILL#ch,0
1278 INK#ch,0:LINE#ch,x,y+8 TO x-6,y TO x,y-8 TO x+6,y TO x,y+8
1279 LINE#ch,x,y+8 TO x-2,y-2 TO x,y-8
1280 LINE#ch,x-6,y TO x-2,y-2 TO x+6,y
1281 END DEFine
```

```
1283 REMark New Game / Level Change

1285 DEFine PROCedure MazNew
1286 w=20:h=16:DIM grid(w+1,h+1),cell(w*h,2),pm(5),fv(5)
1287 CLS#3:INK 7:x=w/2:y=h:cell(0,1)=x:cell(0,2)=y:inc=40:cn=1
1288 INK#2,5:CURSOR#2,18,190:PRINT#2,'Direction      Move':INK#2,7
1289 CURSOR#2,76,190:PRINT#2,'←↑↓→':BLOCK#2,18,3,136,194,7
1290 FOR n=1 TO w*h-1
1291   p=0:PAUSE mp                          :REMark p - Pass / mp - 0.5 pause delay
1292   IF x>1 AND grid(x-1,y)=0 : p=p+1:pm(p)=1   :REMark West wall
1293   IF x<w AND grid(x+1,y)=0 : p=p+1:pm(p)=2   :REMark East wall
1294   IF y>1 AND grid(x,y-1)=0 : p=p+1:pm(p)=3   :REMark North wall
1295   IF y<h AND grid(x,y+1)=0 : p=p+1:pm(p)=4   :REMark South wall
1296   IF p=0                                 Note: The Maze algorithms
1297     IF m=1:cn=cn-1:x=cell(cn,1):y=cell(cn,2)        m=1  Recursive Backtracking
1298     IF m=2:cn=0:x=RND(w):y=RND(h):cell(cn,1)=x:cell(cn,2)=y   m=2  Prims Algorithm
1299     IF m=3:x=x+1:IF x>w : x=1 :y=y+1:IF y>h : y=1   m=3  Hunt and Kill Method
1300     IF grid(x,y)=0 : GO TO 1297
1301     GO TO 1292
1302   END IF
1303   r=pm(RND(1 TO p)):cn=cn+1:cell(cn,1)=x:cell(cn,2)=y
1304   IF r=1 : grid(x,y)=grid(x,y)+1:x=x-1:grid(x,y)=2:bx=x*6:by=y*5
1305   IF r=2 : grid(x,y)=grid(x,y)+2:bx=x*6:x=x+1:grid(x,y)=1:by=y*5
1306   IF r=3 : grid(x,y)=grid(x,y)+4:y=y-1:grid(x,y)=8:bx=x*6:by=y*5
1307   IF r=4 : grid(x,y)=grid(x,y)+8:by=y*5:y=y+1:grid(x,y)=4:bx=x*6
1308   IF r=1 OR r=2 :BLOCK#3,8,3,2+bx,1+by,0
1309   IF r=3 OR r=4 :BLOCK#3,4,8,2+bx,1+by,0
1310 END FOR n
1311 END DEFine
```



Note: For faster QL Hardwar/Software Platforms... **mp** set to 0.5 so Maze construction is slowed.

```
1313 DEFine PROCedure MazHall
1314 REMark grid r row/c col:hw hall width:cw cell wall:sf side facing
1315 FOR hall=1 TO 6
1316   tx=RND(4 TO 16):ty=RND(4 TO 12):RESTORE 1328
1317   FOR r=0 TO 1
1318     FOR c=0 TO 2
1319       BLOCK#3,16,8,2+tx*6,1+ty*5,0
1320       BLOCK#3,2,2,6+tx*6,4+ty*5,5 :BLOCK#3,2,2,12+tx*6,4+ty*5,5
1321       FOR hw=1 TO 3
1322         cw=grid(tx+c,ty+r):READ sf
1323         IF Walls(sf,cw)=0:grid(tx+c,ty+r)=grid(tx+c,ty+r)+sf
1324       END FOR hw
1325     END FOR c
1326   END FOR r
1327 END FOR hall
1328 DATA 2,8,8,1,2,8,1,8,8,2,4,4,1,2,4,1,4,4
1329 END DEFine
```

Note: Making alternative routes available by adding further inter-connections.

```
1331 DEFine PROCedure MazLev
1332 BLOCK#2,50,58,132,40,0:BLOCK#4,200,40,300,212,0
1333 FOR i=1 TO 5
1334   IF i=lev:INK#2,5:ELSE INK#2,3
1335   CURSOR#2,16,30+i*11:PRINT#2,i
1336   LINE#2,26,84-i*5 TO 46,84-i*5 TO 36,80-i*5 TO 16,80-i*5 TO 26,84-i*5
1337   IF i<lev:KStone 4,280+i*12,20
1338 END FOR i
1339 IF lev=1:px=10:py=16:ELSE px=RND(3 TO 18):py=RND(2 TO 14)
1340 INK#2,col:CURSOR#2,148,30+10*lev:PRINT#2,glev;' ';gmax
1341 END DEFine
```


```
1343 DEFine PROCedure MazTres
1344 DIM Tres(12,3):n=1
1345 REPeat t_lp
1346   IF n>12:n=1:EXIT t_lp
1347   tx=RND(1 TO w):ty=RND(1 TO h):tn=grid(tx,ty)
1348   FOR i=1 TO n:IF Tres(i,1)=tx AND Tres(i,2)=ty:NEXT t_lp
1349   SELect ON tn=1,2,4,8:Tres(n,1)=tx:Tres(n,2)=ty:n=n+1
1350 END REPeat t_lp
1351 FOR i=1 TO 12:Tres(i,3)=RND(2 TO 6)                    Coins of Karnak(2 to 6)
1352 Tres(3,3)=7:km=0:Tres(11,3)=8:kr=0:Tres(7,3)=9         Mask(7) / Ring(8) / KeyStone
1353 END DEFine
```

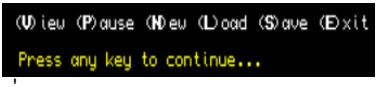
```
1355 DEFine PROCedure MazKey
1356 DIM Mkey(5):RESTORE 1360:ra=RND(24)
1357 FOR i=1 TO 24
1358   READ a,b,c,d:IF i=ra:SKey(1,3)=a:SKey(2,3)=b:SKey(3,3)=c:SKey(4,3)=d
1359 END FOR i
1360 DATA 1,2,3,4, 1,3,2,4, 2,3,1,4, 2,1,3,4, 3,1,2,4, 3,2,1,4
1361 DATA 2,3,4,1, 3,2,4,1, 3,1,4,2, 1,3,4,2, 1,2,4,3, 2,1,4,3
1362 DATA 3,4,1,2, 2,4,1,3, 1,4,2,3, 3,4,2,1, 2,4,3,1, 1,4,3,2
1363 DATA 4,1,2,3, 4,1,3,2, 4,2,3,1, 4,2,1,3, 4,3,1,2, 4,3,2,1
1364 END DEFine
```

Note: Entering the **Sphere of Destiny** the collected **Key Stones** have to be arranged in the same order as those presented in the Sphere. This requires the correct **matching** of both sets of Key Stones. Taking the fifth Stone as already set the other four stones can create 24 different combinations.

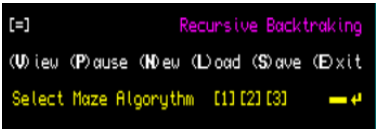
```
1366 REMark Menu Commands

1368 DEFine PROCedure Mes1
1369 BLOCK#2,200,26,240,40,0
1370 INK#2,6:CURSOR#2,240,48:PRINT#2,'Select (N)ew or (L)oad':gck=0
1371 END DEFine

1373 DEFine PROCedure MSel              :REMark Maze Algorithm
1374 IF m=1:Maz$='Recursive Backtraking'
1375 IF m=2:Maz$='Prims Algorithm'
1376 IF m=3:Maz$='Hunt and Kill Method'
1377 INK#2,3:CURSOR#2,332,12:PRINT#2,FILL$(' ',25-LEN(Maz$))&Maz$
1378 END DEFine
```

```
1380 DEFine PROCedure GView
1381 IF gck=0:RETurn
1382 IF tc=0:tc=3:bc=7:sl=lev*5:ELSE tc=0:bc=0:sl=lev
1383 IF TM=1:BLOCK#3,4,3,2+Tres(7,1)*6,1+Tres(7,2)*5,241:PAUSE 20
1384 BLOCK#3,4,3,2+cx*6,1+cy*5,0:cx=px:cy=py:BLOCK#3,4,3,2+cx*6,1+cy*5,bc
1385 FOR n=1 TO 12:IF Tres(n,3)>0:BLOCK#3,4,3,2+Tres(n,1)*6,1+Tres(n,2)*5,tc
1386 END DEFine
```

Note: (V) Toggles **ON**/**OFF** Highlighted Cells seen on the 2D Maze Layout




```
1388 DEFine PROCedure GPause
1389 IF gck=0:RETurn
1390 INK#2,6:CURSOR#2,240,48:PRINT#2,'Press any key to continue...'
1391 GTS=(DATE-Gclk+GTS):PAUSE:BLOCK#2,250,10,240,48,0
1392 END DEFine
```



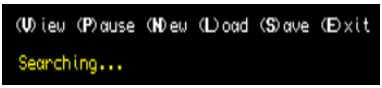
Spacebar ▬ ↵ Enter (Left CURSOR+BLOCK Tail)

```
1394 DEFine PROCedure GNew
1395 GTS=(DATE-Gclk+GTS):INK#2,6:CURSOR#2,200,48
1396 PRINT#2,'Select Maze Algorythm [1][2][3] ←'
1397 BLOCK#2,12,3,460,52,6:BLOCK#2,2,4,480,50,6:INK#2,3:om=m
1398 REPeat New_lp
1399  MSel:k=CODE(INKEY$(-1))
1400  SELect ON k
1401    =49,50,51:m=k-48
1402    =32:BLOCK#2,250,10,240,48,0:m=om:MSel:RETurn
1403    =10:BLOCK#2,250,10,240,48,0:CLS  :EXIT New_lp
1404  END SELect
1405 END REPeat New_lp
1406 gdel=120/RND(3 TO 4):gmax=16:glev=2:lev=1:col=5:MazLev:MazKey
1407 GTS=0:Gclk=DATE:sm=0:snum=2000:Score:gck=1:gst=1
1408 w=20:h=16:mp=.5:MazNew:MazHall:MazTres:MazView:mp=0
1409 END DEFine
```

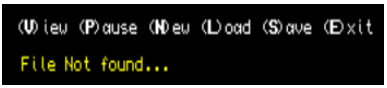

```
1411 DEFine PROCedure SelPath
1412 REMark TS=(DATE-Gclk+GTS):INK#2,6
1413 INK#2,6:CURSOR#2,240,48:PRINT#2,'Select: ↑↓    ';SD$;'  ← →  ←'
1414 BLOCK#2,12,3,412,52,6:BLOCK#2,2,4,432,50,6
1415 REPeat Path_lp
1416  CURSOR#2,300,48:PRINT#2,drv$(dn%):CURSOR#2,384,48:PRINT#2,fnum
1417  k=CODE(INKEY$(-1))
1418  SELect ON k
1419   =192:fnum=fnum -1:IF fnum<0:fnum=9
1420   =200:fnum=fnum+1:IF fnum>9:fnum=0
1421   =208:dn% =dn% -1:IF dn%<0:dn%=dm%
1422   =216:dn%=dn%+1:IF dn%>dm%:dn%=0
1423   = 10:pck=1:EXIT Path_lp
1424   = 32:pck=0:EXIT Path_lp
1425  END SELect
1426 END REPeat Path_lp
1427 END DEFine
```

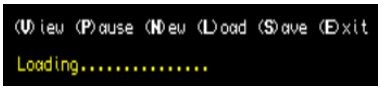
Spacebar ▬ ↵ Enter symbol

```
1429 DEFine PROCedure FCheck
1430 CURSOR#2,240,48:PRINT#2,'Searching...':CLS#2,4
1431 PAUSE 20:DELETE drv$(dn%)&'FList'
1432 OPEN_NEW#99,drv$(dn%)&'FList':DIR#99,drv$(dn%):CLOSE#99
1433 OPEN_IN#99,drv$(dn%)&'FList'
1434 REPeat Dir_lp
1435   IF EOF(#99):CLOSE#99:BLOCK#2,250,10,240,48,0:pck=0:EXIT Dir_lp
1436   INPUT#99,Fchk$:IF Fchk$==SD$&fnum:CLOSE#99:pck=1:EXIT Dir_lp
1437 END REPeat Dir_lp
1438 END DEFine
```

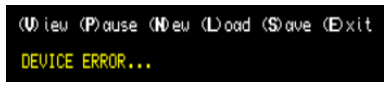


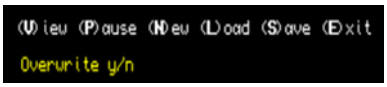

```
1440 DEFine PROCedure MLoad
1441 SelPath:IF pck=0:BLOCK#2,250,10,240,48,0:RETurn :ELSE FCheck
1442 IF pck=0 OR eck=1
1443    CURSOR#2,240,48:PRINT#2,'File Not found...':CLS#2,4
1444    PAUSE 20:BLOCK#2,250,10,240,48,0:eck=0:RETurn
1445 END IF
1446 OPEN_IN#99,drv$(dn%)&SD$&fnum:CURSOR#2,240,48:PRINT#2,'Loading...';
1447 FOR n=1 TO 12:INPUT#99,Tres(n,3):PRINT#2,'.';:PAUSE 2
1448 INPUT#99,m\lev\gmax\glev\km\kr\GTS\sm\snum:CLOSE#99
1449 CLS:MazNew:MazHall:MazTres:MazKey
1450 MSel:col=5:MazLev:Score:MazView
1451 IF km=1:Mask 4,220,20
1452 IF kr=1:Ring 4,250,22
1453 BLOCK#2,250,10,240,48,0:gdel=120/RND(2 TO 4):gst=1:gck=1
1454 END DEFine
```



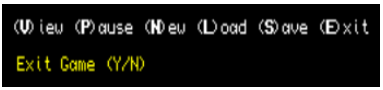



```
1456 DEFine PROCedure MSave
1457 SelPath:IF pck=0 OR gck=0:BLOCK#2,250,10,240,48,0:RETurn :ELSE FCheck
1458 IF eck=1
1459    CURSOR#2,240,48:PRINT#2,'DEVICE ERROR...':CLS#2,4
1460    PAUSE 20:BLOCK#2,250,10,240,48,0:eck=0:RETurn
1461 END IF
1462 IF pck=1
1463    CURSOR#2,240,48:PRINT#2,'Overwrite y/n':PAUSE
1464    IF KEYROW(5)<>64:BLOCK#2,250,10,240,48,0:RETurn
1465 END IF
1466 DELETE drv$(dn%)&SD$&fnum:OPEN_NEW#99,drv$(dn%)&SD$&fnum
1467 CURSOR#2,240,48:PRINT#2,'Saving...';
1468 FOR n=1 TO 12:PRINT#99,Tres(n,3):PRINT#2,'.';:PAUSE 2
1469 PRINT#99,m\lev\gmax\glev\km\kr\GTS\sm\snum:CLOSE#99
1470 BLOCK#2,250,10,240,48,0
1471 END DEFine
```



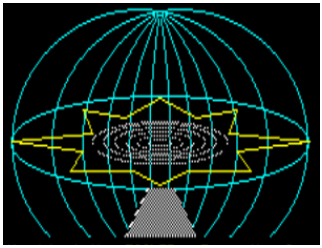


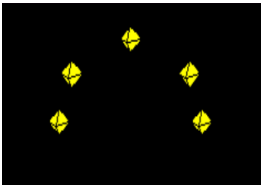

```
1473 DEFine PROCedure GExit
1474 INK#2,6:CURSOR#2,220,48:PRINT#2,'Exit Game (Y/N)':CLS#2,4:PAUSE
1475 IF KEYROW(5)=64:LRUN dn$:ELSE CURSOR#2,200,48:CLS#2,4
1476 END DEFine
```

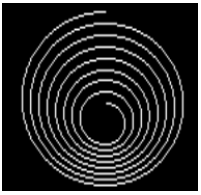


```
1478 REMark Sphere of Destiny - Matching Keystones

1480 DEFine PROCedure KeyStone
1481 KStone 4,340,20:scol=6:SDest:SRing
1482 CURSOR#2,258,44:PRINT#2,'Activate the Sphere of Destiny'
1483 CURSOR#2,240,54:PRINT#2,'by Matching the Sphere and Maze Keys'
1484 INK#2,5:CURSOR#2,250,190:PRINT#2,'Use          to Match and Test Keys'
1485 INK#2,7:CURSOR#2,274,190:PRINT#2,'←↑↓→ ←':BLOCK#2,2,4,310,192,7
1486 check=0:col=0:FOR kp=1 TO 4:ks=kp:GetKey
1487 REPeat key_lp
1488   IF snum<500:snum=0:Score:Mes2:EXIT key_lp
1489   GetKey:k=CODE(INKEY$(-1))
1490   SELect ON k
1491     =192:kp=kp-1:IF kp<1:kp=4
1492     =200:kp=kp+1:IF kp>4:kp=1
1493     =208:ks=ks+1:IF ks>4:ks=1
1494     =216:ks=ks-1:IF ks<1:ks=4
1495     = 10:MatchKey:IF schk<5:snum=snum-500:Score:ELSE Mes3:EXIT key_lp
1496     =248:IF col=0:col=5:ELSE col=0
1497   END SELect
1498 END REPeat key_lp
1499 gst=0:BLOCK#2,250,30,240,40,0:Mes1:QBITS_Maze
1500 END DEFine

1502 DEFine PROCedure MatchKey
1503 schk=1:FOR i=1 TO 4:IF SKey(i,3)=Mkey(i):schk=schk+1
1504 END DEFine

1506 DEFine PROCedure GetKey
1507 IF kp=1:Mkey(1)=ks:c=414
1508 IF kp=2:Mkey(2)=ks:c=430
1509 IF kp=3:Mkey(3)=ks:c=466
1510 IF kp=4:Mkey(4)=ks:c=484
1511 RESTORE 1514:INK#4,col:FOR i=1 TO 4:READ a:CURSOR#4,a,212:PRINT#4,SKey(i,3)
1512 RESTORE 1514:INK#4,5  :FOR i=1 TO 4:READ a:CURSOR#4,a,240:PRINT#4,Mkey(i)
1513 INK#4,6:CURSOR#4,c,240:PRINT#4,Mkey(kp)
1514 DATA 414,430,466,484
1515 END DEFine

1517 DEFine PROCedure Mes2
1518 IF gst=0:RETurn
1519 INK#2,6:CURSOR#2,240,190:PRINT#2,'Hard Luck You FAILED - Try a New Game '
1520 CLS:CLS#3:lev=1:glev=2:MazLev:SEnd:col=0:fil=1:Guard(1):Mes1
1521 END DEFine

1523 DEFine PROCedure Mes3
1524 INK#2,6:CURSOR#2,240,190:PRINT#2,'The Past has Changed - Humanity Saved '
1525 CLS:CLS#3:lev=1:glev=2:MazLev:SEnd:BLOCK#2,250,30,240,40,0:LName:Mes1
1526 END DEFine
```

```
1528 DEFine PROCedure SDest
1529 col=5:ss=8:x=0:y=10:INK 7:FILL 0
1530 REPeat sphere_lp
1531  FOR i=0 TO 1.1  STEP .1
1532    ARC x,y+ss TO x,y-ss,PI*i:ARC x,y+ss TO x,y-ss,-PI*i
1533    INK col:IF col=5:col=0:ELSE col=5
1534  END FOR i
1535  BEEP 2000,8,20,-8,0,0,0:ss=ss+8:IF ss>56:EXIT sphere_lp
1536  PAUSE 5:INK 0:FILL 1:CIRCLE x,y,36+ss,ss*2/100,PI:FILL 0
1537 END REPeat sphere_lp
1538 INK 5:CIRCLE x,y,66,.3,PI/2:y=20:INK 6::FILL 0
1539 LINE x,y+9 TO x-16,y TO x-34,y+4 TO x-30,y-6 TO x-66,y-10
1540 LINE TO x-34,y-14 TO x-40,y-22 TO x-12,y-22 TO x,y-28 TO x+12,y-22
1541 LINE TO x+40,y-22 TO x+34,y-14 TO x+66,y-10 TO x+30,y-6 TO x+34,y+4
1542 LINE TO x+16,y TO x,y+9:FILL 0:INK 248
1543 FILL 1:LINE x-6,y-30 TO x-16,y-50 TO x+16,y-50 TO x+6,y-30
1544 LINE TO x-6,y-30:FILL 0
1545 FOR i=1 TO 6:CIRCLE x,y-10,i*5,.3,PI/2:PAUSE 2
1546 END DEFine

1548 DEFine PROCedure SRing
1549 ch=1:scol=6:FOR i=1 TO 5:KStone 1,SKey(i,1),SKey(i,2):PAUSE 5
1550 END DEFine

1552 DEFine PROCedure SEnd
1553 FOR i=1 TO 24 STEP 2
1554    INK 241:CIRCLE 0,14,i*3,.7,PI/2:BEEP 2000,40,120,90,0,0,0:PAUSE 2
1555 END FOR i
1556 INK 0:FILL 1:CIRCLE,0,14,60:FILL 0:BEEP 30000,1,250,90,-8,15,15:INK 7
1557 FOR i=50 TO 15 STEP -5
1558    ARC 0,i TO 0,-i/2,PI:ARC 0,-i/2 TO 0,i-5,PI:PAUSE i/5
1559 END FOR i
1560 BEEP 10000,4,200,190,0,0,0: PAUSE 20
1561 END DEFine
```





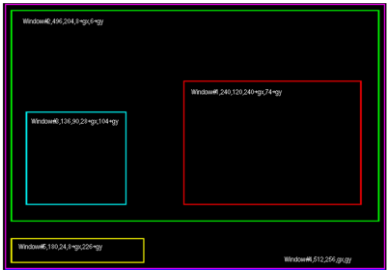


Note: QBITS Solving Code Problems

While entering code, Keyword and Attribute problems are identified by the Line Editor of the SBASIC Interpreter. On starting the program, the parsing of the Interpreter will name any other errors, typos of repeated or missing variables, arithmetic overflows, division by zero etc. In these cases, there is generally a helpful comment such as **At line 1234:1** accompanied with an Error Statement Describing the Problem.

When Program code is not performing as expected, as a first step try Isolating the problem to a PROCedure or FuNction. This can be achieved sometimes by setting associated variables and actioning the Procedure or Function independent from running the whole program. Tracking down a code error it is useful to control **LINES** of code by adding a **PAUSE** to further narrow down the problem. Finally, you may need to add a number of **PRINT** statements that show the step-by-step changes taking place in order to identify what is causing the unexpected result.

```
1563 REMark QBITS Maze Screen Setup
```



```
1565 DEFine PROCedure Init_Win
1566 OPEN#5,scr_:WINDOW#5,180,24,8+gx,226+gy :PAPER#5,0
1567 OPEN#4,scr_:WINDOW#4,512,256,gx,gy      :PAPER#4,0
1568 OPEN#3,scr_:WINDOW#3,136,90,28+gx,106+gy:PAPER#3,5
1569 CSIZE#5,2,1:INK#5,6:BORDER#4,1,3:CLS#4:SCALE#4,240,0,0
1570 WINDOW#2,496,204,8+gx,6+gy      :PAPER#2,0:BORDER#2,1,5
1571 WINDOW#1,240,120,240+gx,74+gy   :PAPER#1,0:SCALE#1,100,-74,-30
1572 WINDOW#0,496,32,gx+8,gy+220     :PAPER#0,0:CSIZE#0,0,0:INK#0,7
1573 END DEFine

1575 DEFine PROCedure Init_Maze
1576 DIM S$(3,70),M$(7,40):CLS#2
1577 S$(1)='The Human Race is under Threat of Extinction from a Rogue DNA Gene'
1578 S$(2)='propagated by an Event in Prehistory.  Beneath the Tombs of Karnak'
1579 S$(3)='     lies the Sphere of Destiny and a Time Portal to the Past.'
1580 M$(1)='Your Mission to go back in Time and'
1581 M$(2)=' prevent the Event from happening.'
1582 M$(3)=' To Activate the Time Portal there'
1583 M$(4)=' are five  Key Stones  each hidden'
1584 M$(5)=' on a different Level of the Tombs'
1585 M$(6)='   however they are protected by'
1586 M$(7)='    Guardians (Phantom Knights)'
1587 CSIZE#2,2,1:OVER#2,1
1588 INK#2,2:FOR i=1 TO 2:CURSOR#2,7+i,3:PRINT#2,'QBITS Karnak Maze'
1589 INK#2,6:FOR i=1 TO 2:CURSOR#2,9+i,4:PRINT#2,'QBITS Karnak Maze'
1590 INK#2,6:FOR i=1 TO 2:CURSOR#2,8+i,70:PRINT#2,'Sphere of Destiny'
1591 CSIZE#2,0,0:OVER#2,0:INK#2,7
1592 INK#2,7:FOR i=1 TO 3:CURSOR#2,44,24+i*10:PRINT#2,S$(i)
1593 INK#2,5:FOR i=1 TO 7:CURSOR#2,12,86+i*10:PRINT#2,M$(i)
1594 Coin 4,140,22:Shield 4,180,32:Sword 4,200,6:Mask 4,220,22:Ring 4,250,22
1595 Score:FOR i=1 TO 4:KStone 4,280+i*12,20
1596 SDest:SRing:INK#2,3
1597 CURSOR#2,24,180:PRINT#2,'Press any key to continue...':PAUSE
1598 CLS:BLOCK#2,490,40,2,28,0:BLOCK#2,220,136,2,60,0
1599 OVER#2,1:col=0:lev=0:MazLev:INK#2,7:Guard(4)
1600 CURSOR#2,236,12:PRINT#2,'[=]'
1601 CURSOR#2,236,30:PRINT#2,'(V)iew (P)ause (N)ew (L)oad (S)ave (E)xit '
1602 CURSOR#2,237,30:PRINT#2,' V     P      N    L     S     E '
1603 INK#2,5:FOR i=0 TO 1:CURSOR#2,6+i,30:PRINT#2,'LEVEL'
1604 OVER#2,0:CURSOR#2,140,30:PRINT#2,'Guardians':gst=0:gck=0:gdel=120:pck=0
1605 INK#4,7:CIRCLE#4,140,74,9,.6,PI/2
1606 INK#2,3:FILL#2,1:LINE#2,65,8 TO 71,12 TO 76,8 TO 65,8:FILL#2,0
1607 END DEFine
```

```
1609 REMark League Table
```


```
1611 DEFine PROCedure LScore
1612 FOR i=1 TO 7
1613   PAUSE 5:BLOCK 200,12*i,20,66-i*6,0
1614   BLOCK 200,2,20,66-i*6,2:BLOCK 200,2,20,66+i*6,2
1615 END FOR i
1616 INK 6:OVER 1:CSIZE 2,1:FOR i=0 TO 1:CURSOR 44+i,28:PRINT 'League Table'
1617 INK 5:OVER 0:CSIZE 0,0:CURSOR 24,50:PRINT 'Points/Moves    Time      Gamer'
1618 FOR a=1 TO 3
1619  CURSOR  30,52+a*12:PRINT FILL$(' ',5-LEN(Grad(a,1)))&Grad(a,1)
1620  CURSOR  66,52+a*12:PRINT FILL$(' ',4-LEN(Grad(a,2)))&Grad(a,2)
1621  CURSOR 154,52+a*12:PRINT name$(a) :HST$=DATE$(Grad(a,3))
1622  CURSOR  98,52+a*12:PRINT HST$(13 TO 20)
1623 END FOR a
1624 END DEFine
```


```
1626 DEFine PROCedure LName
1627 GTS=DATE-Gclk+GTS:Gclk$=DATE$(GTS)
1628 FOR i=1 TO 3: IF Grad(i,1)<snum:Gmr=i:EXIT i:ELSE Gmr=0
1629 IF Gmr=0:LScore:RETurn
1630 IF Gmr>0
1631   Grad(Gmr,1)=snum:Grad(Gmr,2)=sm:Grad(Gmr,3)=GTS:LScore
1632   ch=6:OPEN#ch,con_:WINDOW#ch,60,10,394+gx,126+gy+Gmr*12
1633   PAPER#ch,0:CLS#ch:INK#ch,6:INPUT#ch,name$(Gmr):CLOSE#ch:LSave
1634 END IF
1635 END DEFine

1637 DEFine PROCedure LSave
1638 DELETE dev$&'QBMazeLT':OPEN_NEW#99,dev$&'QBMazeLT'
1639 FOR a=1 TO 3:PRINT#99,name$(a)\Grad(a,1)\Grad(a,2)\Grad(a,3)
1640 CLOSE#99
1641 END DEFine

1643 DEFine PROCedure LLoad
1644 OPEN_IN#99,dev$&'QBMazeLT'
1645 FOR a=1 TO 3:INPUT#99,name$(a)\Grad(a,1)\Grad(a,2)\Grad(a,3)
1646 CLOSE#99
1647 END DEFine

1649 DEFine PROCedure LTDefault                    [CTRL-Spacebar : then type LTDefault  Enter]
1650 REMark League Table Score
1651 name$(1)='QBITS    ':Grad(1,1)=1730:Grad(1,2)=756:Grad(1,3)=1072
1652 name$(2)='         ':Grad(2,1)=0:Grad(2,2)=0:Grad(2,3)=0
1653 name$(3)='         ':Grad(3,1)=0:Grad(3,2)=0:Grad(3,3)=0:LSave
1654 END DEFine

1656 DEFine PROCedure LTReset                      [CTRL-Spacebar : then type LTReset Enter]
1657 REMark League Table Reset
1658 name$(1)='         ':Grad(1,1)=0:Grad(1,2)=0:Grad(1,3)=0
1659 name$(2)='         ':Grad(2,1)=0:Grad(2,2)=0:Grad(2,3)=0
1660 name$(3)='         ':Grad(3,1)=0:Grad(3,2)=0:Grad(3,3)=0:LSave
1661 END DEFine
```

Pandemic Introduction

Games test our abilities and may even educate us by reflecting upon an environmental and/or mental attitude. They can expand our horizons, but also be just an escape and rewarding form of entertainment. The aim of any Game is keeping the rules and direction of flow easy to follow, but with variables that can create elaborate complexities which add to the enjoyment. As in the Game of Chess one 'Learns the Rules to Master the Game'.

The first rule is a Game must comply with and be realistically played within the environment designed around it. There must be an end goal to which it progressively flows, hence has **Direction**. Interactive components, **Actions, Events, Hazards** that are encountered**,** direct the Game to its ultimate conclusion, **Success** or **Defeat**. Further tests to skill and determination are the **boundaries,** set Countdowns, **Time limits** or as a depleting number of **Turns**, or reducing strengths, **Financial** losses, **Energy** levels or simply Game **Lives** lost.

QBITS 2020 Challenge

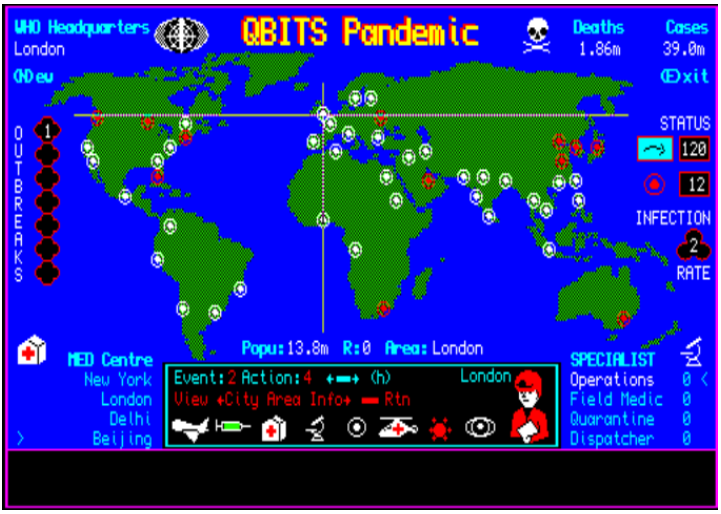
You might have guessed following the lockdown with Covid19 that **Pandemics** became an obvious theme. The Project began as a simple statement: - **QBITS Pandemic** – the World Health Organisation (**WHO**) with a team of **Specialist** set out to limit the **Spread**, find a **Cure**, then **Vaccinate** the population to **Eradicate** a new and **Deadly Virus**.

Artwork began with a **World Map** and **City locations** to which was added an **Outbreak** Tracker, Infection **Rate** and duration of Play **Status** indicators, values derived from a depleting number of City Virus Cards and an increasing Rate of infection. Activity was conceived as a number of **Events** and **Actions** taken.

The Game starts with an initial **Outbreak**, then as the game progresses more city areas become infected. The Challenge was determining a level of difficulty tied into the ability of a Player being able to develop a Game **Strategy.** Increased Opportunities greatly increase the enjoyment of Playing as well as determining the Chances of Winning.

QBITS Pandemic Intro

The Program opens with a Mission Statement outlining the nature of the Game, a World Map and eight WHO Specialists, four of which are randomly chosen to become the team for each New Game. Press any key and the Game screen is displayed.



QBITS Pandemic Game Screen

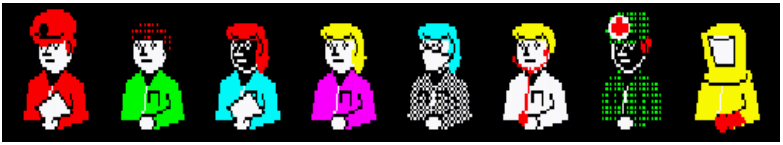
The headings across the top show the selected **WHO Headquarters,** which remains the city of choice at the beginning of each game. Centre page is the **QBITS Title**, followed by the accumulated **Deaths** and confirmed **Cases** which are updated as the Game progresses.

For a (**N**)ew Game press '**N**' or to (**E**)xit press '**E**' (upper or lower case characters apply).

Down the left side is the **OUTBREAKS** Tracker and below this the 4 relocatable **MED** centres. Each time a **MED** centre is Built in an Infected city the **R:** rate is cleared (**R:0**).

On the right side of the screen **STATUS** Boxes show the number of **Turns** left to play and number of **Cities** Infected. Below this is the **Infection Rate Indicator** which scales up each time an **Epidemic** occurs. This escalates the number of Cities with further Infection Increases at the end of a turn until a **Cure Vaccine** is released. If the **R:** factor of a city exceeds 3 then the linked Cities **R:** rates are raised by 1. If any of the linked city **R:** rates rise above 3 this can trigger a cascade of **Outbreaks.**

Bottom right are listed the four **Specialists** chosen from the eight and recruited for each Game.



SPECIALISTS: Operations Planner Dispatcher Researcher Scientist Doctor Field Medic Quarantine

QBITS Pandemic Game

City names are randomly shuffled the first twelve selected to create an initial **Outbreak**. The Game then begins after choosing one of four Cities offered to be WHO Headquarters.

QBITS Pandemic Turns

This can appear quite complex with the decisions that may be taken. A **Specialist**/Player carries out **2 Events** and **4 Actions** that may be taken in any order. The **Event** /**Action** image is selected using the **Left** & **Right Cursor Keys** and by pressing the **Spacebar**. The Red highlight will disappear and a printed **Prompt** is displayed requiring a response. **Enter** will confirm the activity and **Spacebar** will return without the activity taking place The **City** location when required will be identified by a **crosswire** with the **Popu**lation **R:** and **City Name** displayed just above the activity window. Whatever decisions are made all **Events** and **Actions** must be taken to move onto the **Next Turn / Specialist**.

QBITS Pandemic Strategy

The opening rounds are Containment and acquiring **RES**earch credits to activate the release of a **Vaccine,** this is a top priority. Any Specialist with 5 credits can release the Vaccine but only when they are stationed in a City with a **MED Centre**. The exceptions to this rule are the **Researcher** who only requires 4 credits and the **Doctor** who with 5 credits can release the Vaccine from any city. Once a **Vaccine** is released then the Virus **Spread** is halted, but then it becomes a race to clear all cities of infection before running out of **Turns**.

Pressing (**h**) will display the attribute of current **Specialist** and the current **City locations** of all four.

Events: 2 (Select from:)

Flight	Move **Specialist** to another City
Vaccine	Deliver and clear City Infections **R:** (active only after Vaccine Release)
MED	Relocate **MED** Centre to another City
RES	Add **RES**earch Credit (active until Vaccine Release)

Action: 4 (Combination of :)

R:	Reduce the **R:** rate in current City if applicable.
Airlift	Move **Specialist** +/- four cities from current location.
Virus	Release **Vaccine** if current Specialist has required credits
View:	Use **Left** & **Right Cursor keys** to move City to City across the Map shown by crosswires, **Popu**lation, **R:** rate and **City Area names** are displayed.

QBITS Pandemic Activity

Activity such as the **Cure** and **Vaccine** Release are dependent on the state of play. The **R:** Rate is active if the current City Infection is above zero. Once enough **Cure** credits are gathered a **Vaccine** can be released and further city Infections stop. Each **Specialists** carries an *Attribute* see below: -

Flight: evt%-1 *Dispatcher*

This Offers transfer of Specialist to the next City in Card Deck. Useful to reach an Infected City.
Destination extended to any City 1-48

Vaccine: evt%-1 *Field Medic*

Delivers to a group of Cities based around next City in Card Deck. Once delivered City R: is reduced to zero.
Delivery extended to any City 1-48

MED Centre**:** evt%-1 *Operations*

Relocation of MED Centre to next City in Deck. Upon arrival City R: is reduced to zero
Relocation extended to any City 1-48

RESearch Credits**:** evt%-1 *Scientist*

Adds Credits to Specialists Total

Gains an extra Turn added to Countdown

R: rate act%-1 *Planner*

Reduce R: by one for one Action

Free Turn against Countdown

Airlift: act%-1 *Quarantine*

Offers Transfer of Specialist to group of local cities. Once in location can use **R:** rate to reduce Infections
Upon arrival City R: is reduced to zero

Cure: act%-1 *Researcher* *Doctor*

If Specialist has 5 Credits and in City with a MED Centre Vaccine can be released.
As before but requires only 4 not 5 Credits
With 5 Credits can release Vaccine from any City

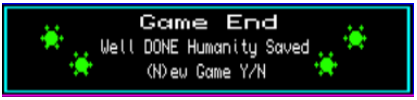
View Info: Access any City Area to show Status & Game (**E**)xit Y/N

QBITS Pandemic Virus Spread

At the end of each **Specialist**/Player **Turn** and until a **Vaccine** is released City Infections continue with the next card form the deck in accordance with **Global Infection Rates 2/3/4**. An **Epidemic** card raises **R:** Rate by 3**.**

QBITS Pandemic Game End

The Game is **Lost** if more than seven **Outbreaks** occur or if the number of **Turns** runs out without managing to **Clear** all **Infected Cities**. For a **Specialist Team** to **Win** all infected Cities have to be cleared of the **Virus**.



QBITS Pandemic Graphics

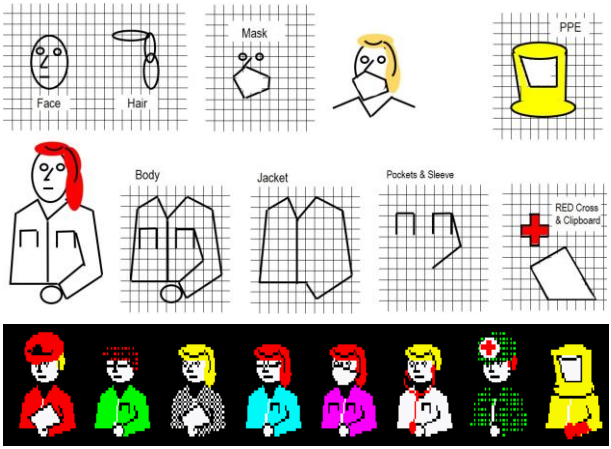
Drawing the characters for QBITS Pandemic and knowing I wanted them to be scalable led to thoughts about the detail and when scaling up or down what might remain recognisable.

When our eyes scan any area, the brain is analysing for pattern recognition. It is one of the reasons why changing shades of light appear to reveal previously hidden images. Looking at cloud formation or staring long enough at the weave of a carpet, your brain in trying to make sense. It will no doubt aspire to link it to a familiar shape such as a face or animal. It is therefore not difficult to create a familiar and recognisable image with only a few lines.

I use a grid to work out the x,y offsets from a common starting point, then add CIRCLE/Ellipse their size or start and end of LINE positions. To this I need only a channel ID, the Window x.y graphic coordinates and a Scale to position on screen.

QBITS Pandemic Specialists

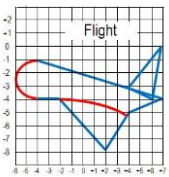
A face for example needs only a round circle with two smaller circle for eyes and three short lines to represent the nose and mouth. Adding hair to a face can be achieved with ellipse shapes, one on top for very short, extending the side with a second for average length and a third lower down to resemble long hair. To help maintain the face outline I overlay with ARCs drawn in black.



The left and right sides of the torso are split to achieve the v neck. This is because the FILL command of S/SuperBASIC cannot be used with re-entrant shapes. Overlaying with black lines divides the jackets into two halves, and outlines the sleeve and pockets.

Changes in colour for jacket and hair with differing lengths can create a variety of Characters. Adding Head Gear, a Mask, Clipboard, Stethoscope and not forgetting PPE gear all help broaden the Roles they might represent.

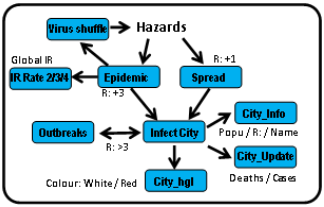
QBITS Pandemic Events Actions



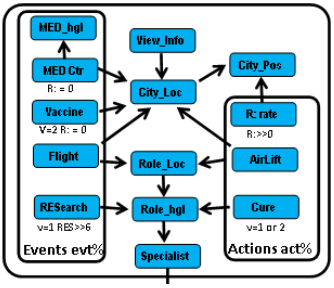
Drawing Object Graphics are a combination of **ARC**'s & **LINE**'s, some areas FILL'd with others just a LINE or ARC to emphasize the image.

QBITS Pandemic Code

```
1000 REMark QBITS_Pandemic_v3 (QBITS Pandemic v3 2021 QPCII)

1002 OPEN_IN#9,'ram2_QBITSConfig':INPUT#9,gx\gy\dn$\dev$:CLOSE#9

1004 REMark Time Aproxx 45min    :To PAUSE - Suspend Actions and/or Events
1005 rate=2:evt%=2:act%=4        :REMark Infection rate : Events : Actions
1006 tnum=120                    :REMark Turns: City Deck+Epidemics
1007 cres=5                      :REMark Number Credits for Vaccine Release

1009 MODE 4:Init_win:lchk=0:QBITS_Pandemic

1011 DEFine PROCedure QBITS_Pandemic
1012 IF lchk=0:Intro_Pandemic:Init_Scrn:Init_city:Init_Info:lchk=1
1013 New_Game:c=1:w=1:v=1:vn=13:gir=2:s=1:Role_HGL 7,s:num=tnum:m%=4
1014 REPeat Game_lp
1015   ActEvent:IF num<=1 OR ob>7 OR ctr=0:Game_End
1016   IF act%=0 AND evt%=0
1017     Act_HGL 2,6:CLS#0:Role_HGL 5,s:s=s+1:IF s>4:s=1          HAZARDS
1018     IF v=1                                                  v=1 Virus active
1019       FOR haz=1 TO rate                                     rate = Global Infection Rate
1020         BLOCK#3,230,10,0,12,0:vn=vn+1:num=num-1
1021         r%=RND(1 TO 8):IF r%=3 OR r%=7:Epidemic:ELSE Spread
1022       END FOR haz
1023     END IF
1024     Act_HGL 0,6:act%=4:evt%=2:m%=4:ActEvent:Role_HGL 7,s    Change to Next Specialist
1025   END IF
1026   en=Team(s,1):Rate_HGL en
1027   Act_HGL 2,m%:k=CODE(INKEY$(-1)):Act_HGL 0,m%              Key  Input
1028   SELect ON k                                              EVENT / ACTION
1029     =  72,104:INK#0,7:Help_Info Role(s)          :REMark Hh  Specialist Help Info
1030     =192:m%=m% -1:IF m%<0:m%=7                              Activity Highlight Left
1031     =200:m%=m%+1:IF m%>7:m%=0                               Activity Highlight Right
1032     =  32:a%=m%+1:INK#4,2:Activity:a%=0:IF lchk=1:CLS#0:num=num-1:lchk=0
1033     =110,78,101,69:CLS#0:GChange:CLS#0       :REMark (N)ew Game or (E)xit
1034     =  61:IF Tck%=0:Tck%=1:ELSE Tck%=0       :REMark TestMode On/Off
1035     =232:IF Tck%=1:act%=0:evt%=0             :REMark F1  Next Specialist
1036     =236:IF Tck%=1:num=tnum                  :REMark F2  Reset Turns
1037     =240:IF Tck%=1:Spread                    :REMark F3  Checks
1038     =244:IF Tck%=1:Epidemic:IF gir=7:gir=2   :REMark F4  Checks
1039     =248:IF Tck%=1:Outbreak:                 :REMark F5  Checks
1040   END SELect
1041 END REPeat Game_lp
1042 END DEFine
```



Note: **Hazards** engaged with a Random element increase the City Infections. **Events** and **Actions** are Choices to be made, in Halting the Spread, Defeat & Eradicate the Virus



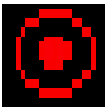

```
1044 DEFine PROCedure GChange
1045 IF k=78 OR k=110:kchk=1:CURSOR#0,230,6:PRINT#0,'New Y/N'
1046 IF k=69 OR k=101:kchk=2:CURSOR#0,226,6:PRINT#0,'Exit Y/N'
1047 PAUSE:IF KEYROW(5)<>64:kchk=0:CLS#0:RETurn
1048 CLS#0:IF kchk=1:QBITS_Pandemic:ELSE LRUN dn$
1049 END DEFine

1051 DEFine PROCedure Game_End
1052 ch=3:CLS#3::INK#3,7
1053 CSIZE#3,1,0:CURSOR#3,88,6:PRINT#3,'Game End':CSIZE#3,0,0
1054 IF ctr=0
1055     cv=4:CURSOR#3,64,14:PRINT#3,'Well DONE Humanity Saved'
1056 ELSE
1057    cv=2:CURSOR#3,48,20:PRINT#3,'You LOSE Virus Not Eradicated'
1058 END IF
1059 CURSOR#3,94,26:PRINT#3,'(N)ew Game Y/N'
1060 AVirus cv,15,17: AVirus cv,25,8:AVirus cv,118,17:AVirus cv,108,8:
1061 PAUSE: IF KEROW(5)=64:QBITS_Pandemic:ELSE LRUN dn$
1062 END DEFine

1064 DEFine PROCedure Act_HGL(col,m%)
1065 BLOCK#3,244,12,0,12,0:x1=1+m%*14.5:x2=15+m%*14.5
1066 INK#3,col:LINE#3,x1,1 TO x1,10 TO x2,10 TO x2,1 TO x1,1
1067 END DEFine
```


```
1069 DEFine PROCedure ActEvent
1070 INK#3,2:CURSOR#3,42,1:PRINT#3,evt%:CURSOR#3,96,1:PRINT#3,act%
1071 INK#2,7:ctr=0:FOR i=1 TO 48:IF city(i,7)>0:ctr=ctr+1
1072 STRIP#2,0:CURSOR#2,479,67:PRINT#2,FILL$(' ',3-LEN(num))&num
1073 CURSOR#2,484,85:PRINT#2,FILL$(' ',2-LEN(ctr))&ctr:STRIP#2,1
1074 END DEFine
```


```
1076 DEFine PROCedure Help_Info(sh)
1077 ln%=LEN(Info$(sh))+10:CURSOR#0,254-ln%/2*6,2:PRINT#0,'Atribute: '&Info$(sh)
1078 ln$='Specialist Locations ':FOR i=1 TO 4:ln$=ln$&'('&i&')'&city$(Team(i,1))&' '
1079 ln%=LEN(ln$):INK#0,4:CURSOR#0,254-ln%/2*6,14:PRINT#0,ln$
1080 END DEFine
:
```

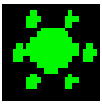
```
1082 DEFine PROCedure Activity
1083  IF a%=1 AND evt%>0                                :City_chg     :REMark Flight
1084  IF a%=2 AND evt%>0 AND v=2                        :Vacc_chg     :REMark Vaccine
1085  IF a%=3 AND evt%>0                                :MED_chg      :REMark MED Centre
1086  IF a%=4 AND evt%>0 AND v=1 AND Team(s,2)<5        :RES_chg      :REMark RESearch Credit
1087  IF a%=5 AND act%>0 AND city(en,7)>0               :Rate_chg     :REMark R: rate
1088  IF a%=6 AND act%>0:cn=Team(s,1)                   :Move_chg     :REMark Airlift
1089  IF a%=7 AND act%>0 AND v=1                        :Cure_chg     :REMark Release Cure
1090  IF a%=8                                           :View_chg     :REMark Info Check
1091  IF vn>48:vn=1
1092 END DEFine

1094 DEFine PROCedure City_chg
1095 INK#3,2:CURSOR#3,4,12:PRINT#3,'Commercial Flight ← Rtn'
1096 BLOCK#3,2,4,118,14,2:BLOCK#3,12,3,126,16,2
1097 cn=virus(vn):cm=cn                                 :REMark Role(s)=3 Dispatcher
1098 IF Role(s)=3:sm=cn-48:em=cn+48:ELSE sm=cn:em=cn
1099 City_loc:IF lchk=1:Team(s,1)=cn:City_Visit cn:vn=vn+1:evt%=evt%-1
1100 END DEFine

1102 DEFine PROCedure Vacc_chg
1103 INK#3,2:CURSOR#3,4,12:PRINT#3,'Vaccine ← City Drop→ ← Rtn'
1104 BLOCK#3,2,4,136,14,2:BLOCK#3,12,3,146,16,2
1105 cn=virus(vn):cm=cn                                 :REMark Role(s)=7 Field Medic
1106 IF Role(s)=7:sm=cn-48:em=cn+48:ELSE sm=cn-RND(1 TO 8):em=cn+RND(1 TO 8)
1107 City_loc:IF lchk=1:city(cn,7)=0:City_Info cn:vn=vn+1:evt%=evt%-1
1108 END DEFine

1110 DEFine PROCedure MED_chg
1111 INK#3,2:CURSOR#3,4,12:PRINT#3,'Relocate MED Centre ← Rtn'
1112 BLOCK#3,2,4,130,14,2:BLOCK#3,12,3,146,16,2
1113 cn=virus(vn):cm=cn                                 :REMark Role(s)=1 Operations
1114 IF Role(s)=1:sm=cn-48:em=cn+48:ELSE sm=cn:em=cn
1115 City_loc
1116 IF lchk=1
1117  MEDCtr(c)=cn:MED_HGL 5,c:city(cn,7)=0:City_Info cn
1118  vn=vn+1:evt%=evt%-1:c=c+1:IF c>4:c=1
1119 END IF
1120 END DEFine

1122 DEFine PROCedure RES_chg
1123 INK#3,2:CURSOR#3,4,12:PRINT#3,'Bank RESearch Credit ← Rtn'
1124 BLOCK#3,2,4,136,14,2:BLOCK#3,12,3,146,16,2:c=s
1125 REPeat RES_lp
1126  IF Team(c,2)<cres:k=CODE(INKEY$(-1)):ELSE EXIT RES_lp
1127  SELect ON k
1128    =32: EXIT RES_lp
1129    =10:Team(c,2)=Team(c,2)+1:vn=vn+1:evt%=evt%-1:lchk=1:EXIT RES_lp
1130  END SELect
1131 END REPeat RES_lp
1132 IF lchk=1 AND Role(s)=5:num=num+2                   :REMark Role(s)=5 Scientist
1133 Cure_HGL 5,c
1134 END DEFine
```

```
1136 DEFine PROCedure Rate_chg
1137 INK#3,2:CURSOR#3,4,12:PRINT#3,'Reduce R: rate ← Rtn'
1138 BLOCK#3,2,4,100,14,2:BLOCK#3,12,3,110,16,2
1139 REPeat Rate_lp
1140  City_pos en:k=CODE(INKEY$(-1)):City_pos en
1141  IF k=10 AND city(en,7)>0:city(en,7)=city(en,7)-1:act%=act%-1:lchk=1
1142  CURSOR#2,252,168:PRINT#2,city(en,7):Rate_HGL en
1143  IF k=32 OR city(en,7)=0 OR act%<1:EXIT Rate_lp
1144 END REPeat Rate_lp
1145 IF lchk=1 AND Role(s)=2:num=num+1                    :REMark Role(s)=2 Planner
1146 END DEFine
```



```
1148 DEFine PROCedure Move_chg
1149 INK#3,2:CURSOR#3,4,12:PRINT#3,'Airlift to ←Next City → ←  Rtn'
1150 BLOCK#3,2,4,148,14,2:BLOCK#3,12,3,158,16,2
1151 cm=cn:sm=cn-act%:em=cn+act%:mact%=act%:City_loc
1152 IF lchk=1
1153  IF cm<Team(s,1):mact%=Team(s,1)-cm                 :REMark < used act% count
1154  IF cm>Team(s,1):mact%=cm-Team(s,1)                 :REMark > used act% count
1155  IF cn=Team(s,1):RETurn
1156  IF Role(s)=8:city(cn,7)=0:City_Info cn             :REMark Role(s)=8 Quarantine
1157  Team(s,1)=cn:City_Visit cn:act%=act%-mact%
1158 END IF
1159 END DEFine
```

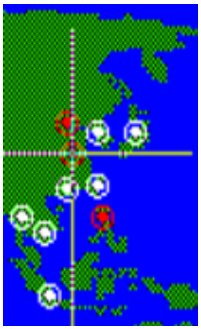
```
1161 DEFine PROCedure Cure_chg
1162 IF v=2:RETurn
1163 IF Role(s)=4 AND Team(s,2)=cres-1:Team(s,2)=cres   :REMark Role(s)=4 Researcher
1164 tchk=0:FOR i=1 TO 5:IF Team(s,1)=MEDCtr(i):tchk=1::EXIT i
1165 IF tchk=0 AND Role(s)<>6:RETurn                    :REMark Role(s)=6 Doctor
1166 IF Team(s,2)<5
1167    INK#3,2:col=2:CURSOR#3,4,12:PRINT#3,'No Vaccine Available     Rtn'
1168 ELSE
1169    INK#3,4:col=4:CURSOR#3,4,12:PRINT#3,'     Vaccine Released     Rtn'
1170    v=2:ch=3:AVirus 4,95,5
1171 END IF
1172 BLOCK#3,12,3,128,16,col:PAUSE
1173 END DEFine
```



```
1175 DEFine PROCedure View_chg
1176 INK#3,2:CURSOR#3,4,12:PRINT#3,'View ←City Area Info→     Rtn'
1177 BLOCK#3,12,3,138,16,2:cm=cn:sm=cn-48:em=cn+48:chk=2:City_loc
1178 END DEFine
```

```
1180 DEFine PROCedure City_Shuffle
1181 DIM virus(48):FOR i=1 TO 48:virus(i)=i
1182 FOR c=48 TO 3 STEP -1
1183  ran=RND(1 TO c-1):temp=virus(c):virus(c)=virus(ran):virus(ran)=temp
1184 END FOR c
1185 END DEFine
```

```
1187 DEFine PROCedure City_loc
1188 REMark cn city number:cm transitional:sm start/ em end city
1189 REPeat Move_lp
1190  IF cm< 1:cn=48+cm       :REMark cm<1  [48+ -cm : 48/47/46 etc]
1191  IF cm>48:cn=cm-48       :REMark cm>48 [cm  -48 :  1/ 2/ 3 etc]
1192  City_pos cn:k=CODE(INKEY$(-1)):City_pos cn
1193  SELect ON k
1194   =192:cm=cm -1:IF cm<sm:cm=sm:ELSE cn=cm
1195   =200:cm=cm+1:IF cm>em:cm=em:ELSE cn=cm
1196   =32:lchk=0:EXIT Move_lp
1197   =10:lchk=1:EXIT Move_lp
1198  END SELect
1199 END REPeat Move_lp
1200 END DEFine
```



```
1202 DEFine PROCedure City_pos(n)
1203 mx=city(n,1):my=city(n,2):INK#2,7:INK#1,7:OVER#1,-1
1204 LINE#1,mx,10 TO mx,86:LINE#1,10,my TO 184,my:OVER#1,0:City_Info n
1205 END DEFine
```


```
1207 DEFine PROCedure Spread
1208 BLOCK#3,230,10,0,12,0:en=virus(vn):Infect en,1
1209 INK#3,2:CURSOR#3,2,12:PRINT#3,'Virus Increase ';city$(en):PAUSE 80
1210 END DEFine

1212 DEFine PROCedure Infect(ci,r)
1213 city(ci,7)=city(ci,7)+r:City_Info ci:City_Update ci:PAUSE 20
1214 IF city(ci,7)>3:city(ci,7)=3:City_Info ci:Outbreak ci
1215 END DEFine
```


```
1217 DEFine PROCedure Outbreak(ci)
1218 BLOCK#3,230,10,0,12,0:ob=ob+1:IF ob>7:Game_End
1219 INK#3,2:CURSOR#3,2,12:PRINT#3,'Outbreak in Surrounding Areas'
1220 BLOCK#2,6,90,4,50,1:INK#2,7
1221 FOR i=1 TO 9:CURSOR#2,4,50+i*9:PRINT#2,ob$(i):PAUSE 3
1222 OVER#2,1:CURSOR#2,12,101-ob*7,-3,-4:PRINT#2,ob:OVER#2,0
1223 PAUSE 50:BLOCK#3,230,12,0,12,0
1224 n1=city(ci,4):IF city(n1,7)<3:Infect n1,1
1225 n2=city(ci,5):IF city(n2,7)<3:Infect n2,1
1226 n3=city(ci,6):IF city(n3,7)<3:Infect n3,1
1227 END DEFine
```


```
1229 DEFine PROCedure Epidemic
1230 BLOCK#3,230,10,0,12,0:en=virus(vn):Infect en,3
1231 INK#3,2:CURSOR#3,2,12:PRINT#3,'Epidemic in ';city$(en);' '
1232 rate=4:IF gir<6:rate=3:IF gir<4:rate=2
1233 City_Shuffle:gir=gir+1:IF gir>7:gir=7
1234 STRIP#2,0:INK#2,7:CURSOR#2,210,60,-3,-4:PRINT#2,rate:STRIP#2,1
1235 PAUSE 80
1236 END DEFine
```

```
1238 DEFine PROCedure City_Info(n)
1239 INK#2,7:p$=city(n,3):pl=LEN(p$)
1240 CURSOR#2,198,168:PRINT#2,p$(1 TO pl-2);'.';p$(pl-1);'m '
1241 CURSOR#2,252,168:PRINT#2,city(n,7):Rate_HGL n
1242 CURSOR#2,302,168:PRINT#2,city$(n)&FILL$(' ',14-LEN(city$(n)))
1243 END DEFine
```


```
1245 DEFine PROCedure City_Update(ca)
1246 City_HGL ca,7,7:PAUSE 5:City_HGL ca,7,2:PAUSE 5:City_HGL ca,2,2
1247 INK#2,7:r=2:ac$='':od$='':IF ac>99900:Game_End
1248 ac=ac+city(ca,3):od=ac:Str$=ac*2:sl=LEN(Str$)
12469 IF sl<4:ac$='.'&Str$(1 TO sl-2):ELSE ac$=Str$(1 TO sl-3)&'.'&Str$(sl-2)
1250 CURSOR#2,460,16:PRINT#2,FILL$(' ',5-LEN(ac$));ac$;'m'
1251 od=ac-RND(500 TO 999):Str$=od:sl=LEN(Str$)
1252 IF sl<4:od$='.0'&Str$(1):ELSE od$=Str$(1 TO sl-4)&'.'&Str$(sl-3 TO sl-2)
1253 CURSOR#2,400,16:PRINT#2,FILL$(' ',5-LEN(od$));od$;'m'
1254 END DEFine

1256 DEFine PROCedure City_Visit
1257 c$=city$(Team(s,1)):INK#3,5:CURSOR#3,160,0:PRINT#3,FILL$(' ',14-LEN(c$))&c$
1258 END DEFine

1260 DEFine PROCedure City_HGL(cn,c1,c2)
1261 x=city(cn,1):y=city(cn,2)
1262 INK#1,c1:CIRCLE#1,x,y,2:INK#1,c2:FILL#1,1:CIRCLE#1,x,y,.8:FILL#1,0
1263 END DEFine

1265 DEFine PROCedure Role_HGL(col,s)
1266 BLOCK#3,36,40,240,0,0:INK#2,col
1267 CURSOR#2,400,174+s*10:PRINT#2,Team$(Role(s)):Cure_HGL 5,s
1268 Specialist 3,126,17,Role(s):City_Visit
1269 END DEFine

1271 DEFine PROCedure MED_HGL(col,w)
1272 BLOCK#2,8,40,4,184,1:INK#2,col:w$=city$(MEDCtr(w))
1273 CURSOR#2,6,174+w*10:PRINT#2,'> ';FILL$(' ',14-LEN(w$));w$
1274 END DEFine
```


```
1276 DEFine PROCedure Rate_HGL(cn)
1277 IF city(cn,7)=0:City_HGL cn,7,7:INK#3,7:ELSE INK#3,2
1278 CIRCLE#3,66,6,3:FILL#3,1:CIRCLE#3,66,6,1:FILL#3,0
1279 END DEFine

1281 DEFine PROCedure Cure_HGL(col,c)
1282 BLOCK#2,12,40,492,184,1:INK#2,col
1283 CURSOR#2,480,174+c*10:PRINT#2,Team(c,2);' <'
1284 END DEFine
```

```
1286 REMark QBPandemic Graphics

1288 DEFine PROCedure Specialist(ch,x,y,d%)
1289 SELect ON d%
1290  =1:Face 0,1,x,y:Hair  2,1,x,y:Body   2,0,x,y:Hood  2,x,y:Clip 7,x,y
1291  =2:Face 0,1,x,y:Hair 16,1,x,y:Body   4,1,x,y
1292  =3:Face 7,0,x,y:Hair  2,2,x,y:Body   5,1,x,y:Clip  7,x,y
1293  =4:Face 0,1,x,y:Hair  6,2,x,y:Body   3,1,x,y
1294  =5:Face 0,1,x,y:Hair  5,2,x,y:Body 248,2,x,y:Mask  x,y
1295  =6:Face 0,1,x,y:Hair  6,1,x,y:Body   7,1,x,y:Scope  x,y
1296  =7:Face 7,0,x,y:Hair  2,1,x,y:Body  32,2,x,y:Hood 32,x,y:Cross x-2,y+4
1297  =8:Body 6,0,x,y:PPE 6,x,y
1298 END SELect
1299 END DEFine
```

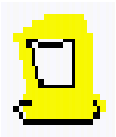
```
1301 DEFine PROCedure Face (col,fil,x,y)
1302 INK#ch,7:FILL#ch,fil:CIRCLE#ch,x,y-.6,4,.8,PI:FILL#ch,0          :REMark Face
1303 INK#ch,col:CIRCLE#ch,x-2.5,y+.5,.5:CIRCLE#ch,x+.5,y+.3,.5      :REMark Eyes
1304 LINE#ch,x-1,y+.4 TO x-2,y-1.2 TO x-.4,y-1.3                    :REMark Nose
1305 LINE#ch,x-1.5,y-2.6 TO x+.5,y-2.6                              :REMark Mouth
1306 END DEFine
```

```
1308 DEFine PROCedure Hair(col,lgth,x,y)
1309 INK#ch,col:FILL#ch,1:CIRCLE#ch,x,y+3,4,.4,PI/2:FILL#ch,0        :REMark Top
1310 IF lgth>0 :FILL#ch,1:CIRCLE#ch,x+3.5,y+1,2,.4,PI:FILL#ch,0     :REMark Back
1311 IF lgth>1 :FILL#ch,1:CIRCLE#ch,x+3.5,y-2,3,.4,PI:FILL#ch,0     :REMark Long
1312 INK#ch,0  :ARC#ch,x-3,y+3 TO x+2,y+3,PI/2:ARC#ch,x+2,y+3 TO x+3,y-2,200
1313 END DEFine
```

```
1315 DEFine PROCedure Body(col,pkt,x,y)
1316 INK#ch,col:FILL#ch,1                                           :REMark Left side
1317 LINE#ch,x-1,y-8 TO x-2,y-14 TO x-6,y-14 TO x-5,y-7 TO x-2,y-5 TO x-1,y-8
1318 FILL#ch,0:FILL#ch,1                                            :REMark Right side
1319 LINE#ch,x-1,y-8 TO x-1,y-14 TO x+1,y-16 TO x+7,y-14 TO x+6,y-7
1320 LINE#ch TO x+2,y-5 TO x,y-8:FILL#ch,0
1321 INK#ch,0:LINE#ch,x-1,y-15 TO x+4,y-12 TO x+3.5,y-9             :REMark Sleeve
1322 INK#ch,0:LINE#ch,x-.5,y-8 TO x-1.5,y-14                        :REMark Mid Divide
1323 IF pkt>0:INK#ch,0:LINE#ch,x+2,y-11 TO x+2,y-9 TO x,y-9 TO x,y-11
1324 IF pkt>1:INK#ch,0:LINE#ch,x-4,y-11 TO x-4,y-9 TO x-2,y-9 TO x-2,y-11
1325 INK#ch,7:FILL#ch,1:CIRCLE#ch,x-.7,y-15,1.6:FILL#ch,0           :REMark Hand
1326 INK#ch,0:CIRCLE#ch,x-.7,y-15,1.6
1327 END DEFine
```

```
1329 DEFine PROCedure Hood(col,x,y)
1330 INK#ch,col:FILL#ch,1:CIRCLE#ch,x-1.6,y+4,3.6,.8,-PI/5:FILL#ch,0
1331 FILL#ch,1:CIRCLE#ch,x+.5,y+4,4,.8,-PI/2:FILL#ch,0:INK#ch,0
1332 ARC#ch,x-4,y+3 TO x+2,y+3,PI/3:FILL#ch,1:CIRCLE#ch,x-2,y+3.4,1:FILL#ch,0
1333 END DEFine
```

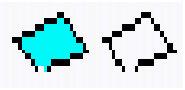
```
1335 DEFine PROCedure PPE(col,x,y)
1336 INK#ch,col:FILL#ch,1:CIRCLE#ch,x-.5,y+3,4,.4,PI/1.8:FILL#ch,0
1337 FILL#ch,1:CIRCLE#ch,x+.5,y-5.5,5,.3,PI/2:FILL#ch,0
1338 INK#ch,0:CIRCLE#ch,x,y-6,5,.3,PI/2     :FILL#ch,1:INK#ch,col
1339 LINE#ch,x-5,y+3 TO x+4,y+3 TO x+4,y-5 TO x-3,y-5 TO x-5,y+3:FILL#ch,0
1340 FILL#ch,1:INK#ch,7
1341 LINE#ch,x-3.5,y+2 TO x+2,y+2 TO x+2,y-3 TO x-2,y-3 TO x-3.5,y+2
1342 FILL#ch,0:INK#ch,0:ARC#ch,x-3.5,y+2 TO x+2,y+2,PI/5
1343 LINE#ch,x+2,y+2 TO x+2,y-3.5 TO x-2,y-3.5 TO x-3.5,y+2
1344 FILL#ch,1:INK#ch,2:CIRCLE#ch,x-.7,y-15,1.4:FILL#ch,0:FILL#ch,1
1345 LINE#ch,x,y-14 TO x+3,y-13 TO x+4,y-15 TO x+1,y-16 TO x+.5,y-14:FILL#ch,0
1346 END DEFine
```



```
1348 DEFine PROCedure Mask(x,y)
1349 FILL#ch,1:INK#ch,7
1350 LINE#ch,x-2,y TO x-4,y-1 TO x-2,y-5 TO x,y-5 TO x+3,y-2 TO x-2,y
1351 FILL#ch,0:INK#ch,0
1352 LINE#ch,x-2,y TO x-4,y-1 TO x-2,y-5 TO x,y-5 TO x+3,y-2 TO x-2,y
1353 END DEFine
```

```
1355 DEFine PROCedure Scope(x,y)
1356 INK#ch,2:ARC#ch,x-4,y-2 TO x+3,y,PI:FILL#ch,1:CIRCLE#ch,x+3,y-1,.6
1357 LINE#ch,x-.5,y-4 TO x-2,y-14:CIRCLE#ch,x-2,y-14,1.2,.5,PI:FILL#ch,0
1358 END DEFine
```

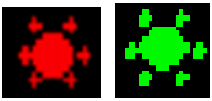
```
1360 DEFine PROCedure Clip(col,x,y)
1361 INK#ch,col:FILL#ch,1
1362 LINE#ch,x-1,y-15 TO x-4,y-12 TO x+1,y-9 TO x+4,y-13 TO x-1,y-15:FILL#ch,0
1363 INK#ch,0 :LINE#ch,x-2,y-15 TO x-4,y-12 TO x+1,y-9 TO x+4,y-13 TO x-1,y-15
1364 END DEFine
```



```
1366 DEFine PROCedure Cross(x,y)
1367 FILL#ch,1:INK#ch,7:CIRCLE#ch,x,y,2.2:FILL#ch,0:FILL#ch,1:INK#ch,2
1368 LINE#ch,x-1.6,y+.4 TO x+1.6,y+.4 TO x+1.6,y-.4 TO x-1.6,y-.4 TO x-1.6,y+.4
1369 LINE#ch,x-.4,y+1.5 TO x+.4,y+1.5 TO x+.4,y-1.5 TO x-.4,y-1.5 TO x-.4,y+1.5
1370 FILL#ch,0:INK#ch,5
1371 END DEFine
```

```
1373 DEFine PROCedure Skull(x,y)
1374 INK#ch,7:LINE#ch,x-4,y-3 TO x+4,y-6:LINE#ch,x-4,y-6 TO x+4,y-3
1375 LINE#ch,x-.4,y-3 TO x-.8,y-3:LINE#ch,x-1.2,y-2.5 TO x+1.6,y-2.5
1376 FILL#ch,1:LINE#ch,x-3,y+1 TO x-2,y-1.5 TO x+2,y-1.5 TO x+3,y+1
1377 ARC#ch, x+3,y+1 TO x-3,y+1,PI:FILL#ch,0:INK#ch,0
1378 FILL#ch,1:LINE#ch,x,y TO x-.8,y-1.5 TO x+.8,y-1.5 TO x,y:FILL#ch,0
1379 FILL#ch,1:CIRCLE#ch,x-1.5,y+.5,.9:FILL#ch, 0
1380 FILL#ch,1:CIRCLE#ch,x+1.7,y+.5,.9:FILL#ch,0:INK#2,5
1381 END DEFine
```

```
1383 DEFine PROCedure AVirus(col,x,y)
1384 LOCal a,b,r,c:RESTORE 1386:INK#ch,col
1385 FOR i=1 TO 7:READ a,b,r:FILL#ch,1:CIRCLE#ch,x+a,y+b,r:FILL#ch,0
1386 DATA  0,0,2,-1.8,3.2,.6,1.8,3.2,.6,-3.2, 0,.6
1387 DATA  3.2,0,.6,-1.8,-2.8,.6,1.8,-2.8,.6
1388 END DEFine
```



```
1390 DEFine PROCedure MEDCamp(x,y)
1391 INK#ch,7:FILL#ch,1:LINE#ch,x-3,y TO x+2,y+3 TO x+5,y+1
1392 LINE#ch TO x+5,y-3.8 TO x+3,y-5 TO x-3,y-5 TO x-3,y:FILL#ch,0
1393 INK#ch,0:LINE#ch,x+3,y-5 TO x+3,y TO x,y+2
1394 LINE#ch,x+3,y TO x+5,y+1.2:Cross x,y-2
1395 END DEFine
```

```
1397 DEFine PROCedure Vaccine(x,y)
1398 INK#ch,7
1399 LINE#ch,x-4,y+1.2 TO x+2,y+1.2 TO x+3,y+.6 TO x+3,y-.6 TO x+2,y-1.2
1400 LINE#ch TO x-4,y-1.2 TO x-4,y+1.2:LINE#ch,x+3,y TO x+6.6,y
1401 LINE#ch,x-6,y TO x-4,y:LINE#ch,x-6,y+2 TO x-6,y-2
1402 FILL#ch,1:INK#ch,4
1403 LINE#ch,x-3,y+.6 TO x+1.8,y+.6 TO x+1.8,y-.6 TO x-3,y-.6 TO x-3,y+.6
1404 FILL#ch,0:INK#ch,7
1405 END DEFine
```

```
1407 DEFine PROCedure Copter(x,y)
1408 INK#ch,7:LINE#ch,x-5,y TO x+6,y:LINE#ch,x,y-1 TO x-5,y-5
1409 FILL#ch,1:LINE#ch,x,y TO x+6,y-4 TO x+4,y-4:ARC#ch TO x,y-6,-PI/4
1410 LINE#ch TO  x-4,y-6:ARC#ch TO x-5,y-5,-PI:LINE#ch TO x-2,y-4 TO x,y
1411 FILL#ch,0:CIRCLE#ch,x+6.5,y-4,1:POINT#ch,x+6.5,y-4
1412 Cross x+1,y-4
1413 END DEFine
```

```
1415 DEFine PROCedure Flight(x,y)
1416 INK#ch,7
1417 FILL#ch,1:ARC#ch,x-4,y-4 TO x-4,y-1,-PI:LINE#ch TO x+7,y-4 TO x+4,y-5
1418 LINE#ch TO x+2,y-8 TO x-2,y-4 TO x-4,y-4:FILL#ch,0
1419 FILL#ch,1:LINE#ch,x+4,y-3 TO x+7,y TO x+6,y-4 TO x+4,y-3:FILL#ch,0
1420 INK#ch,0:ARC#ch,x-2,y-4 TO x+4,y-5,-PI/4
1421 END DEFine
```

```
1423 DEFine PROCedure ViewInfo(x,y)
1424 INK#3,7:CIRCLE#3,x,y,5,.6,PI/2
1425 CIRCLE#3,x,y,3:FILL#3,1:CIRCLE#3,x,y,.8:FILL#3,0
1426 END DEFine
```

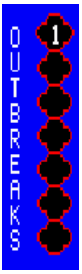
```
1428 DEFine PROCedure RESearch(x,y)
1429 INK#ch,7:ARC#ch,x+2,y TO x+5,y+4,PI:LINE#ch,x,y+1 TO x+4,y+1:FILL#ch,1
1430 LINE#ch,x+2.5,y+2.5 TO x+6,y+5 TO x+5.5,y+5.5 TO x+2,y+3 TO x+2.5,y+2.5
1431 FILL#ch,0:LINE#ch,x+2,y-2.5 TO x+6,y-2.5 TO x+5,y-1 TO x+3,y-1 TO x+2,y-2.5
1432 END DEFine
```

```
1434 DEFine PROCedure WHO_Symbol(x,y)
1435 INK#ch,248:FILL#ch,1 :CIRCLE#ch,x,y,8,.6,PI/2:FILL#ch,0
1436 INK#ch,7:FILL#ch,0   :CIRCLE#ch,x,y,8,.6,PI/2:FILL#ch,0
1437 INK#ch,0:FILL#ch,1   :CIRCLE#ch,x,y,5       :FILL#ch,0
1438 INK#ch,7:FOR i=1 TO 3:CIRCLE#ch,x,y,5-i*1.5
1439 LINE#ch,x-4.5,y TO x+4.5,y:LINE#ch,x-3.5,y+3.5 TO x+3.5,y-3.5
1440 LINE#ch,x+3.5,y+3.5 TO x-3.5,y-3.5
1441 INK#ch,7:FILL#ch,1:CIRCLE#ch,x,y,5,.1,PI:CIRCLE#ch,x,y+4,.8:FILL#ch,0
1442 END DEFine
```

```
1444 REMark Init Pandemic

1446 DEFine PROCedure Init_win
1447 OPEN#4,scr_:OPEN#3,scr_:h=.5
1448 WINDOW#4,512,256,gx,gy      :PAPER#4,0:CLS#4
1449 WINDOW#3,280,42,gx+116,gy+183:PAPER#3,0:SCALE#3,26,0,0 :BORDER#3,1,5
1450 WINDOW#2,508,226,gx+2,gy+1   :PAPER#2,0:SCALE#2,130,0,0:BORDER#2,1,3
1451 WINDOW#1,456,162,gx+28,gy+22 :PAPER#1,1
1452 WINDOW#0,508,30,gx+2,gy+226  :PAPER#0,0:CLS#0:BORDER#0,1,3
1453 END DEFine

1455 DEFine PROCedure Init_Scrn
1456 CLS#3:PAPER#2,1
1457 FOR i=0 TO 8:BLOCK#2,14,224, 0+i*12,0,1:BLOCK#2,14,224,490-i*12,0,1:PAUSE 2
1458 FOR i=1 TO 8:BLOCK#2,18,180,92+i*18,0,1:BLOCK#2,18,180,394-i*18,0,1:PAUSE 2
1459 QBITS_Title:INK#1,224:SCALE#1,96,0,0:m=.5
1460 Init_map:Draw_city:Draw_OB:Draw_Stat:Draw_IR:ch=2
1461 QBold 2,5,1,-2,5,'WHO Headquarters'     :WHO_Symbol 53,122
1462 QBold 2,5,1,396,5,'Deaths    Cases'     :Skull 162,123
1463 QBold 2,5,1,160,168,'Popu:      R:   Area:'
1464 QBold 2,5,1,-5,30,'(N)ew'                :QBold 2,5,1,456,30,'(E)xit'
1465 QBold 2,5,1,36,174,'MED Centre'          :MEDCamp 6,30
1466 QBold 2,5,1,394,174,'SPECIALIST'         :RESearch 206,27
1467 END DEFine

1469 DEFine PROCedure QBITS_Title
1470 CURSOR#2,10,10 :CSIZE#2,2,1:OVER#2,1
1471 INK#2,2:FOR i=0 TO 1:CURSOR#2,164+i,2:PRINT#2,'QBITS Pandemic'
1472 INK#2,6:FOR i=0 TO 1:CURSOR#2,166+i,3:PRINT#2,'QBITS Pandemic'
1473 CSIZE#2,0,0:OVER#2,0:INK#2,7
1474 END DEFine

1476 DEFine PROCedure QBold(ch,col,cs,cx,cy,Str$)
1477 INK#ch,col:OVER#ch,1
1478 FOR a=1 TO LEN(Str$)
1479   FOR b=0 TO cs:CURSOR#ch,cx+b+a*(5+cs),cy:PRINT#ch,Str$(a)
1480 END FOR a:OVER#ch,0
1481 END DEFine

1483 DEFine PROCedure New_Game
1484 LOCal w,x,y:RESTORE 1485:FOR i=1 TO 6:READ w,x,y:BLOCK#2,w,10,x,y,1
1485 DATA 48,4,16, 36,400,16, 36,460,16, 30,198,168, 6,252,168, 84,300,168
1486 STRIP#2,0:BLOCK#2,110,42,0,182,1:BLOCK#2,110,42,394,182,1
1487 ob=1:FOR i=2 TO 7:CURSOR#2,12,101-i*7,-3,-4:PRINT#2,' '          :REMark Outbreaks
1488 rate=2:INK#2,7:CURSOR#2,210,60,-3,-4:PRINT#2,rate                :REMark Infections
1489 STRIP#2,1:BLOCK#2,20,11,479,66,0:BLOCK#2,20,11,479,84,0          :REMark Status
1490 CLS#3:INK#3,7:Draw_city:City_Shuffle:dc=0:ac=0:vn=1              :REMark Deaths Cases
1491 CURSOR#3,40,6:PRINT#3,'WHO Announce New Global Pandemic'
1492 FOR i=1 TO cm:city(i,7)=0
1493 FOR i=1 TO  4:Infect virus(i),1:Infect virus(i+4),2:Infect virus(i+8),3
1494 MEDTeam:Init_Roles:INK#3,5
1495 CURSOR#3,4,1:PRINT#3,'Event:  Action:← →(h)':BLOCK#3,10,3,120,5,5
1496 ch=3:Flight 7,9:Vaccine 22.6,6:MEDCamp 36,6.5:RESearch 48,4
1497 Rate_HGL cn:Copter 79.5,9:AVirus 2,95,5:ViewInfo 109,6           :REMark Activities
1498 END DEFine
```

```
1500 DEFine PROCedure MEDTeam
1501 DIM MEDCtr(5):RESTORE 1513:INK#3,7:CLS#3
1502 CURSOR#3,40, 6:PRINT#3,'Select WHO HQ & Specialists Team'
1503 CURSOR#3,26,22:PRINT#3,'(N)ew York (L)ondon (D)elhi (B)eijing'
1504 k=CODE(INKEY$(-1)):CLS#3:INK#2,7
1505 SELect ON k
1506    =66 ,98:cn=41
1507    =68,100:cn=35
1508    =78,110:cn=13
1509    =REMAINDER :cn=18
1510 END SELect
1511 CURSOR#2,4,16:PRINT#2,city$(cn)&FILL$(' ',8-LEN(city$(cn)))
1512 FOR i=1 TO 4:READ MEDCtr(i):MED_HGL 5,i
1513 DATA 13,18,35,41 :MEDCtr(5)=cn
1514 END DEFine
```

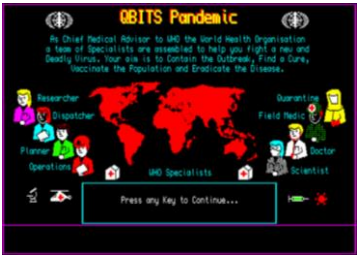
```
1516 DEFine PROCedure Draw_OB
1517 ob$='OUTBREAKS':ob=1:x=12:a=1.9:INK#2,7
1518 FOR i=1 TO 9:CURSOR#2,4,50+i*9:PRINT#2,ob$(i)
1519 FOR i=1 TO 7
1520  y=45+i*7:INK#2,0:FILL#2,1
1521  ARC#2,x-a,y+a TO x+a,y+a,-PI TO x+a,y-a,-PI TO x-a,y-a,-PI TO x-a,y+a,-PI
1522  FILL#2,0:INK#2,2
1523  ARC#2,x-a,y+a TO x+a,y+a,-PI TO x+a,y-a,-PI TO x-a,y-a,-PI TO x-a,y+a,-PI
1524 END FOR i
1525 INK#2,7:OVER#2,1:CURSOR#2,x,y,-3,-4:PRINT#2,'1':OVER#2,0
1526 END DEFine
```




```
1528 DEFine PROCedure Draw_Stat
1529 CURSOR#2,464,54:PRINT#2,'STATUS'
1530 BLOCK#2,22,13,478,65,2:BLOCK#2,20,11,479,66,0
1531 BLOCK#2,22,13,478,83,2:BLOCK#2,20,11,479,84,0
1532 INK#2,2:CIRCLE#2,198,78,3:FILL#2,1:CIRCLE#2,198,78,1.3:FILL#2,0
1533 BLOCK#2,26,13,448,65,2:BLOCK#2,24,11,449,66,5
1534 INK#2,0:ARC#2,195,88 TO 200,88,-PI/2:LINE#2,200,90 TO 201,88 TO 199,87
1535 END DEFine
```


```
1537 DEFine PROCedure Draw_IR
1538 INK#2,7:CURSOR#2,446,102:PRINT#2,'INFECTION'
1539 CURSOR#2,476,130:PRINT#2,'RATE':x=210:y=60:INK#2,0
1540 FILL#2,1:CIRCLE#2,x,y+2,2.5    :FILL#2,0
1541 FILL#2,1:CIRCLE#2,x-2.5,y-2,2.5:FILL#2,0
1542 FILL#2,1:CIRCLE#2,x+2.5,y-2,2.5:FILL#2,0 :INK#2,2
1543 CIRCLE#2,x-2.5,y-2,2.5:CIRCLE#2,x,y+2,2.5:CIRCLE#2,x+2.5,y-2,2.5
1544 FILL#2,1:INK#2,0:CIRCLE#2,x,y,2:FILL#2,0
1545 END DEFine
```

```
1547 DEFine PROCedure Init_Roles
1548 DIM Team$(8,11),Team(8,2),Role(8):RESTORE 1554:ch=2:INK#ch,5
1549 FOR i=1 TO 8:READ t$:Team$(i)=t$:Team(i,2)=0:Role(i)=i
1550 FOR t=8 TO 3 STEP -1
1551  ran=RND(1 TO t-1):temp=Role(t):Role(t)=Role(ran):Role(ran)=temp
1552 END FOR t
1553 INK#2,3:FOR i=1 TO 4:Role_HGL 5,i:Team(i,1)=cn
1554 DATA 'Operations ','Planner   ','Dispatcher ','Researcher '
1555 DATA 'Scientist  ','Doctor    ','Field Medic','Quarantine '
1556 END DEFine

1558 DEFine PROCedure Init_Info
1559 DIM Info$(8,42):RESTORE 1560:FOR i=1 TO 8:READ Info$(i)
1560 DATA 'MED Centre - Relocate to any City (R:>0)'      :REMark Operations
1561 DATA 'R: rate - Reduce with a Free Turn'             :REMark Planner
1562 DATA 'Flight - Extended to any City'                 :REMark Dispatcher
1563 DATA 'Release Vaccine - with one less Credit'        :REMark Researcher
1564 DATA 'Banking Research Credits - Gain extra Turn'    :REMark Scientist
1565 DATA 'Vaccine Release - Can be from any City'        :REMark Doctor
1566 DATA 'Vaccine Delivery - Extended to any City'       :REMark Field Medic
1567 DATA 'R: Rate - Reduced to Zero on entering City'    :REMark Quarantine
1568 END DEFine



1570 DEFine PROCedure Intro_Pandemic
1571 LOCal mx,m$:ch=2
1572 BLOCK#2,504,181,0,0,0:BLOCK#2,112,50,0,174,0:BLOCK#2,112,50,392,174,0
1573 QBITS_Title:STRIP#2,0:INK#2,5:RESTORE 1575
1574 FOR i=1 TO 4:PAUSE 5:READ mx,m$:CURSOR#2,mx,20+i*10:PRINT#2,m$
1575 DATA 60,' As Chief Medical Advisor to WHO the World Health Organisation'
1576 DATA 60,'a team of Specialists are assembled to help you fight a new and'
1577 DATA 60,'Deadly Virus. Your aim is to Contain the Outbreak, Find a Cure,'
1578 DATA 96,'Vaccinate the Population and Eradicate the Disease.'
1579 PAUSE 5:SCALE#1,180,-80,-20:m=.1 :INK#1,2:Init_map
1580 WHO_Symbol 34,121:WHO_Symbol 180,121:ch=2:Intro_Roles
1581 MEDCamp  66,32:RESearch 14,16:Copter 34,20
1582 MEDCamp 148,32:Vaccine 182,15:AVirus 2,196,15
1583 CURSOR#3,54,12:PRINT#3,'Press any Key to Continue...':PAUSE
1584 END DEFine
```

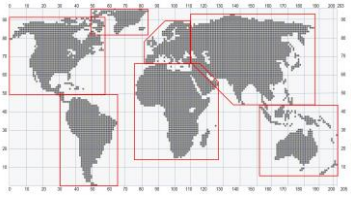



```
1586 DEFine PROCedure Intro_Roles
1587 LOCal x,y,s,c,r:ch=2:RESTORE 1593
1588 INK#2,5:CURSOR#ch,208,166:PRINT#2,'WHO Specialists'
1589 FOR i=1 TO 8
1590   READ x,y,s :Specialist 2,x,y,s:PAUSE 5
1591   READ c,r,r$:INK#2,5:CURSOR#2,c,r:PRINT#2,r$
1592 END FOR i
1593 DATA 12,75,4,48, 90,'Researcher', 204,75,8,392, 90,'Quarantine'
1594 DATA 24,65,3,70,108,'Dispatcher', 192,65,7,366,108,'Field Medic'
1595 DATA 36,55,2,24,144,'Planner',    180,55,6,440,144,'Doctor'
1596 DATA 48,45,1,36,160,'Operations', 168,45,5,412,164,'Scientist'
1597 END DEFine
```

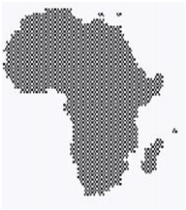
```
1599 REMark Pandemic Maps

1601 DEFine PROCedure Init_city
1602 RESTORE 1617:READ cm              :REMark cm city max:c city number
1603 DIM city$(cm,14),city(cm,7)
1604 FOR i=1 TO cm
1605   READ city$(i)                   :REMark Name
1606   READ city(i,1),city(i,2)        :REMark Grid x,y
1607   READ city(i,3)                  :REMark Population
1608   READ city(i,4),city(i,5),city(i,6)  :REMark Connected Cities
1609      city(i,7)=0                  :REMark Infection Rate
1610 END FOR i
1611 END DEFine
```


```
1613 DEFine PROCedure Draw_city
1614 FOR i=1 TO cm:City_HGL i,7,7
1615 END DEFine

1617 DATA 48                                    :REMark City Info
1618 DATA 'San Francisco',14,66,460,2,3,47,      'Los Angeles',16,62,1310,1,4,10
1619 DATA 'Vancouver',17,75,240,1,5,6,           'Mexico City',26,51,2440,2,7,9
1620 DATA 'Chicago',33,74,800,3,10,12,           'Atlanta',36,62,560,5,6,13
1621 DATA 'Miami',36,57,610,4,9,10,              'Lima',36,32,1000,9,11,15
1622 DATA 'Bogota',40,42,980,4,7,8,              'Washington',40,66,620,5,6,13
1623 DATA 'Santiago',44,17,660,8,14,48,          'Montreal',45,73,410,5,13,18
1624 DATA 'New York',45,69,2370,10,12,16,        'Buenos Aries',54,16,1270,11,15,28
1625 DATA 'Sao Paulo',62,23,2120,8,14,17,        'Madrid',85,68,620,13,18,20
1626 DATA 'Lagos',88,44,1600,15,20,23,           'London',88,76,1380,12,19,22
1627 DATA 'Paris',90,73,1250,18,20,21,           'Algiers',92,65,500,16,17,19
1628 DATA 'Milan',96,70,820,19,24,25,            'Stockholm',98,81,140,18,21,24
1629 DATA 'Kinshasa',98,35,1430,17,28,29,        'St Petersburg',103,81,750,21,22,26
1630 DATA 'Istanbul',105,69,1480,21,27,31,       'Moscow',106,75,1790,19,24,41
1631 DATA 'Cairo',108,57,2040,25,29,32,          'Johannesburg',107,17,800,14,23,34
1632 DATA 'Khartum',111,50,520,23,27,32,         'Baghdad',115,63,870,27,30,33
1633 DATA 'Tehran',120,65,1600,25,26,30,         'Barhrain',121,56,170,29,30,33
1634 DATA 'Karachi',132,57,2750,32,34,35,        'Mumbai',136,51,2770,28,33,36
1635 DATA 'Delhi',138,57,2660,33,37,41,          'Chennai',140,45,1330,34,37,40
1636 DATA 'Kolkata',145,56,1410,35,36,38,        'Bankok',154,50,1450,37,39,42
1637 DATA 'Ho Chi Minh',158,47,1010,38,40,42,    'Jakarta',159,35,3020,36,38,48
1638 DATA 'Beijing',162,68,2490,26,35,44,        'Hong Kong',162,55,720,39,43,46
1639 DATA 'Shanghai',163,62,3400,42,44,45,       'Seoul',167,66,2560,41,43,47
1640 DATA 'Taipia',167,56,700,43,46,47,          'Manila',168,50,2270,42,45,48
1641 DATA 'Tokyo',174,66,3780,1,44,45,           'Sydney',182,14,500,11,40,46
```

```
1643 DEFine PROCedure Init_map
1644 RESTORE 1746:READ map:Draw_map :REMark Greenland
1645 RESTORE 1671:READ map:Draw_map :REMark America North
1646 RESTORE 1736:READ map:Draw_map :REMark Europe
1647 RESTORE 1698:READ map:Draw_map :REMark Asia
1648 RESTORE 1688:READ map:Draw_map :REMark America south
1649 RESTORE 1659:READ map:Draw_map :REMark Africa
1650 RESTORE 1721:READ map:Draw_map :REMark Australasia
1651 END DEFine
```



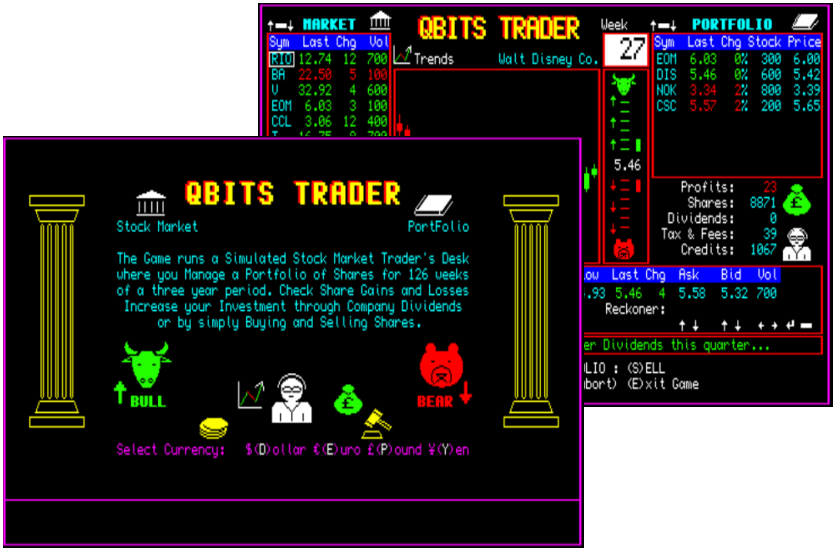
```
1653 DEFine PROCedure Draw_map
1654 FOR i=1 TO map
1655   READ x,y,n:FILL 1:LINE x,y TO x+n,y TO x+n,y-h TO x,y-h TO x,y:FILL 0
1656 END FOR i
1657 END DEFine

1659 DATA 68                          :REMark Africa
1660 DATA 89,65,7,  85,64,11, 104,64,1, 109,64,1,  83,63,14, 82,62,17, 102,62,3
1661 DATA 82,61,24, 82,60,26, 103,60,7, 80,59,30, 79,58,32, 79,57,32, 78,56,34
1662 DATA 78,55,34, 77,54,36, 78,53,35, 78,52,36, 78,51,36, 77,50,38, 77,49,39
1663 DATA 78,48,39, 78,47,39, 120,47,2, 79,46,43, 80,45,42, 80,44,42, 82,43,7
1664 DATA 92,43,29, 94,42,27, 95,41,24, 95,40,23, 94,39,23, 94,38,21, 95,37,20
1665 DATA 96,36,19, 96,35,18, 97,34,17, 97,33,18, 97,32,18, 97,31,18, 97,30,18
1666 DATA 125,32,1, 97,29,18, 120,29,2, 96,28,19, 119,28,3, 96,27,18, 117,27,4
1667 DATA 96,26,16, 117,26,4, 97,25,15, 117,25,3, 97,24,14, 117,24,3, 98,23,14
1668 DATA 116,23,4, 98,22,14, 116,22,4, 98,21,13, 117,21,2, 98,20,12, 98,19,12
1669 DATA 99,18,11, 100,17,8, 100,16,7, 100,15,3, 104,15,1
```



```
1671 DATA 102                         :REMark America North 102
1672 DATA 34,90,4, 41,90,1, 44,90,2, 47,90,2, 50,90,1, 52,90,1, 32,89,9
1673 DATA 46,89,1, 49,89,7, 13,88,9, 29,88,1, 34,88,8, 45,88,2, 51,88,7
1674 DATA 9,87,26, 36,87,2, 43,87,5, 49,87,2, 54,87,4, 10,86,40, 54,86,5
1675 DATA 6,85,40, 51,85,6, 7,84,37, 46,84,2, 53,84,4, 4,83,38, 50,83,2
1676 DATA 54,83,2, 4,82,7, 13,82,27, 49,82,3, 3,81,5, 16,81,23, 48,81,5
1677 DATA 55,81,1, 3,80,2, 6,80,1, 16,80,24, 48,80,8, 1,79,2, 16,79,26
1678 DATA 47,79,10, 17,78,27, 46,78,10, 17,77,27, 46,77,10, 17,76,38, 16,75,1
1679 DATA 18,75,32, 17,74,31, 50,74,2, 56,74,1, 16,73,20, 39,73,11, 55,73,3
1680 DATA 16,72,22, 41,72,12, 15,71,22, 38,71,1, 41,71,7, 50,71,2, 14,70,22
1681 DATA 37,70,10, 14,69,25, 40,69,6, 13,68,31, 13,67,30, 13,66,28, 13,65,28
1682 DATA 13,64,27, 15,63,24, 15,62,23, 15,61,1, 17,61,20, 15,60,1, 17,60,11
1683 DATA 36,60,2, 15,59,2, 18,59,9, 36,59,2, 16,58,1, 18,58,8, 36,58,2
1684 DATA 16,57,1, 19,57,7, 36,57,1, 17,56,1, 19,56,6, 20,55,5, 34,55,5
1685 DATA 20,54,6, 30,54,2, 38,54,2, 20,53,6, 29,53,3, 41,53,2, 21,52,10
1686 DATA 37,52,1, 40,52,3, 45,52,1, 23,51,8
```

```
1688 DATA 55                          :REMark America South
1689 DATA 28,50,6, 29,49,5, 31,48,3, 32,47,2, 39,47,2, 42,47,2, 33,46,2
1690 DATA 35,46,1, 38,46,10, 36,45,14, 37,44,13, 37,43,17, 37,42,18
1691 DATA 37,41,18, 36,40,19, 35,39,21, 35,38,20, 56,38,3, 35,37,27, 35,36,28
1692 DATA 35,35,30, 36,34,30, 36,33,30, 37,32,28, 37,31,27, 38,30,26, 38,29,25
1693 DATA 39,28,24, 41,27,22, 42,26,21, 43,25,20, 43,24,19, 43,23,19, 43,22,16
1694 DATA 43,21,15, 43,20,15, 43,19,15, 43,18,14, 43,17,14, 43,16,13, 44,15,12
1695 DATA 44,14,9, 44,13,9, 44,12,7, 44,11,7, 44,10,5, 45,9,5, 46,8,4, 45,7,5
1696 DATA 46,6,5, 46,5,5, 53,4,1, 47,4,3, 47,3,3, 49,2,2, 50,1,4
```

```
1698 DATA 145                    :REMark Asia 145
1699 DATA 134,92,3, 130,91,11, 128,90,17, 146,90,2, 122,89,2, 125,89,26, 155,89,5
1700 DATA 100,88,7, 122,88,45, 172,88,1, 106,87,4, 112,87,1, 115,87,63, 106,86,5
1701 DATA 112,86,71, 106,85,2, 109,85,72, 182,85,3, 106,84,75, 106,83,76, 106,82,66
1702 DATA 175,82,5, 106,81,63, 175,81,2, 175,81,2, 106,80,62, 175,80,3, 106,79,61
1703 DATA 175,79,4, 106,78,61, 176,78,3, 106,77,64, 177,77,2, 106,76,63, 170,76,1
1704 DATA 178,76,1, 106,75,64, 171,75,1, 106,74,65, 172,74,1, 106,73,65, 172,73,1
1705 DATA 106,72,1, 108,72,2, 112,72,6, 120,72,51, 106,71,1, 109,71,1, 111,71,6
1706 DATA 121,71,49, 173,71,1, 113,70,5, 121,70,49, 173,70,3, 108,69,2, 114,69,5
1707 DATA 122,69,46, 173,69,1, 106,68,13, 121,68,47, 175,68,1, 106,67,13, 122,67,40
1708 DATA 166,67,2, 174,67,1, 107,66,14, 122,66,43, 167,66,2, 174,66,2, 112,65,53
1709 DATA 167,65,3, 173,65,3, 112,64,53, 168,64,1, 171,64,4, 111,63,54, 171,63,2
1710 DATA 111,62,54, 171,62,1, 111,61,55, 111,60,9, 121,60,45, 111,59,9, 122,59,44
1711 DATA 112,58,9, 125,58,41, 113,57,9, 124,57,1, 132,57,34, 167,57,1, 113,56,13
1712 DATA 133,56,32, 166,56,2, 114,55,12, 133,55,30, 166,55,1, 114,54,13, 134,54,1
1713 DATA 136,54,9, 148,54,10, 115,53,11, 136,53,8, 149,53,9, 116,52,9, 136,52,7
1714 DATA 150,52,8, 160,52,1, 167,52,1, 116,51,7, 136,51,6, 150,51,9, 167,51,2
1715 DATA 117,50,5, 137,50,4, 152,50,8, 167,50,2, 117,49,3, 137,49,4, 153,49,7
1716 DATA 168,49,2, 138,48,3, 153,48,1, 155,48,6, 168,48,1, 138,47,3, 153,47,1
1717 DATA 155,47,4, 167,47,1, 169,47,1, 139,46,2, 153,46,1, 157,46,2, 166,46,1
1718 DATA 169,46,1,  171,46,1, 139,45,1, 141,45,1, 153,45,1, 169,45,3, 141,44,2
1719 DATA 154,44,2, 170,44,2, 155,43,2, 155,42,2, 156,41,1

1721 DATA 91                     :REMark Australasia
1722 DATA 165,43,2, 152,42,2, 164,42,3, 153,41,2, 163,41,3, 154,40,2, 161,40,6
1723 DATA 168,40,2, 171,40,1, 173,40,1, 154,39,3, 161,39,6, 168,39,1, 170,39,1
1724 DATA 173,39,1,155,38,3, 161,38,5, 167,38,3, 175,38,2, 155,37,4, 162,37,4
1725 DATA 167,37,3, 173,37,1, 176,37,2, 180,37,3, 189,37,1, 156,36,3, 167,36,1
1726 DATA 169,36,1, 177,36,7, 190,36,1, 157,35,2, 167,35,1, 179,35,7, 187,35,2
1727 DATA 191,35,1, 158,34,5, 180,34,6, 191,34,2, 161,33,4, 166,33,1, 168,33,1
1728 DATA 171,33,1, 180,33,4, 185,33,2, 193,33,2, 167,32,1, 170,32,1,186,32,1
1729 DATA 194,32,2, 195,31,1, 175,30,3, 182,30,1, 174,29,4, 181,29,3, 170,28,8
1730 DATA 181,28,3, 169,27,10, 181,27,3, 168,26,16, 166,25,18, 196,25,1, 164,24,21
1731 DATA 196,24,1, 163,23,23, 197,23,1, 162,22,24, 162,21,25, 162,20,25, 162,19,25
1732 DATA 162,18,25, 162,17,24, 162,16,9, 172,16,14, 161,15,6, 173,15,12, 161,14,3
1733 DATA 175,14,8, 176,13,6, 198,13,1, 176,12,6, 198,12,1, 178,11,1, 197,11,3
1734 DATA 196,10,2, 178,9,2, 194,9,1, 177,8,2, 192,8,2, 190,7,3, 188,6,2
:
1736 DATA 52                     :REMark Europe
1737 DATA 100,88,6, 98,87,8, 97,86,9, 96,85,5, 102,85,4, 95,84,5, 101,84,5
1738 DATA 93,83,6, 101,83,5, 93,82,7, 101,82,5,93,81,7, 103,81,3, 96,80,3, 101,80,5
1739 DATA 86,80,1,87,79,1, 94,79,1, 96,79,2, 101,79,5, 85,78,4, 94,78,2, 98,78,8
1740 DATA 84,77,2, 87,77,3, 93,77,13, 84,76,2, 87,76,3, 91,76,15, 87,75,1
1741 DATA 90,75,16, 88,74,18, 87,73,19, 88,72,18, 88,71,18, 84,70,7
1742 DATA 95,70,3, 99,70,7, 84,69,7, 94,69,1, 97,69,1, 100,69,6, 83,68,6
1743 DATA 94,68,1, 98,68,1, 101,68,2, 83,67,6, 98,67,2
1744 DATA 101,67,2, 83,66,5, 97,66,2, 102,66,2

1746 DATA 23                     :REMark Greenland
1747 DATA 64,95,1, 79,95,2, 52,94,2, 56,94,8, 65,94,20, 51,93,2, 55,93,5
1748 DATA 62,93,21, 52,92,5, 60,92,22, 50,91,5, 64,91,17, 65,90,15, 65,89,15
1749 DATA 66,88,13, 65,87,12, 64,86,9, 64,85,6, 79,85,4, 64,84,5, 79,84,3
1750 DATA 64,83,4, 64,82,2
```



Trader Introduction

Back in the day I attended a short course in Business Management at London University UK. It ended being assigned in groups to act out Roles as Company Directors. Our objective, creation of New Product ranges, devise Advertising and Sales campaigns, Manage Factory Production and Deliveries to imaginary customers. Run over three days, we began by raising start-up capital with a Shares issue. The success or failure was dictated by a Stock Market Simulation run over a three-year period, during which our company progress was monitored and judged. As you would expect the process was dictated by the course lecturers, acting sometimes favourably to our requests and enquiries, but more often than not raising problems on our Company Stratagems, Marketing directions and Financial expectations.

QBITS Trader 'Concept'

I began with some research on Stock Market Trader screens and Board Games with a similar theme. First steps taken were in creating a Stock list of Companies and deciding on the display layouts. A Stock Market list, a Portfolio to place Traders selected Stock and a Traders Account showing current financial position, Profits, Credits etc. Then there was how to control the Buying and Selling and Display of Gains and Losses.

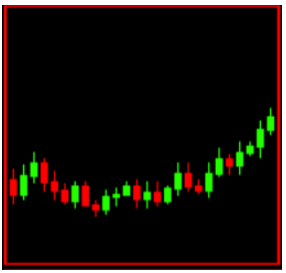
Game design requires developing a background of familiar and repeatedly used functions with random elements that will differ each time the Game is played. Repeated actions help promote the development of stratagems to advance a player's ability to do well. The random elements help to make the Game a more worthwhile Challenge.

QBITS Trader 'Challenge'

The Player is tasked to manage a Stock Market Trader Desk and build a Shares Portfolio, which must be adapted to meet the Price changes of a Simulated Stock Market. The Time limit is a Three-Year period with 10,000 Credits Start-Up Fund. The Score is the accumulating Dividends and Profits made through Buying and Selling Company Shares.

QBITS Trader 'Intro'
Select **Currency (D) (E) (P) (Y)** from the **Intro Screen**, '**£**' is set as the **Default**.

The Trader Screen is displayed and **Market** entries Initialised. The Company Share **Trends** are then Calculated for the first half year (twenty-six weeks). The beginning Phase of the game is to build a **Portfolio** of Company Shares using the opening **10,000 Credits**.



As the game progresses the **Bull** & **Bear** Status show vertically extending bars giving notice to next **End** of **Week Share Price** changes.

Trends are shown using **CandleWick** Graphics.

Scroll the **MARKET** list of Companies with **Up/Down** cursor keys and select with **SpaceBar** to reveal Company **Info**rmation and display Share **Trends** for the preceding twenty-six weeks.

QBITS Trader 'Company Info'
Sym is the abbreviation/index of a Company's Name.

Div (**Dividend)** is a declared % of Face Value (FV) of a Share based on annual Company Profits. A range between 3% to 8% is considered healthy. Irrespective of Market value they are paid out to shareholders quarterly (every 13Wks).

Yld (Yield) is a Rate of Return calculated by subtracting Start value of the investment from its Final value, dividing the Result by the Start value before multiplying by 100.

P/E (**Price/Earnings)** represents the Market value of Stock compared to the Company's earnings. It shows what the market is willing to pay based on Past or Forecast of earnings. This is an indicator to the viability of an Investment.

Open, High, Low, Last are used in analysing a changing **Share Price** and are represented by a **Candle** for the **Open** to **Last** or Closing price range and **Wicks** for the **High** and **Low** of price movement**,** the **Colour** reveals the direction. A **Bull** is shown in **Green** with **Red** for a **Bear**.

Review of patterns formed by the **CandleWic**k graphics can indicate Market Opportunities. They provide insight into the balance between **Buy**ing and **Sell**ing pressures, a steady continuation or Market indecision.

The **Chg** (Change) is the difference between the previous and current weeks **Last Share Price**.

Ask & **Bid** values are calculated from the current weeks **Last Share Price.**

Vol is the Shares Volume being offered.

QBITS Trader 'MARKET'
The Market list is from reviewing Stock Market Lists to create Company Names, Symbols and basic Information. This covered Energy suppliers, Consumer goods, Financial Institutes, Health Care, Industrial Manufactures, Information Technology, Mining/Material Production, Real Estate and Utilities.

QBITS Trader 'Stocks'
Information is held in four Arrays, **CN$(40)** the Company Name, **Stock$(40)** Company **Sym**
Stock(40,6) 1) ??? 2**) Div** 3) **Yld** 4) **P/E** 5) **Price** 6) **Chg**
Trend(40,156,3) 1) **Vol** 2) **Hig**h 3) **Low** 4) **Las**t

```
1365 DEFine PROCedure Init_Stocks
1366 CURSOR#0,76,10:PRINT#0,'Initialising Market':CLS#0,4:RESTORE 1418
1367 FOR a=1 TO 40
1368   READ CN$(a),Stock$(a):FOR b=1 TO 6:READ Stock(a,b):END FOR b
1369   CURSOR#0,184+a*6,10:PRINT#0,'.':PAUSE 1
1370 END FOR a
```

Display of Market Stock

```
1207 DEFine PROCedure MStock(n,my)                    :REMark Market Stock
1208 INK#3,5:CURSOR#3,2,my:PRINT#3,Stock$(n)          :REMark Stock Sym
1209 IF Trend(n,wn,4)>Trend(n,wn-1,4):INK#3,4:INK#7,4:ELSE INK#3,2:INK#7,2
1210 DRJ 3,26,my,2,5,Stock(n,5)+Trend(n,wn,4)         :REMark Stock Last
1211 DRJ 3,60,my,0,3,Trend(n,wn,4)-Trend(n,wn-1,4)    :REMark Stock Chg
1212 INK#7,5:DRJ 3,82,my,0,4,Trend(n,wn,1)            :REMark Stock Vol
1213END DEFine
```

The list has forty entries which are not all shown to screen at the same time so the need to scroll the list Up and Down within its own window space was required. Part of the main program loop includes actions following a key press (**k**) in this case the **Up/D**ow**n** cursors.

```
1018   k=CODE(INKEY$(50)):BLOCK#7,240,10,0,32,0

1022    =208:IF schk=0:mr=mr -1:sn=sn -1:M_Up:ELSE pr=pr-1:IF pr<1:pr=1
1023    =216:IF schk=0:mr=mr+1:sn=sn+1:M_Dn:ELSE pr=pr+1:IF pr>pm:pr=pm
```

HGL highlights the **Sym** of a Company entry and with help of **M_Up** and **M_dn** Scrolls the other entries of the Stock Market list to screen. Added to this the <**Tab**> key switches between **MARKET** and **PORTFOLIO** lists.

```
1030   =  9:IF schk=0:schk=1:ELSE schk=0  [Tab Key]

1051 DEFine PROCedure M_Up
1052 IF mr<1 AND sn>0:SCROLL#3,10:MStock sn,0
1053 IF mr<1:mr=1
1054 IF sn<1:sn=1
1055 END DEFine

1057 DEFine PROCedure M_Dn
1058 IF mr>18 AND sn<41:SCROLL#3,-10:MStock sn,170
1059 IF mr>18:mr=18
1060 IF sn>40:sn=40
1061 END DEFine
```

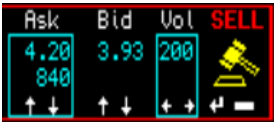
QBITS Trader 'PORTFOLIO'

Up to eight (8) Portfolio entries can be held at any one time. Use, the <**Tab**> key to switch between **Market** and **PorFolio** and cursor keys to Scroll Up/Down to Highlight and Select one of the Entries.

If you **BUY** Shares from the **Market** of a Company already in the **Portfolio** list these will be added to the **Stocks** held and the **Price** will be updated to the new purchase price. The one disadvantage to this is if the new price is lower than the previous it can lower future **Dividend** pay-outs.


```
1119 DEFine PROCedure S_Add(pr,snum)
1120 Asset(pr,1)=sn:Asset(pr,2)=INT(Asset(pr,2)+vol):Asset(pr,3)=cost
1121 sval=sval+snum:sfee=sfee+5+INT(vol/100):cash=cash-snum
1122 Trend(sn,wn,4)=cost-Stock(sn,5):PStock sn,pr:CStock sn
1123 END DEFine
```

If you **SELL** Shares from the **Portfolio** the chosen Volume will be deducted from the Stocks held. If you SELL all of the Stocks held the Company entry will be deleted.




```
1137 DEFine PROCedure S_Del(pr,snum)
1138 Asset(pr,2)=Asset(pr,2)-vol:Asset(pr,3)=cost
1139 sval=sval-snum:sfee=sfee+5+INT(vol/100):cash=cash+snum
1140 Trend(sn,wn,4)=cost-Stock(pn,5):PStock pn,pr:CStock sn
1141 END DEFine
```

Deleting a **PortFolio** entry requires a change to the **Asset** array held information and the screen row of characters to be removed. PROCedure **S_Sort** addresses this requirement by moving held information for the three possible **Asset** list positions, First row Middle rows and Last row. By calculating and setting the CURSOR position and using SCROLL -10 with option 2 the screen area below the cursor moves upward and clears the screen row/entry being deleted.

```
1143 DEFine PROCedure S_Sort(pr,pm)
1144 IF pr<pm
1145   FOR row=pr TO pm-1
1146     FOR c=1 TO 3:Asset(row,c)=Asset(row+1,c)
1147   END FOR row
1148 END IF
1149 FOR c=1 TO 3:Asset(pm,c)=0
1150 IF pr=1:SCROLL#5,-10
1151 IF pr>1:CURSOR#5,0,(pr-2)*10:SCROLL#5,-10,2
1152 pm=pm-1:IF pr>pm:pr=pm
1153 END DEFine
```

QBITS Trader 'Profits & Dividends'

Profits are the difference between the Price when Shares were bought and the Last Share Price and multiplied by the Stock Volumes held. Company Stocks held by the Portfolio who post a div% receive Dividends which are accumulative and pay-out every quarter (13weeks).

QBITS Trader 'Account'

The account details provide a visual status as to how well things are going. As the Game Progresses **Profits** will vary with Current **Share** Price but hopefully climb. The **Dividends** shown are added to the **Credits** and can be used to buy more Shares, potentially more expensive ones.

QBITS Trader 'BUY'

Press '**B**' and use Cursor keys to change the presented '**Bid**' and '**Vol**' values. **Enter** actions your choice, **Spacebar** aborts. Your '**Bid**' may or may not be accepted. It will be rejected if lack of funds '**Credits'.**

QBITS Trader 'Fees'

Each Transaction incurs a '**Fee**', payment of which is deducted from the **Credits**.

QBITS Trader 'SELL'

Once the **PORTFOLIO** has an entry (entries) use <<**Tab**>> to switch between **Market** and **PortFolio** then select with **Up/Dow** Cursor keys to highlight an entry. **Chg** is % difference between **PortFolio Price** shown and **Last Share Price**.

Press '**S**' and use Cursor keys to change the presented '**Ask**' and '**Vol**' values. **Enter** actions your choice, **Spacebar** aborts. Your '**Ask**' Price may or may not be accepted. If not change and try again.

QBITS Trader 'Reuters NEWS Flash'

Once a **PortFolio** has been started as part of the **End** of **Wee**k Share Price a **Reuters NEWS Flash** is displayed. This may change one or more of a Company's Share values held and the calculation of **Share Price** or **Dividends** posted.

QBITS Trader 'Dividends'

The first method of acquiring returns on Stock is to choose those that have **Dividends**. Of the Company Stocks held in the **PortFolio,** those that post a dividend make a pay-out every 13 weeks. To thwart the possibility of buying shares just before a dividend pay-out and reselling to make a quick profit, a Company's **Share Price** is always reduced after a dividend pay-out.

QBITS Trader 'GAME END'

After 156 Weeks the **GAME END**s the **Profits** and **Dividends** are added together to identify **Total Assets**. The Performance Review shows the level of achievement and a Trader Rating.

Press **N** for (**N**)ew Game or **E** to (**E**)xit.

QBITS Trader 'Share Pricing'
The Game opening Initialises the Market entries setting Company Name, Symbol, Dividends, Yields, Price/Earnings. Trends are then Calculated for the first 26 weeks generating a range of changes for Prices and Volume of Shares Offered.

As **QBITS Trader** Game progresses and a **PortFolio** is built, continued changes are made to the **Share Price**, **Volume** and **Dividends**. Future releases may also include changes that reflect **Yield** and **Price/Earnings**.

Altering Stock Prices involves making changes to the integer that represents the Share value and is also used to Calculate the Graphic coordinates that builds the **CandleWick** display. For each Week of Trading the display requires four 'y' coordinate values **Open** (**oy**), **High** (**hy**), **Low** (**ly**) and **Last** (**cy**) close. Open (**oy**) is based on the previous Weeks Last (**cy**) entry.

The array **Trend(sn,wn,1-4)** identifies the Company place in the Stock Market List (1 to 40, the week number (**wn**) and stores integers for ,1) **Vol**ume of Shares ,2) **High** ,3) **Low** ,4) **Las**t.

Trend_Set(wn) generates the **End** of **Week Share Price** changes for all 40 **MARKET** entries.

```
1255 DEFine PROCedure Trend_Set(wn)
1256 CLS#0:CURSOR#0,60,10:PRINT#0,'Calculating Stock Trends':CLS#0,4
1257 IF pm>0:n=Asset(RND(1 TO pm),1):rc=RND(1 TO 8):PRINT#6,CO$(n);' ';RNew$(rc)
1258 FOR a=1 TO 40
1259   CURSOR#0,200+a*6,10:PRINT#0,'.'
1260   Trend(a,wn,1)=100*RND(1 TO 8):ry=RND(2 TO 12):mf=RND(1 TO 2)
1261   IF wn>1:oy=Trend(a,wn-1,4):ELSE oy=84
1262   IF pm>0
1263     IF rc=1 AND a=n:mf=1:ry=12                :REMark Shares ↑
1264     IF rc=2 AND a=n:mf=2:ry=12                :REMark Shares ↓
1265     IF rc=3 AND a=n:Stock(n,2)=RND(8 TO 16)   :REMark Div  ↑
1266     IF rc=4 AND a=n:Stock(n,2)=RND(0 TO 4)    :REMark Div  ↓
1267     IF rc=5 AND a=n:mf=1:ry=6                 :REMark Tax  ↑
1268     IF rc=6 AND a=n:mf=2:ry=6                 :REMark Tax  ↓
1269     IF rc=7:mf=1:ry=8                         :REMark Bull ↑
1270     IF rc=8:mf=2:ry=8                         :REMark Bear ↓
1271   END IF
1272   IF wn MOD 13=0 AND Stock(a,2)>0:mf=2:ry=2+INT(Stock(a,2)/4)
1273   IF mf=1:cy=oy+ry:IF cy>148:cy=oy-RND(2 TO 6)
1274   IF mf=2:cy=oy-ry:IF cy< 20:cy=oy+RND(2 TO 6)
1275   IF cy>oy:hy=cy+RND(2 TO 7):ly=oy-RND(2 TO 7)
1276   IF cy<oy:hy=oy+RND(2 TO 7):ly=cy-RND(2 TO 7)
1277   Trend(a,wn,2)=hy:Trend(a,wn,3)=ly:Trend(a,wn,4)=cy
1278 END FOR a
1279 END DEFine
```

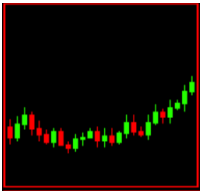
[Variables: **pm** portfolio max number of entries : **rc** row company : **ry** RND y : **mf** +/- math function]

Once the **PortFolio** holds one or more entries at the end of each **Trading Week** there is a selected **Reuters NEWS Flash** of Company held Shares. This adds further elements to the calculation of changes to the **Share Price** and/or **Dividend** % posted pay-outs.

Note: Further Share Price manipulation is not ruled out for Future QBITS Trader Versions.

QBITS Trader 'Stock UpDates'

The **END** of **WEEK** Triggers a new set of Trend Calculations and the following Updates to the Trader **Accounts**, **MARKET**, **PORTFOLIO**, Company **INFO** and **Trend**s displays and a new **Reuters NEWS Flash**.



AStock	Account Info	[Variables **sval sdiv sfee sinv cash**]
PStock **n**, **pm**	Portfolio Info	[Variables **wn py**]
MStock **n**, **my**	Market Info	[Variables **wn**]
CStock **n**	Company Info	[Variables **str$ ops ols nws wk1 wk2**]
STrend **n en**	CandleWicks	[Variables **n en**]

QBITS Trader 'Trend Analysis'

The Buying and Selling Price of Shares varies over a period of time and depends on a number of factors. This essentially comes down to marketable goods or services and maintaining Market Share against competitors.

The volume of turnover can indicate the stability of a company to weather any sudden market depressions. It may also be trying to maintain Market Share and lead to falling profits. It may depend on materials or parts supplied from other suppliers. Other disrupting factors can be the weather, industrial disputes, interest rate changes, government taxes, new restrictions on certain build materials or product use, border control disputes, or military conflicts.

QBITS Trader 'Trading Strategy'

This is summarising Trends with movement displayed as Candle & Wicks. Small **CandleWicks** indicate little price movement and represent consolidation. Large **CandleWicks** show strong buying or selling pressure. These identify the highs and lows of the Trading Share Price in **Bull** and **Bear** Markets.

The Time to **Buy:** a **GREEN CandleWick** showing a lower opening price rising to a high and falling back to close above the opening price. The time to Sell a **RED** CandleWick with wick pointing upwards indicates an opening price rising and then falling to a lower closing price.

Trading Volumes, Price Movements, should be viewed with a Company's strengths and weaknesses, derived from the Dividends, Yields and Price/Earnings.

At the end of the three years (**156 weeks**) the **Game Ends**. Results will reveal if you have made any significant gains. This will be the combination of Dividends and Profits. For the best rewards divide Stock into those with high Dividend pay-outs and those that add Quick Profits by Buying and Selling at the right time.

QBITS Trader 'Strings'

In some ways this took me back to my early days with QL SuperBASIC, printing character strings to screen. The CSIZE command gives scope to creating differing character sets and with the use of OVER can generate the appearance of **Bold** characters with a horizontal pixel offset and by a vertical offset produce a **3D Style**. **Title** used with CSIZE 2,1 is adopted for QBITS Main Title, and with CSIZE 1,0 for Section Headings.

Title used for headings, Windows channel id **ch**, Character INK **col**, CSIZE **w**, **d**, **t** for the offset, Pixel coordinates **tx, ty**, and **str$** to hold character string.

```
1373 Title 2,2,2,1,1,134,2,'QBITS TRADER' x-offset : Title 2,6,2,1,1,136,0,'QBITS TRADER' repeated with y-offset
```

```
1405 DEFine PROCedure Title(ch,col,w,d,t,tx,ty,str$)
1406 CSIZE#ch,w,d:OVER#ch,1:INK#ch,col
1407 FOR i=0 TO t:CURSOR#ch,tx+i,ty:PRINT#ch,str$
1408 CSIZE#ch,0,0:OVER#ch,0:INK#ch,7
1409 END DEFine
```


```
Title 2,5,1,0,2,32,4,'MARKET'
Title 2,5,1,0,1,382,4,'PORTFOLIO'
```

Using **Title** for Characters with CSIZE 0,0 merges the characters and a re-CSIZE within program loops can affect following PRINT lines.

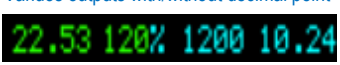
Title **TBold**


```
1172 DEFine PROCedure TBold(ch,col,cs,cx,cy,str$)          TBold adds pixel spacing between Characters
1173 INK#ch,col:OVER#ch,1
1174 FOR a=1 TO LEN(str$)
1175    FOR b=0 TO cs:CURSOR#ch,cx+b+a*(6+cs),cy:PRINT#ch,str$(a)
1176 END FOR a:OVER#ch,0
1177 END DEFine
```

QBITS Trader 'Currency'

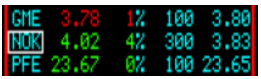
For Currency & Numbers you need to **Right Justify,** add a **decimal point** (**dp**) possibly and apply **spaces** (**sp**). QL SuperBASIC allows coercion of unsuitable data to a type that will allow a specified operation to proceed. Numeric character strings convert to floating point integers and vice versa. This allows the num variable in **DRJ** to accept an integer or numeric string.

```
1179 DEFine PROCedure DRJ(ch,cx,cy,dp,sp,num)              Decimal Right Justified
1180 str$=ABS(INT(num)):sl=LEN(str$)
1181 IF dp>0 AND sl>dp:str$=str$(1 TO sl-2)&'.'&str$(sl-1 TO sl)   Variuos outputs with/without decimal point
1182 IF dp>0 AND sl=dp:str$='.'&str$
1183 IF dp>0 AND sl<dp:str$='.0'&str$
1184 CURSOR#ch,cx,cy:PRINT#ch,FILL$(' ',sp-LEN(str$))&str$
1185 END DEFine
```



QBITS Trader 'Highlight'

This uses LINE and the Graphics coordinates system to draw a box. The SCALE is made the same as the Windows vertical pixel size.




```
1047 DEFine PROCedure HGL(ch,w,d,x,y)                      HiGhLight
1048 OVER#ch,-1:LINE#ch,x,y TO x,y+d TO x+w,y+d TO x+w,y TO x,y:OVER#ch,0
10495 END DEFine
```

This is process is also used to highlight the **Ask**, **Bid** and **Vol** areas.

HGL 7,12,15,88.5,0 : HGL 7,12,15,102,0 : HGL 7,7.4,15,116.4,0`

QBITS Trader PROCedures

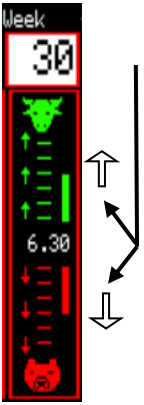
Intro	Opening Introduction to QBTS Trader and Selection of Currency
Init_Trader	Opens windows and sets Trader screen layout
Title	Presents BOLD and 3D Character strings
Init_RNews	The NEWS statements used in the Game
Init _Stocks	Set up the Games Stock Market
QBITS _Trader	Main Program Loop
THelp	Help on Key usage
HGL	**H**i**G**h**L**ights to Aid Game Play
M_Up	**M**arket Scroll **Up**
M_Dn	**M**arket Scroll **D**ow**n**
SMWeek	Print **S**tock **M**arket **Week** Number
Game_End	Review of Assets & Performance
S_Buy	BUY Shares
S_Add	Add Shares to Portfolio
S_Sell	SELL Shares
S_Del	Delete Shares in Portfolio
S_Sort	Sort Portfolio List
S_Vol	Display Change Ask Bid, Volume and manual changes
TBold	Display Bold Charters
DRJ	Draw Right Justified + decimal point
AStock	Display Account Changes
PStock	Display PortFolio Stock entries
MStock	Display Market Stock entries
CStock	Display Company Info of Stock selected
STrend	Display 26 weeks of Price Changes of Company selected
Wick	Display Price changes in Candle Wick format
Trend_Set	Calculate Week changes for the Market Stock
Graphics	
TBuy	Display Buy icon and heading
TSell	Display SELL icon and heading
Market	Market icon
Asset	Asset icon
Chart	Simple Bull/Bear Chart
Money	Coins Display
Trader	Head & shoulders of Trader with headphones
Bear	Bear view full face
Bull	Bull view full face
Pillar	Display Fluted Column

`

QBITS Trader Code

```
1000 REMark QBITS_Trader_v3 (QBITS Trader v3 2021 QPC2)

1002 OPEN_IN#9,'ram2_QBITSConfig':INPUT#9,gx\gy\dn$:CLOSE#9
1003 Tdel=40                              :REMark Time delay for Week End 40 sec
1004 DIM CN$(40,20),Stock$(40,3),Stock(40,6),RNew$(8,40),Trend(40,156,4)

1006 MODE 4:Intro_Trader:Init_Trader:Init_Stocks:Init_RNews:QBITS_Trader

1008 DEFine PROCedure QBITS_Trader
1009 DIM Asset(8,4),Audit(130)
1010 pm=0:FOR wn=1 TO 26:DRJ 8,0,0,0,3,wn:Trend_Set wn:PAUSE 1
1011 sn= 1:FOR mr=1 TO 18:MStock mr,(mr*10)-10
1012 sn=1:mr=1:pm=0:pr=1:pn=1:n=1:cash=10000:sval=0:sdiv=0:sfee=0:sinv=0
1013 schk=0:tnck=0:TMck=0:AStock:THelp:GTm=DATE+Tdel
1014 REPeat Loop
1015   SMWeeK:IF Asset(pr,1)=0:schk=0
1016   IF schk=0:INK#3,7:HGL 3,16,9,0,181-mr*10:n=sn
1017   IF schk=1:INK#5,7:HGL 5,16,9,0, 81-pr*10  :n=Asset(pr,1)
1018   k=CODE(INKEY$(50)):BLOCK#7,240,10,0,32,0
1019   IF schk=0:HGL 3,16,9,0,181-mr*10
1020   IF schk=1:HGL 5,16,9,0, 81-pr*10
1021   SELect ON k
1022     =208:IF schk=0:mr=mr-1:sn=sn-1:M_Up:ELSE pr=pr-1:IF pr<1:pr=1
1023     =216:IF schk=0:mr=mr+1:sn=sn+1:M_Dn:ELSE pr=pr+1:IF pr>pm:pr=pm
1024     =66, 98:IF schk=0:S_Buy  sn:AStock:THelp          :REMark (B)UY
1025     =83,115:IF schk=1:S_Sell pr:AStock:THelp          :REMark (S)ELL
1026     =69,101:GExit:BLOCK#0,20,10,400,14,0              :REMark (E)xit
1027     =78,110:CLS#5:CLS#1:CLS#0:QBITS_Trader            :REMark (N)ew Game
1028     =84,116:TReview                                   :REMark (T)ader Review
1029     = 32:CStock n:STrend n,wn                         :REMark View Info / Trends
1030     =  9:IF schk=0:schk=1:ELSE schk=0                 :REMark Tab Stock<>Portfolio
1031     =61:IF TMck=0:TMck=1:ELSE TMck=0                  :REMark TestMode On/Off
1032     =232,236,240,244,248:IF TMck=1:TestMode           :REMark Test F1/F2/F3/F4/F5
1033   END SELect
1034 END REPeat Loop
1035 END DEFine

1037 DEFine PROCedure GExit
1038 CURSOR#0,400,14:PRINT#0,'Y/N':PAUSE:IF KEYROW(5)=64:LRUN dn$
1039 END DEFine

1041 DEFine PROCedure THelp
1042 CLS#0:CURSOR#0,112,4:PRINT#0,'(B)UY : MARKET <<Tab>> PORTFOLIO : (S)ELL'
1043 CURSOR#0,76,14:PRINT#0,'(N)ew Set Price ↑↓ & Volume ←→ (Rtn  or  Act) (E)xit'
1044 CURSOR#0,318,14:PRINT#0,'← ':BLOCK#0,2,4,324,16,5:BLOCK#0,12,3,290,18,5
1045 END DEFine

1047 DEFine PROCedure HGL(ch,w,d,x,y)             :REMark HiGhLight
1048 OVER#ch,-1:LINE#ch,x,y TO x,y+d TO x+w,y+d TO x+w,y TO x,y:OVER#ch,0
1049 END DEFine
```

```
1051 DEFine PROCedure M_Up
1052 IF mr<1 AND sn>0:SCROLL#3,10:MStock sn,0
1053 IF mr<1:mr=1
1054 IF sn<1:sn=1
1055 END DEFine

1057 DEFine PROCedure M_Dn
1058 IF mr>18 AND sn<41:SCROLL#3,-10:MStock sn,170
1059 IF mr>18:mr=18
1060 IF sn>40:sn=40
1061 END DEFine
```




```
1063 DEFine PROCedure SMWeeK
1064 del=GTm-DATE:IF del<0:del=0
1065 BLOCK#4,4,42-del,28,10+del,4:BLOCK#4,4,42-del,28,70,2
1066 IF GTm<DATE:BLOCK#4,4,48,28,4,0:BLOCK#4,4,48,28,68,0
1067 IF GTm<DATE:Stock_UpDate wn:GTm=DATE+Tdel
1068 END DEFine
```


```
1070 DEFine PROCedure Stock_UpDate(wn)
1071 IF wn<156:wn=wn+1:ELSE TReview:RETurn
1072 Trend_Set wn:CStock n :STrend n,wn:sval=0:snum=0:dnum=0
1073 FOR rs=18 TO 1 STEP -1:MStock (sn-mr)+rs,(rs*10)-10
1074 FOR rp=1 TO pm
1075    rn=Asset(rp,1):pdiv=Stock(Asset(rp,1),2)
1076    snum=INT(Asset(rp,2)*(Stock(rn,5)+Trend(rn,wn,4))/100)
1077    IF wn MOD 13=0 AND pdiv>0
1078       dnum=INT(snum*pdiv/400):sdiv=sdiv+dnum:cash=cash+dnum
1079    END IF
1080    sval=sval+snum:PStock Asset(rp,1),rp
1081 END FOR rp
1082 AStock:PAUSE 10:CLS#0:THelp
1083 END DEFine
```

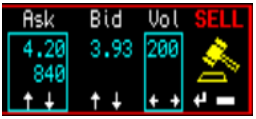
```
1085 DEFine PROCedure TReview
1086 IF pm=0:RETurn :ELSE CLS#1
1087 Title 1,7,0,0,0,36,38,'Performance Review':Audit(0)=10000
1088 INK 7:LINE 18,20 TO 18,90:INK 248:LINE 18,40 TO 160,40:INK 4
1089 FOR i=1 TO wn-26
1090  x1=23+i:x2=24+i:y1=Audit(i-1)/250:y2=Audit(i)/250:LINE x1,y1 TO x2,y2
1091 END FOR i
1092 INK 5:CURSOR 32,110:PRINT 'SpaceBar to Return'
1093 IF wn=156
1094    CURSOR 0,0:Title 1,7,2,1,1,42,10,'GAME END':BLOCK 120,10,32,110,0
1095    score=Audit(130)-10000:cx=48:str$='Your Fired'
1096    IF score> 500:cx=48:str$='Survivable'
1097    IF score>1000:cx=48:str$='Acceptable'
1098    IF score>2500:cx=58:str$='Average'
1099    IF score>4000:cx=58:str$='Skilful'
1100    IF score>8000:cx=60:str$='Expert'
1101    TBold 1,7,1,cx,108,str$
1102 END IF
1103 END DEFine
```


```
1105 DEFine PROCedure S_Buy(sn)
1106 TBuy 7,131,11:TBold 7,4,2,344,0,'BUY':CStock sn:STrend sn,wn
1107 vn=Trend(sn,wn,1):vm=vn:cn=Stock(sn,5)+Trend(sn,wn,4)
1108 S_Vol 102.5,286,2,5,vn,vm,cn:IF k=32:RETurn
1109 IF cost<Stock(sn,5)+Trend(sn,wn,4)-RND(6 TO 12)
1110   CURSOR#7,2,32:PRINT #7,'Unsuccessful Bid - Try Again':PAUSE 20:RETurn
1111 END IF
1112 snum=INT(vol*cost)/100            :REMark cost=Bid
1113 IF snum+20>cash:CURSOR#7,2,32:PRINT#7,'Insufficient Funds':PAUSE 20:RETurn
1114 FOR pr=1 TO pm:IF Asset(pr,1)=sn:S_Add pr,snum:RETurn
1115 IF pm<8:pm=pm+1:S_Add pm,snum:RETurn
1116 CURSOR#7,2,32:PRINT#7,'Sell Some Shares'
1117 END DEFine
```

```
1119 DEFine PROCedure S_Add(pr,snum)
1120 Asset(pr,1)=sn:Asset(pr,2)=INT(Asset(pr,2)+vol):Asset(pr,3)=cost
1121 sval=sval+snum:sfee=sfee+5+INT(vol/100):cash=cash-snum
1122 Trend(sn,wn,4)=cost-Stock(sn,5):PStock sn,pr:CStock sn
1123 END DEFine
```

QBITS **BUY**ing Shares:

Check 1 If Insufficient Funds, Bid too Low or No Portfolio Slots: Rejected.
Check 2 If Company Already in Portfolio Add Shares to Existing Entry.
Check 3 If Company Not in Portfolio and a Free slot, add as New Entry




```
1125 DEFine PROCedure S_Sell(pr)
1126 pn=Asset(pr,1):IF pn<=0:RETurn
1127 TSell 7,133,10:TBold 7,2,2,340,0,'SELL'
1128 STrend pn,wn:CStock pn:vm=Asset(pr,2):IF vm>900:vm=900
1129 vn=vm:S_Vol 88.5,248,2,5,vn,vm,Stock(pn,5)+Trend(pn,wn,4):IF k=32:RETurn
1130 IF cost>Stock(pn,5)+Trend(pn,wn,4)+RND(0 TO 12)
1131   CURSOR#7,0,32:PRINT#7,'Unsuccesful Sale - Try Again':PAUSE 20:RETurn
1132 END IF
1133 snum=INT(vol*cost)/100            :REMark cost=Ask
1134 S_Del pr,snum:IF Asset(pr,2)=0:S_Sort pr,pm
1135 END DEFine
```

```
1137 DEFine PROCedure S_Del(pr,snum)
1138 Asset(pr,2)=Asset(pr,2)-vol:Asset(pr,3)=cost
1139 sval=sval-snum:sfee=sfee+5+INT(vol/100):cash=cash+snum
1140 Trend(sn,wn,4)=cost-Stock(pn,5):PStock pn,pr:CStock sn
1141 END DEFine
```

QBITS **SELL**ing Shares:

Check 1 If Company shares sold and Stock reduced to zero Delete slot.

```
1143 DEFine PROCedure S_Sort(pr,pm)
1144 IF pr<pm
1145   FOR row=pr TO pm-1
1146     FOR c=1 TO 3:Asset(row,c)=Asset(row+1,c)
1147   END FOR row
1148 END IF
1149 FOR c=1 TO 3:Asset(pm,c)=0
1150 IF pr=1:SCROLL#5,-10
1151 IF pr>1:CURSOR#5,0,(pr-2)*10:SCROLL#5,-10,2
1152 pm=pm-1:IF pr>pm:pr=pm
1153 END DEFine
```

```
1155 DEFine PROCedure S_Vol(x,cx,dp,sp,vn,vm,cn)
1156 INK#7,5:HGL 7,12,15, x,0:HGL 7,7.4,15,116.4,0
1157 REPeat Vol_lp
1158  DRJ 7,324,12,0,3,vn:DRJ 7,cx,12,2,5,cn
1159  DRJ 7,cx,22,0,5,INT((cn*vn)/100)
1160  k=CODE(INKEY$(-1))
1161  IF k=192:vn=vn -10:IF vn<  10:vn=  10
1162  IF k=200:vn=vn+10:IF vn>  vm:vn=  vm
1163  IF k=208:cn=cn+  1:IF cn>9990:cn=9990
1164  IF k=216:cn=cn -  1:IF cn<  1:cn=  1
1165  IF k= 32:EXIT Vol_lp
1166  IF k= 10:vol=vn:cost=cn:EXIT Vol_lp
1167 END REPeat Vol_lp
1168 INK#7,5:HGL 7,12,15,x,0:HGL 7,7.4,15,116.4,0
1169 BLOCK#7,30,12,cx,22,0:BLOCK#7,34,32,348,0,0
1170 END DEFine
```

:REMark Volume of Shares

:REMark Ask/Bid Price


```
1172 DEFine PROCedure TBold(ch,col,cs,cx,cy,str$)
1173 INK#ch,col:OVER#ch,1
1174 FOR a=1 TO LEN(str$)
1175   FOR b=0 TO cs:CURSOR#ch,cx+b+a*(6+cs),cy:PRINT#ch,str$(a)
1176 END FOR a:OVER#ch,0
1177 END DEFine
```

```
1179 DEFine PROCedure DRJ(ch,cx,cy,dp,sp,num)
1180 str$=ABS(INT(num)):sl=LEN(str$)
1181 IF dp>0 AND sl>dp:str$=str$(1 TO sl-2)&'.'&str$(sl-1 TO sl)
1182 IF dp>0 AND sl=dp:str$='.'&str$
1183 IF dp>0 AND sl<dp:str$='.0'&str$
1184 CURSOR#ch,cx,cy:PRINT#ch,FILL$(' ',sp-LEN(str$))&str$
1185 END DEFine
```

:REMark Decimal Right Justified

Various outputs with/without decimal point

QBITS Trader 'Summary'

Name	Company Name	1-20 Capital Letters
Sym	Symbol	1-3 Capital Letters identifying the Company
Iss	Issue	Not Used : IPO Initial Public Shares Offer
Div	Dividend	Yearly return as a % paid on each share.
Yld	Yeild	Annual Dividend divided by current stock price
P/E	Price/Earnings	Ratio between stock Price and Company's Earning
Open	Price	Opening Share Price
High	Price	Trade High between Open - Close
Low	Price	Trade Low between Open - Close
Last	Last Price	Stock price at end of Trading Week
Chg	Net Change	Change between previous Last and current Last.
Ask	Price	Share Price Requested
Bid	Price	Share Price Offered
Vol	Volume	The number of Shares Offered for Exchange.

QBITS Trader 'Reuter NEWS Flash'

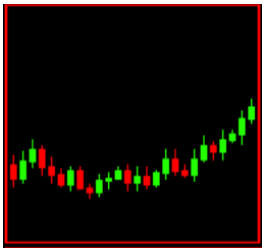
Announcements with potential to affect Share Price or Dividends.

```
1187 DEFine PROCedure AStock                :REMark Account Stock Info
1188 DRJ 2,428,122,0,5,sval                 :REMark Share Value    1-8 Int((stock*last)/100)
1189 DRJ 2,428,132,0,5,sdiv                 :REMark WK13  Dividend 1-8 Int((sval*sdiv)/400)
1190 DRJ 2,428,142,0,5,sfee                 :REMark Fees  each Transaction vol/100
1191 DRJ 2,428,152,0,5,cash+sdiv-sfee       :REMark Cash+Dividens-Fees
1192 Audit(wn-26)=sval+cash+sdiv-sfee:sinv=(cash+sval)-10000
1193 IF sinv<0:INK#2,2:ELSE INK#2,4
1194 DRJ 2,428,112,0,5,sinv:INK#2,5
1195 END DEFine

1197 DEFine PROCedure PStock(n,pr)                              :REMark Portfolio Stock Info
1198 INK#5,5:py=(pr-1)*10:CURSOR#5,2,py:PRINT#5,Stock$(n)       :REMark Stock Sym
1199 sChg=Stock(n,5)+Trend(n,wn,4)-Asset(pr,3):CURSOR#5,78,py:PRINT#5,'%'
1200 IF sChg<0:INK#5,2:ELSE INK#5,4
1201 DRJ 5,26,py,2,5,Stock(n,5)+Trend(n,wn,4)       :REMark Last
1202 DRJ 5,60,py,0,3,(sChg/Asset(pr,3))*100:INK#5,5 :REMark Chg
1203 DRJ 5,90,py,0,4,Asset(pr,2)                    :REMark Stock
1204 DRJ 5,119,py,2,5,Asset(pr,3)                   :REMark Price
1205 END DEFine

1207 DEFine PROCedure MStock(n,my)                  :REMark Market Stock  Info
1208 INK#3,5:CURSOR#3,2,my:PRINT#3,Stock$(n)        :REMark Stock Sym
1209 IF Trend(n,wn,4)>Trend(n,wn-1,4):INK#3,4:INK#7,4:ELSE INK#3,2:INK#7,2
1210 DRJ 3,26,my,2,5,Stock(n,5)+Trend(n,wn,4)       :REMark Stock Last
1211 DRJ 3,60,my,0,3,Trend(n,wn,4)-Trend(n,wn-1,4)  :REMark Stock Chg
1212 INK#7,5:DRJ 3,82,my,0,4,Trend(n,wn,1)          :REMark Stock Vol
1213 END DEFine


1215 DEFine PROCedure CStock(n)                     :REMark Company Stock Info
1216 INK#7,5:CURSOR#7,0,12:PRINT#7,Stock$(n);'  %'  :REMark Sym
1217 REMark DRJ 7, 24,10,0,4,Stock(n,1)             :REMark Unused
1218 DRJ 7, 24,12,0,2,Stock(n,2)                    :REMark Dividend
1219 DRJ 7, 52,12,0,2,Stock(n,3)                    :REMark Yield
1220 DRJ 7, 74,12,0,2,Stock(n,4)                    :REMark Price/Expense
1221 ops=Stock(n,5):ols=Trend(n,wn-1,4):nws=Trend(n,wn,4)
1222 DRJ 7,92,12,2,5,ops+ols                       :REMark Open
1223 DRJ 7,125,12,2,5,ops+ols+Trend(n,wn,2)        :REMark High
1224 DRJ 7,158,12,2,5,ops-(Trend(n,wn,3)-ols)      :REMark Low
1225 IF nws>ols:INK#3,4:INK#7,4:ELSE INK#3,2:INK#7,2
1226 DRJ 7,191,12,2,5,ops+nws                      :REMark Last
1227 DRJ 7,224,12,0,3,nws-ols  :INK#7,5            :REMark Change
1228 DRJ 7,324,12,0,3,Trend(n,wn,1)                :REMark Volume
1229 DRJ 7,286,12,2,5,ops+nws-RND(10 TO 20)        :REMark Bid
1230 DRJ 7,248,12,2,5,ops+nws+RND(10 TO 40)        :REMark Ask
1231 wk1=Trend(n,wn-13,4)*Stock(n,4)*Stock(n,3)/100
1232 wk2=Trend(n,wn,4)*Stock(n,4)*Stock(n,3)/100
1233 IF wk1>wk2:str$='BUY ':ELSE str$='SELL'       :REMark 13wk Trend
1234 CURSOR#7,122,22:PRINT#7,str$
1235 END DEFine
```

Stock Trends:

This monitors the Stock Share movement over a period of time, displays 26 Weeks.

```
1237 DEFine PROCedure STrend(n,en)                    :REMark Stock Trend
1238 CURSOR#2,178,28:PRINT#2,FILL$(' ',20-LEN(CN$(n)))&CN$(n)
1239 INK#4,7:DRJ 4,2,56,2,5,Stock(n,5)+Trend(n,en,4)
1240 DRJ 8,0,0,0,3,en:CLS:INK#4,7:cw=1:CLS
1241 FOR i=en-25 TO en
1242   IF i>1:oy=Trend(n,i-1,4):ELSE oy=84
1243   hy=Trend(n,i,2):ly=Trend(n,i,3):cy=Trend(n,i,4)
1244   IF cy>oy:col=4:y1=cy:y2=oy:ELSE col=2:y1=oy:y2=cy
1245   Wick 1,col,177,hy,ly,y1,y2
1246 END FOR i
1247 IF y1>84:SCROLL#1,(y1-84)*.6:ELSE SCROLL#1,-(84-y1)*.6
1248 END DEFine
```



```
1250 DEFine PROCedure Wick(ch,col,wx,hy,ly,y1,y2)
1251 PAN-7:x1=wx-2:x2=wx+2:INK#ch,col:LINE#ch,wx,hy TO wx,ly
1252 FILL#ch,1:LINE#ch,x1,y1 TO x2,y1 TO x2,y2 TO x1,y2 TO x1,y1:FILL#ch,0
1253 END DEFine
```

```
1255 DEFine PROCedure Trend_Set(wn)
1256 CLS#0:CURSOR#0,60,10:PRINT#0,'Calculating Stock Trends':CLS#0,4
1257 IF pm>0:n=Asset(RND(1 TO pm),1):rk=RND(1 TO 8):PRINT#6,CN$(n);' ';RNew$(rk)
1258 FOR a=1 TO 40
1259   CURSOR#0,200+a*6,10:PRINT#0,'.'
1260   Trend(a,wn,1)=100*RND(1 TO 8):ry=RND(2 TO 12):mf=RND(1 TO 2)
1261   IF wn>1:oy=Trend(a,wn-1,4):ELSE oy=84
1262   IF pm>0
1262     IF rk=1 AND a=n:mf=1:ry=12                :REMark Shares ↑
1263     IF rk=2 AND a=n:mf=2:ry=12                :REMark Shares ↓
1264     IF rk=3 AND a=n:Stock(n,2)=RND(8 TO 16)   :REMark Div  ↑
1265     IF rk=4 AND a=n:Stock(n,2)=RND(0 TO 4)    :REMark Div  ↓
1266     IF rk=5 AND a=n:mf=1:ry=6                 :REMark Tax  ↑
1267     IF rk=6 AND a=n:mf=2:ry=6                 :REMark Tax  ↓
1268     IF rk=7:mf=1:ry=8                         :REMark Bull ↑
1269     IF rk=8:mf=2:ry=8                         :REMark Bear ↓
1271   END IF
1272   IF wn MOD 13=0 AND Stock(a,2)>0:mf=2:ry=2+INT(Stock(a,2)/4)
1273   IF mf=1:cy=oy+ry:IF cy>148:cy=oy-RND(2 TO 6)
1274   IF mf=2:cy=oy-ry:IF cy< 20:cy=oy+RND(2 TO 6)
1275   IF cy>oy:hy=cy+RND(2 TO 7):ly=oy-RND(2 TO 7)
1276   IF cy<oy:hy=oy+RND(2 TO 7):ly=cy-RND(2 TO 7)
1277   Trend(a,wn,2)=hy:Trend(a,wn,3)=ly:Trend(a,wn,4)=cy
1278 END FOR a
1279 END DEFine
```

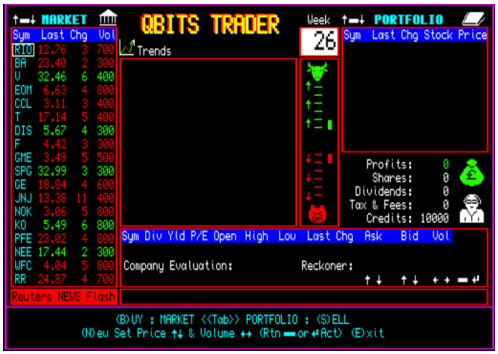
1281 REMark **Intro Screen**

1283 **DEFine PROCedure Intro_Trader**
1284 OPEN#9,scr_:WINDOW#9,512,256,gx,gy:PAPER#9,0:BORDER#9,1,3:CLS#9
1285 BLOCK#9,508,1,0,222,3:SCALE#9,100,0,0
1286 **Title 9,2,3,1,3,155,19,'QBITS TRADER':Title 9,6,3,1,2,158,20,'QBITS TRADER'**
1287 **Market 9,36,88:Chart 9,60,48:Money 9,86,48,``:Assets 9,112,88**
1288 **Bull 9, 36,46:Title 9,4,3,1,1, 90,144,'↑':Title 9,4,1,0,2,110,156,'BULL'**
1289 **Bear 9,112,46:Title 9,2,3,1,1,400,144,'↓':Title 9,2,1,0,2,364,156,'BEAR'**
1290 INK#9,6:**Pillar 9,12,60:Pillar 9,134,60**:INK#9,5
1291 RESTORE 1292:FOR i=1 TO 7:READ str$:CURSOR#9,98,40+i*10:PRINT#9,str$
1292 DATA "Stock Market PortFolio"," "
1293 DATA "The Game runs a Simulated Stock Market Trader's Desk"
1294 DATA "where you Manage a Portfolio of Shares for 126 weeks"
1295 DATA "of a three year period. Check Share Gains and Losses"
1296 DATA " Increase your Investment through Company Dividends"
1297 DATA " or by simply Buying and Selling Shares."
1298 **TBuy 9,58,34:Trader 9,74,40:TSell 9,92,34**:CURSOR#9,98,190
1299 INK#9,3:PRINT#9,"Select Currency: $( )ollar µ( )uro `( )ound ž( )en"
1300 **RESTORE 1301**:INK#9,7:FOR i=1 TO 4:**READ x,c$**:CURSOR#9,x,190:PRINT#9,c$
1301 DATA 224,'D',284,'E',332,'P',386,'Y'
1302 k=CODE(INKEY$(-1)):c$=``:CLS#9
1303 **SELect ON k=68,100**:c$='$' :REMark Shift\4
1304 **SELect ON k=69,101**:c$='µ' :REMark Ctrl\Shift\U or Ctrl\Shift\K ?
1305 **SELect ON k=80,112**:c$=`` :REMark Shift\3
1306 **SELect ON k=89,121**:c$='ž' :REMark Ctrl\shift\.
1307 **END DEFine**

Note: Select currency [D] [E] [P] [Y]

Any other key will set Default Currency c$ = £

1309 REMark **Trader Desk**



```
1311 DEFine PROCedure Init_Trader
1312 WINDOW#0,512,32,gx,gy+224       :PAPER#0,0:INK#0,5:BORDER#0,1,3
1313 WINDOW#1,186,125,120+gx,40+gy:PAPER#1,0:INK#1,7:BORDER#1,1,2
1314 WINDOW#2,500,222,gx+6,gy+2     :PAPER#2,0:INK#2,7:RESTORE 1317
1315 FOR i=3 TO 8:OPEN#i,scr_:READ a,b,c,d:WINDOW#i,a,b,c+gx,d+gy:BORDER#i,1,2
1316 INK#6,7:SCALE#4,80,0,0:SCALE#2,100,0,0:SCALE#1,164,0,0
1317 DATA 112,193,  6, 17,  42,125,308, 40, 154,93,352,17   :REMark Win 3/4/5
1318 DATA 386, 12,120,211, 386, 44,120,166,42,22,308,17   :REMark win 6/7/8
1319 Title 2,2,2,1,1,134,3,'QBITS TRADER':Title 2,6,2,1,1,136,4,'QBITS TRADER'
1320 PAPER#8,7:INK#8,0:CSIZE#8,2,1:CLS#8
1321 BLOCK#7,346,10,0,0,1:INK#7,7 :SCALE#7,20,0,0:OVER#7,1
1322 RESTORE 1323:FOR i=1 TO 16:READ x,y,str$:CURSOR#7,x,y:PRINT#7,str$
1323 DATA 0,0,'Sym',24,0,'Div',48,0,'Yld',72,0,'P/E',96,0,'Open',129,0,'High'
1324 DATA 164,0,'Low',194,0,'Last',224,0,'Chg',254,0,'Ask',292,0,'Bid'
1325 DATA 325,0,'Vol',254,32,' 🡅  🡇',292,32,' 🡅  🡇',325,32,'🡄  🡆',366,32,' 🡄'
1326 BLOCK#7,2,4,372,34,7:BLOCK#7,10,3,352,36,7  :OVER#7,0
1327 INK#7,7:CURSOR#7,2,22:PRINT#7,'Company Evaluation:          Reckoner:'
1328 Chart 2,38,89:Trader 2,161,34:Money 2,161,48,c$:BLOCK#9,112,12,4,210,2
1329 STRIP#9,2:INK#9,0:CURSOR#9,5,211:PRINT#9,'Reuters NEWS Flash':STRIP#9,0
1330 Title 2,5,1,0,2,380,5,'PORTFOLIO':Assets 2,162,99
1331 INK#5,7:STRIP#5,1:PRINT#5,'Sym  Last Chg Stock Price':STRIP#5,0
1332 WINDOW#5,150,80,354+gx,29+gy:SCALE#5,80,0,0
1333 INK#4,4:FOR i=1 TO 3:CURSOR#4,4,i*14:PRINT#4,'🡅 '
1334 Bull 4,8,76:FOR i=1 TO 6:BLOCK#4,8,1,14,14+i*6,4
1335 INK#4,2:FOR i=1 TO 3:CURSOR#4,4,54+i*14:PRINT#4,' 🡇'
1336 Bear 4,8,7:FOR i=1 TO 6:BLOCK#4,8,1,14,64+i*6,2
1337 Title 2,5,1,0,2,32,5,'MARKET':Market 2,34,99
1338 INK#3,7:STRIP#3,1:PRINT#3,'Sym  Last Chg  Vol':STRIP#3,0
1339 WINDOW#3,108,180,8+gx,29+gy:SCALE#3,180,0,0
1340 RESTORE 1341:FOR i=1 TO 9:READ x,y,str$:CURSOR#2,x,y:PRINT#2,str$
1341 DATA 2,4,' 🡅  🡇',346,4,' 🡅  🡇',132,28,'Trends',310,5,'Week'
1342 DATA 378,122,'Shares:',360,132,'Dividends:',354,142,'Tax & Fees:'
1343 DATA 372,112,'Profits:',372,152,'Credits:'
1344 BLOCK#2,10,3,10,9,7:BLOCK#2,10,3,354,9,7
1345 END DEFine

1347 DEFine PROCedure Title(ch,col,w,d,t,tx,ty,str$)
1348 CSIZE#ch,w,d:OVER#ch,1:INK#ch,col
1349 FOR i=0 TO t:CURSOR#ch,tx+i,ty:PRINT#ch,str$
1350 CSIZE#ch,0,0:OVER#ch,0:INK#ch,7
1351 END DEFine
```

Note: Title x2 both with horizontal offset and second with a vertical offset to mimic 3D affect

```
1353 DEFine PROCedure Init_RNews
1354 RESTORE 1355:FOR i=1 TO 8:READ RNew$(i)
1355 DATA 'announce release of New Shares...'
1356 DATA 'say bad Sales reduced Share value...'
1357 DATA 'announces Increase in their Dividends...'
1358 DATA 'announce Lower Dividends this quarter...'
1359 DATA 'say Tax Breaks increased Share value...'
1360 DATA 'say Tax Penalties reduced Share value...'
1361 DATA '- Forecasts a Shares Price Rise...'
1362 DATA '- Market Depression Reduces Share value.'
1363 END DEFine
```

```
1368 REMark Set-Up Trader Desk

1370 DEFine PROCedure Init_Trader
1371 WINDOW#0,500,30,gx+6,gy+224    :PAPER#0,0:INK#0,7:RESTORE 1376
1372 WINDOW#1,186,125,120+gx,41+gy:PAPER#1,0:INK#1,7:BORDER#1,1,2
1373 FOR i=3 TO 8
1374   OPEN#i,scr_6x8a0x0:READ a,b,c,d:WINDOW#i,a,b,c+gx,d+gy:BORDER#i,1,2
1375 END FOR i
1376 DATA 112,193,6,18, 42,125,308,41, 154,93,352,18       :REMark Win 3/4/5
1377 DATA 386,12,120,212, 386,44,120,167, 42,22,308,18    :REMark win 6/7/8
1378 PAPER#8,7:INK#8,0:CSIZE#8,2,1:CLS#8:OVER#7,1:INK#7,7:RESTORE 1380
1379 BLOCK#7,346,10,0,0,1:FOR i=1 TO 16:READ x,y,str$:CURSOR#7,x,y:PRINT#7,str$
1380 DATA 0,0,'Sym',24,0,'Div',48,0,'Yld',72,0,'P/E',96,0,'Open',129,0,'High'
1381 DATA 164,0,'Low',194,0,'Last',224,0,'Chg',254,0,'Ask',292,0,'Bid'
1382 DATA 325,0,'Vol',254,32,' ↑  ↓ ',292,32,'↑  ↓ ',325,32,'←  → ',366,32,'←'
1383 BLOCK#7,2,4,372,34,7:BLOCK#7,10,3,352,36,7:OVER#7,0
1384 SCALE#7,20,0,0:SCALE#4,80,0,0:SCALE#2,100,0,0:SCALE#1,164,0,0
1385 Title 2,2,2,1,1,134,3,'QBITS TRADER':Title 2,6,2,1,1,136,4,'QBITS TRADER'
1386 INK#5,7:STRIP#5,1:PRINT#5,'Sym  Last Chg Stock Price':STRIP#5,0
1387 WINDOW#5,150,80,354+gx,30+gy:SCALE#5,80,0,0
1388 INK#3,7:STRIP#3,1:PRINT#3,'Sym  Last Chg  Vol':STRIP#3,0
1389 WINDOW#3,108,180,8+gx,30+gy:SCALE#3,180,0,0
1390 INK#4,4:FOR i=1 TO 3:CURSOR#4,4,i*14:PRINT#4,'    '
1391 Bull 4,8,76:FOR i=1 TO 6:BLOCK#4,8,1,14,14+i*6,4
1392 INK#4,2:FOR i=1 TO 3:CURSOR#4,4,54+i*14:PRINT#4,'    '
1393 Bear 4,8,7:FOR i=1 TO 6:BLOCK#4,8,1,14,64+i*6,2
1394 Title 2,5,1,0,2,32,6,'MARKET':Title 2,5,1,0,2,382,6,'PORTFOLIO':INK#2,7
1395 RESTORE 1396:FOR i=1 TO 9:READ x,y,str$:CURSOR#2,x,y:PRINT#2,str$
1396 DATA 0,6,' ↑  ↓ ',344,6,' ↑  ↓ ',132,28,'Trends',300,6,'Week'
1397 DATA 378,122,'Shares:',360,132,'Dividends:',354,142,'Tax & Fees:'
1398 DATA 372,112,'Profits:',372,152,'Credits:'
1399 BLOCK#2,10,3,8,10,7:Market 2,34,99:BLOCK#2,10,3,352,10,7:Assets 2,162,99
1400 Chart 2,38,89:Trader 2,161,34:Money 2,161,48,c$:BLOCK#9,112,12,4,211,2
1401 STRIP#9,2:CURSOR#9,5,212:PRINT#9,'Reuters NEWS Flash':STRIP#9,0:INK#2,5
1402 INK#7,7:CURSOR#7,2,22:PRINT#7,'Company Evaluation:          Reckoner:'
1403 END DEFine

1405 DEFine PROCedure Title(ch,col,w,d,t,tx,ty,str$)
1406 CSIZE#ch,w,d:OVER#ch,1:INK#ch,col
1407 FOR i=0 TO t:CURSOR#ch,tx+i,ty:PRINT#ch,str$
1408 CSIZE#ch,0,0:OVER#ch,0:INK#ch,7
1409 END DEFine
```

Note: Title x2 both with horizontal offset and second with a vertical offset to mimic 3D affect

```
1411 DEFine PROCedure Init_RNews
1412 RESTORE 1413:FOR i=1 TO 8:READ RNew$(i)
1413 DATA 'announce release of New Shares...'
1414 DATA 'say bad Sales reduced Share value...'
1415 DATA 'announces Increase in their Dividends...'
1416 DATA 'announce Lower Dividends this quarter...'
1417 DATA 'say Tax Breaks increased Share value...'
1418 DATA 'say Tax Penalties reduced Share value...'
1419 DATA '- Forecasts a Shares Price Rise...'
1420 DATA '- Market Depression reduces Share value.'
1421 END DEFine
```

```
1365 DEFine PROCedure Init_Stocks
1366 CURSOR#0,76,10:PRINT#0,'Initialising Market':CLS#0,4:RESTORE 1372
1367 FOR a=1 TO 40
1368   READ CN$(a),Stock$(a):FOR b=1 TO 6:READ Stock(a,b):END FOR b
1369   CURSOR#0,184+a*6,10:PRINT#0,'.':PAUSE 1
1370 END FOR a
1371 REMark Info>> Company Name:Sym:Sector,Status,Div,Yld,P/E,Last,Chg
1372 DATA 'Rio Tinto',              'RIO',0,5,20,30,1200,12
1373 DATA 'Boeing Co.',             'BA ',0,0,20,30,2200,-5
1374 DATA 'Visa',                   'V  ',0,10,12,30,3220,18
1375 DATA 'Exxon Mobil',            'EOM',0,8,20,30,580,-6
1376 DATA 'Carnival Corp.',         'CCL',0,5,20,30,230,-8
1377 DATA 'AT&T Inc.',              'T  ',0,12,20,30,1620,-5
1378 DATA 'Walt Disney Co.',        'DIS',0,0,20,30,490,2
1379 DATA 'Ford Motors',            'F  ',0,4,20,30,370,-10
1380 DATA 'GameStop',               'GME',0,12,20,30,240, 2
1381 DATA 'Simon Property Grp',     'SPG',0,0,20,30,3200, 2
1382 DATA 'General Electric',       'GE ',0,8,20,30,1820, 2
1383 DATA 'Johnson & Johnson',      'JNJ',0,0,20,30,1230, 2
1384 DATA 'Nokia',                  'NOK',0,12,20,30,280, 2
1385 DATA 'Coca-Cola Co.'           ,'KO ',0,0,20,30,470, 2
1386 DATA 'Pfizer Inc',             'PFE',0,10,20,30,2220, 2
1387 DATA 'NextEra Enery',          'NEE',0,0,20,30,1670, 2
1388 DATA 'Wells Fargo',            'WFC',2000,12,20,30,370, 2
1389 DATA 'Rolls Royce Holdings',   'RR',0,10,20,30,2390, 2
1390 DATA 'New Concept Energy',     'GBR',0,0,20,30,3200, 2
1391 DATA 'Drax',                   'DRX',12,10,20,30,1310, 2
1392 DATA 'SilverCrest Metals',     'U1L',0,10,20,30,3200, 2
1393 DATA 'Invinity Enery',         'IES',5,10,20,30,1260, 2
1394 DATA 'Ecolab',                 'ECL',0,12,20,30,1250, 2
1395 DATA 'Locheed Martin Corp.',   'LMT',0,4,20,30,3290, 2
1396 DATA 'PayPal Holdings',        'PYP',0,0,20,30,1200, 2
1397 DATA 'Kinder Morgan',          'KMI',0,5,20,30,1530, 2
1398 DATA 'Vista Gold Grp',         'B1B',0,10,20,30,4200, 2
1399 DATA 'Computer Systems Co.','CSC',0,5,20,30,490, 2
1400 DATA 'Intel Corp.',            'INT',0,5,20,30,385, 2
1401 DATA 'Procter & Gamble Co.',  'PG ',0,12,20,30,890, 2
1402 DATA 'Walmart',                'WMT',0,5,20,30,470, 2
1403 DATA 'Exelon Corp.',           'EXC',0,12,20,30,780, 2
1404 DATA 'Tesco',                  'TSC',0,5,20,30,380, 2
1405 DATA 'Greggs',                 'GRG',0,0,20,30,230, 2
1406 DATA 'Hunting',                'HTG',0,0,20,30,320, 2
1407 DATA 'Centrica',               'CNA',0,5,20,30,670, 2
1408 DATA 'Abingdon Health',        'ABD',0,0,20,30,1200, 2
1409 DATA 'Medica Group P',         'MGP',0,5,20,30,1370, 2
1410 DATA 'Clarkson',               'CKN',0,0,20,30,200, 2
1411 DATA 'Rank',                   'RNK',0,10,20,30,680, 2
1412 END DEFine
```

Variables pm - Portfolio max slots : wn - week num : sn - stock num : mr - Market row

```
1414 REMark Trader Graphics

1416 DEFine PROCedure TBuy(ch,x,y)
1417 INK#ch,6:FILL#ch,1:ARC#ch,x-3.5,y TO x+4,y,-PI/2
1418 LINE#ch TO x+3.4,y-3:ARC#ch TO x-4,y-3,-PI/2:LINE#ch,x-4,y-3 TO x-4,y
1419 FILL#ch,0:INK#ch,0:ARC#ch,x-3,y-.6 TO x+4,y-.6,PI/2
1420 ARC#ch,x-4,y-1.6 TO x+3.6,y-1.6,PI/2:ARC#ch,x-4,y-2.5 TO x+3.6,y-2.5,PI/2
1421 END DEFine
```

```
1423 DEFine PROCedure TSell(ch,x,y)
1424 INK#ch,6:LINE#ch,x-5,y-2.5 TO x-1,y-2.5 TO x,y-4 TO x-6,y-4 TO x-5,y-2.5
1425 FILL#ch,1:LINE#ch,x-5,y TO x-2,y+4 TO x,y+2.8 TO x-3,y-1 TO x-5,y
1426 FILL#ch,0:LINE#ch,x-2,y+1.8 TO x+2.4,y-1 TO x+1.8,y-2 TO x-2,y+.6
1427 INK#ch,0:LINE#ch,x-5,y+1 TO x-2,y-.8:LINE#ch,x-3,y+3.6 TO x,y+1.8
1428 END DEFine
```

```
1430 DEFine PROCedure Market(ch,x,y)
1431 INK#ch,7:FILL#ch,1
1432 LINE#ch,x-3,y-1.5 TO x,y TO x+3,y-1.5 TO x-3,y-1.5:FILL#ch,0
1433 LINE#ch,x-1.3,y-2.2 TO x-1.3,y-4.6:LINE#ch,x-2.5,y-2.2 TO x-2.5,y-4.6
1434 LINE#ch,x+1.3,y-2.2 TO x+1.3,y-4.6:LINE#ch,x+2.5,y-2.2 TO x+2.5,y-4.6
1435 LINE#ch,x,y-2.2 TO x,y-4.6        :LINE#ch,x-3,y-5 TO x+3,y-5
1436 END DEFine
```

```
1438 DEFine PROCedure Assets(ch,x,y)
1439 INK#ch,7:FILL#ch,1:LINE#ch,x+3,y-1 TO x-1.8,y-1
1440 LINE#ch TO x-3.8,y-4 TO x+1,y-4 TO x+3.2,y-1:FILL#ch,0
1441 LINE#ch,x-2.5,y-1 TO x-4.5,y-4:LINE#ch,x+1.2,y-5 TO x+3.5,y-2
1442 ARC#ch,x-4,y-4 TO x-4,y-5,PI:LINE#ch TO x+1,y-5:ARC#ch TO x+1,y-4,-PI
1443 END DEFine
```

```
1445 DEFine PROCedure Chart(ch,x,y)
1446 INK#ch,7:LINE#ch,x,y TO x,y-5 TO x+5,y-5:INK#ch,4
1447 LINE#ch,x+.5,y-4.5 TO x+2,y-2:INK#ch,2:LINE#ch TO x+3.5,y-3.8:INK#ch,4
1448 LINE#ch TO x+5,y :INK#ch,7:LINE#ch,x+3.6,y-1 TO x+5,y TO x+5.2,y-2
1449 END DEFine
```

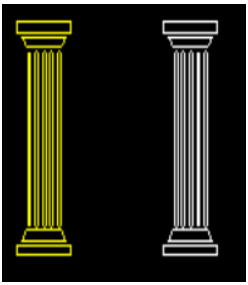
```
1451 DEFine PROCedure Money(ch,x,y,c$)
1452 INK#ch,4:FILL#ch,1:CIRCLE#ch,x,y,1.6,.6,PI/2:FILL#ch,0
1453 INK#ch,4:FILL#ch,1:CIRCLE#ch,x,y-4,3:FILL#ch,0
1454 INK#ch,4:FILL#ch,1:CIRCLE#ch,x,y-5,4,.6,PI/2:FILL#ch,0
1455 INK#ch,0:LINE#ch,x-1,y-1 TO x,y-2 TO x+1,y-1
1456 STRIP#ch,4:INK#ch,0:CSIZE#ch,2,0
1457 CURSOR#ch,x,y,-7,+5:PRINT#ch,c$:STRIP#ch,0:CSIZE#ch,0,0
1458 END DEFine
```

```
1460 DEFine PROCedure Trader(ch,x,y)
1461 INK#ch,7:FILL#ch,1:CIRCLE#ch,x,y+.5,2.2:FILL#ch,0
1462 ARC#ch,x+2.8,y+.6 TO x-2.8,y+.6,PI:INK#ch,0:LINE#ch,x-.5,y-1 TO x+.5,y-1
1463 INK#ch,7:FILL#ch,1:ARC#ch,x-4,y-4 TO x-1,y-3,-PI:LINE#ch TO x+1,y-3
1464 ARC#ch TO x+4,y-4,-PI:LINE#ch TO x+4,y-6 TO x-4,y-6 TO x-4,y-4:FILL#ch,0
1465 INK#ch,0:CIRCLE#ch,x-1,y+.6,1,.5,PI/2:CIRCLE#ch,x+1,y+.6,1,.5,PI/2
1466 LINE#ch,x-1.8,y-2 TO x,y-4 TO x+1.8,y-2:LINE#ch,x,y-4 TO x,y-6
1467 LINE#ch,x-2.4,y-4.5 TO x-2.4,y-6:LINE#ch,x+2.4,y-4.5 TO x+2.4,y-6
1468 END DEFine
```

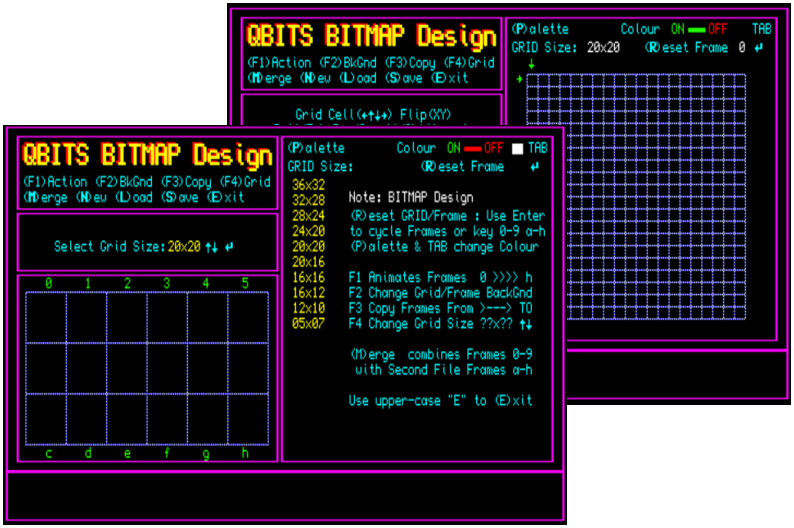
```
1470 DEFine PROCedure Bear(ch,x,y)
1471 INK#ch,2:FILL#ch,1:ARC#ch,x-3,y+1 TO x-1.6,y+1,-80:FILL#ch,0
1472 LINE#ch,x-3,y+1 TO x+3,y+1:FILL#ch,1:ARC#ch,x+1.6,y+1 TO x+3,y+1,-80
1473 ARC#ch TO x+3,y-5,-PI/2 TO x-3,y-5,-PI/3 TO x-3.2,y+1,-PI/2:FILL#ch,0
1474 INK#ch,0:ARC#ch,x-1.5,y-5.4 TO x+1.5,y-5.4,-80
1475 LINE#ch,x-1.2,y-5 TO x+1,y-3.5 TO x-1,y-3.5 TO x+1.2,y-5
1476 LINE#ch,x-2,y-1 TO x-1.4,y-1:LINE#ch,x+1.4,y-1 TO x+2,y-1
1477 END DEFine
```

```
1479 DEFine PROCedure Bull(ch,x,y)
1480 INK#ch,4:FILL#ch,1:ARC#ch,x-1,y+1 TO x+1,y+1,PI/4
1481 ARC#ch TO x+2.8,y, PI/2 TO x+.8,y-6,-PI/4 TO x-.8,y-6,PI/4
1482 ARC#ch TO x-2.8,y,-PI/4 TO x-1,y+1,PI/2:FILL#ch,0
1483 FILL#ch,1:ARC#ch,x-2,y+1 TO x-4,y+3,-PI/2 TO x-2,y+1, PI:FILL#ch,0
1484 FILL#ch,1:ARC#ch,x+2,y+1 TO x+4,y+3, PI/2 TO x+2,y+1,-PI:FILL#ch,0
1485 LINE#ch,x-3,y TO x-5,y-1 TO x-3,y-1:LINE#ch,x+3,y TO x+5,y-1 TO x+3,y-1
1486 INK#ch,0:LINE#ch,x-1.8,y-1.5 TO x-1,y-1.5:LINE#ch,x+1.8,y-1.5 TO x+1,y-1.5
1487 CIRCLE#ch,x-.8,y-4.8,.4:CIRCLE#ch,x+.8,y-4.8,.4
1488 END DEFine
```

```
1490 DEFine PROCedure Pillar(ch,x,y)
1491 LINE#ch, x-5,y+21 TO x+6,y+21 TO x+6,y+23 TO x-5,y+23 TO x-5,y+21
1492 LINE#ch, x-3,y+18 TO x+4,y+18 TO x+5,y+20 TO x-4,y+20 TO x-3,y+18
1493 LINE#ch, x-3,y-18 TO x+4,y-18 TO x+5,y-20 TO x-4,y-20 TO x-3,y-18
1494 LINE#ch, x-5,y-21 TO x+6,y-21 TO x+6,y-23 TO x-5,y-23 TO x-5,y-21
1495 FOR c=1 TO 5
1496   x1=x-4+c*1.5-.3:x2=x-4+c*1.5+.3:y1=y+17:y2=y-17
1497   ARC#ch,x1,y1 TO x2,y1,-PI:LINE#ch TO x2,y2
1498   ARC#ch,x2,y2 TO x1,y2,-PI:LINE#ch TO x1,y1
1499 END FOR c
1500 END DEFine
```



```
1502 DEFine PROCedure TestMode
1503 SELect ON k
1504    =232:Stock_UpDate 25:wn=26                    :REMark Resets to Start
1505    =236:Stock_UpDate wn                          :REMark Increments Weeks
1506    =240:Wick 1,col,177,90,70,85,75:IF col=4:col=2:ELSE col=4
1507    =244:FOR i=1 TO 8:PRINT#6,CN$(i);' ';RNew$(i):PAUSE 50:CLS#6
1508    =248:Audit(0)=10000:FOR i=1 TO 130:Audit(i)=audit(i)+10000+i*RND(-3 TO 70):END FOR i:wn=156:TReview
1509 END SELect
1510 END DEFine
```



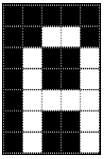
BITMAP Design Introduction

Whether you call them **Sprites, Glyphs or Bitmaps** these are the collection of Bytes arranged in columns and rows, stored or generated to identify individual Pixel colours of an object. Initially used to handle graphical objects separate from the video memory, such as printing Characters to screen, they were quickly developed into Game Sprites and during the 1980's and much of the 1990s, became the standard way to integrate images used in what is now referred to as Retro or Classic Computer Games.

Compared with early consoles, that generated a few thousand screen Pixels the modern computer utilises millions of bytes and works with dedicated **3D Video chips** rendering **3D Objects** as Vectored Graphics more efficiently than **2D BITMAP Sprites**.

Character Sets

Character Fonts represented by a two-dimensional array are often referred to as a **dot matrix**. In the day with screen of low resolution, character sets were created with minimal columns and rows as the one shown.



```
DATA 0,0,0,0,0
DATA 0,0,1,1,0
DATA 0,1,0,0,1
DATA 0,1,0,0,1
DATA 0,1,1,1,1
DATA 0,1,0,0,1
DATA 0,1,0,0,1
```

Grid 5x7

In this example the 1-bit Colour scheme show
DATA fields with 0's as Black
and 1's as White.

Characters Fonts represent different Styles, Sizes as well as aspects such as **Bold,** *Italics* etc. Internationally accepted standards for character codes permitted worldwide interchange of text in electronic form. Character sets such as **ASCII** use 8-bit encoding whereas **Unicode** uses variable bit encoding. **Unicode** represents most of the characters used in the written languages of today's world. While **ASCII** is still widely used, it is based upon the West's Latin script.

Game Sprites

The eighties computers could only keep track of a few moving **Sprites** at a time and the Characters and Objects look very pixelated by today's standards. Animation was created from a sequence of **Sprites** with typically a grid of 8x16 pixels and four colours, yet even within these limitations programmers produced some lovable characters. The best **Sprite Designers** when working with small fields of only six, seven, eight pixels across could imply a face with minimal detail.

The creating and modifying of Pixel Characters or Objects soon became a recognised art form.

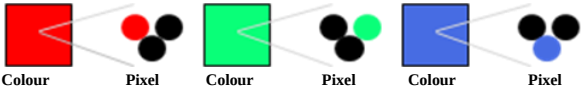
Resolution & Colour

Resolution is the measure of **Pixel density**, referenced as **dots per inch (dpi)**. So for every inch (25mm) of width or height there will be a set number of Pixels. Each **Pixel** has colour depth: **hue** (its quality of colour or wave length) and **value** (relative darkness or lightness - its energy) and represented by a binary code. The number of bits indicates how many colours are available for each Pixel. For a Black and White image, only 1-bit is needed either 0 OFF or 1 ON, for 2-bit colour: 00, 01, 10, 11. Increasing the bits per pixel then the greater colour and depth is achievable. With true colour a **Pixel** is generated from three colours (**Red**, **Green**, and **Blue**) and the colours seen are due to different combinations and light intensities.

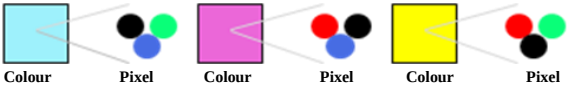
White Grey Black

For a White section of a screen all three colours are active with about the same relative intensities as in sunlight. Gray parts of the screen have all three colours producing light, but at a much lower intensity. Black is the lack of any emitted light.

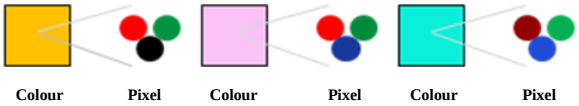
Primary Colour: Red Green Blue



Colour Combinations: Cyan Magenta Yellow



Mixtures of two or three primary colours with different intensities create the other colours. The combinations for Orange (Red with a little Green) - Pink (Red with a little Green and a little Blue) and Turquoise (Blue and Green with a little Red) as shown.



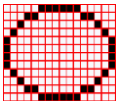
BITMAP Design
Bitmaps are sometimes referred to as "raster" graphics, created from rows of different coloured Pixels that collectively form an image. One problem with Bitmaps images is that they cannot be enlarged to a great degree without the effect of pixilation, this is where the individual squares become visible rather than having a smooth finish. Despite this drawback, bitmap images are extremely useful when designing interfaces and in printing images.

The mapping of a colour assigned to each individual square in a Bitmap image is sometimes referred to as the key. Essentially this is what is stored in a .bmp file - the data that tells the rendering part of the software how many squares there are and the colouring of each. Apart from the BMP file format, the most common and widely used primarily for web transfers are JPG, GIF, TIFF, PSD, PNG, etc. Other popular formats are RAW and EXIF, which contain not only the information about the image and how to render it but metadata on how it was captured.

QBITS BITMAP Considerations
The QL Vector Graphics System draws the object relative to its coordinate System. However, using Pixel Graphics to create a BITMAP image, it is not always as straight forward as imagined. Monitor screens tend to be rectangular not square in shape with a different number of horizontal pixels to vertical ones, so to obtain graphical balance with say a circle the number of columns can be greater than the number of rows.

To represent this the BITMAP horizontal is offset making it oval in shape. But when displayed on screen the **Pixel** image appears circular.

Grid 16x12



In the same way ARCs and angled direction of LINEs will require some careful consideration or at best a trial by error approach until the desired result is obtained. Mapping the Pixel colours of a design requires a working GRID. Then a Frame view to present the Object at Pixel level, so any distortions caused by Pixel Graphic or Colour variances can then be adjusted in the GRID area to better achieve that which was anticipated. Having a number of Frames to view then opens the possibilities for developing Game Animation.

Hobbyist programs in the eighties employed simple methods for storage. The Bitmap saved as a string of Bytes literally dumped from an array or straight from memory. To LOAD they reversed the process, Reading the file contents directly into memory or to a Dimensioned Array. A second method was to SAVE as DATA Lines so as to MERGE them with other programs.

Seen as necessary for developing a variety of Sprites, different working GRID Sizes would be needed as opposed to an all-encompassing approach. Then options to switch Gride Sizes and Merge different BITMAP Files seemed a logical next step.

For QBITS Storage and Retrieval a **BITMAP File** should hold a series of **Frames**, defined by **Columns** x **Rows** with each cell assigned a **colour** value, then options to Save in **GRID** Format or as a set of **DATA** Lines.

BITMAP File = Filename, Frame, Columns, Rows, Colour

Next was reviewing QL Colour arrangements for developing a QBITS BITMAP Colour Palette.

1980's QL Screen Organisation

The original QL has 32k of **Screen RAM** and the **Pixel** coordinate system to define the position and size of Windows. The **Screen RAM** being organised as a series of **16 Bit words** starting from address **Hex 2000** and progressing in the order of the raster scan.

Hex 2000 - 2800 (Dec 131072 - 163840) 128 Bytes x 256 rows

High Byte								**Low Bite**								
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**	**Bit**
G	G	G	G	G	G	G	G	R	R	R	R	R	R	R	R	**512**
G	F	G	F	G	F	G	F	R	B	R	B	R	B	R	B	**256**

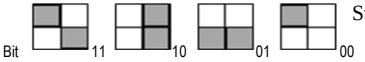
QDOS Screen MODE

The colour of each pixel on screen is a combination of **Green** and Red in **MODE 4** (**512**) plus **Blue** in **MODE 8** (**256**). Using certain bit values and setting the bits **ON** or **OFF**, the result is a **Primary** and XOR **Contrast** making a Combined Colour plus a **Stipple** pattern.

Bit	**Value**	**Colour**	**Bit Pattern**	**Colour Combination**	**Final Colour**	**MODE 8**	**MODE 4**
0	1	**Blue**	000	No colour	**Black**	0	0
1	2	**Red**	001	**Blue**	**Blue**	1	
2	4	**Green**	010	**Red**	**Red**	2	2
3	8	**Blue**	011	**Red+Blue**	**Magenta**	3	
4	16	**Red**	100	**Green**	**Green**	4	4
5	32	**Green**	101	**Green+Blue**	**Cyan**	5	
6	64	Stipple	110	**Green+Red**	**Yellow**	6	
7	128	Stipple	111	**Green+Red+Blue**	**White**	7	7

SuperBasic Colour Components

The default is a Checkerboard **Stipple** pattern with the **Contrast** set the same as **Primary**



Stipple Contrast XOR Green Red Blue

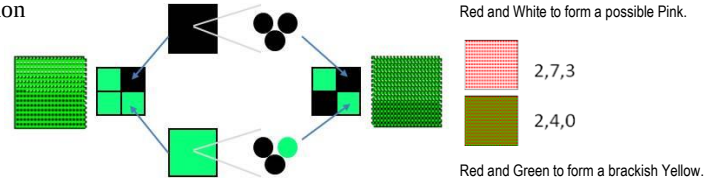
To Calculate a composite colour from zero:

add +1 Primary Blue + 2 Red, + 4 Green
then +8 Contrast Blue +16 Red, +32 Green
Stipple 0 +64 horizontal +128 vertical, +192 Checkerboard

Exploring QL Colour Palette

The original QL Mode 4 has only Black, Red, Green, White, but by exploring the use of **Stipple** combinations it is possible to create a change in light intensity and the semblance of additional colours by manipulating values in the 0-255 Colour range:

Mode 4 High Resolution



QPC Colour Palettes

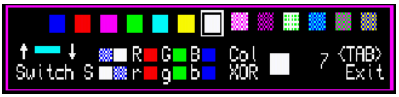
For faster machines with expanded memory the **SMSQ\E** O/S has extended the range of colours and has higher screen resolutions with improved performance. **SBASIC** keywords now include **COLOUR_NATIVE** (machine dependant), **COLOUR_QL** (standard QL colours), **COLOUR_PAL** (8-bit 256 Colour Palette) and **COLOUR_24** (True 24-bit Colour Palette) together with **PALETTE_QL** and **PALETTE_8** which allow changing the colours by mapping them to alternative ones.

QBITS BITMAP Colour (P)alette

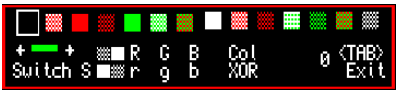
Using the 8-bit colour (Mode 16) option of QPC11 Setting, and operating with SBASIC Mode 4 Colours displayed with Contrast and Stipple combination are in the range 0 to 255. The bits are identified using DIV and MOD functions:

Stipple (num DIV 64); **Contrast** (num MOD 64 DIV 8); Primary (num MOD 64 MOD 8).

In **Select Mode** use Left & Right cursor key to highlight and choose one of the 14 combinations of **Primary**, **Contrast** & **Stipple**.



Switch with **Spacebar** to **Component Mix**, where **Primary RGB** & **Contrast rgb** Colours can be switched **ON/OFF** with 'R,r,G,g,B,b' keys and the **Stipple** Pattern by cycling through with '**S**'.

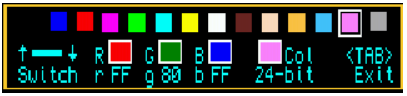


Spacebar returns to Palette **Select Mode** showing any change. Use **TAB** to Exit back to **GRID.** Spacebar now turns GRID Colour Mode **ON/OFF** and **TAB** will cycle through the changed Palette displaying the relevant colour combinations.

QBITS True Colour Option

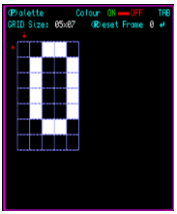
As a considered option a **Colour_24** (**P**)alette version has no Stipple just **Red Green** & **Blue** each with a range of 0-255 or in Hex 00-FF. Note Keywords such as WINDOW, INK, BLOCK etc. with Pixel attributes would be more easily written in Hex numbers ($num).

000000	**Red**	=	num DIV 65536
TO	**Green**	=	num MOD 65536 DIV 256
FFFFFF	**Blue**	=	num MOD 65536 MOD 256

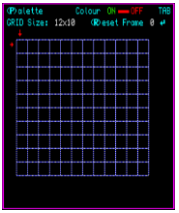


QBITS BITMAP GRID Sizes

Setting individual **Cell Sizes** for a range of GRID designs dictates what can be fitted within a 512x256 Pixel range. As the GRID Cell number increases it was necessary to re-calculate **Cell** width/height so as to best utilise the screen GRID space especially when trying to leave enough space to show a meaningful set of Frames.



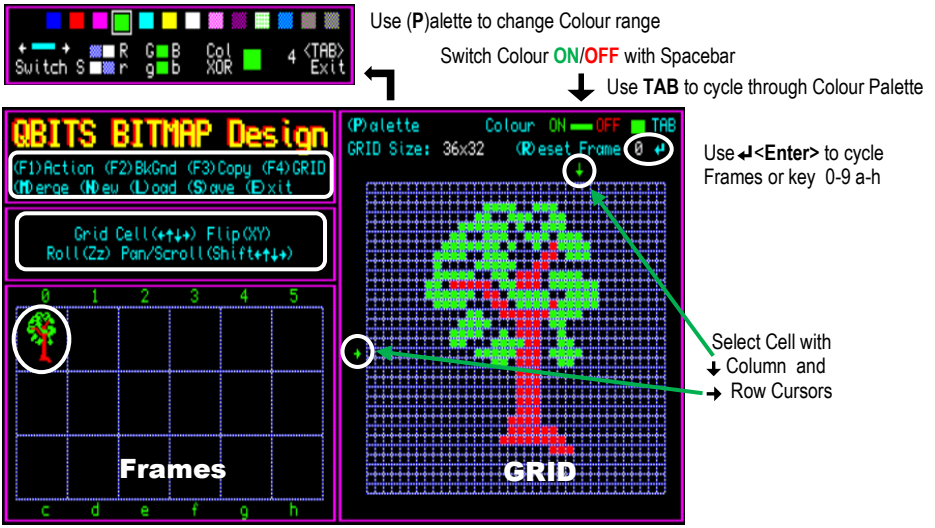
GRID Sizes began with a **5x7** Cells for a Character Set smaller than the 6x10 SuperBASIC CSIZE 0,0. The next **12x10** is equivalent to the Mode 8 CSIZE 2,0. The rest ranking up to **36x32** are stepped increases using multiples of four.



QBITS BITMAP Layout

The decided screen layout shown below has a **BITMAP GRID** area for **O**bject Designs of up to 36x32 Pixels and **Frames** in three rows of six identified as 0 to 9 & a to h. This can lead to some interesting usage, which will be viewed later.

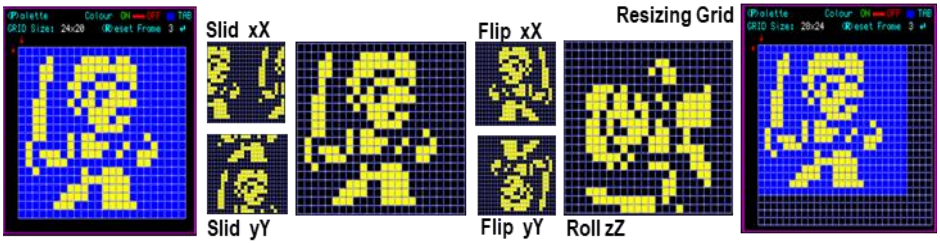
Each GRID Cell is filled with a background colour, default being Black (0) use **(F2)BkGnd** to choose an alternative from the Colour Palette. The Colour Palette has 14 Colours each of which can be changed. Refer back to the Colour (P)alette arrangements delt with on previous page.



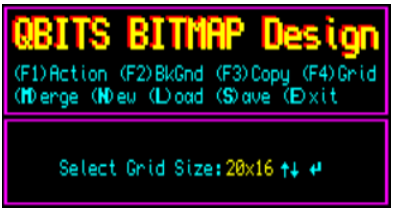
Flip(XY) Swaps Left-Right / Top-Bottom **Roll** (**zZ**) Turns Grid 90^0 Clockwise / Anti-clockwise **Pan/Scroll (Shift ← ↑ ↓ →)** moves the columns horizontally and rows vertically. (**R**)**eset** clears all the Cells of selected **Frame** back to **Black** (Colour = 0). (**N**)ew prompts for a Save before clearing all Frames.

QBITS BITMAP GRID Flip Roll & Slid

As the **GRID** is stored as an Array swapping or moving entries (column/row) with a **Flip** left or right, top to bottom or **Slid** with Pan and Scroll a column or row at a time is relatively simple. A 90^0 **Roll** requires a little more when relocating a cell entry corner to corner to corner. A resized GRID allows an expansion of Sprite area the left Grid 24x20 has been transposed into the 28x24 on the right leaving unused areas to the right and bottom.



QBITS BITMAP Commands
On start-up you are prompted to select a GRID Size. In addition to the (**N**)ew (L)oad (S)ave & (E)xit is:-

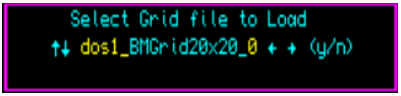


QBITS BITMAP Merge
This allows two GRID Files of same or different GRID Sizes to combine their Frames [**0 TO 9**] .

Note: If Files different Grid sizes, (**L**)oad smaller First, then **Resize** with (**F4**)**Grid** command.

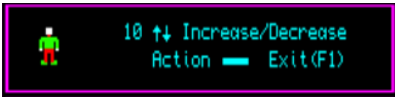
QBITS BITMAP New
This will reset all Frames, but will first present the opportunity to **(S)ave** current Frame Set

QBITS BITMAP Load
You can only Load a BITMAP File that has the same Selected GRID Size.



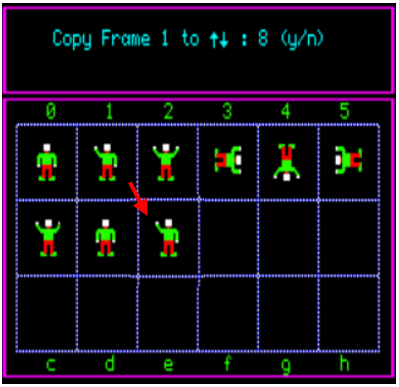
QBITS BITMAP Save
You can Save BITMAP Files as a GRID in Array Format or as Lines of DATA.

(F1)Action - this runs the sequence of Frames. Set PAUSE to -1 and Step though at your own pace or set a 1-20 Frame delay and run the sequence...



(F2)BkGnd – Select a Background colour. Clear by Selecting same Background colour again.

(F3)Copy - from current Frame to selected Frame.



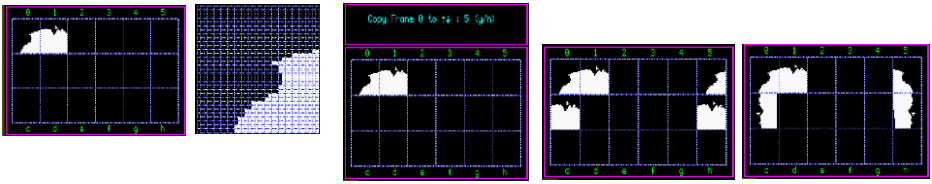
Note: Animation
While developing this code what came to light was an option to create a short action sequence. The Frames show a simple Sprite in various positions. Using **(F3) Copy** Frame (0) is copied to Frame (1) and the left arm position changed. Copying this to Frame (2) the next change is made and so on. In copying Frames the **Flip, Roll** and **Slid** commands can be used to quickly build up a sequence of altered images.

(F4)GRID - use to select any of the available Grid Sizes with Up/Down Cursors to same Size as BITMAP File to be Loaded or about to be Saved.

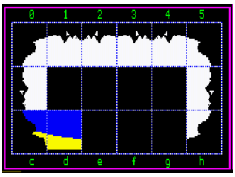
Note: In Saving to a smaller Grid Size may delete Pixels on the edges of the current design.

QBITS BITMAP Background Scenes

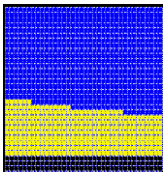
The three rows of six Pixel Frames can now be explored as a means of creating backgrounds. Taking Frame (**0**) and use of (**F2**) **BkGnd** to select **White**, then change some of the top and side to Black to form jagged pattern. Then Copy Frames (0) & (1) to Frames (5), (6) & (b) as shown. Use the **Flip**(**X**) and **Rotate** (**zZ**) to build the outer edges to a background scene.



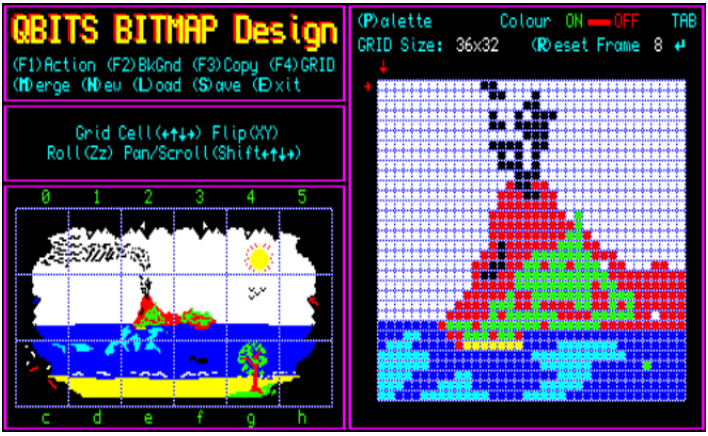
Using an uneven **36x32 Grid** when you Rotate the Frame the two columns to left and right are filled with the background colour (the colour contained in cell 0,0). Therefore, these cells will need colouring in.



Selecting Frame (c) we add a blue sea and yellow sandy beach then continue with this theme in (d) to (g). Above the seas we have sky so for Frames (7) to (a) use (**F2**)**BkGnd** again to set to **White**.



Then it's just a question of completing the unfinished Frames with sky, sea and sand. Then adding a few extras, a blazing Sun, a flock of birds, a shady tree, then on the horizon an island with an erupting Volcano.



QBITS BITMAP Files

BMGrid05x07_0 Character Fonts 0-9A-H
BMGrid20x16_0 Animation Sprite
BMGrid24x20_0 Sprites Collection
BMGrid36x32_0 Background Scene

QBITS BITMAP Design Procedures

BMDesign	Access to Grid Functions and Main Commands
GReset	Reset current Frame Pixels & Grid Cells all to zero
Gflip	Flip horizontally or vertically the Pixel Frame & Grid
GRoll	Rotate 90^0 Clockwise or Anticlockwise Pixel Frame & Grid
GSlid	Pan / Scroll pixel frame & Grid
Ghelp	Displays keys used for Grid Functions
FAction	Step through sequence of Frames simulating Animation
GBkGnd	Select an alternative Background colour
CBkGnd	Changes the background with new colour
FCopy	Copy current Frame and Overwrite in another Frame
GSize	Selection of Grid Size activated by (N)ew or (R)esize
FCalc	Calculate position [0,0] x,y of Pixels Frame position
SDraw	Draws a whole Frame and/or Grid display
GDraw	**P**rints individual Pixel and/or Grid blocks
DGrid	Draws Grid lines and sets Grid Attributes
GTemp	Copy **SGrid**(**p**,**c**,**r**) to **TGrid**(**p**,**c**,**r**)
GLive	Copy **TGrid**(**p**,**c**,**r**) to **SGrid**(**p**,**c**,**r**)
CPQL	Colour Palette for QL Colour/ Contrast/Stipple mix
CPSel	Highlight a Palette Colour
CMode	Switch between palette selection and colour mix
CRead	Reads and identifies Colour/Contrast/Stipple
CPrnt	Prints colour components of colour/contrast/stipple
GMerge	Select and Load new Grid - Merge with current Grid
GNew	Option to save current Grid then Select new Grid
GFSel	Identifies Load/Merge/Save the latter for Grid or Data
GFChk	Checks file exists for Load or overwrite (y/n) for Save
GLoad	Loads selected file into the **SGrid**(**p**,**c**,**r**)
GSave	Saves Array dump or DATA statements to selected file
QExit	LRUN dn$ return to QBITS Progs Menu
Init_Win	Set WINDOW paramiters
Init_Title	Display QBITS Title and main Commands New/Load/Save etc.
Init_CPQL	Sets the Colour Palette/Contrast/Stipple parameters
Init_Grid	Gets Grid attributes, opens Grid & Frame Windows
BITMAP _Intro	Instructions on Keywords

QBITS BITMAP and True Colour Palette

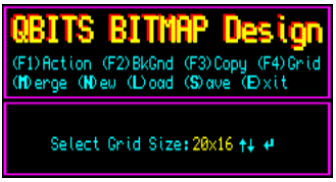
Init_CP24

CP24

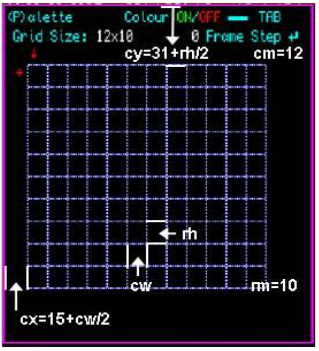
QBITS BITMAP Design code

```
1000 REMark QBITS_BITMAPS_v3 (QBITS BITMap Design v3 2021)

1002 OPEN_IN#9,'ram2_QBITSConfig':INPUT#9,gx\gy\dn$\dev$\dn%\dm%
1003 DIM Drv$(15,5):FOR d=0 TO 15:INPUT#9,Drv$(d):END FOR d: CLOSE#9

1005 WHEN ERRor
1006   eck=1:CONTINUE
1007 END WHEN

1009 MODE 4:Init_Win:BMDesign

1011 DEFine PROCedure BMDesign
1012 gm=5:sm=0:GSize:fp=1:p=0:FCalc
1013 REPeat Des_lp
1014   IF cur=4:SGrid(p,x,y)=CP(cs%):c=x:r=y:fp=1:gp=1:FCalc:GDraw
1015   BLOCK#4,236,8,8,26,0 :INK#4,cur:CURSOR#4,cx+x*cw,24:PRINT#4,'↓'
1016   BLOCK#4,12,170,4,34,0:INK#4,cur:CURSOR#4,8,24+cy+y*rh:PRINT#4,'→ '
1017   BLOCK#4,16,3,166,6,cur:BLOCK#4,10,7,210,4,CP(cs%):k=CODE(INKEY$(-1))
1018   SELect ON k
1019     =48 TO  57:p=k-48:fp=1:gp=1:FCalc     :SDraw  :REMark 0 to 9
1020     =97 TO 104:p=k-87:fp=1:gp=1:FCalc     :SDraw  :REMark a to h
1021     =232:CLS#7                  :FAction  :GPaint  :REMark (F1)Frame Action
1022     =236:CLS#7                  :GBkGnd :GPaint  :REMark (F2)Grid BackGnd
1023     =240:CLS#7:n=p              :GCopy    :GPaint  :REMark (F3)Copy
1024     =244:CLS#7:IF gm<10:sm=2  :GSize    :GPaint  :REMark (F4)Grid Resize
1025     =80,112:CLS#7                :CPQL    :GPaint  :REMark (P)alette
1026     = 9:IF cur=4:cs%=cs%+1:IF cs%>13:cs%=0        :REMark <TAB>Change Colour
1027     =32:IF cur=2:cur=4:ELSE cur=2                 :REMark <SpaceBar> Colour ON/OFF
1028     =10:IF p=pm:p=0:gp=1:FCalc:SDraw:ELSE p=p+1:gp=1:FCalc:SDraw
1029     =192:x=x -1:IF x< 0:x=0
1030     =200:x=x+1:IF x=cm:x=cm-1                     :REMark (← ↑↓→) x,y Grid Pointers
1031     =208:y=y -1:IF y< 0:y=0
1032     =216:y=y+1:IF y=rm:y=rm-1
1033     =88,120:xf=cm-1:yf=0:zx=-1:zy=1       :GFlip  :REMark (X)Flip Grid
1034     =89,121:yf=rm -1:xf=0:zy=-1:zx=1      :GFlip  :REMark (Y)Flip Grid
1035     = 90:rxm=rm-1+cs:rt=-1:rym=0:yt= 1    :GRoll  :REMark (z)Roll ClockWise
1036     =122:rxm=cs:rt=1 :rym=rm-1   :yt=-1   :GRoll  :REMark (Z)Roll Anti-CW
1037     =196:pa=cm-1:pb=0:pc=-1:pd=0:md=0     :GSlid  :REMark <Shift ← →>Pan Grid
1038     =204:pa=0:pb=cm-1:pc=0:pd=-1:md=0     :GSlid
1039     =212:sa=rm-1:sb=0:sc=-1:sd= 0:md=1    :GSlid  :REMark <Shift ↑ ↓>Scroll Grid
1040     =220:sa=0:sb=rm-1:sc= 0:sd=-1:md=1    :GSlid
1041     = 69:QExit:BLOCK#0,20,10,212,34,0             :REMark (E)xit
1042     =77,109:ck=0:dg=2:CLS#7  :GMerge    :GPaint  :REMark (M)erge Grid
1043     =78,110:ck=0:dg=3:sm=1    :GSize    :GPaint  :REMark (N)ew
1044     =76,108:ck=0:dg=1:CLS#7  :GFSel     :GPaint  :REMark (L)oad
1045     =83,115:ck=0:dg=0:CLS#7  :GFSel     :GPaint  :REMark (S)ave Data/Grid
1046     =83,114:col=0:CLS#7        :GReset   :GPaint  :REMark (R)eset Grid
1047   END SELect
1048 END REPeat Des_lp
1049 END DEFine
```

```
1051 DEFine PROCedure QExit
1052 CURSOR#6,212,34:PRINT#6,'Y/N':PAUSE:IF KEYROW(5)=64:LRUN dn$
1053 END DEFine

1055 DEFine PROCedure GSize
1056 REMark sm=0 Resize Grid size sm=1 New grid
1057 CLS#7:INK#7,5:IF sm=1:GFSel
1058 CURSOR#7,32,16:PRINT#7,'Select GRID Size:      ↑↓ ← '
1059 BLOCK#7,2,4,194,18,5:INK#7,6
1060 REPeat Glp
1061  IF gm=0:c1$='05':r1$='07':ELSE c1$=GA(gm,0):r1$=GA(gm,1)
1062  CURSOR#7,136,16:PRINT#7,c1$;'x';r1$:k=CODE(INKEY$(-1))
1063  SELect ON k
1064    =208:gm=gm+1:IF gm>9:gm=9
1065    =216:gm=gm-1:IF gm<0:gm=0
1066    = 10:c$=c1$:r$=r1$:EXIT Glp
1067    = 32,78,110,244:IF sm>0:CLS#7:RETurn
1068  END SELect
1069 END REPeat Glp
1070 IF sm=2
1071    FOR p=0 TO pm:GTemp
1072    DGrid:DIM SGrid(p,cm-1,rm-1):FCls
1073    FOR p=0 TO pm:GLive:fp=1:FCalc:SDraw
1074    p=0:fg=0:gp=1:FCalc:SDraw
1075 END IF
1076 IF sm=1
1077    DGrid:DIM SGrid(pm,cm-1,rm-1),TGrid(17,36,32):FCls
1078    obg=0:nbg=0:col=7:cur=2:pn=0:fp=0:gp=0                    :REMark Set Variables
1079 END IF
1080 IF sm=0:sm=1:DGrid:DIM SGrid(pm,cm-1,rm-1),TGrid(17,36,32)
1081 CLS#7:GPaint
1082 END DEFine
```

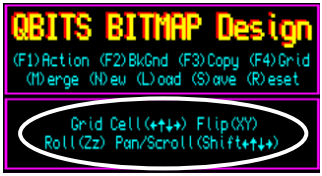


Note: **GA** **G**rid **A**ttributes Identify the Grid Size column **cm** & rows **rm** with cell width **cw** and cell height **rh** which are used to calculate the offsets for the Pointers **cx** & **cy**.

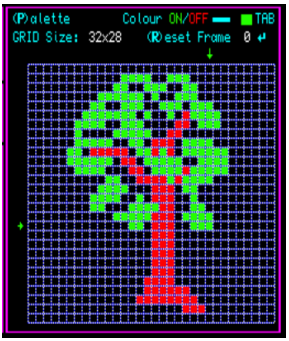



```
1084 DEFine PROCedure DGrid
1085 pm=17:cm=GA(gm,0):rm=GA(gm,1):BLOCK#4,230,172,16,34,0
1086 cw=GA(gm,2):rh=GA(gm,3):cx=GA(gm,4):cy=GA(gm,5)
1087 cur=2:BLOCK#4,220,162,18,36,0:BLOCK#4,30,10,72,14,0
1088 IF cm>rm:cs=2:ELSE cs=0
1089 IF cm=5:c$='05':r$='07':ELSE c$=cm:r$=rm
1090 INK#4,7:CURSOR#4,73,14:PRINT#4,c$;'x';r$
1091 FOR c=0 TO cm:BLOCK#4,1,rm*rh,18+c*cw,36,241
1092 FOR  r=0 TO rm:BLOCK#4,cm*cw,1,18,36+r*rh,241
1093 END DEFine

1095 DEFine PROCedure GTemp
1096 FOR r=0 TO rm-1
1097   FOR c=0 TO cm-1:TGrid(p,c,r)=SGrid(p,c,r)          Temporary Copy of GRID Design
1098 END FOR r
1099 END DEFine
```

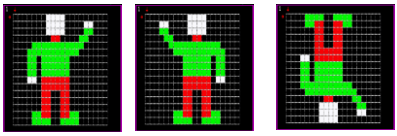
```
1101 DEFine PROCedure GLive
1102 FOR r=0 TO rm-1
1103   FOR c=0 TO cm-1:SGrid(p,c,r)=TGrid(p,c,r)          Update Sprite Frame
1104 END FOR r
1105 END DEFine

1107 DEFine PROCedure FCls
1108 FOR pr=0 TO 2
1109   FOR pc=0 TO 5:BLOCK#5,36,32,7+pc*37,11+pr*33,0     Clear all Frames
1110 END FOR pr
1111 END DEFine

1113 DEFine PROCedure FCalc
1114 IF p<10:p$=p:ELSE p$=CHR$(p+87)
1115 INK#4,7:CURSOR#4,212,14:PRINT#4,p$                  Print Frame ID (0-9 a-h)
1116 IF fp=1                                             WINDOW ch=5
1117    ch=5:pc=p:pr=0                                   Frames 0 - 5
1118    IF p> 5:pc=p -6:pr=33                            Frames 6 - b
1119    IF p>11:pc=p-12:pr=66                            Frames c - h
1120    px=7+INT((36-cm)/2)+pc*37:py=11+INT((32-rm)/2)+pr Calc position px,py
1121 END IF
1122 END DEFine

1124 DEFine PROCedure SDraw
1125 FOR r=0 TO rm-1:FOR c=0 TO cm-1:GDraw:END FOR c:END FOR r   Draw Sprite
1126 fp=0:gp=0:cur=2
1127 END DEFine

1129 DEFine PROCedure GDraw                              SGrid(p,c,r) Pixel Colour
1130 IF gp=1:BLOCK#4,cw-1,rh-1,19+c*cw,37+r*rh,SGrid(p,c,r)   Draw GRID Sprite
1131 IF fp=1 OR fp=3:BLOCK#ch,1,1,px+c,py+r,SGrid(p,c,r)     Draw Frame Sprite
1132 END DEFine

1134 DEFine PROCedure GReset
1135 CLS#7:CURSOR#7,20,6:PRINT#7,'Reset/Clear Grid: ';p$;' (y/n)'   Clear Current Sprite GRID /Frame
1136 IF INKEY$(-1)<>'y':CLS#7:RETurn
1137 FOR r=0 TO rm-1:FOR c=0 TO cm-1:SGrid(p,c,r)=col:END FOR c:END FOR r   col = Background
1138 fp=1:FCalc:BLOCK#5,36,32,7+pc*37,11+pr,0:fp=0:gp=1:SDraw:cur=2:CLS#7
1139 END DEFine

1141 REMark BITMAP GRID Design

1143 DEFine PROCedure GPaint
1144 ch=7:CLS#7:INK#ch,5
1145 CURSOR#ch,48, 8:PRINT#ch,'Grid Cell( ← ↑ ↓ → ) Flip(XY)'
1146 CURSOR#ch,28,18:PRINT#ch,'Roll(Zz) Pan/Scroll(Shift ← ↑ ↓ →' :ch=5
1147 END DEFine
```



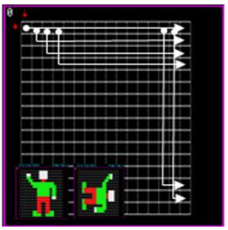
To create an image, locate cell with Cursors, switch Colour **ON** / **OFF** use **TAB** key to cycle for selected colour.



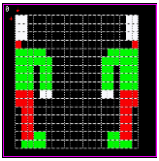
Note: The following Flip, Roll, Slid are activated by Xx,Yy,Zz and Shift Cursor keys. Flip and Slid are Horizontal and Vertical Cell Transfers top to bottom left to right. Roll rotates cell data by ninety degrees

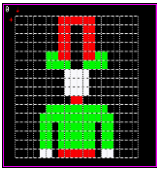


```
1149 DEFine PROCedure GFlip
1150 FOR r=0 TO rm-1
1151   FOR c=0 TO cm-1:TGrid(p,xf+c*zx,yf+r*zy)=SGrid(p,c,r)
1152 END FOR r
1153 GLive:fp=1:gp=1:FCalc:SDraw
1154 END DEFine
```

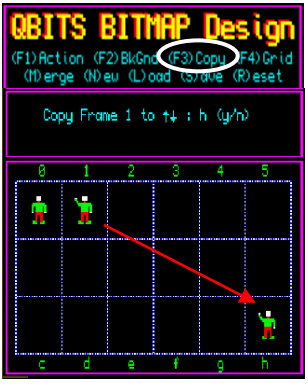


```
1156 DEFine PROCedure GRoll
1157 IF cm=5 OR rm=10:RETurn
1158 FOR r=0 TO rm-1
1159   rx=rxm+(r*rt):ry=rym
1160   FOR c=0 TO cm-1:TGrid(p,c,r)=SGrid(p,0,0)
1161   FOR c=cs TO cs+rm-1:TGrid(p,c,r)=SGrid(p,rx,ry):ry=ry+yt
1162 END FOR r
1163 GLive:fp=1:gp=1:FCalc:SDraw
1164 END DEFine
```





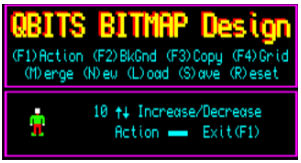
```
1166 DEFine PROCedure GSlid
1167 IF md=0
1168   FOR r=0 TO rm-1
1169     FOR c=1 TO cm-1
1170       TGrid(p,pa,r)=SGrid(p,pb,r):TGrid(p,c+pc,r)=SGrid(p,c+pd,r)
1171     END FOR c
1172   END FOR r
1173 ELSE
1174   FOR r=1 TO rm-1
1175     FOR c=0 TO cm-1
1176       TGrid(p,c,sa)=SGrid(p,c,sb):TGrid(p,c,r+sc)=SGrid(p,c,r+sd)
1177     END FOR c
1178   END FOR r
1179 END IF
1180 GLive:fp=1:gp=1:FCalc:SDraw
1181 END DEFine
```




```
1183 DEFine PROCedure GCopy
1184 CURSOR#7,28,8:PRINT#7,'Copy Frame ';p$;' to ↑ ↓: ';n$;' (y/n)'
1185 REPeat C_lp
1186   IF n<10:n$=n:ELSE n$=CHR$(n+87)
1187   CURSOR#7,154,8:PRINT#7,n$
1188   k=CODE(INKEY$(-1))
1189   SELect ON k
1190     =208:n=n-1:IF n<0:n=0
1191     =216:n=n+1:IF n>pm:n=pm
1192     =89,121:IF n=p:CLS#7:RETurn :ELSE EXIT C_lp
1193     =78,110,240:CLS#7:RETurn
1194   END SELect
1195 END REPeat C_lp
1196 FOR r=0 TO rm-1
1197   FOR c=0 TO cm-1:SGrid(n,c,r)=SGrid(p,c,r)
1198 END FOR r
1199 p=n:fp=1:gp=0:FCalc:SDraw
1200 END DEFine
```

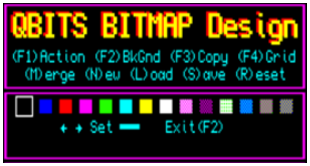
Note: A series of Frames are cycled through with a set delay to give the appearance of an animation.

```
1202 DEFine PROCedure FAction
1203 ch=7:CLS#7:del=10:px=9+INT(32-cm)/2:py=3+INT(28-rm)/2
1204 CURSOR#7,88,6:PRINT#7,'      ↑↓Increase/Decrease'
1205 CURSOR#7,88,18:PRINT#7,'Action      Exit(F1)'
1206 BLOCK#7,16,3,132,22,5:p=0:fp=3:FCalc:SDraw
1207 REPeat Ani_lp
1208   CURSOR#7,70,6:PRINT#7,FILL$(' ',2-LEN(del))&del
1209   k=CODE(INKEY$(-1))
1210   SELect ON k
1211     =208:del=del+2:IF del>20:del=20
1212     =216:del=del-2:IF del< 0:del=-1
1213     = 32:FOR p=0 TO pm:PAUSE del:fp=3:FCalc:SDraw
1214     =232:p=0:CLS#7:EXIT Ani_lp
1215   END SELect
1216 END REPeat Ani_lp
1217 END DEFine
```



Frames Display

```
1219 DEFine PROCedure GBkGnd
1220 CLS#7:FOR i=0 TO 13:BLOCK#7,10,8,10+i*16,3,CP(i)
1221 CURSOR#7,42,14:PRINT#7,' ← → Set      Exit(F2)'
1222 BLOCK#7,16,3,90,17,5:ch=7:xg%=7:x1%=cs%:y1%=1:p1=CP(cs%)
1223 REPeat Bk_lp
1224   CPSel:k=CODE(INKEY$(-1))
1225   SELect ON k
1226     =192:cs%=cs%-1:IF cs%< 0:cs%=13
1227     =200:cs%=cs%+1:IF cs%>13:cs%=0
1228     =236:CLS#7:ch=5:EXIT Bk_lp
1229     = 32:Bk=CP(cs%):CSwap:fp=1:gp=1:FCalc:SDraw:ch=7
1230   END SELect
1231 END REPeat Bk_lp
1232 END DEFine
```

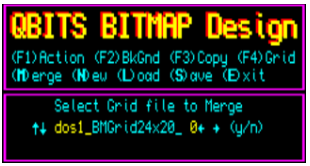



```
1234 DEFine PROCedure CSwap
1235 FOR r=0 TO rm-1
1236   FOR c=0 TO cm-1
1237     IF SGrid(p,c,r)=0 :SGrid(p,c,r)=16
1238     IF SGrid(p,c,r)=Bk:SGrid(p,c,r)=32
1239     IF SGrid(p,c,r)=16:SGrid(p,c,r)=Bk
1240     IF SGrid(p,c,r)=32:SGrid(p,c,r)=0
1241   END FOR c
1242 END FOR r
1243 END DEFine
```

Swap blank Cell ‘ ‘ space with Background colour

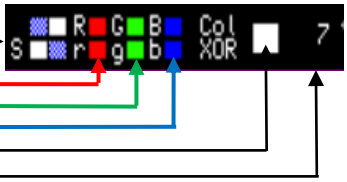
Note: Merges Frames a - h of current GRID Designs with Frames 0 - 9 from a second GRID File:.If of different GRID Sizes Load / Work on smaller GRID Size first. Then change GRID Size and Load second GRID File.

```
1245 DEFine PROCedure GMerge
1246 FOR p=10 TO 17:GTemp
1247 GFSel:IF ck=0:RETurn
1248 FOR p=10 TO 17:GLive
1249 p=0:fp=0:gp=1:FCalc:SDraw
1250 END DEFine
```



```
1252 REMark Colour Palette

1254 DEFine PROCedure CPQL
1255 ch=7:CLS#ch:INK#ch,7
1256 CURSOR#ch,76,16:PRINT#ch,'R  G  B   Col      <TAB>'
1257 CURSOR#ch,46,24:PRINT#ch,'S   r  g  b   XOR      Exit'
1258 CURSOR#ch,4,24:PRINT#ch,'Switch'
1259 FOR i=0 TO 13:BLOCK#ch,10,8,10+i*16,2,CP(i)
1260 xg%=7:x1%=cs%:y1%=1:p1=CP(cs%):cp1=p1:col=cp1
1261 CRead:CPrnt:pck=1:CMode
1262 REPeat CP_lp
1263   IF pck=0:CPSel:ELSE CPrnt:cp1=col
1264   k=CODE(INKEY$(-1))
1265   SELect ON k
1266     =  66:IF BP%=1:BP% =0:ELSE BP%=1
1267     =  98:IF bc% =1:bc%  =0:ELSE bc%=1
1268     =  71:IF GP%=4:GP%=0:ELSE GP%=4
1269     =103:IF gc% =4:gc%  =0:ELSE gc%=4
1270     =  82:IF RP%=2:RP% =0:ELSE RP%=2
1271     =114:IF rc%  =2:rc%   =0:ELSE rc%=2
1272     =83,115:stn%=stn%+1       :IF stn%>3:stn%=0
1273     =192:IF pck=0:cs%=cs% -1:IF cs%<0 :cs%=13
1274     =200:IF pck=0:cs%=cs%+1:IF cs%>13:cs%=0
1275     =208:IF pck=1:cp1  =cp1+1:CRead
1276     =216:IF pck=1:cp1  =cp1 -1:CRead
1277     =  32:CMode
1278     = 9,80,112     :FCalc:EXIT CP_lp
1279   END SELect
1280 END REPeat CP_lp
1281 END DEFine

1283 DEFine PROCedure CPSel
1284 BLOCK#7,14,12,xg%+x1%*16,y1%,0:BLOCK#7,10,8,xg%+2+x1%*16,y1%+2,p1
1285 p1=CP(cs%):x1%=cs%:BLOCK#7,14,12,xg%+x1%*16,y1%,7
1286 BLOCK#7,12,10,xg%+1+x1%*16,y1%+1,0:BLOCK#7,10,8,xg%+2+x1%*16,y1%+2,p1
1287 END DEFine

1289 DEFine PROCedure CMode
1290 IF pck=0
1291   pck=1:cp1=CP(cs%):CRead
1292   CURSOR#ch,6,14:PRINT#ch,' ← →':BLOCK#ch,16,3,16,17,7
1293 ELSE
1294   pck=0:CP(cs%)=cp1:CRead
1295   CURSOR#ch,6,14:PRINT#ch,' ↑  ↓':BLOCK#ch,16,3,16,17,7
1296 END IF
1297 END DEFine

1299 DEFine PROCedure CRead
1300 IF cp1<0:cp1=255
1301 IF cp1>255:cp1=0
1302 stn%=cp1 DIV 64
1303 Con%=cp1 MOD 64 DIV   8:rc% =Pal(Con%,0):gc% =Pal(Con%,1):bc%=Pal(Con%,2)
1304 Maj% =cp1 MOD 64 MOD 8:RP%=Pal(Maj%,0):GP%=Pal(Maj%,1):BP%=Pal(Maj%,2)
1305 END DEFine
```

```
1307 DEFine PROCedure CPrnt
1308 col=(RP%+GP%+BP%)+(rc%+gc%+bc%)*8+stn%*64
1309 r1%=RP%:r2%=rc%:g1%=GP%:g2%=gc%:b1%=BP%:b2%=bc%
1310 IF col<8:r2%=r1%:g2%=g1%:b2%=b1%:st%=3
1311 BLOCK#7,8,6,  56,18,Stp(stn%,0):BLOCK#7,8,6,65,18,Stp(stn%,1)
1312 BLOCK#7,8,6,  56,26,Stp(stn%,2):BLOCK#7,8,6,65,26,Stp(stn%,3)
1313 BLOCK#7,8,6,  84,18, r1%:BLOCK#7,8,6,  84,26,r2%
1314 BLOCK#7,8,6,102,18,g1%:BLOCK#7,8,6,102,26,g2%
1315 BLOCK#7,8,6,120,18,b1%:BLOCK#7,8,6,120,26,b2%
1316 BLOCK#7,12,10,162,20,col
1317 CURSOR#7,180,19:PRINT#7,FILL$(' ',3-LEN(col))&col
1318 END DEFine
```



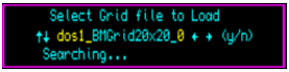
Note: PRINT FILL$(' ',3-LEN(cp1))&cp1;' ';stn%;' gc%;' ';rc%;' ';bc%;' ';GP%;' 'RP%;' 'BP% **7 0000421**

```
1320 REMark QBITS File System
```

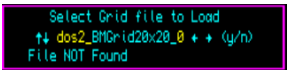
Note: GRID File Select on dg Data Grid Options:
dg=0 Save - as Data or Grid File
dg=1 Load - Grid File
dg=2 Merge- a Second file (Frames a-h) with Frames 0 -9
dg=3 Save - Grid file ; IF (N)ew - Save Frames
dg=4 Save - as a Data file


```
1322 DEFine PROCedure GFSel
1323 REPeat lp
1324 IF dg=0
1325   CURSOR#7,26,13:PRINT#7,'Save as (D)ata or (G)rid file'
1326   IF KEYROW(3)=64:dg=3:a$='Grid ':b$='Save':CLS#7:EXIT lp
1327   IF KEYROW(4)=64:dg=4:a$='Data':b$='Save':CLS#7:EXIT lp
1328 END IF
1329 IF dg=1:a$='Grid':b$='Load'  :EXIT lp
1330 IF dg=2:a$='Grid':b$='Merge':EXIT lp
1331 IF dg=3:a$='Grid':b$='Save'  :EXIT lp
1332 END REPeat lp
1333 CURSOR#7,38,1:PRINT#7,'Select ';a$;' file to ';b$;
1334 CURSOR#7,20,13:PRINT#7,'↑↓     BM';a$;c$;'x';r$;'_    ←→(y/n)'
1335 INK#7,6
1336 REPeat Sel_lp
1337   CURSOR#7,38,13:PRINT#7,Drv$(dn%):CURSOR#7,146,13:PRINT#7,pn
1338   k=CODE(INKEY$(-1))
1339   SELect ON k
1340     =192:pn=pn-1:IF pn<0:pn=0                    :REMark File  0-9
1341     =200:pn=pn+1:IF pn>9:pn=9
1342     =208:dn%=dn%+1:IF dn%>15:dn%=0                :REMark Device mdv1_ Dos1_
1343     =216:dn%=dn%-1:IF dn%<0:dn%=15
1344     =89,121:ck=1:SFile$='BM'&a$&c$&'x'&r$&'_'&pn:EXIT Sel_lp
1345     =78,110:INK#7,5:CLS#7:RETurn
1346   END SELect
1347 END REPeat Sel_lp
1348 IF dg=1 OR dg=2:GFChk:GLoad:CLS#7
1349 IF dg=3 OR dg=4:GFChk:GSave:CLS#7
1350 END DEFine
```

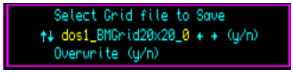
```
1352 DEFine PROCedure GFChk
1353 INK#7,5:CURSOR#7,20,24:PRINT#7,'  Searching...  '
1354 PAUSE 20:DELETE Drv$(dn%)&'FList'
1355 OPEN_NEW#9,Drv$(dn%)&'FList':DIR#9,Drv$(dn%):CLOSE#9
1356 OPEN_IN#9,Drv$(dn%)&'FList'
1357 REPeat dir_lp
1358   IF eck=1 OR EOF(#9):CLOSE#9:ck=0:EXIT dir_lp
1359   INPUT#9,CFile$:IF CFile$==SFile$:CLOSE#9:ck=1:EXIT dir_lp
1360 END REPeat dir_lp
1361 END DEFine
```

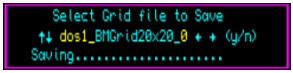



```
1363 DEFine PROCedure GLoad
1364 IF ck=0:CURSOR#7,20,24:PRINT#7,'File NOT Found':PAUSE 50:RETurn
1365 OPEN_IN#9,Drv$(dn%)&SFile$:CURSOR#7,20,24:PRINT#7,'  Loading...';:CLS#7,4
1366 PAUSE 20:fp=1:gp=0:INPUT#9,pm\rm\cm:pt=pm:IF dg=2:pm=9
1367 FOR p=0 TO pm
1368   PRINT#7,'.';:FCalc
1369   FOR r=0 TO rm-1
1370     FOR c=0 TO cm-1:INPUT#9,SGrid(p,c,r):GDraw
1371   END FOR r
1372 END FOR p
1373 CLOSE#9:pm=pt:p=0:FCalc:fp=0:gp=1:SDraw
1374 END DEFine
```




```
1376 DEFine PROCedure GSave
1377 IF eck=1:
1378   eck=0:CURSOR#7,20,24:PRINT#7,'DEVICE ERROR  ':PAUSE 50:RETurn
1379 END IF
1380 IF ck=1
1381   CURSOR#7,20,24:PRINT#7,'   Overwrite (y/n)'
1382   IF INKEY$(-1)=='y':DELETE Drv$(dn%)&SFile$:ELSE RETurn
1383 END IF
1384 OPEN_NEW#9,Drv$(dn%)&SFile$:CURSOR#7,20,24:PRINT#7,'Saving...';:CLS#7,4
1385 PAUSE 20:num=2000
1386 IF dg=4:PRINT#9,num&' DATA '&pm&','&rm&','&cm:ELSE PRINT#9,pm\rm\cm
1387 FOR p=0 TO pm
1388   PRINT#7,'.';:IF dg=4:num=num+1:PRINT#9,num&' :'
1389   FOR r=0 TO rm-1
1390     IF dg=4
1391       num=num+1:PRINT#9,num&' DATA ';
1392       FOR c=0 TO cm-1:PRINT#9,SGrid(p,c,r);',';
1393       PRINT#9,SGrid(p,cm-1,r)
1394     ELSE
1395       FOR c=0 TO cm-1:PRINT#9,SGrid(p,c,r)
1396     END IF
1397   END FOR r
1398 END FOR p
1399 CLOSE#9:p=0
1400 END DEFine
```



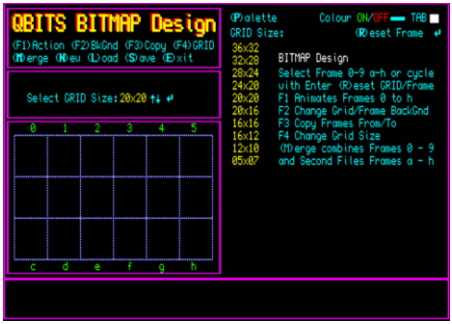


1402 REMark **QBITS BITMAP SetUp**


```
1404 DEFine PROCedure Init_Win
1405 WINDOW#0,512, 32,gx,gy+224:PAPER#0,0:BORDER#0,1,3:CLS#0
1406 WINDOW#1,496,216,gx+8,gy+2:PAPER#1,0
1407 WINDOW#2,512,224,gx,gy       :PAPER#2,0:BORDER#2,1,3:CLS#2
1408 OPEN#4,scr_:WINDOW#4,251,212,gx+252,gy+ 4:BORDER#4,1,3
1409 OPEN#5,scr_:WINDOW#5,240,122,gx+ 10,gy+94:BORDER#5,1,3
1410 OPEN#6,scr_:WINDOW#6,240,  48,gx+ 10,gy+ 4:BORDER#6,1,3
1411 OPEN#7,scr_:WINDOW#7,240,  38,gx+ 10,gy+54:BORDER#7,1,3
1412 Init_Menu:Init_Grid:Init_CPQL:BITMAP_Intro
1413 END DEFine
```


```
1415 DEFine PROCedure Init_Menu
1416 ch=6:CSIZE#ch,2,1:OVER#ch,1:INK#ch,6
1417 INK#ch,2:FOR i=0 TO 1:CURSOR#ch,1+i,1:PRINT#ch,'QBITS BITMAP Design'
1418 INK#ch,6:FOR i=0 TO 1:CURSOR#ch,3+i,2:PRINT#ch,'QBITS BITMAP Design'
1419      CSIZE#ch,0,0:OVER#ch,0:INK#ch,5
1420 CURSOR#ch,2,24:PRINT#ch,'(F1)Action (F2)BkGnd (F3)Copy (F4)GRID'
1421 CURSOR#ch,2,34:PRINT#ch,'(M)erge (N)ew (L)oad (S)ave (E)xit':OVER#ch,1
1422 CURSOR#ch,3,34:PRINT#ch,' M       N     L      S      E'    :OVER#ch,0
1423 END DEFine

1425 DEFine PROCedure Init_CPQL
1426 DIM CP(13)         :RESTORE 1428          :REMark Colour Palette 0-11
1427 FOR cr=0 TO 13  :READ c:CP(cr)=c
1428 DATA   0,1,2,3,4,5,6,7,227,216,31,225,251,254
1429 DIM Pal(7,3)       :RESTORE 1431          :REMark Colour/Contrast
1430 FOR c=0 TO 7    :READ r,g,b:Pal(c,0)=r:Pal(c,1)=g:Pal(c,2)=b
1431 DATA 0,0,0,0,0,1,2,0,0,2,0,1,0,4,0,0,4,1,2,4,0,2,4,1
1432 DIM Stp(3,3)       :RESTORE 1436          :REMark Stipple 0-3
1433 FOR s1=0 TO 3
1434    FOR b1=0 TO 3:READ s:Stp(s1,b1)=s
1435 END FOR s1
1436 DATA 7,241,7,7,7,7,241,241,241,7,241,7,7,241,241,7
1437 END DEFine
```

```
1439 DEFine PROCedure Init_Grid
1440 LOCal a,b,c:cs%=0:x=0:cx=0:y=0:cy=0:pn=0:fp=0:gp=0
1441 DIM a$(4),b$(5),c$(2),r$(2),p$(2),n$(2)
1442 DIM TGrid(17,32,28),SFile$(20),CFile$(20)
1443 DIM GA(9,5):RESTORE 1446
1444 FOR a=0 TO 9:FOR b=0 TO 5:READ c:GA(a,b)=c:END FOR b:END FOR a
1445 DATA 5,7,18,16,25,13     ,12,10,16,14,20,12          :REMark  5x7  12x10
1446 DATA 16,12,13,12,19,11 ,16,16,12,10,19,11           :REMark 16x12 16x16
1447 DATA 20,16,11,9,19,11   ,20,20,10,8,19,10           :REMark 20x16 20x20
1448 DATA 24,20,9,8,19,10    ,28,24,8,7,19,10            :REMark 24x20 28x24
1449 DATA 32,28,7,6,19,10    ,36,32,6,5,19,10            :REMark 32x28 36x32
1450 ch=4:       REMark GRID Window
1451 INK#ch,5:CURSOR#ch,   2,2:PRINT#ch,'(P)alette      Colour          TAB'
1452 INK#ch,4:CURSOR#ch,150,2:PRINT#ch,'ON'  :BLOCK#ch,16,3,166,6,2
1453 INK#ch,2:CURSOR#ch,184,2:PRINT#ch,'OFF' :BLOCK#ch,10,7,210,4,7
1454 INK#ch,5:CURSOR#ch,  4,14:PRINT#ch,'GRID Size:       (R)eset Frame  ↵'
1455 OVER#ch,1:CURSOR#ch,9,2:PRINT#ch,'P'
1456 CURSOR#ch,131,14:PRINT#ch,'R':OVER#ch,0 :BLOCK#ch,2,4,232,16,5
1457 ch=5:       REMark Frame WIndow
1458 FOR i=0 TO 3:BLOCK#ch,222,1,6,10+i*33,241
1459 FOR i=0 TO 6:BLOCK#ch,1,100,6+i*37,10,241
1460 FOR i=0 TO 5:CURSOR#ch,24+i*36,0:PRINT#ch,i
1461 FOR i=0 TO 5:CURSOR#ch,24+i*36,110:PRINT#ch,CHR$(i+99)
1462 END DEFine
```




```
1464 DEFine PROCedure BITMAP_Intro
1465 RESTORE 1478   :BLOCK#4,232,180,8,26,0:INK#4,7
1466 CURSOR#4,60, 34:PRINT#4,'Note: BITMAP Design':INK#4,5
1467 CURSOR#4,60, 46:PRINT#4,'(R)eset GRID/Frame : Use Enter'
1468 CURSOR#4,60, 56:PRINT#4,'to cycle Frames or key 0-9 a-h'
1469 CURSOR#4,60, 66:PRINT#4,'(P)alette & TAB change Colour'
1470 CURSOR#4,60, 86:PRINT#4,'F1 Animates Frames  0 >>>> h'
1471 CURSOR#4,60, 96:PRINT#4,'F2 Change Grid/Frame BackGnd'
1472 CURSOR#4,60,106:PRINT#4,'F3 Copy Frames From >---> TO'
1473 CURSOR#4,60,116:PRINT#4,'F4 Change Grid Size ??x?? ↑↓ '
1474 CURSOR#4,60,136:PRINT#4,'(M)erge  combines Frames 0-9'
1475 CURSOR#4,60,146:PRINT#4,' with Second File Frames a-h'
1476 CURSOR#4,60,166:PRINT#4,'Use upper-case "E" to (E)xit'
477 INK#4,6:FOR i=1 TO 10:READ str$:CURSOR#4,8,16+i*10:PRINT#4,str$
1478 DATA '36x32','32x28','28x24','24x20','20x20'
1479 DATA '20x16','16x16','16x12','12x10','05x07'
1480 END DEFine
```

QBITS BITMAPS Storage
Each BITMAP file opens with the Format settings 0-17 Frames, GRID size in columns x Rows. Then beginning with Frame 1 Grid position 0/0 through to Colum max and Row max lists the Pixel square Colour Settings

QBITS BITMAP_v3 Filename - QBSGrid20x20_0-9 or QBSData20x20_0-9 (**S**tipple Colours)
QBITS BITMAP_vCP24 Filename - QBTGrid20x20_0-9 or QBTData20x20_0-9 (**T**rue Colours)

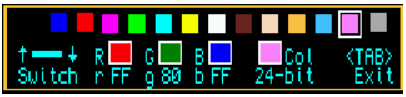
QBITS BITMAP True Colour
The CP24 Code version was an opportunity not to be missed so as to appreciate the QPC **COLOUR_24** MODE. The main coding differences being driven by the **Colour Palettes. QBITS BITMAPS_v3** uses the 8-bit **COLOUR_QL** MODE with colour values 0-255. Setting for Windows BORDER, PAPER and INK etc. The 24-bit True Colour require values between 0-16,777,215 or 00000 to FFFFFF.

QBITS BITMAP Palette 24
For QPC **COLOUR_24**, **RED**, **GREEN** & **BLUE** each have a range of 0 - 255 or Hex 00 - FF with RED occupying the highest Byte, **GREEN** the Middle Byte and **BLUE** the lowest Byte. The BITMAPS Palette of 0-13 Colours are Selected with Left/Right cursors as before with Spacebar switching between Palette and Colour Mix. Upper/Lower case keys **R, r**, **G, g, B, b** Increase/Decrease the values for **Red Green Blue** colours.

ie. Black=0, **Red**=$FF0000, **Green**=$FF00 **Blue**=$FF and **White**=$FFFFFF

num =0 - 16,777,215 or in Hex 0 - $FFFFFF

Red = num DIV 65536
Green = num MOD 65536 DIV 256
Blue = num MOD 65536 MOD 256



The Default settings for the Colour 24 Palette values [0 to 13] **RP% GP% BP%**

$00,$00,$00 Black	$00,$00,$FF BLUE	$FF,$00,$FF Magenta
$FF,$00,$00 RED	$00,$FF,$00 GREEN	$00,$FF,$FF Cyan
$FF,$FF,$00 Yellow	$FE,$FE,$FE Off White	$92,$92,$92 Grey
$FF,$92,$00 Orange	$FF,$B6,$DB Pink	$B6,$6D,$6D Brick
$6D,$24,$24 Brown	$00,$49,$92 Ultramarine	

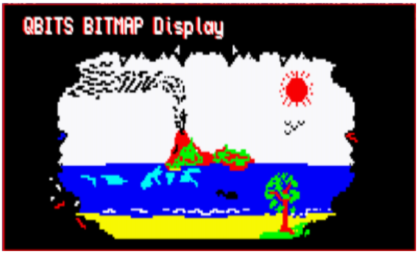
Note: Hex('FFFFFF') returns 1.677722E7 whereas HEX$(1.677722E7,24) return 000004 not FFFFFF as expected. This is to do with the way floating point numbers are rounded up. Setting the upper limit for each of the Colours to $FE provides a simple resolve this problem with a minimum loss to setting a true colour.

QBITS Colour Palette Conversion
Swapping a **COLOUR_QL Palette** number for an equivalent **COLOUR_24** or visa versa with the basic Colour set 0 to 7 is relatively straightforward, but is much more complex when interpreting Colour, Contrast and Stipple combinations.

If this is desired my suggestion is load the File into an array where values are compared wth the contents of a look up table to work out the desired RGB / Contrast / Stipple equivalents.

QBITS Background Display

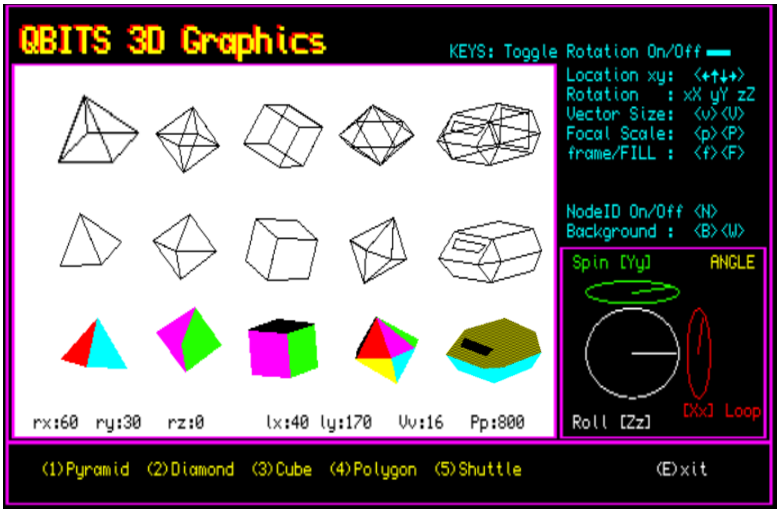
Saving the 18 frames as a Data File, this Prog will Merge the file; Dimension an Array into which it will READ the Lines of DATA; then Draw the Background scene in Pixel Size (**en**=1) or enlarged to twice the Pixel Size (**en**=2).




```
100 REMark QBITS_BMDisplay (QBITS BITMap DATA Display v01 2019)

102 km=4:en=2:Init
103 MERGE dos1_QBPData32x28_0              :REMark Collect DATA File info
104 DLoad

106 DEFine PROCedure Init
107 WINDOW#1,496/(km/4),212,6,6:BORDER#1,1,2:PAPER#1,0:CLS#1
108 CSIZE#1,2,1:OVER#1,1:INK#1,4
109 INK#1,2:FOR i=0 TO 1:CURSOR 20+i,8+i:PRINT 'QBITS BITMAP Display'
110 INK#1,7:FOR i=0 TO 1:CURSOR 22+i,8+i:PRINT 'QBITS BITMAP Display'
111 CSIZE#1,0,0:OVER#1,0
112 CLS#0:CURSOR#0,20,6:PRINT#0,'Reading DATA...'
113 END DEFine

115 DEFine PROCedure DLoad
116 num=2000:RESTORE num:READ pm,rm,cm:DIM DGrid(pm,cm-1,rm-1)
117 FOR p=0 TO pm
118   FOR r=0 TO rm-1
119     FOR c=0 TO cm-1:READ a$:DGrid(p,c,r)=a$
120   END FOR r
121 END FOR p
122 CLS#0:BLOCK 2+192*en,2+en*84,49,29,7:FOR p=0 TO pm:DCal:DDraw
123 END DEFine

125 DEFine PROCedure DCal
126 ch=1:pc=p:pr=0
127 IF p> 5:pc=p -6:pr=28*en                :REMark Image Size en=1 or 2
128 IF p>11:pc=p-12:pr=56*en
129 px=50+pc*32*en
130 py=30+pr
131 END DEFine

133 DEFine PROCedure DDraw
134 FOR r=0 TO rm-1
135   FOR c=0 TO cm-1:BLOCK#ch,en,en,px+c*en,py+r*en,DGrid(p,c,r)
136 END FOR r
137 END DEFine
```

Merge dos1_QBPData32x28_0

```
2000 DATA 17,28,32                          REMark  pm, rm, cm
2001 :
2002 DATA 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
2003 DATA 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
2004 DATA 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0  Continue…to End of Bitmap Data File
```



3D Graphics Introduction

Those early days of seeking the illusion of movement, Magic Lanterns with their hand drawn cards, later replaced with photographic stills. In 1878 British photographer Eadweard Muybridge created the zoopraxiscope, which projected up to 200 sequence photographs creating the illusion of movement in real-time. The Kinetoscope (a peep-hole motion picture viewer) introduced at the US Chicago World Fair of 1893 was developed by W. K. L. Dickson an employee of Edison. In France the Lumiere brothers made their first public screening of ten short films in 1895 and as they say the motion picture industry was born.

In much the same way the availability of home computing in the 1980's sparked myself and others interest in creating moving graphics, especially those involving the manipulation of 3D images. Unfortunately, back in my early days of programming such things as 3D Graphics were a little out of my league and probably still are. At the time I did jot down some notes amongst my future aspirations. Having renewed my interest in QL SuperBASIC, I thought it was time to give 3D Graphics a bit of a spin (sorry for the pun).

QBITS 3D Graphics

My goal, the basic code for creating a simple 3D Wireframe object. Repositioning it around the screen, ability to zoom in and out, plus the prize, rotational movement. In a two-dimensional screen objects drawn with the illusion of Perspective. That has something to do with focal scale, but more about that later. As a finishing touch show only visible surfaces of the wireframe to create what appears to be a solid object and then add coloured surfaces.

Depending on what source you refer to or your own background you might come across a few variations on the terms used for rotating a three-dimensional object. The most common being Roll, Pitch and Yaw associated with flying. I thought of others Circulate, Orbit, Spin, Loop. For QBITS 3D Graphics, I decided on Roll, Spin & Loop. All conveniently four-letter words, giving a little conformity in computer programming always being a good thing.

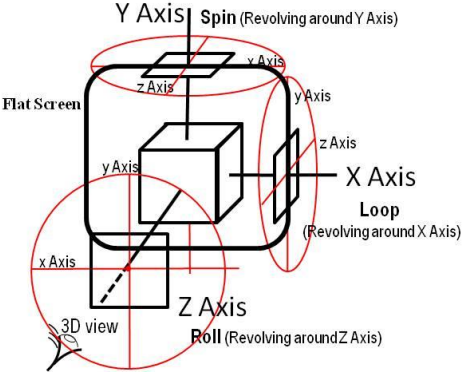
QBITS Exploring 3D Graphics

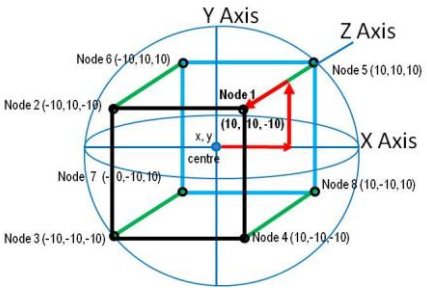
Where to start... the location of a two-dimensional object uses xy coordinates. Moving position, alters the xy coordinate values, a number of x points to Left or Right across the screen and a number of y points up or down the screen. When an object is moved to a new position, without changing its shape or size, this is a translation.

A three-dimensional object requires an additional coordinate, usually designated as z. Rotation of an object changes the orientation within each of its relative axis. This alters its shape and size and is known as a transformation. Converting three-dimensional objects onto a two-dimensional plane requires conversion of 3D coordinates into 2D coordinates. The coding for such requires a number of steps and Yes it involves some basic trigonometry.

Imaginary Eye View

Reviewing the diagram opposite it is easy to identify x y coordinates on a flat screen. For a three-dimensional object the z coordinates lie in front and behind the screen. Using a Cube as our object, half is sticking out from the front of the screen surface, the other half lying behind.





QBITS Nodes xyz Coordinates

Be it a simple Cube or multisided polyhedron, each point of reference for a 3D Object, will be identified as a **Node**. These describe the Objects coordinates for a Wireframe referenced to each of its three axis xyz.

The centre of the Cube is given as the x,y zero position. Following the red arrows Node (1) is shown as x+10 on X axis, y+10 on the Y axis and looking down from above it lies in front of the screen on the Z axis. This places the object closer so is given a value of z -10.

```
                        DATA  8          :REMark Number of Nodes
Node (1) xyz is  10, 10,-10   DATA  10, 10,-10
Node (2) xyz is -10, 10,-10   DATA -10, 10,-10
Node (3) xyz is -10,-10,-10   DATA  10,-10,-10
Node (4) xyz is  10,-10,-10   DATA  10,-10,-10
Node (5) xyz is  10, 10, 10   DATA  10, 10, 10
Node (6) xyz is -10, 10, 10   DATA -10, 10, 10
Node (7) xyz is -10,-10, 10   DATA -10,-10, 10
Node (8) xyz is  10,-10, 10   DATA  10,-10, 10
```

As a set of DATA lines, the above can be used for basic configuration information. This will apply not only to our Cube, but with any polyhedron and its multiple Nodes.

.

QBITS Vector Calculations

A Vector is a distance in a particular direction. For QBITS 3D Graphics, Vectors represent an Object Nodes relating to its wireframe image. **Vectors vx,vy** are the screen coordinates derived from a central location **lx,ly** and calculated from an Objects Node **xyz** coordinates.

In creating a 3D Object trigonometry is used to find the position of a rotating point (x y) set around a central origin at a distance (r) and by degrees (a).

x = r * COS(a)
y = r * SIN (a)

If we then rotate further the angle to b:

x' = r * COS(α + b)
y' = r * SIN (α + b)

By using trigonometric addition of each equation:

x' = r * COS(a) COS(b) - r * SIN (a) SIN(b)
y' = r * SIN(a) COS(b) + r * COS(a) SIN(b)

Then substituting in the values for x and y above, we get an equation for the new coordinates as a function of the old coordinates and extended angle of rotation:

x' = x × COS(b) - y * SIN(b)
y' = y × COS(b) + x * SIN(b)

The above describes one plane we have three XYZ. By combining the required function for COS and SIN of the angle it can be used with each plane:

ra=+.5 : c = COS(ra) : s = SIN(ra)

Then the code for position in each plane is as follows:

yt = y : y = c * yt – s * z : z = s * yt + c * z X axis (y, z planes)
xt = x : x = c * xt + s * z : z = s * xt + c * z Y axis (x, z planes)
xt = x : x = c * xt – s * y : y = s * xt + c * y Z axis (x, y planes)

Where yt, xt hold the previous x, y coordinate values. The x y z are updated with new values. Following this the 3D coordinates are transposed into 2D screen positions with the following:

vx = lx + (x * fs) / (z + fs)
vy = ly + (y * fs) / (z + fs)

Variables lx,ly are the central location coordinates, fs is a factor of scale, that determines the zoom in or out from an imaginary view point. Imagine a large building as seen from a distance, it has a fairly uniform shape. Yet standing beneath one corner, the height immediately above as opposed to the height of the building further down the street appears out of proportion to its true measurement. This is **Perspective,** the appearance of things relative to one another determined by their distance and is the technique of displaying 3D objects on a two-dimensional surface. Rotation of an Object that has a low fs can distort and extend the image into weird shapes.

The Vector vx(n) and vy(n) calculation for each Nodes screen coordination can be set within a FOR loop and stored in a Dimensioned Array.

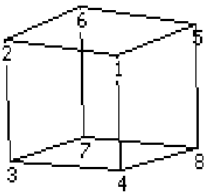
DIM vx(n),vy(n) where n is the total number of Nodes

QBITS Frames

Frames are the linking of **Vectors** to create a wireframe. As with any polyhedron first comes the number of **Nodes** and their **xyz** values from which **Vector** values **vx, vy** can be calculated. Then each **Frame** and its group of **Nodes** / **Vector** coordinates can be identified and linked.

For example, a **Cube** has six **Frames** and eight **Nodes**:

	DATA 6	Frames
Frame (1) Vector a - b - c - d	DATA 8,7,6,5	Nodes 8-7-6-5
Frame (2) Vector a - b - c - d	DATA 2,6,7,3	
Frame (3) Vector a - b - c - d	DATA 4,3,7,8	
Frame (4) Vector a - b - c - d	DATA 5,1,4,8	
Frame (5) Vector a - b - c - d	DATA 5,6,2,1	
Frame (6) Vector a - b - c - d	DATA 1,2,3,4	Nodes 1-2-3-4

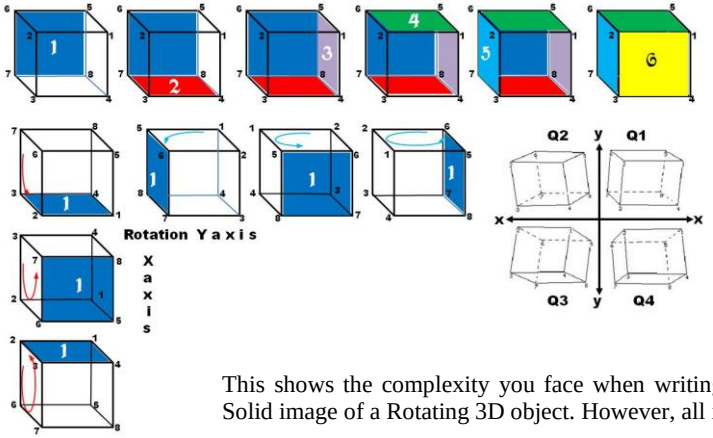


The DATA Set provides the means to identify linked Nodes in constructing the 3D Objects planes (sides) of the Wireframe using the LINE command.

QBITS Wireframe to Solid Objects

A **Frame** is by definition a closed area, therefore it can be left unfilled as a Wireframe or coloured in with FILL to create a Solid Object. The problem arise as to what is visible and what is out of view as seen from the viewpoint.

The frame sequence above loads those Frames hidden from view first with the ones covering the viewed surfaces last. The problem is as the Object is rotated away from initial settings in any of its three axes then the sequence of Frames hidden from view and those that come into view will change. The Cube images below show the initial load and display of Frame surfaces and the back frame as it **Spins** and **Loops** to different positions on screen.



This shows the complexity you face when writing code to display the Solid image of a Rotating 3D object. However, all is not lost…

In Exploring 3D Rotation Graphics, I have used planar polygons of which each Frame surface has a unique property. It has two sides, one which looks internally and the other outwardly. Therefore, by determining the outward direction of a frames surface we can then use this to identify if it is pointing away or towards our view point.

QBITS Hidden Surface Removal

When determining hidden surface removal there are two basic types, Object-space, used for three-dimensional processing and Image-space, used for two-dimensional processing. What is needed to remove an Objects hidden surfaces (Frames) is an algorithm that identifies those surfaces that are not seen from the view point. The most common computing method used to carry out this action is called the Plane Equation Method.

In simple terms a Vector Normal is computed for a Frame surface such that its value indicates whether it is facing away from or towards the viewer. QBITS Prog uses the counter or anti clockwise coordinates system for defining Frames. This is known as the left-handed rule for the Plane Equation see below. (The is an alternative called the right-handed or clockwise system.)

This is based on the following equation: Ax+By+Cz+D=0

where the Vector Normal (N) to the plane is N=[A B C]
and where C > 0 is a surface facing away.
and where C<=0 is a surface facing towards the view point.

Obtaining the Vector Normal is based on the plane passing through three points:

P1= (x1, y1, z1), P2= (x2, y2, z2), P3= (x3, y3, z3):

x – x1 y1 – y1 z – z1
x2 – x1 y2 – y1 z2 – x1 = 0
x3 – x1 y3 – y1 z3 – x1

This is equivalent to: Ax+By+Cz+D=0

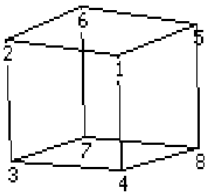
and where C = (x2 – x1) * (y3 – y1) – (x3 – x1) * (y2 – y1)

C is the value we are interested in to determine the outward facing direction of the Frame surface and whether it is facing towards or away from the view point.

QBITS Anti Clockwise Coordinate system

Going back to our Frame DATA lists you will notice that the Node ID's for the front facing surface are 1,2,3,4 and are ordered in an anticlockwise manner. The back face 5,6,7,8 in the DATA list is ordered as 8,7,6,5 or clockwise. However, if you were to view this surface rotated 180 degrees to the front 8,7,6,5 is counting in an anticlockwise direction.

DATA 8,7,6,5,bg2 :REMark back Frame [bg2 = INK colour]
DATA 2,6,7,3,2
DATA 4,3,7,8,4 **Note:** Frame [Four Nodes + Surface Colour]
DATA 5,1,4,8,3
DATA 5,6,2,1,5
DATA 1,2,3,4,bg2 :REMark front Frame



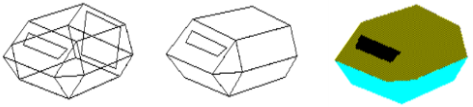
The Vector Normal for Frame surface P1, P2, P3 from the equation are substituted with three Frames Node xy coordinates. In this case x(a), y(a) - x(b), y(b) - x(c), y(c)

C=(x(b)-x(a)) * (y(c)-y(a))-(x(c)-x(a)) * (y(b)-y(a)) [C > 0 Hidden Frame Removal]

QBITS 3D Object Mode

When all of the Wireframe is drawn including hidden planes, cull setting **cset=1**. For a Solid Frame where hidden frames are culled **cset=2**. If a Solid Wireframe is displayed and Frame surfaces are coloured **cset=3**. Surface **col**ours are the fifth value of a **Frame** DATA line.

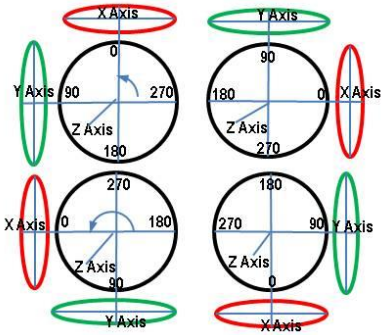
DATA 1,2,3,4,col



QBITS Rotation

Rotary movement is a change of angle in one of the three planes **Roll/Spin/Loop**. Toggle random **Rotation** On/Off with <**Spacebar**>. To **Loop/Spin/Roll** use < **xXyYzZ** > keys, once an Object has been Rotated from its initial position the Roll/Loop and Spin key commands can act differently to what maybe expected.

If **Z,X,Y** axis is changed, direction of rotation is altered in the other planes. An example of this is where the actions of **xX** (Loop) and **yY** (Spin) can be reversed. As shown in the example opposite Rotation around the Z axis changes the actions of Spin and Loop as it moves through each quadrant. Hopefully my diagram explains this better than I can put into words.



QBITS Relocation

Change Objects **lx,ly** coordinates with <←↑↓→>

QBITS Perspective

Use keys **Pp** to increase or decrease values from between 80 and 800. Low values distort and extends parts of the Object as it is Rotated. Values below 80 are just too weird.

QBITS Vector Size

Use keys **Vv** to Enlarge or Reduce the size of an Object. The process of reading and storing the 'Nodes xyz' values gave me the idea of adding a multiplier that could be changed in a uniform manner. Vector size '**vs'** has a range from 0.5 to 1.5 in 0.1 increments.

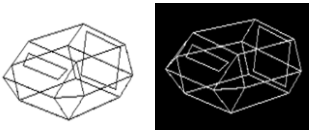
QBITS Node ID

At this point it would seem logical to include the ability to identify the Nodes displayed in their screen positions as part of an Objects image. Pressing the **N** key toggles On/Off **nset,** which actions the print of Node ID's. For this I make use of the CURSOR graphics coordinate system:

IF **nset**=1 :CURSOR vx(**n**),vy(**n**),-2,2 :PRINT **n** (**n** being the Node number)

QBITS Background

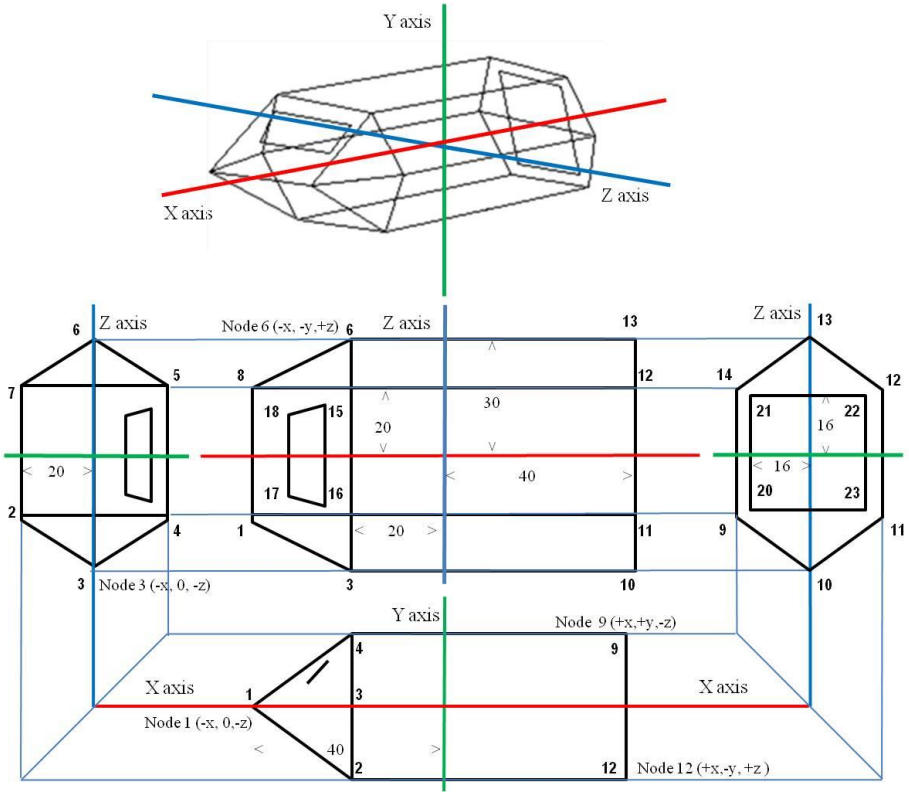
Pressing **B** or **W** keys changes PAPER colour (bg1) and INK (bg2) either a Black background with white INK, or White background with black INK. Solid Objects use INK for a Frame Colour.



QBITS Wireframe Design

Expanding on the simple Wireframe objects, I decided to include one I call the Space Shuttle. The three-dimensional sketch is drawn showing the different planes overhead, front, rear and side elevation. These are linked with the XYZ planes to identify the Nodes (xyz) and their relevant units of distance +/- values from the central coordinates lx,ly. Each Nodes xyz value can then be entered into a Data List.

The layout design for the QBITS Space Shuttle.

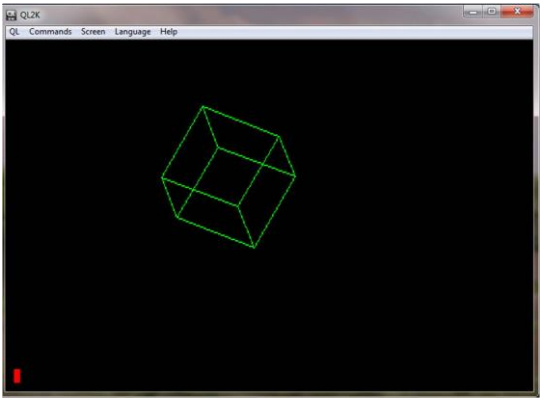


The Frame Data list are READ and used by the Plane Equation to determine if the outward surface of the polygon is facing towards or away from point of observation. It is therefore important they are ordered correctly, that is in an anticlockwise direction.

New Object DATA Lines can be added and called by adding or making changes to PROCedure **Obj_Name** by adding or changing an Objects number (**?**) & **Name.** In **Obj_Init** set RESTORE points for Node '**nres**' & Vector '**vres**' DATA Lines. To understand DATA Format for Node and Frame settings, check out Objects DATA Lines included with this Program.

The basic Program for 3D Rotation Graphics

```
100 REMark QB3D_Cube (Rotating Cube)

104 MODE 4:WINDOW 512,200,0,0:PAPER 0:INK 4:CLS:SCALE 100,0,0
106 DIM x(8),y(8),z(8),vx(8),vy(8)
108 vl=16:fs=10000:ra=0.1                    :REMark Vector length : Focal point : Rotation angle

112 CLS

114 x(1)=-vl:y(1)=-vl:z(1)=-vl               :REMark Nodes 1 - 8
116 x(2)=-vl:y(2)=+vl:z(2)=-vl
118 x(3)=+vl:y(3)=+vl:z(3)=-vl
120 x(4)=+vl:y(4)=-vl:z(4)=-vl
122 x(5)=-vl:y(5)=-vl:z(5)=+vl
124 x(6)=-vl:y(6)=+vl:z(6)=+vl
126 x(7)=+vl:y(7)=+vl:z(7)=+vl
128 x(8)=+vl:y(8)=-vl:z(8)=+vl

132 ra=ra+0.1:c=COS(ra):s=SIN(ra)             (0.1 increments)

136 FOR np=1 TO 8
138   REMark Rotation on X Axix
140   yt=y(np):y(np)=c*yt-s*z(np):z(np)=s*yt+c*z(np)
142   REMark Rotation on Y Axis
144   xt=x(np):x(np)=c*xt+s*z(np):z(np)=s*xt+c*z(np)
146   REMark Rotation on Z Axis
148   xt=x(np):x(np)=c*xt-s*y(np):y(np)=s*xt+c*y(np)
150   REMark Points Projections and Translations to Screen Coordinates
152   vx(np)=80+(x(np)*fs)/(z(np)+fs)
154   vy(np)=50+(y(np)*fs)/(z(np)+fs)
156 END FOR np

160 LINE vx(1),vy(1) TO vx(2),vy(2)          :REMark Vectors - Draws A Cube
162 LINE vx(2),vy(2) TO vx(3),vy(3)
164 LINE vx(3),vy(3) TO vx(4),vy(4)
166 LINE vx(4),vy(4) TO vx(1),vy(1)
168 LINE vx(5),vy(5) TO vx(6),vy(6)
170 LINE vx(6),vy(6) TO vx(7),vy(7)
172 LINE vx(7),vy(7) TO vx(8),vy(8)
174 LINE vx(8),vy(8) TO vx(5),vy(5)
176 LINE vx(1),vy(1) TO vx(5),vy(5)
178 LINE vx(2),vy(2) TO vx(6),vy(6)
180 LINE vx(3),vy(3) TO vx(7),vy(7)
182 LINE vx(4),vy(4) TO vx(8),vy(8)

186 PAUSE 5

188 GO TO 112
```

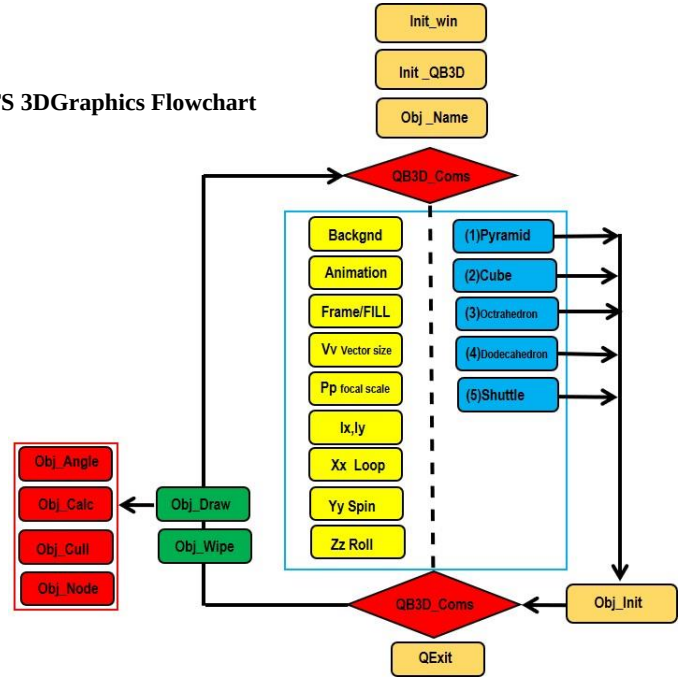


QBITS 3DGraphics Procedures

Set up of Screen windows and common variables
+ Data sets for Pyramid, Cube, Hexagon, Shuttle

Inir_win	Set Window attributes
Init_QB3D	Sets screen layout and KEY information.
QExit	Exit Prog call QBITSProgs Menu .
QB3D_Coms	Serves as main Menu to functions.
lx,ly	Screen Location
V v	Enlarge / Reduce Object Vector size
P p	Increase / Decrease Perspective (focal scale)
f F	Toggle Wireframe (default) <**f** >Solid frame <**F**>Surface FILL
XxYyZz	Loop / Spin / Roll Object
B W	Change Screen background (**B**lack/**W**hite)
Obj_Node	Loads Node xyz and sets vector size.
Obj_Auto	Sets auto Loop/ Spin/ Roll of Object
Obj_Ang	Updates and Draws Loop/ Spin/ Roll Angle Graphics.
Obj_Calc	Calculates new vx,vy coordinates of Object.
Obj_Draw	Draws Object to screen.
Obj_Wipe	Wipes existing Object
Obj_Cull	Identifies hidden frames
Obj_Name	Select of Object Name
Obj_Init	Sets DATA RESTORE references

QBITS 3DGraphics Flowchart



QBITS 3D Graphics Code

```
1000 REMark QBITS_3DGraphics_v3 (Exploring QL 3D Rotation Graphics v3 2021 QPCII)

1002 OPEN_IN#9,'ram2_QBITSConfig':INPUT#9,gx\gy\dn$:CLOSE#9

1004 wx=140:wy=120:fs=800:vs=.8              :REMark Angle:win xy:focal scale:vector size
1005 aset=-1:cset=1:nset=1:iset=1            :REMark Toggle switches
1006 bg1=0:bg2=7:k=49:Wset=512               :REMark Screen settings

1008 Mode 4:Init_win:Init_QB3D:Obj_Name:QBITS_3DComands

1010 DEFine PROCedure Init_win
1011 OPEN#4,con_10x10a10x10_4:OPEN#3,scr_
1012 WINDOW#3,138,98,370+gx,124+gy:PAPER#3,0:SCALE#3,100,0,0:CSIZE#3,0,0
1013 WINDOW#2,512,224,gx,gy         :BORDER#2,1,3:PAPER#2,0:CLS#2
1014 WINDOW#1,364,192,4+gx,30+gy:BORDER#1,1,3:PAPER#1,0:INK#1,7:SCALE 200,0,0
1015 WINDOW#0,512,32,gx,224+gy    :BORDER#0,1,3:PAPER#0,0:INK#0,7:CLS#0
1016 END DEFine

1018 DEFine PROCedure Init_QB3D
1019 ch=2:CSIZE#ch,2,1:OVER#ch,1:str$='QBITS 3D Graphics'
1020 INK#ch,2:FOR i=0 TO 1:CURSOR#ch,5+i,7:PRINT#ch,str$
1021 INK#ch,6:FOR i=0 TO 1:CURSOR#ch,7+i,8:PRINT#ch,str$
1022 CSIZE#ch,0,0:OVER#ch,0
1023 INK#ch,5:CURSOR#ch,294,18:PRINT#ch,'KEYS: Toggle Rotation On/Off'
1024 BLOCK#ch,16,3,466,22,5
1025 CURSOR#ch,372, 30:PRINT#ch,'Location xy:  <🡐 🡑🡓 🡒>' :REMark Cursor keys
1026 CURSOR#ch,372, 40:PRINT#ch,'Rotation   : xX yY zZ'
1027 CURSOR#ch,372, 50:PRINT#ch,'Vector Size:  <v><V>' :REMark Reduce/Enlarge
1028 CURSOR#ch,372, 60:PRINT#ch,'Focal Scale:  <p><P>' :REMark Reduce/Enlarge
1029 CURSOR#ch,372, 70:PRINT#ch,'frame/FILL :  <f><F>'
1030 CURSOR#ch,372,100:PRINT#ch,'NodeID On/Off <N>'
1031 CURSOR#ch,372,110:PRINT#ch,'Background :  <B><W>'
1032 ch=3:BORDER#ch,1,3:CLS#ch
1033 INK#ch,2:CURSOR#ch,78,78:PRINT#ch,'[Xx] Loop':CIRCLE#ch,70,44,24,.25,0
1034 INK#ch,4:CURSOR#ch, 6, 2:PRINT#ch,'Spin [Yy]':CIRCLE#ch,36,77,24,.25,PI/2
1035 INK#ch,7:CURSOR#ch, 6,84:PRINT#ch,'Roll [Zz]':CIRCLE#ch,36,44,24,1,0
1036 INK#ch,6:CURSOR#ch,98, 2:PRINT#ch,'ANGLE'
1037 ch=1:CSIZE#ch,0,0:INK#ch,4
1038 INK#0,7:CURSOR#0,430, 10:PRINT#0,'(E)xit'
1039 END DEFine

1041 DEFine PROCedure QExit
1042 CURSOR#0,472,10:PRINT#0,'Y/N':PAUSE:IF KEYROW(5)=64:LRUN dn$
1043 END DEFine
```

```
1045 DEFine PROCedure QBITS_3DComands
1046 REPeat lp
1047  SELect ON k
1048  =69,101:QExit:BLOCK#0,20,10,470,4,0          :REMark (E)xit
1049  =66,98 :bg1=0:bg2=7:PAPER#1,0:CLS#1          :REMark (B)lack background
1050  =87,119:bg1=7:bg2=0:PAPER#1,7:CLS#1          :REMark (W)hite background
1051  =49,50,51,52,53   :iset=1:Obj_Ang:Obj_Init   :REMark Load Object DATA
1052  =  32    :IF aset=-1:aset=5:ELSE aset=-1     :REMark Toggle animation
1053  =102     :IF cset= 1 OR cset=3:cset=2:ELSE cset=1 :REMark (f)rame On/Off
1054  =  70    :IF cset= 1 OR cset=2:cset=3:ELSE cset=1 :REMark (F)ILL  On/Off
1055  =78,110:IF nset= 1:nset=2:ELSE nset=1        :REMark (N)ode ID On/Off
1056  = 86:vs=vs+.1 :IF vs>=1.5 :vs=1.5            :REMark (V)Increase Vector size
1057  =118:vs=vs-.1 :IF vs<=  .5 :vs= .5           :REMark (v)Decrease Vector size
1058  =  80:fs=fs+10 :IF fs> 800 :fs=800           :REMark (P)Increase Focal scale
1059  =112:fs=fs -10 :IF fs<  80 :fs=  80          :REMark (p)Decrease Focal scale
1060  =192:wx=wx -10 :IF wx<=  10 :wx=  10        :REMark ← move left
1061  =200:wx=wx+10 :IF wx>=270 :wx=270          :REMark → move right
1062  =208:wy=wy+10 :IF wy>=190 :wy=190          :REMark ↑ move up
1063  =216:wy=wy -10 :IF wy<=  30 :wy=  30        :REMark ↓ move down
1064  =  88:iset=1:Obj_Ang:rx=rx -5:IF rx<   0:rx=rx+360 :REMark (X) Clockwise Loop
1065  =120:iset=1:Obj_Ang:rx=rx+5:IF rx>360:rx=rx -360 :REMark (x) Anti- Loop
1066  =  89:iset=1:Obj_Ang:ry=ry -5:IF ry<   0:ry=ry+360 :REMark (y) Clockwise Spin
1067  =121:iset=1:Obj_Ang:ry=ry+5:IF ry>360:ry=0  :REMark (Y) Anti- Spin
1068  =  90:iset=1:Obj_Ang:rz=rz -5:IF rz<   0:rz=rz+360 :REMark (z) Clockwise Roll
1069  =122:iset=1:Obj_Ang:rz=rz+5:IF rz>360:rz=rz -360 :REMark (Z) Anti- Roll
1070  END SELect
1071  Obj_Wipe:Obj_Draw:INK bg2
1072  IF aset=5:iset=1:Obj_Auto:ELSE Obj_Ang
1073  CURSOR 160,178:PRINT 'lx:'&wx&'  ly:'&wy&'   Vv:'&(vs*20)&'   Pp:'&fs&'  '
1074  CURSOR  12,178:PRINT 'rx:'&rx&'  ry:'&ry&'   rz:'&rz&'   '
1075  k=CODE(INKEY$(#4,aset))
1076 END  REPeat lp
1077 END DEFine
```

```
Rotation On/Off ▬
Location xy:  ⟨←↑↓→⟩
Rotation    : xX yY zZ
Vector Size:  ⟨v⟩⟨V⟩
Focal Scale:  ⟨p⟩⟨P⟩
frame/FILL :  ⟨f⟩⟨F⟩


NodeID On/Off ⟨N⟩
Background :  ⟨B⟩⟨W⟩
```

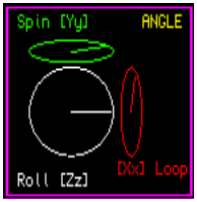
Note: On startup the Pyramid Object (**1**) is displayed with **rx, ry, rz** angles, Location **lx,ly,** Vector size **Vv** and Perspective **Pp.** Press Keys (**B**)lack or (**W**)hite to change background. Keys (**1**),(**2**),(**3**),(**4**) displays Pyramid, Cube, Hexagon and Shuttle Object.
Press <**Spacebar**> to toggle On/Off random Rotation, Loop, Spin, Roll of Object.
Press <**f**> Object toggles between Wireframe & Solid. <**F**> FILL's surfaces with colours
Press <**N**> to toggle Node ID On/Off.
Press <**V**> enlarge or <**v**> reduce Vector size of Object.
Press <**P**> increase or <**p**> decrease Perspective (Focal scale).
Change Objects position use < ←↑↓→ > for Loop/Spin/Roll use <**Xx Yy Zz** > keys.

```
1079 DEFine PROCedure Obj_Node
1080 LOCal lp,a,b,c:RESTORE nres:READ no
1081 FOR lp=1 TO no
1082    READ a,b,c:x(lp)=a*vs:y(lp)=b*vs:z(lp)=c*vs
1083 END FOR lp
1084 END DEFine
```

The Procedures for the 3D Rotation Graphics.

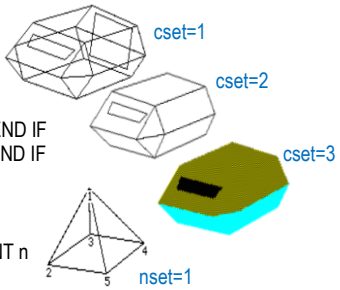
```
1086 DEFine PROCedure Obj_Auto
1087 rx=rx+5:IF rx>=360:rx=0
1088 ry=ry+5:IF ry>=360:ry=0
1089 rz=rz+5:IF rz>=360:rz=0
1090 END DEFine
```

```
1092 DEFine PROCedure Obj_Ang
1093 ch=3:INK#ch,0:FILL#ch,1:CIRCLE#ch,36,44,23,1,0:FILL#ch,0
1094 FILL#ch,1:CIRCLE#ch,36,77,21,.23,PI/2     :FILL#ch,0
1095 FILL#ch,1:CIRCLE#ch,70,44,21,.25,0        :FILL#ch,0
1096 IF iset=1:zink=0:yink=0:xink=0:ELSE zink=7:yink=4:xink=2
1097 INK#ch,zink:LINE#ch,36,44 TO 36+23*COS(RAD(rz)),44+23*SIN(RAD(rz))
1098 INK#ch,yink:LINE#ch,36,77 TO 36+23*COS(RAD(ry)),77+  5*SIN(RAD(ry))
1099 INK#ch,xink:LINE#ch,70,44 TO 70+  5*COS(RAD(rx)),44+23*SIN(RAD(rx)):ch=1
1100 END DEFine
```




```
1102 DEFine PROCedure Obj_Calc                    Calculates changes to Vector coordinates
1103 cx=COS(RAD(rx)):sx=SIN(RAD(rx))
1104 cy=COS(RAD(ry)):sy=SIN(RAD(ry))
1105 cz=COS(RAD(rz)):sz=SIN(RAD(rz))
1106 FOR np=1 TO no
1107   yt=y(np):y(np)=cx*yt-sx*z(np):z(np)=sx*yt+cx*z(np)
1108   xt=x(np):x(np)=cy*xt+sy*z(np):z(np)=sy*xt+cy*z(np)
1109   xt=x(np):x(np)=cz*xt-sz*y(np):y(np)=sz*xt+cz*y(np)
1110   vx(np)=wx+(x(np)*fs)/(z(np)+fs)
1111   vy(np)=wy+(y(np)*fs)/(z(np)+fs)
1112 END FOR np
1113 END DEFine
```

```
1115 DEFine PROCedure Obj_Draw
1116 LOCal lp,v,a,b,c,d,i:RESTORE vres:READ v:iset=2:Obj_Calc
1117 FOR lp=1 TO v
1118   READ a,b,c,d,i:IF cset=1:INK bg2:FILL 0:END IF
1119   IF cset=2:Obj_Cull:IF c1>0:GO TO 1133:END IF :INK bg2:FILL 0:END IF
1120   IF cset=3:Obj_Cull:IF c1>0:GO TO 1133:END IF :INK   i  :FILL 1:END IF
1121   LINE vx(a),vy(a) TO vx(b),vy(b) TO vx(c),vy(c) TO vx(d),vy(d)
1122   LINE TO vx(a),vy(a):FILL 0
1123 END FOR lp
1124 IF nset=2:Obj_Node:FOR n=1 TO no:CURSOR vx(n),vy(n),-2,2:PRINT n
1125 END DEFine
```



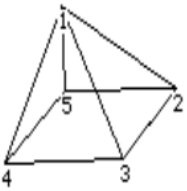

```
1127 DEFine PROCedure Obj_Wipe                    Clears screen of Object & Node ID
1128 LOCal lp,v,a,b,c,d,i:RESTORE vres:READ v:INK bg1
1129 FOR lp=1 TO v
1130   READ a,b,c,d,i:FILL 1:LINE vx(a),vy(a) TO vx(b),vy(b) TO vx(c),vy(c)
1131   LINE TO vx(d),vy(d) TO vx(a),vy(a):FILL 0
1132 END FOR lp
1133 Obj_Node:FOR n=1 TO no:CURSOR vx(n),vy(n),-2,2:PRINT n
1134 END DEFine
```

```
1136 DEFine PROCedure Obj_Cull                    Identifies Hidden surfaces
1137 c1=(x(b)-x(a))*(y(c)-y(a))-(x(c)-x(a))*(y(b)-y(a))
1138 END DEFine
```

```
1200 REMark QBITS 3D Wire Data

1202 DEFine PROCedure Obj_Name
1203 CSIZE#0,0,0:INK#0,6
1204 CURSOR#0, 20,8:PRINT#0,'(1)Pyramid'
1205 CURSOR#0, 90,8:PRINT#0,'(2)Diamond'
1206 CURSOR#0,160,8:PRINT#0,'(3)Cube'
1207 CURSOR#0,212,8:PRINT#0,'(4)Polygon'
1208 CURSOR#0,282,8:PRINT#0,'(5)Shuttle'
1209 END DEFine

1211 DEFine PROCedure Obj_Init
1212 REMark WARNING maintain correct nres:vrs:fres numbers
1213 IF k=49:nres=2001:vres=2008:rx= 60:ry=30:rz=0
1214 IF k=50:nres=2016:vres=2024:rx= 15:ry=30:rz=0
1215 IF k=51:nres=2035:vres=2045:rx=115:ry=30:rz=0
1216 IF k=52:nres=2054:vres=2064:rx= 15:ry= 0:rz=0
1217 IF k=53:nres=2079:vres=2103:rx= 0:ry=60:rz=0
1218 RESTORE nres:READ n:DIM x(n),y(n),z(n),vx(n),vy(n)        Note: n num of Nodes
1219 RESTORE vres:READ v:DIM fr(v,6):CLS#1                    Note: v num of Vector Frames
1220 END DEFine
```

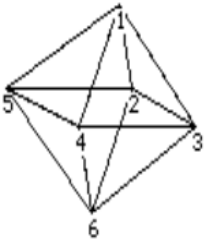
Note: Create New Objects add DATA for Nodes & Frames update **nres** & **vres**

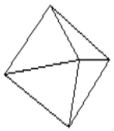
references



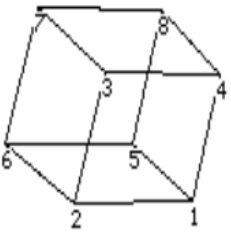

```
2000 REMark Pyramid
2001 DATA 5                :REMark Nodes
2002 DATA   0,  0,-20      :REMark Node 1
2003 DATA  20, 20, 20
2004 DATA  20,-20, 20
2005 DATA -20,-20, 20
2006 DATA -20, 20, 20

                2008 DATA 5            :REMark Frames
                2009 DATA 1,2,3,1,2
                2010 DATA 1,3,4,1,4
                2011 DATA 1,4,5,1,3
                2012 DATA 1,5,2,1,5
                2013 DATA 5,4,3,2,bg2
```

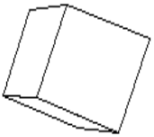
```
2015 REMark Diamond / OctraHedron
2016 DATA 6                :REMark Nodes x,y,z
2017 DATA   0, 30,  0      :REMark Node 1
2018 DATA  20,  0,-20
2019 DATA  20,  0, 20
2020 DATA -20,  0, 20
2021 DATA -20,  0,-20
2022 DATA   0,-30,  0

2024 DATA 8                :REMark Frames
2025 DATA 1,2,3,1,2
2026 DATA 6,3,2,6,2
2027 DATA 1,3,4,1,4
2028 DATA 6,4,3,6,4
2029 DATA 1,4,5,1,3
2030 DATA 6,5,4,6,3
2031 DATA 1,5,2,1,5
2032 DATA 6,2,5,6,5
```



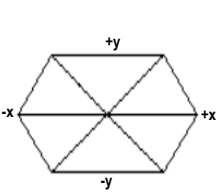


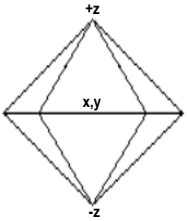
```
2034 REMark Cube
2035 DATA  8                :REMark Nodes x,y,z
2036 DATA  20, 20, 20      :REMark Node 1
2037 DATA -20, 20, 20
2038 DATA -20, 20,-20
2039 DATA  20, 20,-20      :REMark Node 4
2040 DATA  20,-20, 20      :REMark Node 5
2041 DATA -20,-20, 20
2042 DATA -20,-20,-20
2043 DATA  20,-20,-20      :REMark Node 8
2044 :
2045 DATA  6               :REMark Frames
2046 DATA 8,7,6,5,bg2      :REMark back Frame
2047 DATA 2,6,7,3,2
2048 DATA 4,3,7,8,4
2049 DATA 5,1,4,8,3
2050 DATA 5,6,2,1,5
2051 DATA 1,2,3,4,bg2      :REMark front Frame
```



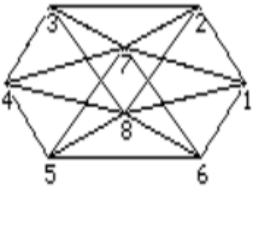


Note: Vector Graphics used in Gamming generally create Objects out of triangular polygon shaped planes.

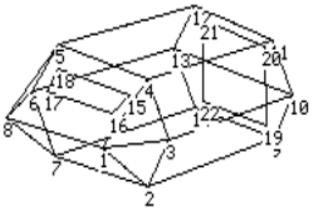


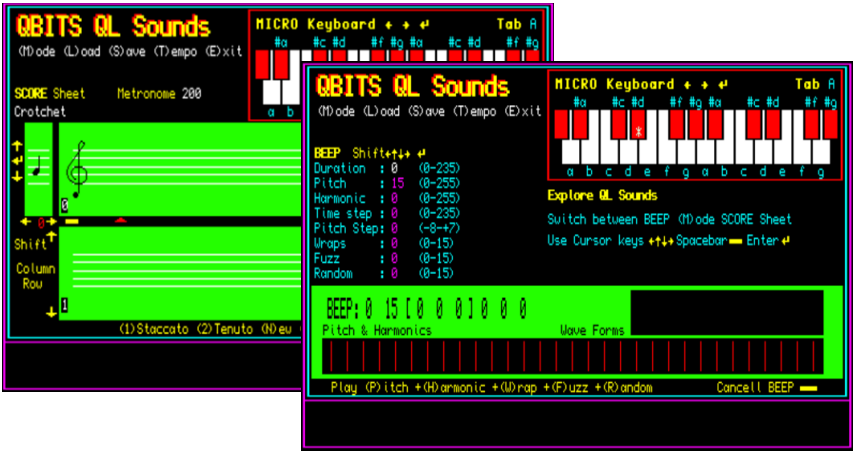



```
2053 REMark Dodecahedron
2054 DATA  8               :REMark Nodes x,y,z
2055 DATA  32,  0,  0      :REMark Node 1
2056 DATA  16, 24,  0
2057 DATA -16, 24,  0
2058 DATA -32,  0,  0
2059 DATA -16,-24,  0
2060 DATA  16,-24,  0      :REMark Node 6
2061 DATA   0,  0,-32      :REMark Node 7
2062 DATA   0,  0, 32      :REMark Node 8
2063 :
2064 DATA  12              :REMark Frames
2065 DATA 1,2,7,7,6        :REMark 1
2066 DATA 2,3,7,7,5
2067 DATA 3,4,7,7,4
2068 DATA 4,5,7,7,3
2069 DATA 5,6,7,7,2
2070 DATA 6,1,7,7,bg2      :REMark 6
2071 DATA 2,1,8,8,2
2072 DATA 3,2,8,8,3
2073 DATA 4,3,8,8,bg2
2074 DATA 5,4,8,8,5
2075 DATA 6,5,8,8,6
2076 DATA 1,6,8,8,4        :REMark 12
```



```
2078 REMark Space Shuttle
2079 DATA  22                :REMark Nodes
2080 DATA -40,  0, 20        :REMark Node 1
2081 DATA -20,-20, 20        :REMark Node 2
2082 DATA -20,  0, 30
2083 DATA -20, 20, 20
2084 DATA -20, 20,-20
2085 DATA -20,  0,-30
2086 DATA -20,-20,-20        :REMark Node 7
2087 DATA -40,  0,-20        :REMark Node 8
2088 DATA  40,-20, 20        :REMark Node 9
2089 DATA  40,  0, 30
2090 DATA  40, 20, 20
2091 DATA  40, 20,-20
2092 DATA  40,  0,-30
2093 DATA  40,-20,-20        :REMark Node 14
2094 DATA -24, 14, 16        :REMark Node 15
2095 DATA -30,  8, 14
2096 DATA -30,  8,-14
2097 DATA -24, 14,-16        :REMark Node 18
2098 DATA  40,-16, 16        :REMark Node 19
2099 DATA  40, 16, 16
2100 DATA  40, 16,-16
2101 DATA  40,-16,-16        :REMark Node 22
2102 :
2103 DATA  16                :REMark Frames
2104 DATA 9,10,13,14,5       :REMark Rear Frames
2105 DATA 10,11,12,13,240
2106 DATA 19,20,21,22,191 :REMark Rear Door
2107 DATA 2,9,14,7,5         :REMark Side Frames
2108 DATA 6,7,14,13,5
2109 DATA 5,6,13,12,240
2110 DATA 5,12,11,4,240
2111 DATA 4,11,10,3,240
2112 DATA 3,10,9,2,5
2113 DATA 3,2,1,3,5          :REMark Front Frames
2114 DATA 1,2,7,8,5
2115 DATA 7,6,8,7,5
2116 DATA 8,6,5,8,240
2117 DATA 4,1,8,5,240
2118 DATA 1,4,3,1,240
2119 DATA 15,16,17,18,0      :REMark Pilot Window
```



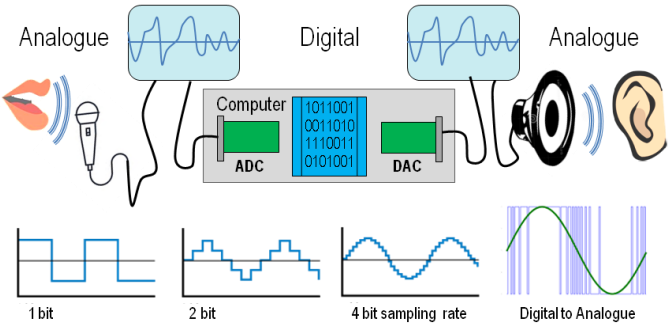


QL Sounds Introduction

In exploring the QL BEEP command I had to review a few facts as to not having any musical background. My Grandfather in his day was a bit of an entrepreneur running a small troop of entertainers. He played the piano and performed on stage in various song and dance routines. Regrettably I have to admit his musical talents were not passed on and in polite terms, I have told that I'm somewhat tone deaf (my misspent youth listening to loud rock!!!). I also confess to not being a skilled programmer although in my Projects Manager role I have been involved with computer code development. So hopefully I inherited some of his entrepreneurial skills.

Digital Audio

The human ear registers vibrations, sounds waves that are analogue in nature. In recording and reproduction systems, digital audio is the encoded representation of these sounds for processing, storage or transmission. The analogue wave length is sampled in regular time slots and the varying amplitude represented as a series of precise numbers. This allows editing and mixing to be introduced adding special effects to simulate reverberation, the enhancement of certain frequencies or change in pitch.

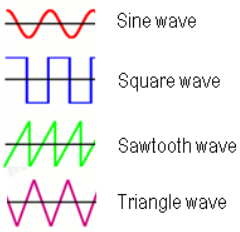


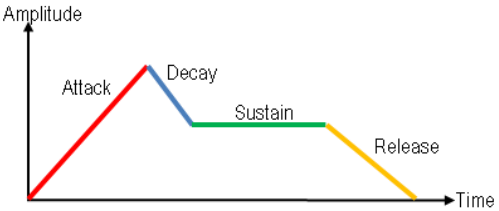
Called the Nyquist frequency, this is twice the highest frequency present in a signal, being the minimum sample rate without introducing errors. Digital representation is conveyed as a number of On/Off's, High's & Low's or in binary logic 1's and 0's.

Sound Synthesizer

Electronic synthesizers use basic components working together to reproduce the audio sensation of a sound. Oscillators generate the Waveform and changes in Pitch, filtering removes certain frequencies in the wave to change the Timbre, amplification controls the Volume, and varying modulation to create effects.

A pure Note or Pitch will be in the form of a sine wave. Mixed with sympathetic vibrations enriches the tonal blend creating differing waveforms. Timbre is perceived as the quality of these different sounds, the characteristic representing the pre-conscious identity, and based on information gained from the frequency Transients, Noisiness, unsteadiness, perceived Pitch and spread of Harmonics in a specific time frame. Wow! Really! This simply means they make distinguishable the same Pitch from being played on a Piano as opposed to a Violin. The main Pitch is called the fundamental frequency and the related frequencies the Harmonics. The wave envelope is the relationship of amplitude to time, this is called the pattern of Attack, Decay, Sustain and Release **ADSR**.

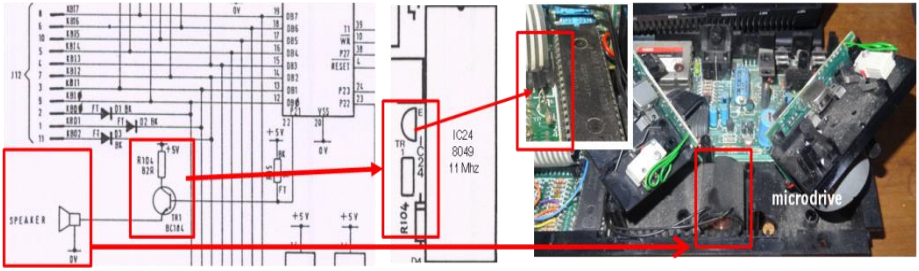




QL Sound Generation

The QL 68008 CPU (Central-Processor-Unit) communicates with a slave processor called the IPC (Intelligent-Peripheral-Controller). This **Intel 8049** co-processor works alongside the custom chip ZX8302 ULA (Uncommitted Logic Array), acting as a Keyboard buffer / Joystick buffer, RS232 Receive buffer and Sound Generator directing the output to the internal Loudspeaker. The 8049 Chip contains a 2kx8 Programable memory, a 128x8 RAM memory, 27 I/O lines, 8-bit Timer/Counter in addition to its on-board Oscillator and Clock circuits.

Some QLs have an 8749 chip, which is the EPROM version. The 8049 can also be replaced by the Hermes Co-Processor manufactured by Tony Firshman, which provides improvements to sound, serial ports and further eliminates problems of keyboard bounce.

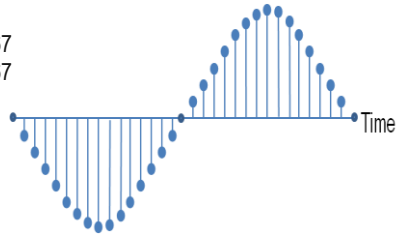


In executing a **BEEP** instruction, the IPC writes to port 2, P21 switching On/Off transistor TR1. The emitter follower configuration is used as a voltage buffer amplifier to drive the 60-ohm 23mm loudspeaker situated between the two microdrives.

QL IPC Communication

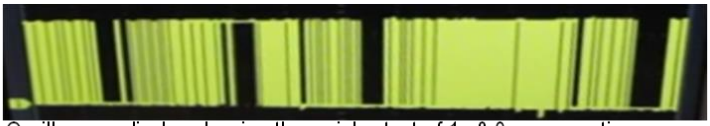
A command sent to the IPC uses the Manager Trap **MT.IPCOM** in the form of a header describing the command followed by any parameters. For audio output this is to Initiate Sound, and has 8 parameters

8 bits	pitch_1	range 0 to 255
8 bits	pitch_2	range 0 to 255
16 bits	interval between steps	range -32768 to +32767
16 bits	duration	range -32768 to +32767
4 bits	step in pitch	range -8 to 7
4 bits	wrap	range 0 to 15
4 bits	randomness of step	range 0 to 15
4bits	fuzziness	range 0 to 15



Frequency and Amplitude

Two key features of a sound wave is the frequency (how many times the wave vibrates in one second) and is broadly related to the Pitch of the sound we hear. The amplitude (Volume) of a sound is related to the amount of energy that the sound wave carries. As the frequency increase the shorter the wave length, this is represented by the number of changes within a time frame. The higher the energy, the louder it sounds, the higher a waves amplitude. Digitally the amplitude is represented in binary 1s and 0s as a precise count or weight. A wave form is therefore generated by the number of ones in the binary record processed by the ***DAC*** *(Digital to Analogue Converter) on each cycle and related to the sample rate.*



Oscilloscope display showing the serial output of 1s & 0s representing a wave

QL Sound Output

The QL Sound is produced by switching the voltage fed to Transistor TRI via the IPC Port 2 Pin 21. In each output cycle the differing number of changes coupled with the device inherent latency, produces an output more recognisable as an analogue wave. This digital to analogue conversion being derived from the BEEP parameters sent as part of the IPC Instruction.

The one thing the IPC doesn't appear to have is any separate control over Wave Amplitude. This gives a partial explanation as to why higher pitches sound louder than their lower counterparts, the strings of 1s and 0s being closer together add to an overall Amplitude.

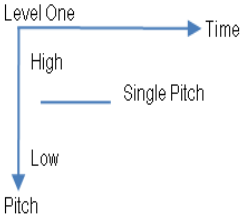
If four, instead of one IPC I/O Port had been use and the Fuzziness 4-bit Attribute used to vary the voltage Amplitude feeding TR1 Transistor, much more control could have been applied to the resulting waveform.

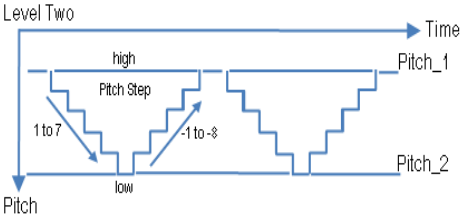
QL Sound Attributes

Does the QL Sound Generator fit the bill? It has two pitches and a method to ramp up and down between the two frequencies producing a range of harmonics. Adding harmonics can build the fundamental frequency from sinusoidal to more that of a square wave. Then there's Wrap which I believe is intended to create an output similar to a sawtooth wave. There is also randomness and fuzziness to further change the output waveform.

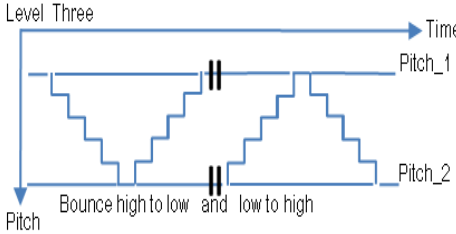
QL Sound Concepts
The QL Guide describes Sound being generated by the IPC (8049) as controlled by specifying a number of parameters, allowing the stage-by-stage build-up of more complex sounds.

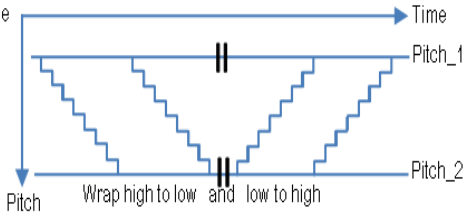
The first level is a single pitch active for a specified time, which is a pure sound at a set frequency. Once the **IPC 8049** has been instructed it will itself carry on for a specified duration. Given a value of zero will run until a following **BEEP** command cancels it out or changes the parameter setting for a different sound. The duration is carried out in units of 72 microseconds the range 1 to 32767 or again from -32768 to -1 (for -1 or 32767 being 2.36 seconds).





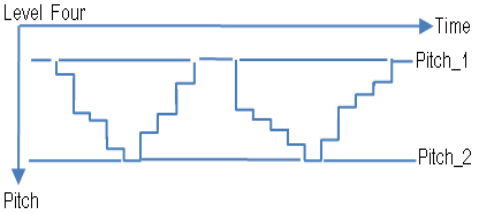
At level two a second pitch is added and the rate at which the sound ramps between the two pitches allegedly can produce semi musical beeps, spiralling or rippling tones, growls, zaps and moans. The number of steps and direction can be configured high to low or low to high.





The third level controls the sound after reaching one of the pitches. The sound is left to bounce or Wrap a number of times. Depending on step direction this can be high to low or visa verse.

Level four introduces a deviation from the specified step or gradient in moving between pitches with Random larger or smaller steps. This Random element can generate a wide and unexpected range of sounds.



Fuzzy is **level five** and is described as a further variation that adds changes to the pitch being generated and tends to make the sound end up being more like a buzz.

Note: Unfortunately, being able to vary the Amplitude of the output is not a parameter option.

QL SuperBASIC BEEP Command
The QL User Guide list the parameters as Duration, Pitch, Pitch_2, Grad_x, Grad_y, Wraps, Fuzzy and Random. They are grouped as Duration + Pitch, Pitch_2 + Grad_x + Grade_y, Wrap, Fuzzy, Random. All can be used in different combinations and values to build complex sounds.

Duration
The minimum cycle used by the **IPC (8049)** processor is 72 microseconds. But what would be the shortest multiple to perceive a sound? One is the Pitch or Frequency, the other Amplitude. Once a sound wave reaches the human ear, the brain can perceive it in around 50 milliseconds. The Stapes Reflex is where the ear protects itself from very loud noises, here perception can be in as little as 25 milliseconds. In the relationship between hearing a sound and Pitch perception this would be in the order of 100ms or slightly less for a higher Pitch.

When a number of **BEEP** Instructions are carried out sequentially there is concern that any following sound will be actioned before the previous one is fully executed. The length of a sound being an important factor an alternative to the duration parameter is to use the **SuperBASIC PAUSE** command to control when to activate a following **BEEP** instruction to the **IPC 8049** processor. The **PAUSE** command uses multiples of 20 milliseconds for example: **BEEP 0,3 : PAUSE 100 : BEEP** (will last two seconds).

Pitch
The sensation of a vibration is the perceptual property of sounds and allows the ordering of their frequency to be judged as higher or lower. **BEEP** duration and first pitch produces a single fundamental frequency. The **Pitch** range climbs from 255 its lowest to highest at 0.

Harmonic
Pitch_2 can be referred to as the **Harmonic,** add a **Time Interval** (**Grad_x**) and a **Pitch Step Grad_y**), they create a sequence of sound variations ordered by the time duration and number of steps between the main Pitch and second Pitch. The Time Interval again is in multiple units of 72 microseconds for each note in the sequence. The Pitch Step range is -8 to 7 where step 1 to 7 scales downwards high to low pitch and -8 to 0 starts the sequence scaling upwards from low to a higher pitch. The sequence continues by bounces between the two pitches. A **Harmonic** without a **Time Interval** and/or **Pitch Step** has no affect. Adding a **Pitch** step of 1 when **Harmonic** and **Time Interval** are both 0 identifies the pitch as a high zero. **Harmonic** plus a **Pitch** step with **Time Interval** 0 just changes Main **Pitch** to the **Harmonic**.

Wraps
Wraps repeat the sequence of harmonics produced by the **Pitch_1**, **Pitch_2**, **Grad_x**, **Grad_y** parameters a number of times. Zero continues the bounce effect of the harmonic. Increasing values 1 to 7 creates scaling high too low, 8 to 15 creates scaling low to high.

Fuzzy & Random
Fuzzy decreases the purity of the Pitch, Random, randomises the steps until little of the original sequence is evident. Both of these have a range 0 to 15, Zero has no effect and the active range is more like 8 to 15. Increasing the Fuzzy range simply blurs the pitch to a buzz.

BEEPING
This **SuperBASIC function** detects if the QL hardware is producing a sound and simply returns as true or false. IF **BEEPING** THEN **BEEP**. If true this will cancel any QL Sound output.

QBITS QL Sound Output

As a starting point in coding for a QL Sound output, I added a few modifications to the **BEEP Exerciser** from **QL SuperBASIC - The Definitive Handbook by Jan Jones**.

Variable	Parameter	Value	Description
d	duration	0 to 235	[0 increments of d*1e4/72]
p	pitch	0 to 255	[0 highest descending to 55]
h	harmonic	0 to 255	[0 highest descending to 255]
t	time interval	0 to 235	[0 increments of t*1e4/72]
s	pitch step	0 to 15	[effective range 1 to 7 low to high 8 to 15 high to low]
w	wrap	0 to 15	[effective range 1 to 15]
f	fuzz	0 to 15	[effective range 8 to 15]
r	random	0 to 15	[effective range 8 to 15]

```
100 REMark QLBeepv1   (QBITS Exploring QL Sounds 2018)

104 MODE 4:BInit : BMenu

108 DEFine PROCedure BInit
110 WINDOW 492,200,8,8:PAPER 7:INK 0:CSIZE 0,0:CLS
112 CURSOR 8,6:PRINT 'Play (P)itch +(H)armonic +(W)rap +(F)uzz +(R)andom'
114 CURSOR 8, 50:PRINT 'Duration     :       (0 to 235)'      :REMark d
116 CURSOR 8, 60:PRINT 'Pitch        :       (0 to 255)'      :REMark p
118 CURSOR 8, 70:PRINT 'Harmonic     :       (0 to 255)'      :REMark h
120 CURSOR 8, 80:PRINT 'Time Step    :       (0 to 235)'      :REMark t
122 CURSOR 8, 90:PRINT 'Pitch Step   :       (-8 to 7)'       :REMark s
124 CURSOR 8,100:PRINT 'Wraps        :       (0 to 15)'       :REMark w
126 CURSOR 8,110:PRINT 'Fuzz         :       (0 to 15)'       :REMark f
128 CURSOR 8,120:PRINT 'Random       :       (0 to 15)'       :REMark r
130 CURSOR 8,134:PRINT 'Edit  use    ←↑↓→  Space cancels BEEP <Esc> quit menu'
132 DIM BPm(7):INK 2:FOR ipm=0 TO 7:BRead
134 END DEFine

138 DEFine PROCedure BMenu
140 INK 0:ipm=0:BPrt
142 REPeat lp
144  CSIZE 0,1:CURSOR 8,24:PRINT 'BEEP: ';d;'  ';p;' [ ';h;'  ';t;'  ';s;' ] ';w;'  ';f;'  ';r;' ':CLS 4
146  k=CODE(INKEY$(-1))                    :REMark Read Keyboard
148  SELect ON k
150    =208:IF ipm>0:BChange -1            :REMark Up
152    =216:IF ipm<7:BChange 1             :REMark Down
154    =192:BPm(ipm)=BPm(ipm)-1:BRead      :REMark Left
156    =200:BPm(ipm)=BPm(ipm)+1:BRead      :REMark Right
158    = 80,112:BEEP d,p                   :REMark (P)itch
160    = 72,104:BEEP d,p,h,t,s             :REMark +(H)armonic
162    = 87,119:BEEP d,p,h,t,s,w           :REMark +(W)rap
164    = 70,102:BEEP d,p,h,t,s,w,f         :REMark +(F)uzz
166    = 82,114:BEEP d,p,h,t,s,w,f,r       :REMark +(R)andom
168    = 32:BEEP
170    = 27:BEEP:STOP
172  END SELect
174 END REPeat lp
176 END DEFine
```

```
180 DEFine PROCedure BPrt
182 CSIZE 0,0:CURSOR 80,50+ipm*10:PRINT BPm(ipm) TO 14:CSIZE 0,1
184 END DEFine

188 DEFine PROCedure BChange(change)
190 INK 4:BPrt
192 ipm=ipm+change
195 INK 7:BPrt
196 END DEFine

200 DEFine PROCedure BRead
202 BPrt: d=INT(BPm(0)*10000/72): t=INT(BPm(3)*10000/72)
204 p=BPm(1):h=BPm(2): s=BPm(4):w=BPm(5):f=BPm(6):r=BPm(7)
206 END DEFine
```

```
Play (P)itch +(H)armonic +(W)rap +(F)uzz +(R)andom

BEEP: 0 1[ 3 10000 7 ]0 0 0

Duration  :0          (0 to 235)
Pitch     :1          (0 TO 255)
Harmonic  :3          (0 TO 255)
Time Step :72         (0 to 235)
Pitch Step:7          (-8 to 7)
Warps     :0          (0 to 15)
Fuzz      :0          (0 to 15)
Random    :0          (0 to 15)
Edit use +↑↓+ Space cancels BEEP <Esc> to quit menu
```

To my untrained ear the above example gives a passable representation of a police panda car siren.

QBITS QL BEEP Parameters

Using **BEEP** parameters with the QL Internal speaker arrangement, it has to be said, is more a trial-and-error process rather than any constructed methodology. However, the program supplied here allows setting the various parameters and switching them on sequentially to hear the effects that take place.

Listening to Musical Notes

The next step was to look more deeply into Pitch frequency and their Harmonics. The frequency range of the human ear can be as low as 20 cycles per second or as high as 20,000 cycles per second (20Hz to 20kHz). The higher the frequency, the higher the pitch – double the frequency and the pitch goes an octave higher. For example, 260Hz is approximately middle C on a piano keyboard 720Hz is C an octave higher, 4186Hz is the highest C8 and A0 the lowest key is 27.5Hz. The AC mains hum of 50Hz in Europe is close to the Pitch of G1 = 48.99Hz.

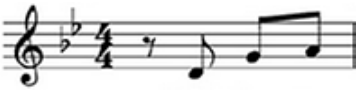
This whole process of relationship between Frequencies and their Harmonics, short and long-time intervals with rising or falling Pitches and the Wave Envelope they produce create Tonal quality. In music this can be represented by symbols that allow a range of Notes and the way each is to be played. A Notes differing tones are dependent on the instrument being played. Therefore, at this point I made a quick overview of musical representation, Stave, Clefs, Notes, their meaning and relationship.

Identifying a Music Note

In evaluating Pitch in musical terms, I began by identifying the notes and their representation. Letters or symbols are used making it easier to write and quicker to read. Pitch classes are represented by letters of the alphabet (A,B,C,D,E,F,G) or by Do-Re-Me-Fa-Sol-La-Ti.

The letter definition and corresponding notes are:

C –> do
D –> re
E –> mi
F –> fa
G –> so
A –> la
B –> ti (**H** in German)

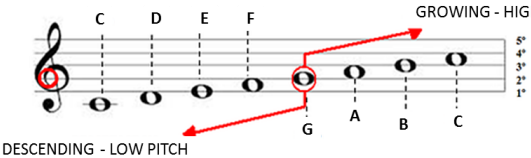


Measure (bar) Separator

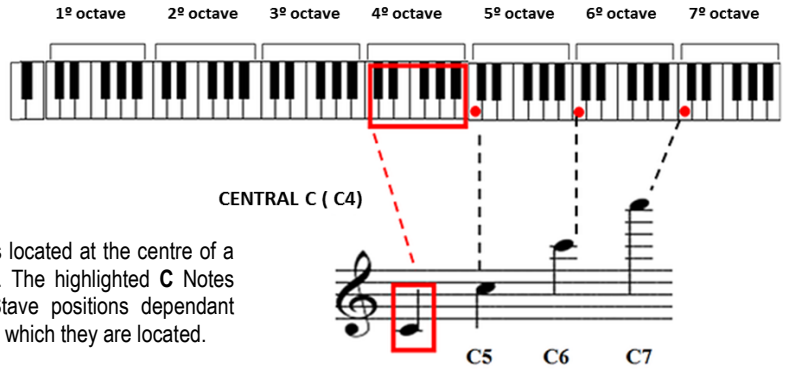
Sheet music registers the harmonic, rhythmic and melodic ideas. Notes are positioned and written in the form of musical symbols.

Music Stave

The five lines (1st, 2nd, 3rd, 4th and 5th) of a **Stave** are where each line and each space between represents a different Note of Scale.



To read sheet music - is the sequence of Notes, forwards and backwards!



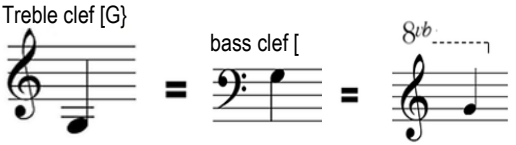
Middle **C (C4)** is located at the centre of a Piano keyboard. The highlighted **C** Notes hold different Stave positions dependant on the Octave in which they are located.

Ledger Lines

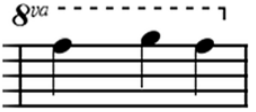
Where the Stave can't handle the representation of the Notes for a full range of Octaves, Ledger lines are used. These lines are nothing more than the continuation of the Stave, they are used to represent Notes that surpass the bottom or upper limits.

Treble Clef

Musicians throughout history have assigned different positions for their Notes. **Clefs** were created as symbols serving to sign the Note and the line of reference adopted. The most common Clef for guitar, piano and voice is the **Treble Clef** also known as the **G Clef** because the design of the Clef encircles the second Stave line which is G.



The symbol 8*v* is followed by the letter "*b*", which means "below", "8*va*" would be for Octaves above.



Interpretation of the notes (F, G, F) should be played one Octave above the position that it is in the Stave.

Accidentals

To show the increase or decrease of a Notes Pitch by one half step, symbols called Accidentals are used. When these same symbols appear at the very beginning of the music score, they are specifying a Key Signature. They stay in effect for all of the Notes of the same Pitch for the rest of the measure.

Flats lower the Pitch of the Note by one half step.

Sharps raise the Pitch of the Note by one half step.

Naturals cancel out any previous Sharps or Flats. The Pitch returns to normal.

Slurs smoothly connect Notes of different Pitches.

Articulations

These effect how the Note is played and include the slur, ties, phrase mark, staccato, staccatissimo, accent, sforzando, rinforzando, and legato.

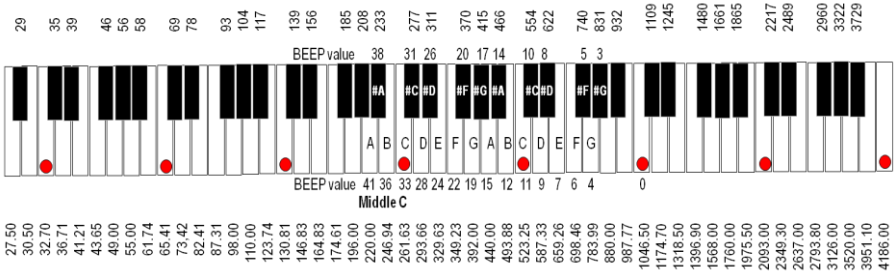
This means to play the Notes without breaks.

Ties connect Notes of the same Pitch, forming essentially one longer Note.

Key Notes & BEEP values

I thought it might be useful to review the range of Piano keys and associated Notes or Pitches to their related frequencies. Then with a little help from a **QLUB** Prog and again with my untrained ear I cross referenced **BEEP** Pitch to values around the middle **C** shown on my chart.

BEEP Pitch_1 = 0 to my untrained ear equates to a C6 or 1046.50Hz.

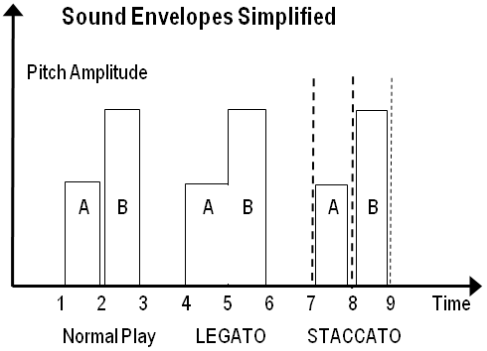


QLUB Music Micro Please!

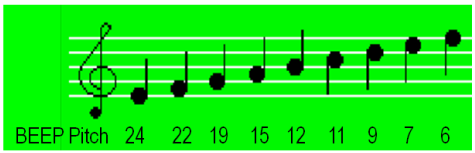
The **QLUB** edition of Mar/April 1985 carried an article with a short program which displayed to screen a **Stave** a **G-Clef** and added **crotchet** symbols to selected Pitches. It described musical Notes over two Octaves that could be reproduced with a **BEEP pitch_1** numbered equivalent.

The article displayed the music symbols and their beat values from **Breve** to **hemi-demi-semi-quaver** (8 beats down to 1/16). Also drawn were Simplified **Sound envelopes** showing **Pitch / Amplitude** of Normal Playing time against **Legato** and **Staccato**.

```
100 REMark QLUBMicrov1    (QLUB Music Micro QBITS - 2018)

104 DIM pitch(18)
106 MODE 4:WINDOW 448,200,32,16:PAPER 4:CLS
108 WINDOW#0,448,20,32,216:PAPER#0,7:INK#0,0:CLS#0
110 PRINT#0,'auto or manual ? (a/m)'
112 IF INKEY$(-1)=='m' THEN yourself=1:ELSE yourself=0

116 REPeat loop
118  up=50:across=16:inc=0:CLS
120  Stave:Draw_Clef
122  FOR note=1 TO 18
124    Pick_Note yourself
126    Display_Note
128    pitch(note)=p
130    across=across+8
132  END FOR note
134  Play_Tune
136  CLS#0:PRINT#0,'Another tune ? (y/n)'
138  again$=INKEY$(-1)
140  IF again$=='n' THEN EXIT loop
142 END REPeat loop
144 STOP
```

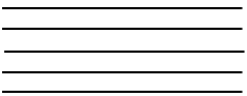



```
148 DEFine PROCedure Pick_Note(yourself)
150 IF yourself
152 REPeat check
154   CLS#0:INPUT#0, 'Note number (1 to 9)';choice
156  ELSE
158   choice=RND(1 TO 9)
160  END IF
162  SELect ON choice
164   =1:p=24:inc=0
166   =2:p=22:inc=1.5
168   =3:p=19:inc=3
170   =4:p=15:inc=4.5
172   =5:p=12:inc=6
174   =6:p=11:inc=7.5
176   =7:p= 9:inc=9
178   =8:p= 7:inc=10.5
180   =9:p= 6:inc=12
182   =REMAINDER :END REPeat check
184  END SELect
186 END DEFine
```



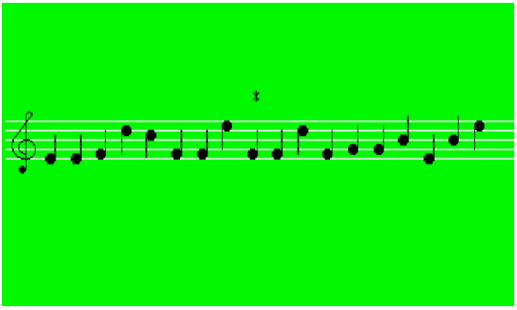
```
190 DEFine PROCedure Display_Note
192 FILL 1:CIRCLE across, up+inc,1.5:FILL 0
194 IF p<12
196   LINE across-1.5,up+inc TO across-1.5,up+inc-8
198 ELSE
200   LINE across+1.5,up+inc TO across+1.5,up+inc+8
202 END IF
204 BEEP-1,p:PAUSE#0:BEEP
206 END DEFine
```

```
210 DEFine PROCedure Stave
212 INK 7
214 FOR ledger=0 TO 12 STEP 3
216   LINE 2,up+ledger TO 165,up+ledger
218 END FOR ledger
220 INK 0
222 END DEFine
```



```
226 DEFine PROCedure Draw_Clef
228 LINE 8,up+1.5
230 ARC_R TO 0,4.5,-PI
232 ARC_R TO 0,-6,-PI TO -3,7,-3*PI/4
234 LINE_R TO 5,7:ARC_R TO -2,0,PI
236 LINE_R TO 0,-18
238 FILL 1:CIRCLE_R -1,0,1:FILL 0
240 END DEFine
```

```
244 DEFine PROCedure Play_Tune
246 CLS#0:PRINT#0,'Press any key to Play!'
248 PAUSE
250 x=116:y=72
252 FOR note=1 TO 18
254   Blip x,y
256   BEEP -1,pitch(note)
258   PAUSE 20
260   Blip x,y:x=note*8+16
262   BEEP
264 END FOR note
266 CLS#0:PRINT#0,'Play again(y/n)'
268 IF INKEY$(-1)=='y':Play_Tune
270 END DEFine
```



```
274 DEFine PROCedure Blip(x,y)
276 INK 4:OVER -1:CURSOR x,y,0,0 :PRINT '*' :OVER 0
278 END DEFine
```

The author of the **QLUB** publication was not given, however later that year 1985; the **QL User** magazine published **James Lucy's QL COMPOSER** which appeared to have a connection.

Bearing in mind the quizzical nature of the QL Sound generator it seemed logical to explore and retain sets of BEEP parameters, each key defined either as a musical Note or everyday sound or even weird futuristic effects. To create a Score, with keyboard Notes offering differing sounds it now required a selection of Symbols to identify timing and how they might be played.

QBITS Setting the Beat

Determining a Time Signature or **Tempo**. Music has a regular pulse or rhythm identified as the **beat**. Written after the **Clef** two numbers at the beginning of a Score establish the number of beats in each uniformed section or measure. The top number is the beats in a measure; the lower number the note combination for each **beat**. For example, a **4/4** timing would be four beats to the metre and each beat represented by a Crotchet, but potentially other note combinations, two Minim or a single Semibreve.

QBITS Metronome

The **Beat** is calculated against the rate to be played per minute. This is used to calculate an individual **Note** or **Rest** value in determining the duration used with the **PAUSE** Command (*see below*). To provide a distinct separation between **Notes** or **Rests** from any previously played, a **delay PAUSE** is inserted. For normal playback **80 milliseconds** is chosen as a **standard break**. For **Staccato** (separated) this delay is increase to **120 milliseconds** to make the Note more distinctive. For **Legato** (lengthened) this is reduced to **40 milliseconds** so it appears to merge with any following Note.

To recognise differing Pitches the Human ear requires around 100 milliseconds of sound. Therefore, the shortest duration based on a Semiquaver (a quarter Note) would need to be in the order of 180ms. Calculating Beats per minute (**bpm**), 60 seconds is multiplied by the **PAUSE** value for one second ie. 60*50=3000. This is then multiplied by the **Note** or **Rest** value (4 to 0.25). The result is then further divided by the Metronome rate. The PAUSE duration for a Crotchet with a max of 240bpm would be (60*50*1)/240=12.5, for a Semiquaver (60*50*0.25)/240=3.125. A duration **PAUSE** of 3 is only 60ms and not nearly enough time for the human ear to differentiate a change in pitch. For the **ADSR** of a **Note's** playback in this proposed arrangement the **Attack, Decay, Sustain** is covered by the **BEEP** command with its attributes and set by the **PAUSE** duration that follows. A second cancelling **BEEP** followed by a further **PAUSE** delay creates the **Release** before any following Note.

Accents or **Articulations** explain how each **Note** is to be performed, **Staccato** with the note short and detached, **Tenuto** holding the Note for its full value blending into the next as in playing **Legato**. Another way to extending the duration and by half as much time again, is by placing an **Augmentation Dot** after the Note. For modes of play such as **Staccato** or **Tenuto**/**Legato** and the **Augmentation Dot** the **duration** and **delay** can be adjusted to reflect the change by increasing and decreasing their lengths.

BEEP *d,p,h,t,s,w,f,r.*:**PAUSE** *duration* : **BEEP** :**PAUSE** *delay*

Staccato marked by a dot placed above or below the Note head

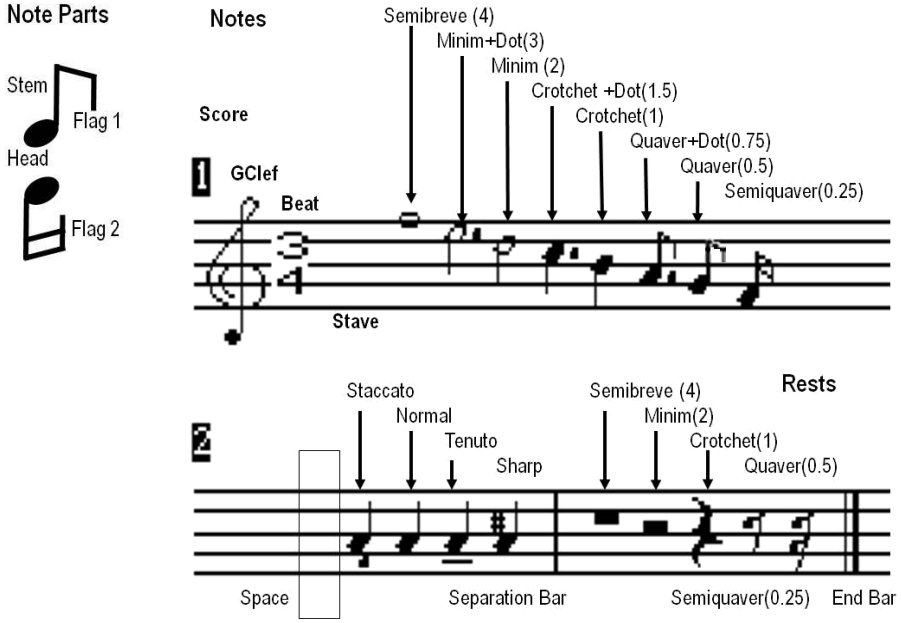
Tenuto marked by a line placed above or below the Note head.

Dot placed after the **note** adds half of the value of the **note** to itself.

As important are the **Rests** where there is no **BEEP** command just a **PAUSE** *duration + delay*. The **Space**, **Separation Bar** and **End Bar** are given a zero time.

QBITS Notes and Scores

The **QBITS Notes** chosen range is from the **Semibreve** down to the **Semiquaver** (a value of 4 beats down to 1/4 of a beat), this includes **Notes** with a **Dot Argumentation**. Then **Rests** to hold equivalent time slots where no music is played. **Sharps** that increase a Note by half a Pitch step and **Staccato** and **Tenuto** to further emphasise the length of how a Note is to be played**.** Other symbols include a **Separation bar** for the measures or metre and an **End bar** to complete a musical Score.



QBITS Musical Symbol Generation

Considerations are a **Note's** positions either below, between or on a **Stave line**, and if additional **Ledgers** are required. The **Stave** and components that make up the Musical Symbols **Notes, Rests** etc. use **SuperBASIC** commands that operate with the **Graphics Coordinate System**. Each Symbol of a Score will therefore require progressive positioning along the Stave as well as vertical positioning relative to the scale of a Note being displayed.

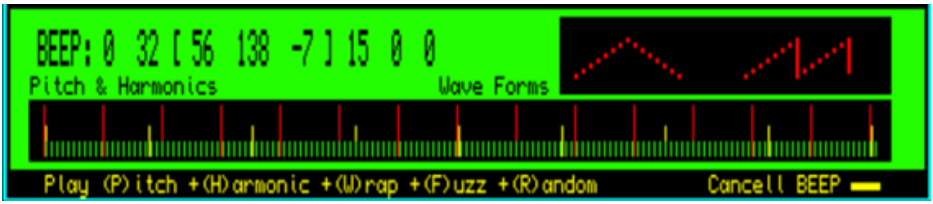
First is the **Space** which clears a position and redraws the **Stave** lines. The **Space** is a **FILL**ed rectangle drawn with the **LINE** command. Further use of the **LINE** command then displays the **Stave** lines. For example, the combination of a **Space** and selected **Beat** value after the **GClef** allows the signature **Beat** to be changed or optionally to show **No Beat**.

The **Separation Bar, End Bar, Semibreve** and **Minim Rests** make further use of the **LINE** and **Fill** commands. The **Crotchet**, **Quaver** and **Semiquaver Rests** required a little more engineering. The actual **Notes** are built up from parts, **Head, Stem Tails** as shown above. **Accidentals #Sharps**, **Articulations, Staccato/Tenuto Dots - Lines** and following **Augmentation Dots** are added as required.

QBITS Menu Considerations
I decided on two (**M)odes** the **BEEP** where changes to the Attributes could be explored and assigned as **Notes** or **Sounds** from which entries one could construct a written **Score** for replay. Added to the usual **(L)oad (S)ave** and **(E)xit.** is the **(T)empo** which allows changes to the setting of the **Beat** and **Metronome** values in **SCORE** Mode.

'**m**' =0 **BEEP Mode** =1 **SCORE Mode**

QBITS BEEP Mode
I decided on some enhancement to the **BEEP Prog** given earlier, namely to provide some additional Graphics to show how the differing parameters might have cause and effect on the Waveform output. The bottom part of the display shows the **Pitch** and **Harmonics** and interspaced sub frequencies created by the **grad_X**, **grad_y** parameters (**QBITS Time/Step**) used with the **BEEP** command. Derived from the **QL Sound Concepts** this is also shown as the First Wave. The Second Wave represents the assumed effects of the **Wrap** parameter. My hope in exploring the **BEEP** command parameters by listening and identifying their effects will lead to a more methodical way of constructing useful sound outputs.



QBITS Program Arrays & Variable Assignment
For the **BEEP Exploration** with use of the **Micro Keyboard** the option was to load an array so each **Key** is set with **BEEP parameter information**. These would then be available for use as **Notes** in constructing a **Score** for playback.

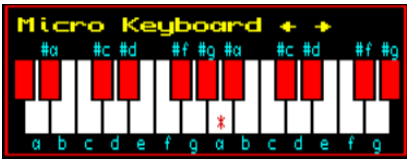
For the Micro Keyboard **Mkey(kg,kn,kp)**
kg 0-1 Micro key Groups **(Note: A/B Groups could be extended)**
kn 0-23 Micro key Number
kp 0-8 Micro Key Parameters **d,p,h,t,s,w,f,r,so**
ie. **d** duration, **p** pitch, **h** harmonic, **t** time interval, **s** step, **w** warp, **f** fuzzy, **r** random, **so** Stave offset

For the Score Sheet **Score(sl,sn,sp)**
sl 0-9 Score Lines **(Note: 10 x 24 = 240 Symbols can be assigned)**
sn 0-23 Score Note position
sp 0-5 Score Parameters **kg,kn,ds,vn,as,ar**
kg Micro Keyboard Group (A/B) **, kn** Micro key number, **ds** display symbol (0-15),
vn value (4 to 0.25) of **Note** or **Rest** , **as** Spare, **ar** Articulations (**Staccato Tenuto/Legato**)

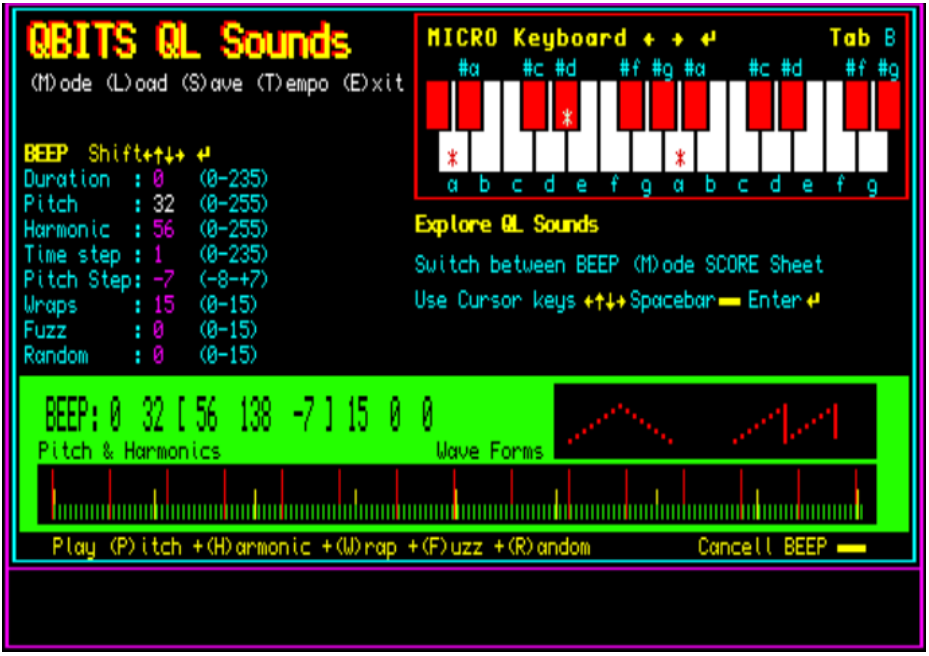
QBITS Coordinates
For **WINDOW x,y graphic coordination's** the conventions in previous **QL Sound Progs** used variables **across** and **up**, so for the **Micro Keyboard** this became **ka, ku** and for the **Score Sheet sa, su**. For **Notes** dependant on keyboard location on the **Stave** the offset **so** is added to **su** for the displayed symbol.

QBITS Micro Keyboard

Reviewing past Progs aimed at using the **QL BEEP command** I hadn't seen any graphical representation of a keyboard. This display is based on the key values from the QLUB article described earlier.




```
100 REMark QBSMicro_v1 (QBITS Micro Keyboard Graphics 2018)

102 MODE 4:CLS:Init_Keyboard:Select_Key

104 DEFine PROCedure Init_Keyboard
105 OPEN#3,scr_276x74a224x6 :PAPER#3,0:BORDER#3,1,2:CLS#3
106 FOR i=0 TO 13:BLOCK#3,16,36,12+i*18,26,7
107 FOR i=0 TO 14
108   IF i=2 OR i=5 OR i=9 OR i=12                    Note no action (Keys without upper keys)
109   ELSE
110     BLOCK#3,16,20,3+i*18,26,0
111     BLOCK#3,12,19,5+i*18,26,2
112   END IF
113 END FOR i
114 OVER#3,1:CSIZE#3,2,0:INK#3,6
115 FOR i=1 TO 2:CURSOR#3,4+i,4:PRINT#3,'Micro Keyboard ¼ ½'
116 OVER#3,0:CSIZE#3,0,0:INK#3,5
117 CURSOR#3,16,16:PRINT#3,' #a   #c #d   #f #g #a   #c #d   #f #g'
118 CURSOR#3,16,62:PRINT#3,'a  b  c  d  e  f  g  a  b  c  d  e  f  g'
119 END DEFine

121 DEFine PROCedure Select_Key
122 kp=0:ka=16:ku=54
123 OVER#3,-1:CURSOR#3,ka,ku:PRINT#3,'*':OVER#3,0
124 REPeat klp
125 k=CODE(INKEY$(-1))                    :REMark Read Keyboarded
126 SELect ON k
127   =192:IF kp> 0 :Change_Key -1        :REMark Left
128   =200:IF kp<23:Change_Key  1         :REMark Right
129   = 27:CLOSE#3:EXIT klp
130 END SELect
131 END REPeat klp
132 END DEFine

134 DEFine PROCedure Change_Key(change)
135 OVER#3,-1:CURSOR#3,ka,ku:PRINT#3,'*':OVER#3,0
136 kp=kp+change:ka=16
137 IF kp> 2:ka=26
138 IF kp> 7:ka=35
139 IF kp>14:ka=44
140 IF kp>19:ka=53
141 ka=ka+kp*9
142 SELect ON kp:=1,4,6,9,11,13,16,18,21,23:ku=36
143 SELect ON kp:=0,2,3,5,7,8,10,12,14,15,17,19,20,22:ku=52
144 OVER#3,-1:CURSOR#3,ka,ku:PRINT#3,'*':OVER#3,0
145 END DEFine
```



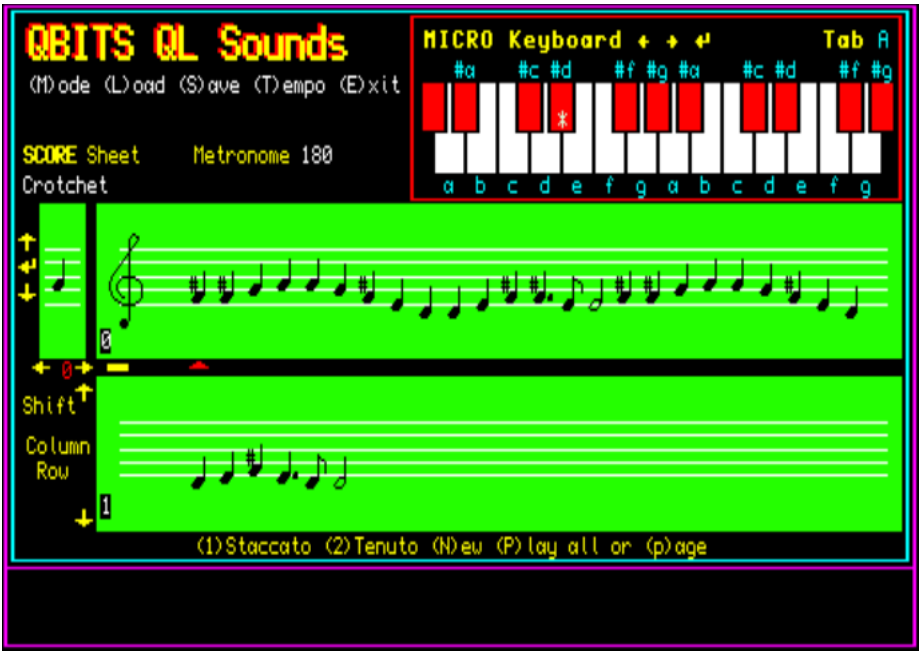
QBITS Layout Design
The two Mode approach taken is for exploring the **BEEP** parameters with the opportunity to Graphically show the Pitch & Harmonic frequencies and Waveforms proposedly produced by Wrap, Fuzz and Random. The second Mode was for Musical Scores. The **Micro Keyboard** is present in both displays and provides default Sounds for BEEP and Musical Score settings.

QBITS Menu
Top Left the Main Menu (**M**)ode (**L**)oad (**S**)ave (**T**)empo (**E**)xit. (**M**) mode alternates between BEEP and Musical Score Displays. For Storage and Retrieval of data files QBSDat_0 to 9 use (**S**)ave and (**L**)oad. (**T**)empo is only active with Musical Score Mode. (**E**)xit prompts with Y/N.

QBITS BEEP Mode Controls
To use **MICRO Keyboard**, Tab A/B extends the number of Octaves, to Select a **Note** use Left/Right Cursors keys and Enter. The BEEP Parameters, **Pitch** & **Harmonics** Frequencies and **Wave Forms** are then updated and displayed in the lower part of the screen.

Navigate the BEEP Parameters with Shift Up/Down Cursor keys. Decrease/Increase values with Shift Left/Right cursors then Press Enter to reset the BEEP values and Graphics displayed, Red for Pitch, with Pitch_2 the Harmonic in Yellow and Time Steps in Green.

The Sound is activated with Keys (**P**) (**H**) (**W**) (**F**) and (**R**). Just the Pitch with (**P**) then progressively to Random (**R**) to play all parameters. Spacebar terminates the BEEP Sound.



QBITS SCORE Mode Controls
Press (**T**) Tempo: change the Beat with Left/Right Cursor from Blank 2/2 2/4 4/4 3/4 3/8 to 6/8 and the Metronome range 30 to 240 in 10-point steps with Up/Down Cursors. Set with Enter.

Shift Up/Down Cursors changes the Score Sheet Rows shown on screen. The Shift Left/Right cursors move the Red Position marker and Spacebar toggles between upper and lower Rows.

The labelled Stave displays current selected Notes, Rests or Bars, select with Up/Down cursors. To set Pitch use Left/Right cursors of MICRO Keyboard, if a Note Symbol is displayed this will move up or down with respect to its Ledger bar position. Pressing Enter will place the Symbol selected on the Score sheet at marker position. Key (**1**) Staccato adds a dot to the Note, when played the PAUSE Delay is shortened. Key (**2**) Tenuto adds a line to the Note, when played the PAUSE Delay is lengthened.

(**N**)ew clears the Score rows of any Notes. (**P**)lay will read and play all rows displaying them as it progresses. (**p**)age reads only the two rows currently displayed.

QBITS QL Sound Data Files
Files **QBSDat_0** TO **9** used with (**S**)ave & (**L**)oad, holds arrays for the MICRO keyboard and has two sets A/B each with 0 to 23 entries, and SCORE Sheet entries with 0 to 9 rows each with 0 to 23 entries (potentially values for 240 Notes).

QBITS PROCedures

QLSounds		Init Windows, Welcome Instructions, Select default storage device	
Init_Keyboard		Setup Micro Keyboard display	
Init_keys		Seed Micro Key default configurations	
QMenu		Main Menu	
KMode		**m=0** 'Explore Sounds' : **m=1** 'Score Sheet'	
SLoad		Load from selected filename into array Key(,ln,a)	
SSave		Save to selected filename from array key(l,n,a)	
SelPath		Select device & Data file name **QBSDat_0 to 9**	
FCk		Filename check against **DIR** File List	
Tempo		Set Beat and Metronome values	
KChange(change)		Change key on Micro Keyboard range 0 to 23	
SChange(change)		Change Score Symbol	
SNew		Resets array Score (sl,sn,sa)=0	
NChange(change)		Movement of Highlight marker for Score sheets	
SEnt		Enters Symbol on Score sheet and updates Score Array	
PScore		Plays all Score sheets (P) or (p) page (Displayed Staves)	
LChange(change)		Changes lines and actions Score symbol info displayed on page.	
DSymbol		Displays Symbols selected	
SSpace		Clears display position	
Ledger		Add ledger lines when required to extend Stave	
EBar		Draws End Bar	
SBar		Draws Separation bar	
Head, Stem, Hook1, **Hook2**		Draws Parts that make up a Note Symbol	
Semibreve	**SBRest**	Draws Symbol	Beat value of 4
Minim	**MRest**	Draws Symbol	Beat value of 2
Crochet	**CRest**	Draws Symbol	Beat value of 1
Quaver	**QRest**	Draws Symbol	Beat value of ½
Semiquaver	**SQRest**	Draws Symbol	Beat value of ¼
Sharp		Pitch raised by ½ pitch	
Dot		Beat played 1½ times normal length (Beat values 3,1½, ¾)	
Staccato		Notes played Pause = 8/10 of Beat (distinct)	
Tenuto		Notes played Pause = Whole Beat (lengthy)	
GClef		Draws the G-Clef	
Tempo		Select Beat & Metronome values	
TPrt		Screen displsy updates and Prints Tempo selection	
TChange		Toggles between KMkey arrays A & B	
PChange(change)		Selects parameters of BEEP	
PPram		Screen display updates and Prints selected parameter	
BRead		Reads Beep parameters d,p,h,,t,s,w,f,r	
AChange(change)		Updates BEEP attribute checks boundaries	
BPrt		Screen Display Prints BEEP attributes	
BWave		Calculates and updates BEEP wave Info and displays to screen	

QBITS QL Sounds Code

```
1000 REMark QBITS_QLSounds_v3 (QBITS Exploring QL Sounds v3 2021 QPCII)

1002 OPEN_IN#9,'ram2_QBITSConfig':INPUT#9,gx\gy\dn$\dev$\dn%\dm%
1003 DIM Drv$(15,5):FOR d=0 TO 15:INPUT#9,Drv$(d):END FOR d:CLOSE#9

1005 WHEN ERRor
1006   eck=1:CONTINUE
1007 END WHEN

1009 MODE 4:Init_win:Init_Keys:QBITS_QLSounds

1011 DEFine PROCedure QBITS_QLSounds
1012 ac=0:ar=0:sl=0:sn=0:ds=12:bn=1:mn=200
1013 mp=1:kg=0:kn=12::m=0:BMode:fnum=0:SD$='QBSDat_'
1014 OVER#3,-1:CURSOR#3,ka,ku:PRINT#3,'*':OVER#3,0
1015 REPeat Mlp
1016 IF m=0 AND sx=1:BPrt:BWave
1017 k=CODE(INKEY$(-1))
1018  SELect ON k
1019   =192:IF kn>  0:KChange -1:BRead:SChange 0          :REMark Cursor Left
1020   =200:IF kn<23:KChange  1:BRead:SChange 0          :REMark Cursor Right
1021   =208:IF  m=1 AND ds> 0  :SChange -1                :REMark Cursor Up
1022   =216:IF  m=1 AND ds<15  :SChange  1                :REMark Cursor Down
1023   =196:IF  m=0:AChange -1 :ELSE NChange mp,-1        :REMark Cursor Shift Left
1024   =204:IF  m=0:AChange  1 :ELSE NChange mp, 1        :REMark Cursor Shift Right
1025   =212:IF  m=0:PChange -1 :ELSE RChange -1           :REMark Cursor Shift Up
1026   =220:IF  m=0:PChange  1 :ELSE RChange  1           :REMark Cursor Shift Down
1027   =  9:IF kg=0:kg=1:ELSE kg=0:END IF :TChange        :REMark Tab Toggle A/B Grp
1028   = 10:IF  m=0:BPrt:BWave:ELSE SEnt                  :REMark Enter Sounds/Score
1029   = 32:IF m=0:BEEP:ELSE IF mp=1:mp=-1:ELSE mp=1:END IF :NChange mp,0
1030   =66, 98:BRead:Bep d,p,h,t,s,w,f,r:PAUSE 20:BEEP   :REMark (B)eep Test Note
1031   =77,109:IF m=0:m=1:SMode:ELSE m=0:BMode            :REMark (M)ode BEEP/Score
1032   =76,108:SelPath:FCheck:KLoad:IF m=1:RChange 0     :REMark (L)oad
1033   =83,115:SelPath:FCheck:KSave                      :REMark (S)ave
1034   =84,116:IF m=1:Tempo                              :REMark (T)empo
1035   =69,101:GEXIT :BLOCK 48,10,172,36,0               :REMark (E)xit
1036   =78,110:IF m=1:SNew:l=1                           :REMark (N)ew
1037   =80,112:IF m=0:BEEP d,p:ELSE PScore               :REMark (P)itch/(P)lay
1038   =72,104:IF m=0:BEEP d,p,h,t,s                     :REMark +(H)armonic
1039   =87,119:IF m=0:BEEP d,p,h,t,s,w                   :REMark +(W)raps
1040   =70,102:IF m=0:BEEP d,p,h,t,s,w,f                 :REMark +(F)uzz
1041   =82,114:IF m=0:BEEP d,p,h,t,s,w,f,r               :REMark +(R)andom
1042   =49:IF ar=0:ar=1:ELSE ar=0:END IF :SChange 0      :REMark (1)Staccato
1043   =50:IF ar=0:ar=2:ELSE ar=0:END IF :SChange 0      :REMark (2)Tenuto
1044  END SELect
1045 END REPeat Mlp
1046 END DEFine
```

Note: The Main Menu gives access to nearly all functions both in **BEEP Mode** when exploring the parameters or in **Score Mode** when building a piece of music. Select (**M**)ode and further actions by pressing Keys enclosed in brackets. Navigate by use of the Cursor keys with/without Shift, action with Spacebar and Enter.

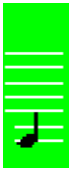
```
1048 DEFine PROCedure GEXIT
1049 INK 7:CURSOR 172,36:PRINT 'Exit Y/N':PAUSE:IF KEYROW(5)=64:LRUN dn$
1050 END DEFine

1052 DEFine PROCedure KChange(change)          Note: Controls the Micro Keyboard Key Changes
1053 OVER#3,-1:CURSOR#3,ka,ku:PRINT#3,'*':OVER#3,0
1054 kn=kn+change:ka=16
1055 IF kn> 2:ka=26
1056 IF kn> 7:ka=35
1057 IF kn>14:ka=44
1058 IF kn>19:ka=53
1059 ka=ka+kn*9:ar=0:SChange 0
1060 SELect ON kn:=1,4,6,9,11,13,16,18,21,23:ku=36:ac=1
1061 SELect ON kn:=0,2,3,5,7,8,10,12,14,15,17,19,20,22:ku=52
1062 OVER#3,-1:CURSOR#3,ka,ku:PRINT#3,'*':OVER#3,0
1063 IF m=0:INK 3:FOR kp=0 TO 7:PPram
1064 IF m=0:kp=0:INK 7:PPram
1065 END DEFine

1067 DEFine PROCedure SChange(change)         Note: Controls the Selection of Symbol Changes
1068 na=10:nu=56:IF m=0:RETurn
1069 BLOCK 26,62,14,76,4:BLOCK 100,10,0,64,0:INK 7
1070 FOR i=0 TO 12 STEP 3:LINE 7,56+i TO 15,56+i
1071 ds=ds+change:DSymbol:INK 7:CURSOR 4,64:PRINT N$
1072 END DEFine

1074 DEFine PROCedure SNew                    Note: Clears displayed score and Score Array
1075 BLOCK 444,62,46, 76,4:FOR i=0 TO 12 STEP 3:LINE 20,56+i TO 198,56+i
1076 BLOCK 444,62,46,145,4:FOR i=0 TO 12 STEP 3:LINE 20,18+i TO 198,18+i
1077 FOR l=0 TO 9
1078  FOR n=0 TO 23
1079    FOR a=0 TO 4:Score(l,n,a)=0
1080  END FOR n
1081 END FOR l
1082 GClef:l=0:CURSOR 48,78:PRINT l:CURSOR 48,146:PRINT l+1
1083 sn=0:NChange 1,0:bn=1:mn=200:TPrt
1084 END DEFine

1086 DEFine PROCedure NChange(mp,change)      Note: Controls position of Note Change
1087 sn=sn+change:IF sn<0 OR sn>23:sn=0
1088 BLOCK 426,6,64,139,0:INK 2:ma=42+sn*6.5:mu=42
1089 BLOCK 12,10,20,138,0:CURSOR 20,138:PRINT FILL$(' ',2-LEN(sn))&sn
1090 FILL 1:LINE ma-2,mu TO ma,mu+mp TO ma+2,mu TO ma-2,mu:FILL 0:INK 7
1091 END DEFine

1093 DEFine PROCedure SEnt                    Note: Enters Selected Note /Symbol into Score
1094 IF mp=1:nu=56:l=sl
1095 IF mp=-1:nu=18:l=sl+1
1096 na=42+sn*6.5:Score(l,sn,0)=kn:Score(l,sn,1)=ds
1097 IF ar=1 OR ar=2:Score(l,sn,3)=ar:ELSE Score(l,sn,3)=0
1098 DSymbol:Score(l,sn,4)=nv:ac=0:ar=0:SChange 0
1099 sn=sn+1:IF sn>23:sn=23:NChange mp,0:ELSE NChange mp,0
1100 END DEFine
```

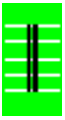
```
1102 DEFine PROCedure PScore                    Note: Plays back Score
1103 LOCal l:mp=1:nt=kn
1104 IF k=112:lmin=sl:lmax=sl+1:ELSE lmin=0:lmax=9:sl=0:LChange 0
1105 FOR l=lmin TO lmin+1
1106  IF l=lmin:mp=1:ELSE mp=-1
1107  FOR n=0 TO 23
1108    ds=Score(l,n,1):IF ds<3:NEXT n
1109    kn=Score(l,n,0)
1110    nv=Score(l,n,4):dur=3000*nv/mn:sn=n:NChange mp,0:del=5
1111    IF Score(l,n,3)=1:del=8:dur=dur-1
1112    IF Score(l,n,3)>1:del=2:dur=dur+2
1113    IF ds<8:PAUSE dur+del
1114    IF ds>7:BRead:BEEP d,p,h,t,s,w,f,r:PAUSE dur:BEEP:PAUSE del
1115  END FOR n
1116 END FOR l
1117 IF lmax=9 AND lmin<8:lmin=lmin+2:RChange 2:GO TO 1105
1118 BEEP:kn=nt:sn=0:ds=12:SChange 0:mp=1:NChange 1,0
1119 END DEFine
```

QBSDat_0 Demo file [the extract below from Beethoven's Ninth Symphony + Plus others Lines 2/3 4/5 6/7]


```
1121 DEFine PROCedure RChange(change)          Note: Row Change displays Score Lines
1122 sl=sl+change:tn=kn
1123 IF sl<0:sl=0
1124 IF sl>8:sl=8
1125 IF sl<9:CURSOR 48,126:PRINT sl:CURSOR 48,192:PRINT sl+1
1126 FOR sn=0 TO 23
1127   na=42+sn*6.5
1128   nu=56:kn=Score(sl,sn,0):ds=Score(sl,sn,1):ar=Score(sl,sn,3):DSymbol
1129   nu=18:kn=Score(sl+1,sn,0):ds=Score(sl+1,sn,1):ar=Score(sl+1,sn,3):DSymbol
1130 END FOR sn
1131 kn=tn:ds=12:sn=0:NChange 1,0:TPrt
1132 END DEFine
```

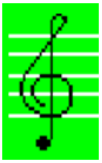
```
1134 DEFine PROCedure DSymbol
1135 SSpace:INK 0:ac=0
1136 IF ds>7:SELect ON kn=1,4,6,9,11,13,16,18,21,23:ac=1
1137 IF ds>7 AND MKey(0,kn,8)= -3   :lu=nu-3:Ledger
1138 IF ds>7 AND MKey(0,kn,8)= -4.5:lu=nu-3:Ledger:lu=lu-3:Ledger
1139 IF ds>7 AND MKey(0,kn,8)= -6   :lu=nu-3:Ledger:lu=lu-3:Ledger
1140 IF ds>7:nu=nu+MKey(0,kn,8)
1141 SELect ON ds
1142  =  0:nv=0                    :N$='Space'
1143  =  1:nv=0   :EBar            :N$='End Bar'
1144  =  2:nv=0   :SBar            :N$='Bar Seperator'
1145  =  3:nv=4   :SBRest          :N$='Semibreve Rest'
1146  =  4:nv=2   :MRest           :N$='Minim Rest'
1147  =  5:nv=1   :CRest           :N$='Crotchet Rest'
1148  =  6:nv=  .5:QRest           :N$='Quaver Rest'
1149  =  7:nv=.25:QRest:SRest      :N$='SemiQuaver Rest '
1150  =  8:nv=4   :Semibreve       :N$='Semibreve'
1151  =  9:nv=3   :Minim:Dot       :N$='Minim+Dot'
1152  =10:nv=2   :Minim            :N$='Minim'
1153  =11:nv=1.5:Crotchet :Dot     :N$='Crotchet+Dot'
1154  =12:nv=1   :Crotchet         :N$='Crotchet'
1155  =13:nv=.75:Quaver:Dot       :N$='Quaver+Dot'
1156  =14:nv=  .5:Quaver           :N$='Quaver'
1157  =15:nv=.25:Semiquaver       :N$='Semiquaver'
1158 END SELect
1159 IF ds>7
1160   IF ac=1:Sharp:ac=0
1161   IF ar=1:Staccato
1162   IF ar=2:Tenuto
1163 END IF
1164 END DEFine

1166 DEFine PROCedure SSpace
1167 LOCal x,y,su:x=na-2.5:y=nu+22:INK 4
1168 FILL 1:LINE x,y TO x+6.5,y TO x+6.5,y-34 TO x,y-34 TO x,y:FILL 0
1169 INK 7:FOR su=0 TO 12 STEP 3:LINE x,nu+su TO x+7,nu+su
1170 END DEFine

1172 DEFine PROCedure Ledger
1173 INK 7:LINE na-2,lu TO na+5,lu:INK 0
1174 END DEFine

1176 DEFine PROCedure EBar
1177 SBar:FILL 1
1178 LINE na+1,nu TO na+1,nu+12 TO na+1.6,nu+12 TO na+1.6,nu TO na+1,nu:FILL 0
1179 END DEFine

1181 DEFine PROCedure SBar
1182 LINE na,nu TO na,nu+12.5
1183 END DEFine
```





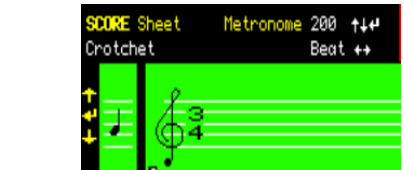
```
1185 DEFine PROCedure SBRest                    Semibreve Rest
1186 FILL 1:LINE na-1,nu+7.5:LINE_R TO 3,0 TO 0,1 TO -3,0 TO 0,-1:FILL 0
1187 END DEFine

1189 DEFine PROCedure MRest                     Minim Rest
1190 FILL 1:LINE na-1,nu+6.2:LINE_R TO 3,0 TO 0,1 TO -3,0 TO 0,-1:FILL 0
1191 END DEFine

1193 DEFine PROCedure Crest                     Crocket Rest
1194 LINE na+2,nu+4.5:FILL 1:ARC_R TO -2,-3,3*PI/4 TO 1,4,-3*PI/4:LINE_R TO -2,1
1195 ARC_R TO 0,4.5,3*PI/4:LINE_R TO 3,-2:ARC_R TO 1,-4.5,3*PI/4:FILL 0
1196 END DEFine

1198 DEFine PROCedure QRest                     Quaver Rest
1199 LINE na+.8,nu+3 TO na+2,nu+9:LINE_R TO -2,-2,
1200 FILL 1:CIRCLE_R 0,.6,.6:FILL 0
1201 END DEFine

1203 DEFine PROCedure SRest                     Semiquaver Rest
1204 LINE na,nu TO na+1.5,nu+6 TO na-.5,nu+3.5
1205 FILL 1:CIRCLE_R 0,.8,.6:FILL 0
1206 END DEFine

1208 DEFine PROCedure Head
1209 CIRCLE na,nu,1.5,.6,-PI/4
1210 END DEFine

1212 DEFine PROCedure Stem
1213 IF kn>13
1214  LINE na-1.1,nu-.5 TO na-1.1,nu-6
1215 ELSE
1216  LINE na+1.2,nu+.5 TO na+1.2,nu+6
1217 END IF
1218 END DEFine

1220 DEFine PROCedure Flag1
1221 IF kn>13
1222  LINE_R TO 2,1.5  TO 0,2.5
1223 ELSE
1224  LINE_R TO 2,-1.5 TO 0,-2.5
1225 END IF
1226 END DEFine

1228 DEFine PROCedure Flag2
1229 IF kn>13
1230   LINE_R TO 0,-1 TO -2,-1
1231 ELSE
1232   LINE_R TO 0,1 TO -2,1
1233 END IF
1234 END DEFine
```

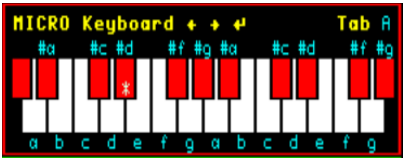
```
1236 DEFine PROCedure Semibreve
1237 CIRCLE na,nu,1.4,.7,PI/2
1238 END DEFine

1240 DEFine PROCedure Minim
1241 Head:Stem
1242 END DEFine

1244 DEFine PROCedure Crotchet
1245 FILL 1:Head:FILL 0:Stem
1246 END DEFine

1248 DEFine PROCedure Quaver
1249 Crotchet na,nu:Flag1
1250 END DEFine

1252 DEFine PROCedure Semiquaver
1253 Quaver na,nu:Flag2
1254 END DEFine

1256 DEFine PROCedure Sharp
1257 OVER 1:CURSOR na-2.5,nu+5,0,0:PRINT '#':OVER 0
1258 END DEFine

1260 DEFine PROCedure Dot
1261 FILL 1:CIRCLE na+2.5,nu,.6:FILL 0
1262 END DEFine

1264 DEFine PROCedure Staccato
1265 INK 0:FILL 1
1266 IF kn>13:CIRCLE na+1,nu+2.8,.6:ELSE CIRCLE na+.3,nu-2.8,.6
1267 FILL 0:INK 7
1268 END DEFine

1270 DEFine PROCedure Tenuto
1271 IF kn>13:LINE na,nu+4:ELSE LINE na,nu-3
1272 INK 0:FILL 1:LINE_R TO 2,0 TO 0,-.5 TO -2,0 TO 0,.5:FILL 0:INK 7
1273 END DEFine

1275 DEFine PROCedure GClef
1276 INK 0:LINE 25,57.5:ARC_R TO 1,4.5,-PI
1277 ARC_R TO 0,-6,-PI TO -3,7,-3*PI/4:LINE_R TO 5,7:ARC_R TO -2,0,PI
1278 LINE_R TO 0,-18:FILL 1:CIRCLE_R -1,0,.8:FILL 0:INK 7
1279 END DEFine
```



```
1281 DEFine PROCedure Tempo                    Note: Change of Beat / Metronome value
1282 CURSOR 160,52:PRINT 'Beat ← →':CURSOR 188,64:PRINT '↑↓ ←':BLOCK 2,4,206,54,7
1283 REPeat Tlp
1284  k=CODE(INKEY$(-1))
1285  SELect ON k
1286    =192:IF bn>1:bn=bn-1:TPrt
1287    =200:IF bn<7:bn=bn+1:TPrt
1288    =208:IF mn<240:mn=mn+10:TPrt
1289    =216:IF mn> 30:mn=mn-10:TPrt
1290    = 10:EXIT Tlp
1291  END SELect
1292 END REPeat Tlp
1293 BLOCK 52,10,160,64,0:BLOCK 24,10,188,52,0
1294 END DEFine
```


```
1296 DEFine PROCedure TPrt                     Note: Print Beat value on m1score sheet
1297 CURSOR 160,52:PRINT mn;' '
1298 SELect ON bn
1299   =1:na=32:nu=56:SSpace:RETurn
1300   =2:b1$='2':b2$='2'
1301   =3:b1$='2':b2$='4'
1302   =4:b1$='4':b2$='4'
1303   =5:b1$='3':b2$='4'
1304   =6:b1$='3':b2$='8'
1305   =7:b1$='6':b2$='8'
1306 END SELect
1307 CSIZE 2,0:INK 0:STRIP 4
1308 CURSOR 30,67,0,0:PRINT b1$:CURSOR 30,62,0,0:PRINT b2$
1309 CSIZE 0,0:INK 7:STRIP 0
1310 END DEFine
```





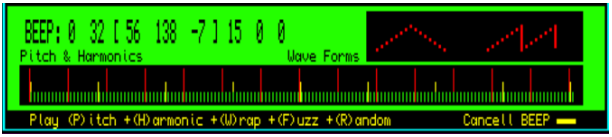

```
1312 DEFine PROCedure TChange          Note: Tab Change
1313 IF kg=0:T$='A':ELSE T$='B'
1314 CURSOR#3,260,4:PRINT#3,T$:KChange 0
1315 END DEFine

1317 DEFine PROCedure PChange(change)          Note: BEEP Parameter Change
1318 INK 3:PPram:kp=kp+change:IF kp<0 OR kp>7:kp=0:END IF :INK 7:PPram
1319 END DEFine

1321 DEFine PROCedure PPram                    Note: Print BEEP Parameter
1322 CURSOR 76,62+kp*10:PRINT '   ':CURSOR 76,62+kp*10:PRINT MKey(kg,kn,kp) TO 14
1323 END DEFine

1325 DEFine PROCedure BRead
1326 d=MKey(kg,kn,0):p=MKey(kg,kn,1):h=MKey(kg,kn,2)
1327  t=MKey(kg,kn,3):s=MKey(kg,kn,4):w=MKey(kg,kn,5)
1328  f=MKey(kg,kn,6):r=MKey(kg,kn,7)
1329 END DEFine
```

```
1331 DEFine PROCedure AChange(change)
1332 MKey(kg,kn,kp)=MKey(kg,kn,kp)+change:BRead
1333 IF d < 0 OR d > 235:d=0:MKey(kg,kn,0)=d
1334 IF p < 0 OR p > 255:p=0:MKey(kg,kn,1)=p
1335 IF h < 0 OR h > 255:h=0:MKey(kg,kn,2)=h
1336 IF t  < 0 OR  t > 235 :t=0:MKey(kg,kn,3)=t
1337 IF s< -8 OR s >    7 :s=0:MKey(kg,kn,4)=s
1338 IF w< 0 OR w >  15 :w=0:MKey(kg,kn,5)=w
1339 IF f < 0 OR  f >   15  :f=0:MKey(kg,kn,6)=f
1340 IF r < 0 OR  r >   15  :r=0:MKey(kg,kn,7)=r
1341 INK 7:PPram
1342 END DEFine
```

Note: Attribute Change of BEEP Parameter values


```
1344 DEFine PROCedure BPrt
1345 CURSOR#4,8,4:BRead:d=INT(d*10000/72):t=INT(t*10000/72)
1346 PRINT#4,'BEEP: ';d;'  ';p;' [ ';h;'  ';t;'  ';s;' ] ';w;'  ';f;'  ';r;' ':CLS#4,4
1347 END DEFine
```

BEEP Attributes



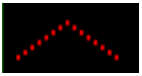
Note: Graphics to describe BEEP Waveforms

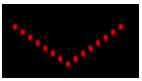
```
1349 DEFine PROCedure BWave
1350 BLOCK 468,24,12,180,0:BLOCK 180,30,300,148,0:STRIP 4:INK 0
1351 CURSOR 12,170:PRINT 'Pitch & Harmonics                Wave Forms'
1352 FOR i=0 TO 452 STEP p:BLOCK 1,20,20+i,182,2
1353 IF h>0 AND s<>0
1354    bs=SQRT(s*s):h2=INT((h-p)/bs):IF bs>h-p:h2=1
1355    FOR i=0 TO 450 STEP h :BLOCK 1,12,21+i,190,6
1356    FOR i=0 TO 450 STEP h2:BLOCK 1, 6,21+i,196,4
1357 END IF
1358 IF s>0
1359    bs=s:ba=308:bu=156
1360    FOR i=0 TO bs:BLOCK 2,2,ba+i*4,bu+i*2,2
1361    ba=308+bs*4:bu=156+bs*2
1362    FOR i=0 TO bs:BLOCK 2,2,ba+i*4,bu-i*2,2
1363 END IF
1364 IF s<0
1365    bs=SQRT(s*s):ba=308:bu=156+bs*2
1366    FOR i=0 TO bs:BLOCK 2,2,ba+i*4,bu-i*2,2
1367    ba=308+bs*4:bu=156
1368    FOR i=0 TO bs:BLOCK 2,2,ba+i*4,bu+i*2,2
1369 END IF
1370 IF w<=8 AND w>0
1371   bw=w:wa=bw*4:wu=156
1372   FOR i=0 TO bw:BLOCK 2,2,400+i*4,wu+i*2,2
1373   BLOCK 2,2*bw,400+bw*4,wu,2:BLOCK 2,2*bw,400+bw*8,wu,2
1374   FOR i=0 TO bw:BLOCK 2,2,400+wa+i*4,wu+i*2,2
1375 END IF
1376 IF w>8
1377    bw=w-8:wa=bw*4:wu=156+bw*2
1378    FOR i=0 TO bw:BLOCK 2,2,400+i*4,wu-i*2,2
1379    BLOCK 2,2*bw,400+bw*4,156,2:BLOCK 2,2*bw,400+bw*8,156,2
1380    FOR i=0 TO bw:BLOCK 2,2,400+wa+i*4,wu-i*2,2
1381 END IF
1382 STRIP 0:INK 6
1383 END DEFine
```

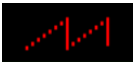
Pitch Red

1st Harmonic Yellow

2nd harmonic Green

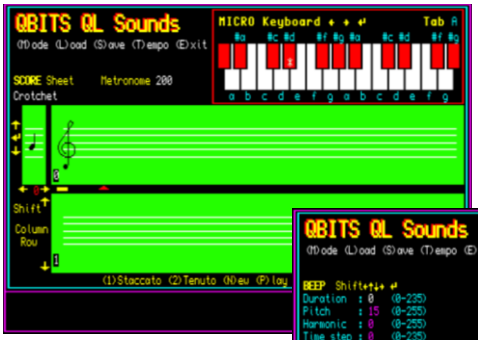




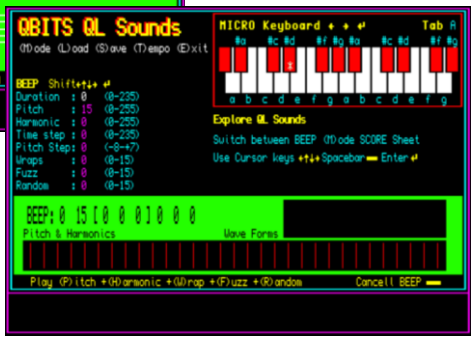


```
1385 DEFine PROCedure SMode
1386  BLOCK 220,168,0,50,0:BLOCK 452,62,46,76,4:BLOCK 452,62,46,145,4:INK 7
1387  FOR i=0 TO 12 STEP 3:LINE 24,56+i TO 200,56+i
1388  FOR i=0 TO 12 STEP 3:LINE 24,18+i TO 200,18+i
1389  INK 6 :CURSOR 4,52:PRINT 'SCORE Sheet     Metronome ';mn;' '
1390  OVER 1:CURSOR 5,52:PRINT 'SCORE':OVER 0:CSIZE 2,0
1391  CURSOR 8,136:PRINT '¼ ½':CURSOR 32,146:PRINT '¾':CURSOR 32,196:PRINT '¿'
1392  CURSOR 0, 86:PRINT '¾\'\'¼\'¿':BLOCK 2,4,10,98,6 :BLOCK 16,3,52,140,6
1393  CSIZE 0,0:BLOCK 498,10,0,208,0
1394  CURSOR 4,152:PRINT 'Shift':CURSOR 0,168:PRINT ' Column'\'  Row'
1395  CURSOR 100,208:PRINT '(1)Staccato (2)Tenuto (N)ew (P)lay all or (p)age'
1396  GClef:SChange 0:RChange 0:KChange 0
1397 END DEFine
```

SCORE Sheet Mode



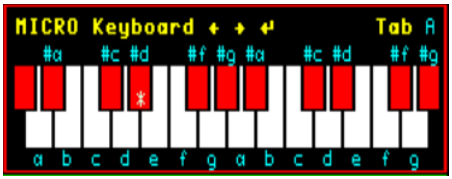
BEEP Mode




```
1399 DEFine PROCedure BMode
1400  BLOCK 220,26,0,50,0:BLOCK 498,142,0,76,0:BLOCK 496,62,2,145,4:INK 6
1401  OVER 1:FOR i=0 TO 1:CURSOR 222+i,80:PRINT 'Explore QL Sounds'
1402  CURSOR 4,52 :PRINT 'BEEP  Shift¼¾¿½ ¼':BLOCK 2,4,106,54,6
1403  CURSOR 5,52 :PRINT 'BEEP':OVER 0
1404  CURSOR  20,208:PRINT 'Play (P)itch +(H)armonic +(W)rap +(F)uzz +(R)andom'
1405  CURSOR 380,208:PRINT 'Cancell BEEP':BLOCK 16,3,458,212,6:INK 5
1406  CURSOR 4,  62:PRINT 'Duration   :   (0-235)'          :REMark d
1407  CURSOR 4,  72:PRINT 'Pitch      :   (0-255)'          :REMark p
1408  CURSOR 4,  82:PRINT 'Harmonic   :   (0-255)'          :REMark h
1409  CURSOR 4,  92:PRINT 'Time step  :   (0-235)'          :REMark t
1410  CURSOR 4,102:PRINT 'Pitch Step :   (-8-+7)'          :REMark s
1411  CURSOR 4,112:PRINT 'Wraps       :   (0 -15)'          :REMark w
1412  CURSOR 4,122:PRINT 'Fuzz        :   (0 -15)'          :REMark f
1413  CURSOR 4,132:PRINT 'Random   :   (0 -15)'          :REMark r
1414  CURSOR 222, 96:PRINT 'Switch between BEEP (M)ode SCORE Sheet'
1415  CURSOR 222,110:PRINT 'Use Cursor keys    Spacebar   Enter':INK 6
1416  CURSOR 316,110:PRINT '¼¾¿½':CURSOR 440,110:PRINT '¼'
1417  BLOCK 2,4,446,112,6:BLOCK 12,3,392,114,6:BRead:KChange 0:BPrt:BWave
1418 END DEFine
```

```
1420 DEFine PROCedure Init_win
1421 OPEN#5,scr_:WINDOW#5,512,256,gx,gy:PAPER#5,0:BORDER#5,1,3:CLS#5
1422 OPEN#4,scr_:WINDOW#4,284,24,gx+14,gy+150:PAPER#4,4:INK#4,0:CSIZE#4,0,1
1423 WINDOW#2,504,220,gx+4,gy+4:PAPER#2,0:INK#2,7:CSIZE#2, 0,0:CLS#2
1424 WINDOW#1,504,220,gx+4,gy+2:PAPER 0:BORDER 1,5
1425 WINDOW#0,512,32,gx,gy+224:PAPER#0,0:BORDER#0,1,3:INK#0,7:CSIZE#0,0,0
1426 CSIZE 2,1:OVER 1
1427 INK 2:FOR i=0 TO 1:CURSOR 4+i,2:PRINT 'QBITS QL Sounds'
1428 INK 6:FOR i=0 TO 1:CURSOR 6+i,3:PRINT 'QBITS QL Sounds'
1429 Init_Keyboard:CSIZE 0,0:OVER 0:SCALE 120,0,0:INK 7
1430 CURSOR 6,24:PRINT '(M)ode (L)oad (S)ave (T)empo (E)xit'
1431 END DEFine
```

The **Micro Keyboard** WINDOW now includes the active **Group Array** as **A or B,** toggle between with **Tab** key.

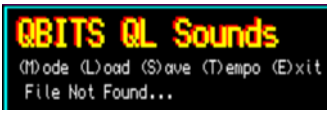



```
1433 DEFine PROCedure Init_Keyboard
1434 OPEN#3,scr_:WINDOW#3,276,74,gx+228,gy+4:PAPER#3,0:BORDER#3,1,2:CLS#3
1435 FOR i=0 TO 13:BLOCK#3,16,36,12+i*18,26,7
1436 FOR i=0 TO 14
1437   IF i=2 OR i=5 OR i=9 OR i=12
1438   ELSE
1439     BLOCK#3,16,20,3+i*18,26,0:BLOCK#3,12,19,5+i*18,26,2
1440   END IF
1441 END FOR i
1442 OVER#3,1:CSIZE#3,1,0:INK#3,6
1443 FOR i=1 TO 2:CURSOR#3,4+i,4:PRINT#3,'MICRO Keyboard ← → ←          Tab'
1444 OVER#3,0:CSIZE#3,0,0:INK#3,5:BLOCK#3,2,4,164,6,6
1445 CURSOR#3,16,16:PRINT#3,' #a   #c #d   #f #g #a   #c #d   #f #g'
1446 CURSOR#3,16,62:PRINT#3,'a  b  c  d  e  f  g  a  b  c  d  e  f  g'
1447 END DEFine
```

QBITS Programs use **SelPath** for **Load** & **Save** to choose from a list of allocated **Devices** and **Data Files** and **FCheck** to search the Selected Device File **DIR**ectroy returning ck=0 if **NOT Found** or ck=1 if **Found**.

```
1449 DEFine PROCedure SelPath
1450 INK 5:CURSOR 10,36:PRINT 'Select:↑↓    ';SD$;' ←→  ←'1451 REPeat FSel
1451 BLOCK 12,3,170,40,5:BLOCK 2,4,190,38,5:INK 7
1452 REPeat FSel
1453   CURSOR 68,36:PRINT Drv$(dn%):CURSOR 142,36:PRINT fnum:k=CODE(INKEY$(5))
1454   SELect ON k
1455     =192:IF fnum>0:fnum=fnum-1
1456     =200:IF fnum<9:fnum=fnum+1
1457     =208:dn%=dn%+1:IF dn%>15:dn%=0
1458     =216:dn%=dn%-1:IF dn%<0:dn%=15
1459     = 10:ck=1:BLOCK 212,10,0,36,0:EXIT FSel
1460     = 32:ck=0:BLOCK 212,10,0,36,0:EXIT FSEL
1461   END SELect
1462 END REPeat FSel
1463 DFname$=Drv$(dn%)&SD$&fnum:Gf$=SD$&fnum
1464 END DEFine
```

```
1466 DEFine PROCedure FCheck
1467 BLOCK 200,10,0,36,0:IF ck=2:RETurn:ELSE CURSOR 12,36:PRINT 'Searching...'
1468 PAUSE 20:DELETE Drv$(dn%)&'FList'
1469 OPEN_NEW#99,Drv$(dn%)&'FList':DIR#99,Drv$(dn%):CLOSE#99
1470 OPEN_IN#99,Drv$(dn%)&'FList'
1471 REPeat dir_lp
1472  IF EOF(#99):CLOSE#99:BLOCK 212,10,0,36,0:ck=0:EXIT dir_lp
1473  INPUT#99,fchk$:IF fchk$==Gf$:CLOSE#99:ck=1:EXIT dir_lp
1474 END REPeat dir_lp
1475 END DEFine
```




```
1477 DEFine PROCedure KLoad
1478 SelPath:BLOCK 212,10,0,36,0:IF ck=0:RETurn :ELSE FCheck
1479 IF ck=0 OR eck=1
1480    CURSOR 12,36:PRINT 'File Not Found...':PAUSE 50
1481    eck=0:BLOCK 212,10,0,36,0:RETurn
1482 END IF
1483 OPEN_IN#99,DFname$:CURSOR 12,36:PRINT 'Loading...'
1484 FOR kg=0 TO 1
1485  FOR kn=0 TO 23:FOR kp=0 TO 8:INPUT#99,MKey(kg,kn,kp):END FOR kp
1486 END FOR kg
1487 INPUT#99,mn\bn:kg=0:kn=12:kp=0:TChange
1488 FOR sl=0 TO 9
1489   CURSOR 66+sl*6,36:PRINT '.':PAUSE 5
1490   FOR sn=0 TO 23:FOR sp=0 TO 4:INPUT#99,Score(sl,sn,sp):END FOR sp
1491 END FOR sl
1492 CLOSE#99:sl=0:sn=0:sp=0:ck=0:KChange 0:PAUSE 50:BLOCK 212,10,0,36,0
1493 END DEFine
```

```
1495 DEFine PROCedure KSave
1496 SelPath:IF ck=0:RETurn :ELSE FCheck
1497 IF eck=1
1498    eck=0:CURSOR 12,36:PRINT 'DEVICE ERROR...'
1499    PAUSE 50:BLOCK 212,10,0,36,0:RETurn
1500 END IF
1501 IF ck=1
1502   CURSOR 12,36:PRINT 'Overwrite y/n':PAUSE
1503   IF KEYROW(5)<>64:BLOCK 212,10,0,36,0:RETurn
1504 END IF
1505 DELETE DFname$:OPEN_NEW#99,DFname$:CURSOR 12,36:PRINT 'Saving...'
1506 FOR kg=0 TO 1
1507  FOR kn=0 TO 23:FOR kp=0 TO 8:PRINT#99,MKey(kg,kn,kp):END FOR kp
1508 END FOR kg
1509 PRINT#99,mn\bn:kg=0:kn=12:kp=0:TChange
1510 FOR sl=0 TO 9
1511   CURSOR 66+sl*6,36:PRINT '.':PAUSE 5
1512   FOR sn=0 TO 23:FOR sp=0 TO 4:PRINT#99,Score(sl,sn,sp):END FOR sp
1513 END FOR sl
1514 CLOSE#99:sl=0:sn=0:sp=0:ck=0:KChange 0:PAUSE 50:BLOCK 212,10,0,36,0
1515 END DEFine
```

```
1517 DEFine PROCedure Init_Keys
1518 REMark Mkey(kg,kn,kp)      = kg(0-1):=kn(0-23):=kp(0-8)
1519 REMark Mkey(kg,kn,0 - 1)  = d duration : =p Pitch
1520 REMark Mkey(kg,kn,2 - 4)  = h harmonic: =t time: =s Step
1521 REMark Mkey(kg,kn,5 - 7)  = w Wrap: =f Fuzzy : =r random
1522 REMark Mkey(kg,kn,8)      = so Stave Offset -6 to +13.5 S
1523 REMark Score(sl,sn,sp)    = sl(0-9):=sn(0-23):=sp(0-4)
1524 REMark Score(sl,sn,0)     = kn Note number(0-23)
1525 REMark Score(sl,sn,1)     = ds display symbol =0 to 23
1526 REMark Score(sl,sn,2)     =   Spare
1527REMark Score(sl,sn,3)      = ar Articulation 1=Staccato 2=Tenuto(Legato)
1528 REMark Score(sl,sn,4)     = nv Note/Rest value =0 or 4,3,2,1.5,1,0.75,0.5,0.25
1529 DIM MKey(1,23,8),Score(9,23,4)              :REMark Tab A/B keys
1530 DATA 41,38,36,33,31,28,26,24,22,20,19,17,15,14,12,11,10,9,8,7,6,5,4,3
1531 DATA -6,-6,-4.5,-3,-3,-1.5,-1.5,0,1.5,1.5,3,3
1532 DATA 4.5,4.5,6,7.5,7.5,9,9,10.5,12,12,13.5,13.5
1533 RESTORE 1530:FOR kn=0 TO 23:READ p  :MKey(0,kn,0)=0:MKey(0,kn,1)=p
1534 RESTORE 1531:FOR kn=0 TO 23:READ so:MKey(0,kn,8)=so
1535 DATA 0,82,164,1, 7,15,0,0      :REMark kn=0
1536 DATA 0,76,152,1,-7, 7,0,0      :REMark kn=1
1537 DATA 0,72,144,1, 7,15,0,0      :REMark kn=2
1538 DATA 0,66,132,1,-7, 7,0,0      :REMark kn=3
1539 DATA 0,62,124,1, 7,15,0,0      :REMark kn=4
1540 DATA 0,56,112,1,-7, 7,0,0      :REMark kn=5
1541 DATA 0,52,104,1, 7,15,0,0      :REMark kn=6
1542 DATA 0,48, 96,1,-7, 7,0,0      :REMark kn=7
1543 DATA 0,44, 88,1, 7,15,0,0      :REMark kn=8
1544 DATA 0,40, 80,1,-7, 7,0,0      :REMark kn=9
1545 DATA 0,38, 76,1, 7,15,0,0      :REMark kn=10
1546 DATA 0,34, 68,1,-7, 7,0,0      :REMark kn=11
1547 DATA 0,30, 60,1, 7,15,0,0      :REMark kn=12
1548 DATA 0,28, 56,1,-7, 7,0,0      :REMark kn=13
1549 DATA 0,24, 48,1, 7,15,0,0      :REMark kn=14
1550 DATA 0,22, 44,1,-7, 7,0,0      :REMark kn=15
1551 DATA 0,20, 40,1, 7,15,0,0      :REMark kn=16
1552 DATA 0,18, 36,1,-7, 7,0,0      :REMark kn=17
1553 DATA 0,16, 32,1, 7,15,0,0      :REMark kn=18
1554 DATA 0,14, 28,1,-7, 7,0,0      :REMark kn=19
1555 DATA 0,12, 24,1, 7,15,0,0      :REMark kn=20
1556 DATA 0,10, 20,1,-7,14,0,0      :REMark kn=21
1557 DATA 0, 8, 16,1, 7,15,0,0      :REMark kn=22
1558 DATA 0, 6, 12,1,-7, 7,0,0      :REMark kn=23
1559 RESTORE 1535                   :REMark Tab B Keys
1560 FOR kn=0 TO 23
1561   READ d,p,h,t,s,w,f,r:MKey(1,kn,0)=d:MKey(1,kn,1)=p
1562   MKey(1,kn,2)=h:MKey(1,kn,3)=t:MKey(1,kn,4)=s
1563   MKey(1,kn,5)=w:MKey(1,kn,6)=f:MKey(1,kn,7)=r
1564 END FOR kn
1565 kg=0:ka=143:ku=52;kn=12:TChange
1566 END DEFine
```

Notes on BEEP Parameters
The DATA lines for the MICRO Keyboard Tab B are experimental. When running the program these can be overwritten as can the Tab A to create a set of personalised sounds.

Sound Construction, **duratio**n should be in steps of 200 milliseconds. For **Pitch_1 & Pitch_2, Fundamental** and **Harmonic,** the following pages identify the QL Sound range of frequencies. Typically **Pitch_2** is a multiple of **Pitch_1** then by increasing the **Time** and/or **Step** (**Grad_x, Grad_y**) alters the composite sound output by the number of interim frequencies. **Wrap** creates a changing pattern of rising or falling scale.

Note: Table for own use...

	Duration	Pitch	Harmonic	Time	Step	Wrap	Fuzzy	Random	Remark
0									
1									
2									
3									
4									
5									
6									
7									
8									
9									
10									
11									
12									
13									
14									
15									
16									
17									
18									
19									
20									
21									
22									
23									

Pitch vs Frequency on the **Sinclair QL by Marq**

If the **QL BEEP** highest pitch 0 = 1313Hz and lowest pitch 255 is a frequency of 43Hz. Assuming that the formula is of the form *a/(x+c)*, we get approximately the following relationship between frequency (f) and Pitch (p): f = 11336.256 / (p + 8.634) and p = 11336.256 / f - 8.634

Playable Notes and their BEEP value:

F1	251
F#1	236
G1	222
G#1	210
A1	197
A#1	187
B1	175
C2	165
C#2	155
D2	146
D#2	137
E2	129
F2	121
F#2	114
G2	107
G#2	101
A2	94
A#2	89
B2	83

(From here; off about 0.5 Hz max.)

C3	78
C#3	73
D3	69
D#3	64
E3	60
F3	56
F#3	53
G3	49
G#3	46
A3	43
A#3	40
B3	37

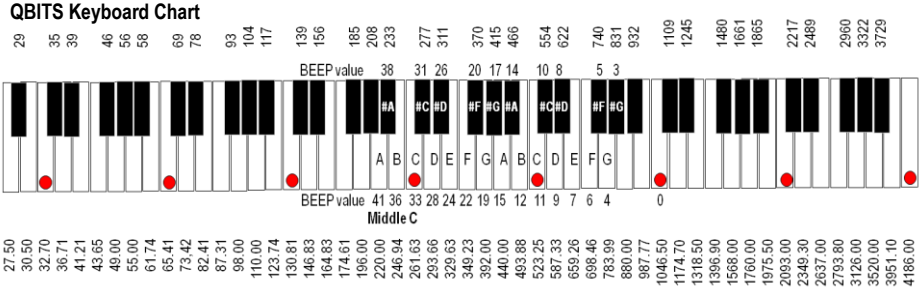
(Deviations have grown to 1 Hz+)

C4	35	Middle C 261.63
C#4	32	
D4	30	
D#4	28	
E4	26	
F4	24	
F#4	22	
G4	20	
G#4	19	
A4	17	
A#4	16	
B4	14	

(At this point the notes have very little to do with the periods, but let's keep going for the sake of completeness...)

C5	13
C#5	12
D5	11
D#5	10
E5	9
F5	8
F#5	7
G5	6
G#5	5
A5	4
B5	3
C6	2
D6	1
E6	0

QBITS Keyboard Chart



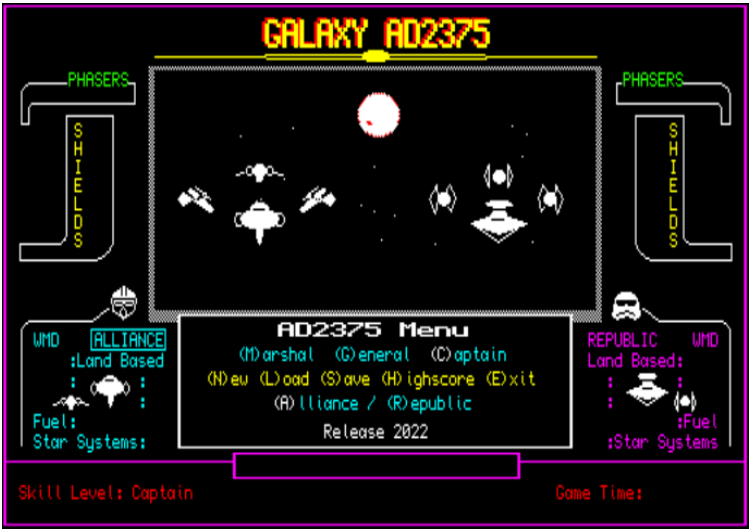
Beep Pitch frequencies supplied by Marq: **QBITS highlighted Octave Notes**

0	1313.00	47	203.77	**94**	**110.45 A3**
1	1176.71	48	200.17	95	109.39
2	**1066.05 C6**	49	**196.69 G3**	96	108.34
3	974.42	50	193.34	97	107.32
4	897.29	51	190.10	98	106.31
5	**831.48 G5#**	52	186.96	99	105.32
6	**774.66 G5**	53	183.93	**100**	**104.35 G2#**
7	**725.11 F5#**	54	180.99	101	103.40
8	**681.52 F5**	55	178.15	102	102.47
9	**642.87 E5**	56	175.39	103	101.55
10	**608.37 D5#**	57	172.72	104	100.65
11	**577.38 D5**	58	170.13	105	99.76
12	**549.40 C5#**	59	167.61	106	98.89
13	**524.01 C5**	60	165.17	**107**	**98.04 G2**
14	**500.85 B5**	61	162.80	108	97.20
15	479.66	62	160.49	109	96.37
16	**460.19 A5#**	63	158.25	110	95.56
17	**442.24 A5**	64	156.07	111	94.76
18	425.63	65	153.95	112	93.97
19	**410.23 G4#**	66	151.89	113	93.20
20	**395.90 G4**	67	149.88	114	92.44
21	382.54	68	147.93	115	91.69
22	**370.06 F4#**	69	146.02	116	90.96
23	**358.36 F4**	70	144.17	117	90.23
24	347.38	71	142.35	118	89.52
25	337.05	72	140.59	119	88.82
26	**327.32 E4**	73	138.87	120	88.13
27	318.13	74	137.19	121	87.45
28	**309.45 D4#**	75	135.55	122	86.78
29	301.22	76	133.94	123	86.12
30	**293.43 D4**	77	132.38	124	85.47
31	286.02	**78**	**130.85 C3**	125	84.83
32	**278.99 C4#**	79	129.36	126	84.20
33	272.28	80	127.90	127	83.58
34	265.90	81	126.47	128	82.97
35	**259.80 C4**	82	125.08	129	82.37
36	253.98	83	123.71	130	81.77
37	**248.42 B4**	84	122.38	131	81.19
38	243.09	85	121.07	132	80.61
39	237.99	86	119.79	133	80.04
40	**233.09 A4#**	87	118.54	134	79.48
41	228.40	**88**	**117.31 A3#**	135	78.92
42	223.89	89	116.11	136	78.38
43	**219.55 A4**	90	114.93	137	77.84
44	215.38	91	113.78	138	77.31
45	211.36	92	112.65	139	76.79
46	**207.50 G3#**	93	111.54	140	76.27

141	75.76	181	59.78	221	49.37
142	75.26	182	59.47	**222**	**49.15 G1**
143	74.76	183	59.16	223	48.94
144	74.27	184	58.85	224	48.73
145	73.79	185	58.54	225	48.52
146	73.31	**186**	**58.24 A2#**	226	48.31
147	72.84	187	57.95	227	48.11
148	72.37	188	57.65	228	47.91
149	71.92	189	57.36	229	47.70
150	71.46	190	57.07	230	47.50
151	71.01	191	56.79	231	47.31
152	70.57	192	56.50	232	47.11
153	70.14	193	56.22	233	46.92
154	69.70	194	55.94	234	46.72
155	69.28	195	55.67	235	46.53
156	68.86	196	55.40	236	46.34
157	68.44	**197**	**55.13 A2**	237	46.15
158	68.03	198	54.86	238	45.96
159	67.63	199	54.60	239	45.78
160	67.22	200	54.34	240	45.59
161	66.83	201	54.08	241	45.41
162	66.44	202	53.82	242	45.23
163	66.05	203	53.57	243	45.05
164	65.67	204	53.31	244	44.87
165	**65.29 C2**	205	53.06	245	44.70
166	64.91	206	52.82	246	44.52
167	64.54	207	52.57	247	44.35
168	64.18	208	52.33	248	44.17
169	63.82	209	52.09	249	44.00
170	63.46	210	51.85	250	43.83
171	63.11	211	51.61	251	43.66
172	62.76	212	51.38	252	43.49
173	62.41	213	51.15	253	43.33
174	62.07	214	50.92	254	43.16
175	61.73	215	50.69	**255**	**43.00 F1**
176	61.40	216	50.47		
177	61.07	217	50.24		
178	60.74	218	50.02		**32.70 C1**
179	60.42	219	49.80		
180	60.10	220	49.58		

QBITS QL BEEP Summary

Although the BEEP parameters produce a large range of sounds and potentially some musical Notes, it lacks the range of controls or mix of a basic electronic keyboard. However, I feel learning more about the QL BEEP command has been an experience worth pursuing and I've enjoyed tinkering with the code in putting together this Prog...



Galaxy Introduction

Back in the day as they say Computer Games always held an interest, but mostly in trying to understand the code around the screen Graphics and action. Through exploring the coding, I was keen to start writing and try out my own. High on my list was a Space adventure and so early attempts on **QBITS Galaxy** began **i**n the later part of the nineteen eighties. It was a time when I had perhaps an overly enthusiastic interest in Science Fiction and played a lot of board games with family and friends.

However, in June 1987 Sinclair QL World Magazine published **Stellaris** by David Cormona. I now realise this had a big impact on what I was trying to achieve and a possible reason why I didn't pursue my Game at the time. A few years back while reviving and expanding on the early code my thoughts centred on whether there was any connection with the nineteen-eighties QL Game and release of **Stellaris Galaxy** by Paradox Development Studio in 2016.

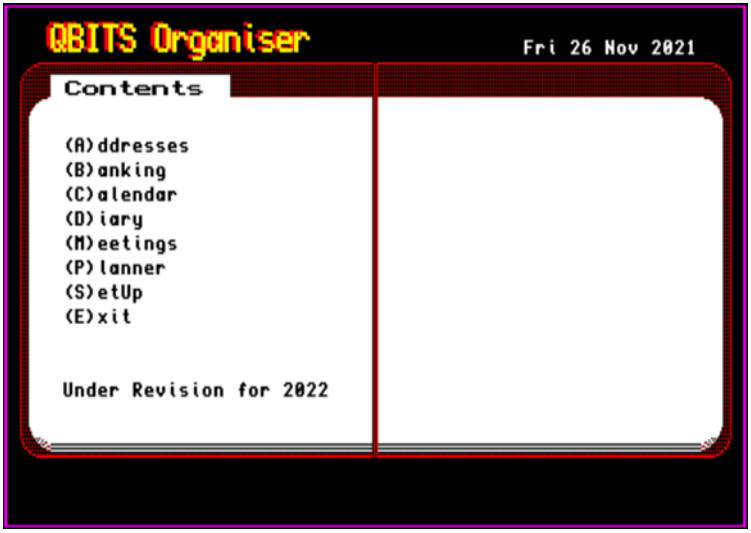
QBITS Galaxy AD2370v5

This version having reached a level of maturity, I returned to my original workings. Although the present screen layout work well enough there were a number of modifications I wanted to make. In the end it seemed more relevant to move the Game up to a New Time Line.

QBITS Galaxy AD2375

Hence by adding five years, there are changes to the Layout and the sides are now Alliance and Republic with more sophisticated Space craft. A fuel bar has been added, the Trade options expanded and more work put into Phasor and Shield Graphics for the Encounter section.

My aim is to complete the Work in Progress by the first quarter 2022.



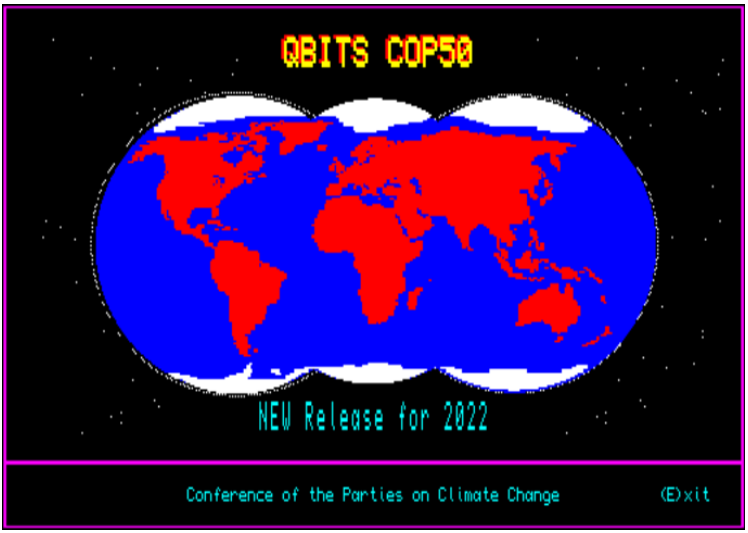
Organiser Introduction

Nineteen eighty Calculations for a **QBIT Organiser** revealed a year's storage of DATA might be in the order of 300Kbytes. The Program itself was edging to around 30KBytes for the thousand lines of code. This prompted dividing the DATA storage requirement into monthly SAVE/LOAD files, so as to reduce the RAM allocation to between 60Kbytes and 80Kbytes.

What was almost out of reach just a few short decades ago seems insignificant with modern computing power and their Gigabytes of RAM and Storage space. The upgraded versions of QL hardware and even more so the Software Emulators running on today's high performance platforms far exceed that of the QL's humble beginnings.

QBITS Organiser Contents began as a collection of programs, each interesting in their own way but unfortunately residing on floppies that were largely defunct. I did manage to copy some across and add in the missing code. Then carried out a number of updates before presenting a version in recent postings to the QLForum.co.uk.

After further investigations it has prompted another Revision which I hope to Release in 2022.



COP50 Introduction

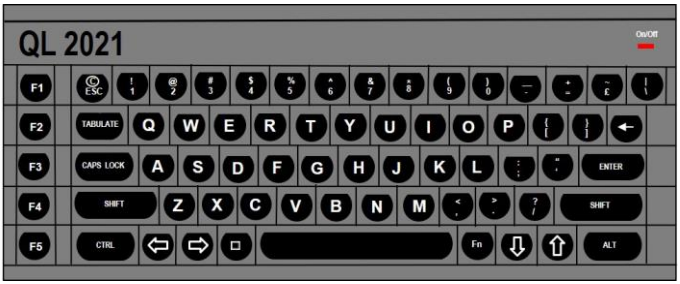
The COVID19 Pandemic made many more aware of how Internationally we are all connected and depend on each other's reaction to posed threats to our existence. In that light Observations on Climate Change stated at COP26 has again awakened the urgency of extending Global efforts to reduce Carbon Emissions and oceanic pollution affecting our Planetary ecosystem. The expert's analysis has given a clear warning that a rise in Global temperatures above 1.5 degrees if not checked will bring about extreme and irreversible changes to weather patterns affecting millions for the decades and centauries to come. And therein lies the possible demise of modern humanity.

QBITS COP50 Concept

Some background reading and the noticeable weather changes discussed by family and friends has already helped in my own education on Climate Change. So it will come as no surprise that my thoughts have become focused on creating a Game Scenario covering these Global issues.

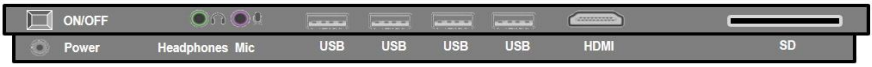
The family response was favourable and not without a few suggestions. I'm looking forward to the challenge QBITS COP50 will present. Hopefully it will reflect the scale of changes necessary and complex difficulties in accomplishing the required International Cooperation. The Pledges made and complied with and the possible disaster that awaits us all if they aren't!

This will be the QBITS Project for 2022.



QBITS QL2021

Last but not least, aspirations of a Quantum Leap unfulfilled. An envisaged 2021 design that epitomises the original concept of a keyboard computer. The old QL Microdrive section removed leaving the keyboard layout, but with three extra keys. A reduced Enter allowing the adding of a Delete, and a reduced Spacebar with a Start & Function set either side. The bottom housing replaced with a slimmer one with a slight gradient low at the front rising to higher at the back. Hardware a quad processor with 4/8GByte of RAM and 32/64 GByte for Storage.



Along the back edge a slot for a memory card, then an HDMI video/audio port, four USB ports, sockets for Mic and Stereo Earphones and the Power socket with an on/off button. The HDMI output capable of handling HD 4K screen resolutions. An upgraded Stereo output that automatically cuts sound output to HDMI socket if headphones are plugged in. WiFi, Bluetooth included or externally attached via USB. Other USB ports used for mouse, a Printer, possibly an external hard drive, CD/DVD player or a Floppy disk drive for backwards compatibility…

One option would be modifying a QL Keyboard top casing, replacing the bottom with a new housing maybe built with a 3D printer. The Internal hardware an advanced SuperGold card or a Raspberry PI with a few add-on's…

On power up SMSQ/E and SBASIC Interpreter loaded. Two additional commands CRUN which would self-Compile a SBASIC program before running as a Job. CSAVE would likewise Compile and Save the code for later loading as an EXEC file.