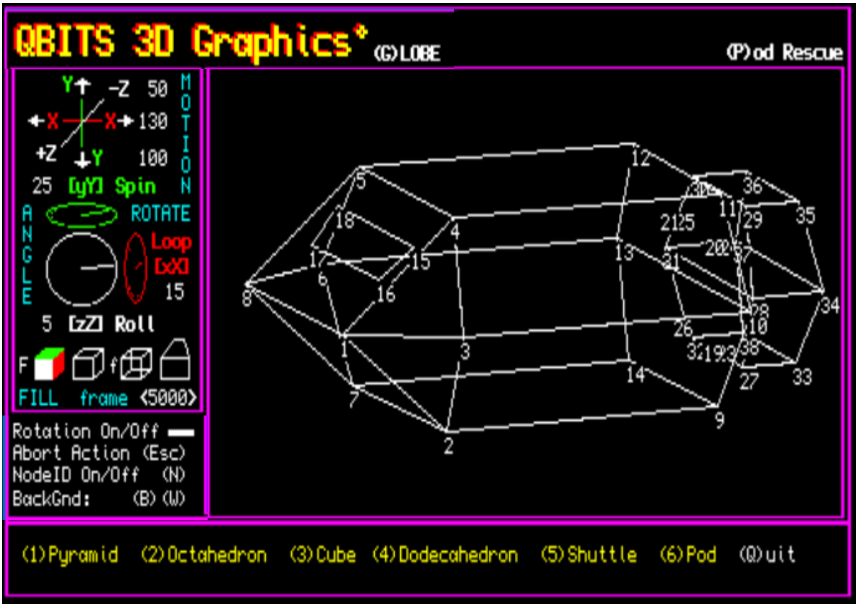
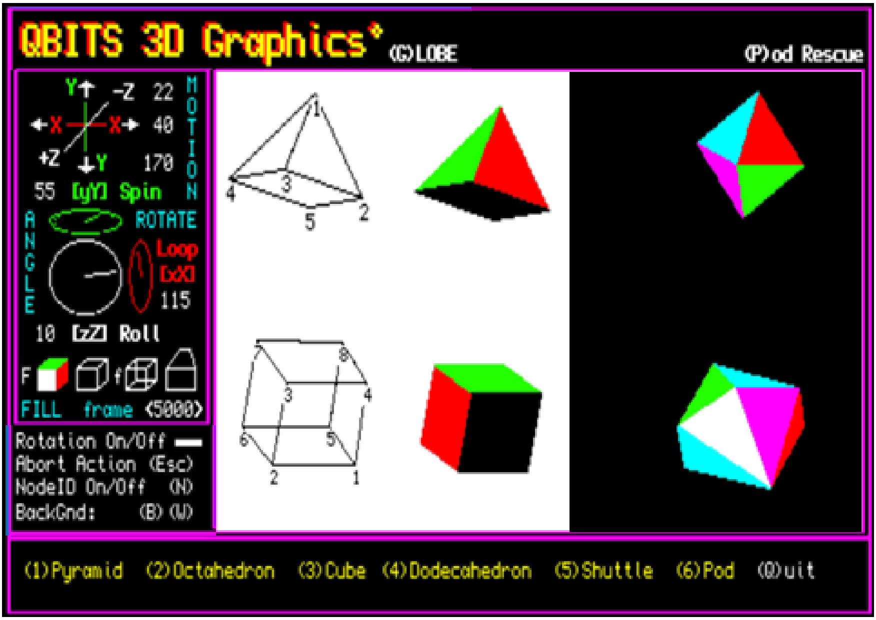
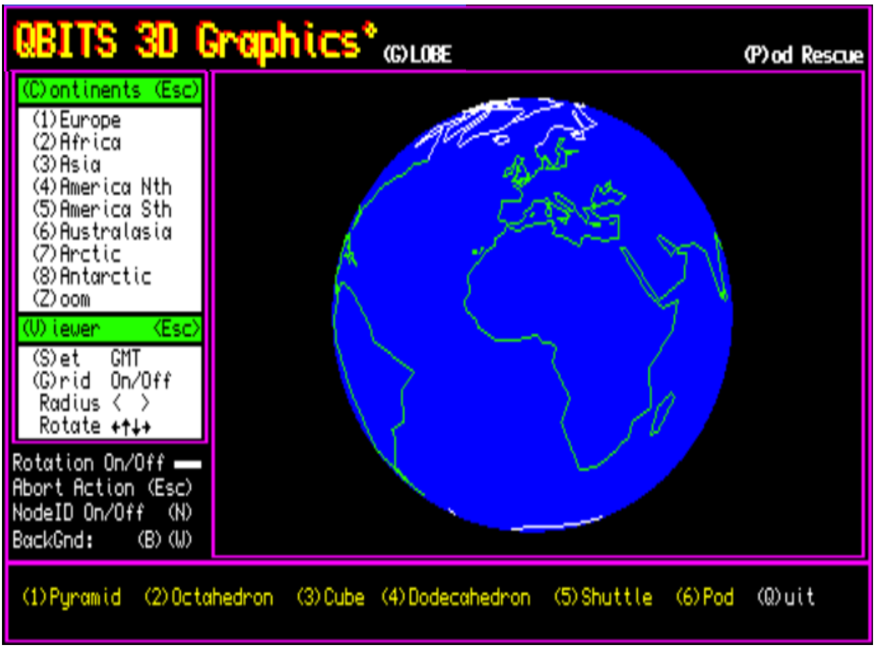
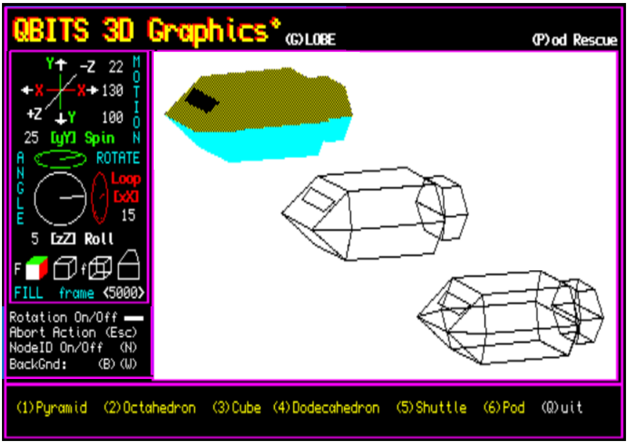
Sinclair QL 3D Rotation Graphics



Sinclair QL 3D Rotation Graphics


QBITS 3D Graphics*
(G)LOBE
(P)od Rescue
MOTION
ANGLE
[yY] Spin
ROTATE
Loop [xX]
[zZ] Roll
FILL frame (5000)
Rotation On/Off
Abort Action (Esc)
NodeID On/Off (N)
BackGnd: (B) (W)
(1)Pyramid (2)Octahedron (3)Cube (4)Dodecahedron (5)Shuttle (6)Pod (Q)uit


QBITS 3D Graphics*
(G)LOBE
(P)od Rescue
(C)ontinents (Esc)
(1)Europe
(2)Africa
(3)Asia
(4)America Nth
(5)America Sth
(6)Australasia
(7)Arctic
(8)Antarctic
(Z)oom
(V)iewer (Esc)
(S)et GMT
(G)rid On/Off
Radius < >
Rotate
Rotation On/Off
Abort Action (Esc)
NodeID On/Off (N)
BackGnd: (B) (W)
(1)Pyramid (2)Octahedron (3)Cube (4)Dodecahedron (5)Shuttle (6)Pod (Q)uit



Introduction

The availability of home computing in the 1980's sparked myself and others interest in creating computer generated graphics, especially when it involved the manipulation of 3D images. My early attempt to create the illusion of movement began in the mid-eighties writing QL SuperBASIC code to display a 3D Rotating Object a simple Cube. My original QLs performance did not inspire me to continue and the coding was probably a little out of my league at the time, however I did jot down some notes for future review.

QBITS Progs

2023 and my preferred QL Platform is the **QPC2 Emulator**, where in recent years I have upgraded and developed new QBITS Progs. I now have a group of QL SuperBASIC programs accessed via a Menu program named **QBITSProgs.** Each program when actioned by the Menu Loads and starts by importing a common set of variables from a **QBITSConfig** file. This has led to a code Review where future QBITS Progs will be Released with a single format that hopefully runs across a variety of QL platforms.

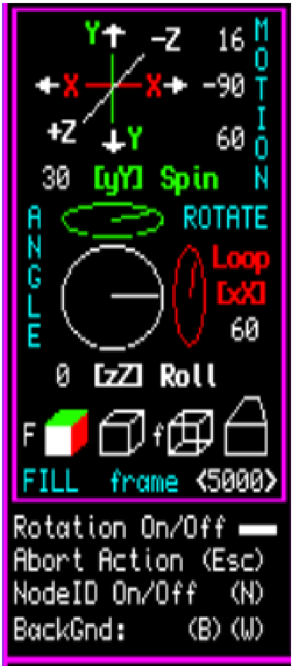
QBITS 3DGraphics SE

With any upgrade one looks to improve and add more features. This version of Exploring 3D Graphics has a revised display for the action keys and updates of variables as the 3D Object moves through changing positions. A Rescue Pod has been added that can be viewed separately or in combination with the Space Shuttle.

This Special Edition explores further aspects of 3D Rotating Graphics. It includes a simulation where you take control and manoeuvre the Space Shuttle to Dock and Lock with the revolving Rescue Pod [Not as easy a task as you might think

Globe displays a revolving World Map with a Continents Menu to select different areas and a Zoom to enlarge them. Also a Viewer that allows changes to other aspects of the display size, orientation or with/without a longitude/latitude Grid.

QBITS 3DGraphics SE – Control Keys
Images convey more than words so they say, for this revised and Special Edition of Exploring Three-Dimensional Illustrations, Motion and Angle of Rotation are displayed more graphically in a side window.



For **MOTION** along either of the **XX:YY:ZZ** axis lines. **X:Y** is controlled by Left, Right, Up and Down Cursor keys, for **Z** distance near '**+**' or far '**-**' keys. The Current **Z X Y** values shown to the right.

For **ANGLE** of Rotation; nn **[yY]Spin** about the **YY** axis shown as a horizontal ellipse above a Circle depicting the nn **[zZ]Roll** about the **ZZ** Axis. To the right nn **[xX]Loop** about the **XX** axis is shown as a vertical ellipse. The current values **nn** displayed with Title.

Upper case '**F**' Fills the viewable Fames with Ink Colour. Lower case '**f**' Switches between a Solid or Wireframe.
Use Left/Right '**<**'5000'**>**' Chevron keys to Decrease Increase the Focal Scale range from 250 to 7500.

Spacebar Toggle On/Off Auto **Rotation** of Object.

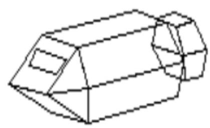
(**Esc**) Abort Globe / Pod Rescue.

'N' Toggles On/Off **Node ID's.**

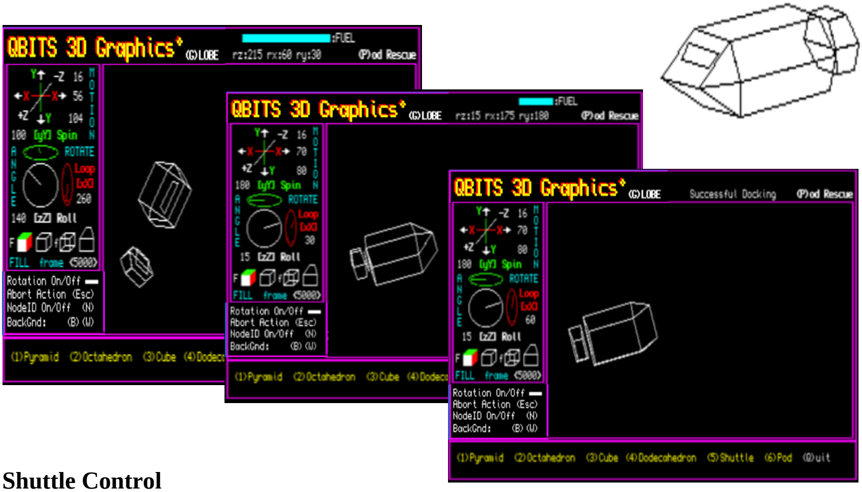
'**B**' '**W**' Switch **BackGnd** between Black or White.

QBITS 3DGraphics SE – Wireframe Objects
Upon selecting one of the Objects (1)(2)(3)(4)(5)(6) the Program first sets up the **Node xyz** coordinates and **Frame** sequences. The 2D Conversion of Vectors are then calculated so that the Wireframe can be displayed to screen. The object can then be manipulated to new positions and display its changing faces (Frames) as it is rotated.

Object (5) & (6) can be displayed separately or together. Choose (5) Shuttle or (6) Pod to view their shapes and manoeuvrability. and check of Node ID's. Then by selecting the other (6) Pod or (5) Shuttle the two are now shown linked together.

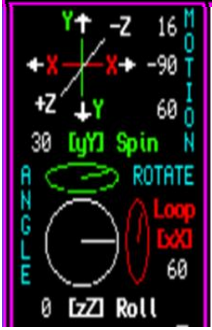


QBITS 3DGraphics SE - Pod Rescue



Shuttle Control
Pod Rescue is where you have to bring under control the Shuttle as it Rotates and moves back and forth across the screen, then manoeuvrer it to Dock with the Rescue Pod.

To take control and reposition the Shuttle you will need to fire the Shuttles directional jets so as to reorientate its position for docking with the Rescue Pod. However, each has to be countered with an opposite firing to prevent getting into further uncontrollable Motions and Rotations.



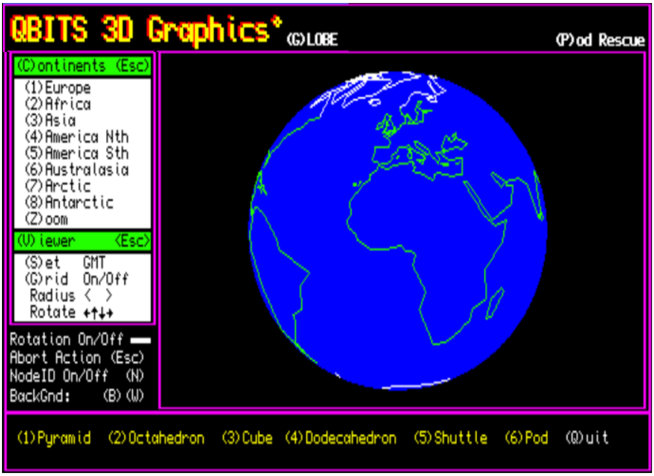
To begin **Pod Rescue** press '**P**', a **:FUEL** bar is displayed which will deplete each time a Shuttle directional Jet is fired. The window positioning **wx wy** coordinates of the Pod are not shown and have to be judged, use Motion Cursor keys and +/- for distance with values shown in side window. The **rx ry rz** Rotation angles of the Pod are shown below the fuel bar. Two are static, **rz** and **ry**, which you can match by use of **zZ** and **yY** keys; Angles displayed in side window. Having positioned and orientated the Shuttle for docking you must now set the Shuttle to Loop in Phase with the Rescue Pod using **xX**.

A successful docking ends the simulation.

If your unsuccessful you may try again.

Return or Abort Press (Esc) key.

QBITS 3DGraphics SE - Globe

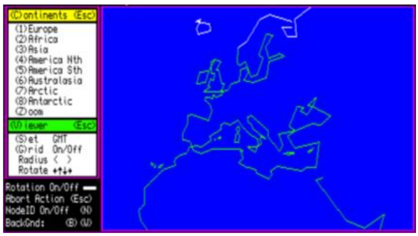


Globe Display

To access press ‘G’ the main window changes to show a World Map of Planet Earth that reveals the different continents as it revolves. Auto Rotation On/Off is controlled by the Spacebar (▬). The Side window clears to shows a Menu the (**C**)ontinents allowing selection of different areas and (**V**)iewer to change other aspect of the display.

Continents

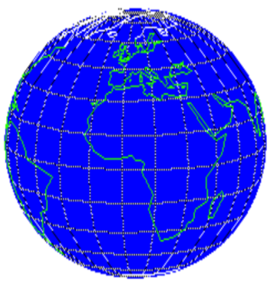
Select list by pressing ‘**C**’, the Title bar changes colour to indicate it has been actioned. Then chose an area by pressing (1) to (8). Once an area is chosen you can then Zoom in to enlarge the display.



Exit by pressing (Esc) key.

Viewer

Select by pressing ‘**V**’, the Title bar changes colour to indicate it has been actioned. Pressing ‘S’ [(S)et GMT] returns Map to the prime meridian at Greenwich.



Pressing ‘G’ [(G)rid] toggles Longitude and Latitude Grid lines On/Off.

To reduce or increase Global size use the Left and Right Chevrons [Radius < >]. To Rotate the Globe in various directions, use Left Right Up Down Cursor keys.

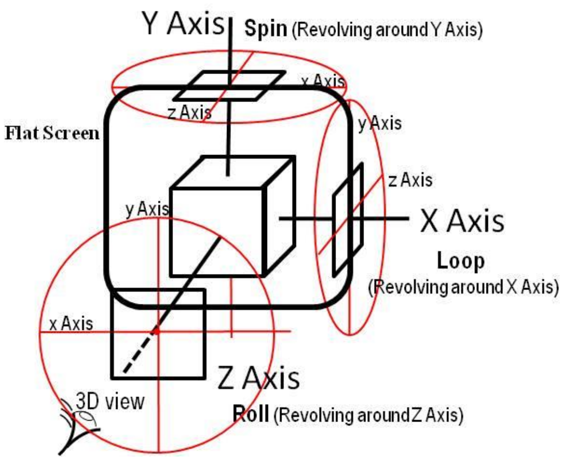
Return by using (Esc) key.

Exploring 3D Graphics

Starting with a two-dimensional object, its outline points of reference are depicted by its x y coordinates. By changing the x y coordinates values a number of x points across the screen (left to right) and by the number of y points (up or down), the object displayed is moved to a new position, this without changing its shape or size is called a translation.

For a three-dimensional object a third coordinate, usually assigned as z is added. Three-dimensional Rotation changes the orientation within each of XX:YY:ZZ relative axis. This alters the shape and size viewed and is known as a transformation. Converting a Three-Dimensional object onto a Two-Dimensional screen image requires transforming of 3D coordinates into 2D coordinates. The coding for such requires a number of steps and involves basic trigonometry.

Depending on what source you refer to or your own background you might come across a few variations on the terms used for 3D rotation. The most common being Roll, Pitch and Yaw associated with flying. I thought of others Rotate, Circulate Orbit, Spin, Loop. For my 3D Rotation Graphics, I decided on Loop, Spin & Roll. All just happen to be four letter words, a little conformity in computer coding always a good thing.



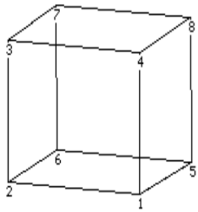
Imaginary Eye View

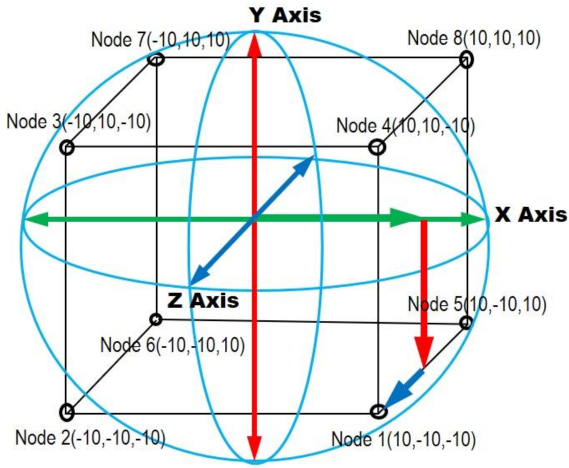
In viewing the diagram, for a flat screen it is easy enough to imaging the x y coordinates. For three-dimensional space, we need to look at points that lie in front and behind the screen. Using a Cube as our object in space, half of which is sticking out the front of the screen surface, while the other half is lying behind. Looking face on to the screen, you see a square, when you stand over the screen and look straight down you also see a square (half poking out the front, half poking out the back). Looking directly from left or right of the screen, again you see a square half out the front and half out the back.

For each point of reference that connect a 3D Object, be it a simple Cube as shown or a multisided polyhedron, shall be referred to as a Node. These points (Nodes) identify the Objects coordinates so as to Draw a 2D Wireframe as referenced to each of its axis.

Initialising xyz coordinates

The centre of the Cube is given as a Global x y position. Following the Arrows <see below> Node (1) is shown on the X axis as +x units from gy [x=0]. On the Y axis it drops below the gx [y=0] by +y units. Looking down from above we can also see it lies in front of the screen on the Z axis, this places the object closer to us so here we can give it a value of –z units.





	DATA 8	:REMark Number of Nodes
Node (1) xyz	DATA 10,-10,-10	
Node (2) xyz	DATA -10,-10,-10	
Node (3) xyz	DATA -10,-10, 10	
Node (4) xyz	DATA 10,-10, 10	
Node (5) xyz	DATA 10, 10,-10	
Node (6) xyz	DATA -10, 10,-10	
Node (7) xyz	DATA -10, 10, 10	
Node (8) xyz	DATA 10, 10, 10	

The Node coordinates can now be written as a set of DATA lines, which can used as the basic configuration information. This will apply to not only to our Cube but with any polyhedron and its multiple Nodes.

DIM x(n),y(n),z(n) where **n** is the number of **Nodes** of our polyhedron.

This is the first step to creating our screen image. The next is to consider how these points of reference might move in front of out view point. Drawing a line between points if we Roll the cube on its ZZ axis then we see a square surface turning through 360 degrees. If we Spin the cube around the YY axis then the surface presented changes to show two changing rectangular surfaces before returning to a square. A similar view is presented by Looping around the XX axis. Turning the cube on both XX YY axis the number of surfaces and their shapes change again.

Vector Calculations

To represent a 3D Object onto a 2D screen we use **Vector** Coordinates. These are the calculated **x y** 2D screen positions derived from the Global **x y** set at the centre of our object and correlates to each individual **Node x y z** coordinates.

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**).

$$x = r \times COS(a)$$
$$y = r \times SIN(a)$$

If we then rotate further the angle to b:

$$x' = r \times COS(\alpha + b)$$
$$y' = r \times SIN(\alpha + b)$$

By using trigonometric addition of each equation:

$$x' = r \times COS(a)\ COS(b) - r \times SIN(a)\ SIN(b)$$
$$y' = r \times SIN(a)\ COS(b) + r \times COS(a)\ SIN(b)$$

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 angle of rotation:

$$x' = x \times COS(b) - y \times SIN(b)$$
$$y' = y \times COS(b) + x \times SIN(b)$$

The above describes one plane we have three XYZ. For now, we can combine the required function for COS and SIN of the angle to 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 x yt – s x z : z = s x yt + c x z     X axis (y, z planes)
xt = x : x = c x xt + s x z : z = s x xt + c x z     Y axis (x, z planes)
xt = x : x = c x xt – s x y : y = s x xt + c x 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. The 3D coordinates are then transposed into 2D screen positions:

```
vx = wx + (x x fs) / (z + fs)
vy = wy + (y x fs) / (z + fs)
```

Where wx wy are the window coordinates and fs is a scale factor that determines how much we have zoomed in or out from an imaginary focal point.

The above **Vector** calculation for each Node **vx(n)** and **vy(n)** screen coordination again can be stored in a Dimensioned Array.

DIM vx(n),vy(n) where **n** is the same as the number of **Nodes**

We now have our second step whereby 3D positions can be calculated to be represented in a 2D environment. Next is to further process angular movement with changes to the **wx wy** horizontal and vertical positioning and the objects distance as viewed from the view point. This is conveyed by reducing the Object size as it moves away and making it appear bigger as it moves towards the view point.

QBITS 3D Movement & Conversion

Movement is accomplished in various ways. Rotary movement as shown is a change of angle in one of the three planes **xy zy zx Roll/Spin/Loop**. The **zZ xX yY** keys are used by the program to alter the angle for its corresponding plane lower case **zxy** for Anticlockwise and **ZXY** upper case for Clockwise.

For Global repositioning of the Object the **Cursor Left Right Up Down** keys are used to move the **wx wy** coordinates. Distance requires reducing or enlarging the screen image. The process of reading and storing the Nodes **x y z** values gave me the idea of adding a multiplier and thereby being able to increase or reduce an Objects size in a uniform manner. The vector size **vs** is simply that with a range 0.5 to 2.5 in 0.1 increments and controlled by **+/-** keys.

```
DEFine PROCedure Obj_Node
LOCal lp,a,b,c:RESTORE nres
FOR lp=sn TO mn
  READ a,b,c:x(lp)=a*vs:y(lp)=b*vs:z(lp)=c*vs
END FOR lp
END DEFine

DEFine PROCedure Obj_Calc
 cx=COS(RAD(rx)):sx=SIN(RAD(rx))
 cy=COS(RAD(ry)):sy=SIN(RAD(ry))
 cz=COS(RAD(rz)):sz=SIN(RAD(rz))
 FOR np=sn TO mn
    yt=y(np):y(np)=cx*yt-sx*z(np):z(np)=sx*yt+cx*z(np)
    xt=x(np):x(np)=cy*xt+sy*z(np):z(np)=sy*xt+cy*z(np)
    xt=x(np):x(np)=cz*xt-sz*y(np):y(np)=sz*xt+cz*y(np)
    vx(np)=wx+(x(np)*fs)/(z(np)+fs)
    vy(np)=wy+(y(np)*fs)/(z(np)+fs)
 END FOR np
 END DEFine
```

Part of the Object calculations incorporate the Perspective or Focal Scale (**fs**). Imagine a large building from a distance its shape is fairly uniform. Standing at one corner, the height above us as opposed to the height of the building further down the street appears out of proportion to its true measurement. This is what we understand as Perspective, the appearance of things relative to one another as determined by their distance from the viewer and is part of the technique of representing three-dimensional objects on a two-dimensional surface.

Using the **< >** chevron keys **fs** is Decreased or Increased between 250 and 7500. The effect of **fs** at its lower vales enlarges and distorts the Object and can look a little weird.

The fourth step is to correlate the progress made so far. We have **Nodes** and **Vector** representation, **Global Repositioning**, **Axis Rotation** but now need to bring these together and create our Object to screen. To achieve this each side or plane of our object has to be constructed as a Frame.

QBITS 3D Nodes, Vectors & Frames
Displaying a Cube, we begin by reviewing its components. A Cube has eight coordinate points (**Nodes**) and six sides (**Frames**). As with any polyhedron we need to identify the number of **Nodes**, their **xyz** values from which we calculate their **Vector** values **vx vy** for the 2D screen coordinates. Having these we can create each **Frame** from the list of Node coordinates.

QBITS 3D Screen Display
A Frame is the area contained within a set of linked Nodes. A DATA set is used to identify these linked Nodes for the program. The SuperBASIC LINE function can then be used to draw the shape of each to construct a Wireframe of the Object.

	vres	DATA 6
Frame (1) Vector a - b - c - d		DATA 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

RESTORE vres :READ vn
FOR Frames=1 TO vn [ie. 6 for Cube]
  READ a,b,c,d
  LINE vx(**a**),vy(**a**) TO vx(**b**),vy(**b**) TO vx(**c**),vy(**c**) TO vx(**d**),vy(**d**) TO vx(**a**),vy(**a**)
END FOR node

A FOR loop with **READ** function calls upon the lines of DATA that provide the instruction set to build the Wireframe. The order in which they are presented has a significance that will be explained later when exploring how Wireframe images are turned into Solid images.

QBITS 3D 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. For this 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:

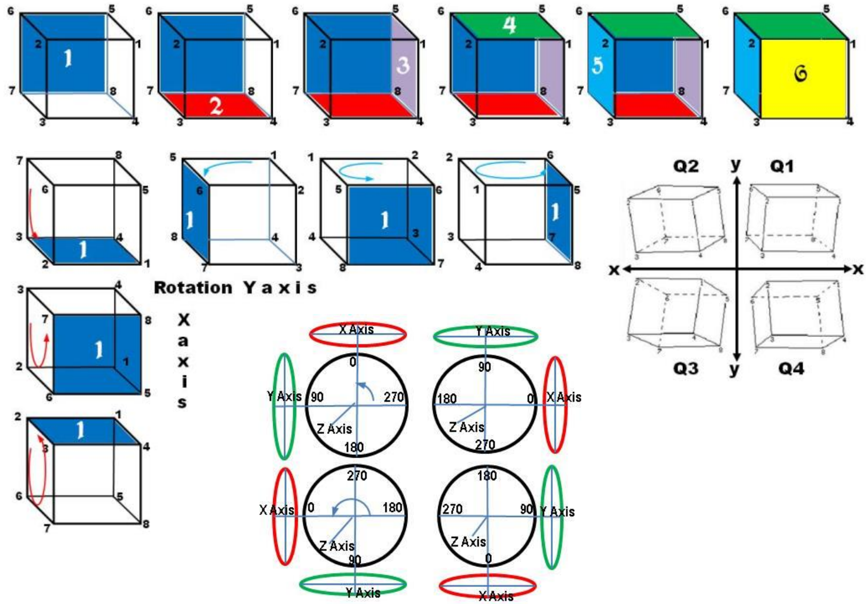
```
IF nset=2:FOR n=sn TO mn:CURSOR vx(n),vy(n),-2,2:PRINTn
```

[**sn**= start node : **mn**=max node : **n** being the Node number]

Note: When using the **xXyYzZ** keys to Loop/Spin/Roll respectively, once an Object has been rotated from its initial position the Roll/Loop and Spin key commands can act differently to what maybe expected. The positioning of the **ZXY** axis are changed and so rotate in altered planes. An example of this is where the actions of **xX** (Loop) and **yY** (Spin) or **xX** and **zZ** act in reversed to each other's original action.

QBITS Notes on XYZ Rotation

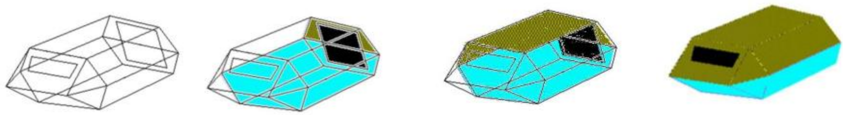
The Frame sequence as hinted before loads those Frames hidden from view first with the ones covering the viewed surfaces last. The problem is as an Object is rotated away from initial settings in any of its three axis then the sequence of Frames hidden from view and those that come into view change. The row of images below shows the initial build and display of Frame surfaces for our Cube, and then the back frame as it **Spins** and **Loops** to different positions on screen, some hidden and some in view.



The screen object is rotated by the actions of **xXyYzZ** keys. In the example shown Rotation is around the Z axis. The actions of Spin and Loop change as it moves through each quadrant. Hopefully my diagrams above explain this better than I can put into words. This gives some indication of the complexity you may face when writing code to display the viewable surfaces of a 3D rotating object.

QBITS Wireframe to Solid Object

As a Frame is by definition a closed area, we have the option to leave it unfilled as a Wireframe or coloured in to create a Solid Object using the SuperBASIC FILL function.



This brings us to a fifth step, that is how to remove those Hidden Frames???

QBITS Hidden Surface Removal

In Exploring QL 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 it can be used to see if it is pointing away or towards our view point.

The two basic types of hidden surface removal are Object-space for Three-Dimensional processing and Image-space used for Two-Dimensional processing when determining hidden surfaces.

As the above heading implies a method is sought to remove those hidden surfaces (Frames) of an object to provide a more realistic representation. Namely we seek an algorithm that identifies those Frame surfaces of an object that are not seen from the view point. The most common method used for carrying out this action in computing is called the **Plane Equation Method**.

In simple terms you compute a Vector Normal to a plane (Frame surface) such that its value indicates whether it is facing away from or towards the viewer. I have used the counter or anti-clockwise coordinates system for defining hidden QBITS Frames. This is known as the **Left-handed rule** for the Plane Equation shown below. (There is an alternative called the right-handed or clockwise system)

These are based on the 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 viewer.

Obtaining the Vector Normal we use an equation based on the plane passing through three points: **P1**=(x1, y1, z1), **P2**= (x2, y2, z2), **P3**=(x3, y3, z3):

$$\begin{matrix} x - x1 & y1 - y1 & z - z1 \\ x2 - x1 & y2 - y1 & z2 - x1 \\ x3 - x1 & y3 - y1 & z3 - x1 \end{matrix} = 0$$

The matrix equation above is equivalent to: $Ax+By+Cz+D=0$

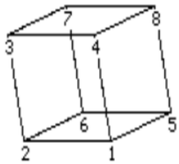
where $C = (x2 - x1)*(y3 - y1) - (x3 - x1)*(y2 - y1)$

C is the value we are interested in to determine the outward facing direction of the Frame surface and whether it is towards or away from the view point.

The last step is to Draw our Object with the option of displaying as a Wireframe or as Solid without//with the viewable planes coloured in. To achieve this, we need to understand how the culling of unwanted Frames are delt with.

QBITS Anti Clockwise Method

Going back to our Frame DATA lists you will note that the Nodes for the front facing surface are 1,2,3,4 and are ordered in a Clockwise manner and last in the list. The back face 5,6,7,8 is first in the DATA list and ordered as 8,7,6,5 or anti-clockwise. However, if you were to view this surface rotated 180 degrees to the front 8,7,6,5 is then counted in a Clockwise direction and Frame surface 1,2,3,4 is now counted anti-clockwise.



DATA 8,7,6,5,**bg2** back Frame [bg2 = Frame surface Colour]
DATA 2,6,7,3,**2**
DATA 4,3,7,8,**4**
DATA 5,1,4,8,**3**
DATA 5,6,2,1,**5**
DATA 1,2,3,4,bg2 front Frame

QBITS 3D Obj_Draw

We now have all the elements required to draw our objects image to screen, the **Node xyz** coordinates, the calculated **Vector vx vy** coordinates, the **Frame** instruction set and a method of eliminating **Hidden** frames.

```
DEFine PROCedure Obj_Draw
 LOCal lp,v,a,b,c,d,i:Obj_Node:RESTORE vres:iset=2:Obj_Calc
 FOR lp=1 TO vo
   READ a,b,c,d,i : IF cset=1:INK bg2:FILL 0:END IF
   IF cset=2:Obj_Cull:IF c1>0:GO TO 1167:END IF :INK bg2:FILL 0:END IF
   IF cset=3:Obj_Cull:IF c1>0:GO TO 1167:END IF :INK i    :FILL 1:END IF
   LINE vx(a),vy(a) TO vx(b),vy(b) TO vx(c),vy(c) TO vx(d),vy(d)  TO vx(a),vy(a)
   FILL 0
 END FOR lp
 IF nset=2:FOR n=sn TO mn:CURSOR vx(n),vy(n),-2,2:PRINT n
END DEFine
```

QBITS 3D Obj_Cull

To calculate the Vector Normal of a Frames surface the points **P1,P2,P3** are substituted with three of a Frames Node **xy** coordinates ie. x(a), y(a) : x(b), y(b) : x(c), y(c)

```
DEFine PROCedure Obj_Cull
  c1=(x(b)-x(a))*(y(c)-y(a))-(x(c)-x(a))*(y(b)-y(a))
END DEFine
```

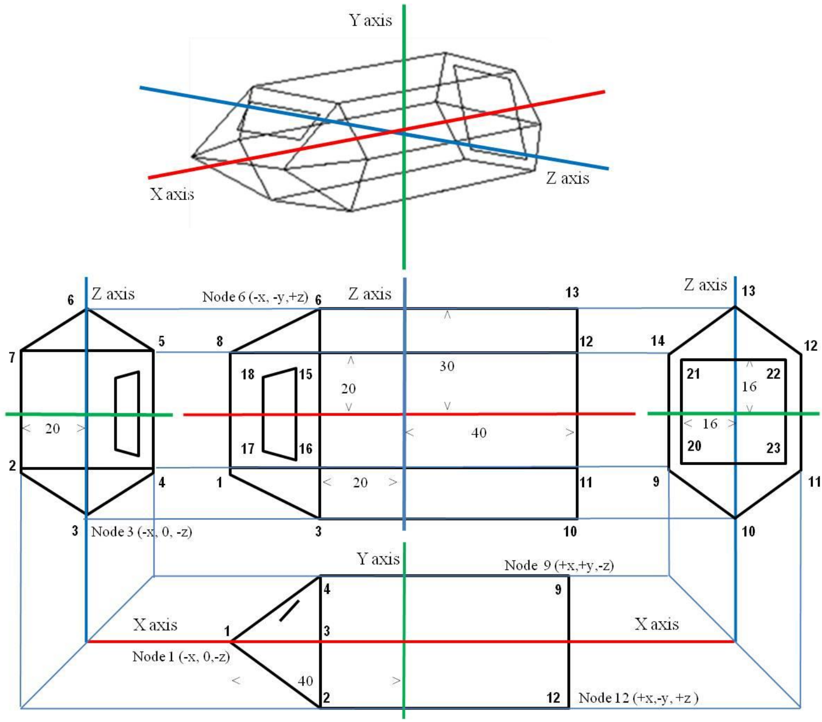
QBITS 3D Wireframe Settings

The Wireframe by default outlines all the Frames of an Object with **cset=1.** If **cset=2** the procedure **Obj_Cull** is used to eliminate hidden Frames and a Solid is displayed. If **cset=3** again **Obj_Cull** is used to eliminate hidden frames, but the viewed Frame surfaces are now **FILL**ed. The colour for a Frame surface is the Fifth value entered on the **Frame** DATA Lines (see DATA lines for the Cube above).

For Node ID display '**N**' toggles **nset** Off **= 1** On **= 2**. For development of designs the Nodes displayed can be change with **sn** start node & **mn** max node.

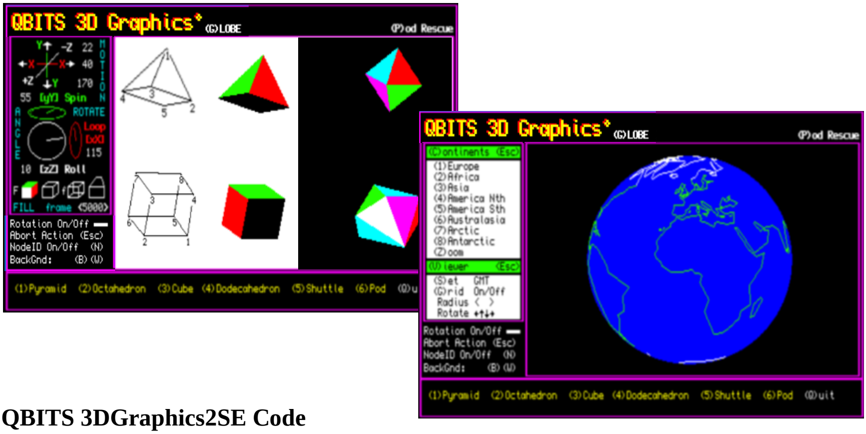
QBITS 3D Wireframe Design
To expand on the simple wireframe objects of Pyramid, Diamond, Cube, I have included a simple Space Shuttle design. First the object is drawn schematically shown with front and side elevations. This is then Mapped to the objects **XYZ** planes, with the Nodes (**xyz**) and their relevant units of distance **+/-** values.

Here's the basic layouts for the Space Shuttle showing values in the **XYZ** planes.



DATA List One can be made for each of the **Node xyz** values. For the Wireframe a **DATA List Two** is required linking Nodes to form **Frames**. These are also READ and used by the Plane Equation of **Obj_Cull** to determine if the outward surface of the polygon is facing towards or away from point of observation. It is therefore important they are seen as ordered correctly, that is Counter or Anticlockwise to viewpoint.

These lists can be added to or created as new **3D DATA Lists** following the Format presented in Program Lines 2000 onwards. Also remember to add in the **RESTORE** references **nres, vres** etc. as part of **Obj_Shape** and their Object **names** into **the Obj_Name** Procedure for screen display and action (number). The action number is entered as part of the **Menu_3DCommnads** (see Line 1081).



QBITS 3DGraphics2SE Code

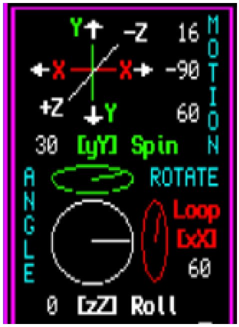
1000 REMark **QBITS_3DGraphics2SE_bas** (Exploring QL 3D Rotation Graphics2 SE 2023)

```
1002 MODE 4:gx=0:gy=0              :REMark gx:gy Screem High Res Platforms

1004 wx=0:wy=0:fs=800:vs=.8        :REMark 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              :REMark Screen settings

1008 WHEN ERRor : CONTINUE : END WHEN
```

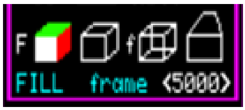
Note: QBITS 3DGraphics is one of a group of QBITS Progs that uses QBITSConfig to input common variables.

```
1010 OPEN_IN#9,'ram2_QBITSConfig':INPUT#9,gx\gy\dn$:CLOSE#9

1012 Init_win:Init_QB3D:Obj_Name:Menu_3DCommands

1014 DEFine PROCedure Init_win
1015 OPEN#4,con_10x10a10x10_4
1016 OPEN#3,scr_:WINDOW#3,116,150,4+gx,26+gy
1017 WINDOW#2,512,224,gx,gy     :BORDER#2,1,3:PAPER#2,0:CLS#2
1018 WINDOW#1,386,196,122+gx,26+gy:BORDER#1,1,3:PAPER#1,0:INK#1,7
1019 WINDOW#0,512,32,gx,224+gy    :BORDER#0,1,3:PAPER#0,0:INK#0,7:CLS#0
1020 ch=2:CURSOR#ch,0,0:OVER#ch,1
1021 CSIZE#ch,2,1:str$='QBITS 3D Graphics©'
1022 INK#ch,2:FOR i=0 TO 1:CURSOR#ch,2+i,3:PRINT#ch,str$
1023 INK#ch,6:FOR i=0 TO 1:CURSOR#ch,4+i,4:PRINT#ch,str$
1024 CSIZE#ch,0,0 :INK#ch,7
1025 CURSOR#ch,2,178:PRINT#ch,'Rotation On/Off':BLOCK#ch,16,3,98,182,7
1026 CURSOR#ch,2,188:PRINT#ch,'Abort Action (Esc)'
1027 CURSOR#ch,2,198:PRINT#ch,'NodeID On/Off  (N)'
1028 CURSOR#ch,2,209:PRINT#ch,'BackGnd:    (B)(W)'
1029 OVER#ch,0:ch=3:SCALE#ch,170,0,0:BORDER#ch,1,3
1030 INK#0,7:CURSOR#0,440,8:PRINT#0,'(Q)uit'
1031 END DEFine
```

```
1033 DEFine PROCedure Init_QB3D
1034 LOCal a,b,c,d,e,f,g,h,j,k
1035 OVER#ch,1:INK#ch,7:CSIZE#ch,2,0:mx=34:my=70:RESTORE 1036
1036 FOR i=1 TO 4:READ a,b,str$:CURSOR#ch,mx,my,a,b:PRINT#ch,str$
1037 DATA -34,-69,'←',+20,-69,' →',-6,-84,' ↑',-6,-52,' ↓'
1038 OVER#ch,1:CSIZE#ch,0,0:INK#ch,7
1039 FOR i=1 TO 11
1040   READ a,b,c,str$:INK#ch,c
1041   CURSOR#ch,mx,my,a,b:PRINT#ch,str$
1042   CURSOR#ch,mx+1,my,a,b:PRINT#ch,str$
1043 END FOR i
1044 DATA -12,-84,4,'Y',+6,-52,4,'Y',-21,-68,2,'X',+14,-68,2,'X',-28,-54,7,'+Z'
1045 DATA +16,-82,7,'-Z',-10,+20,7,'[zZ] Roll',-10,-40,4,'[yY] Spin'
1046 DATA +42,-5,2,'[xX]',+42,-16,2,'Loop',+39,+52,7,'<    >'
1047 OVER#ch,0:INK#ch,5:CURSOR#ch,70,58:PRINT#ch,'ROTATE'
1048 INK#ch,2:LINE#ch,mx-9,my+74 TO mx+11,my+74        :REMark XX
1049 INK#ch,4:LINE#ch,mx,my+62 TO mx,my+86             :REMark YY
1050 INK#ch,7:LINE#ch,mx-10,my+62 TO mx+12,my+86       :REMark ZZ
1051 INK#ch,7:CIRCLE#ch,mx,my,18,1,0                   :REMark Roll
1052 INK#ch,4:CIRCLE#ch,mx,my+27,18,.32,PI/2:          :REMark Spin
1053 INK#ch,2:CIRCLE#ch,mx+28,my,17,.32,0:INK#ch,5     :REMark Loop
1054 str$='MOTION':FOR i=1 TO 6:CURSOR#ch,100,-8+i*9:PRINT#ch,str$(i)
1055 str$='ANGLE' :FOR i=1 TO 1055:CURSOR#ch,4,49+i*9:PRINT#ch,str$(i)
```




```
1056 INK#ch,7:CURSOR#ch,2,124:PRINT#ch,'F       f'
1057 FOR i=1 TO 12
1058   READ a,b,c,d,e,f,g,h,j,k : INK#ch,j          :FILL#ch,k
1059   LINE#ch,a,b TO c,d TO e,f TO g,h TO a,b:FILL#ch,0
1060 END FOR i
1061 DATA 10,15,10,25,20,25,20,15,7,1, 20,15,20,25,25,30,25,20,2,1
1062 DATA 10,25,15,30,25,30,20,25,4,1, 30,15,30,25,40,25,40,15,7,0
1063 DATA 40,15,40,25,45,30,45,20,7,0, 30,25,35,30,45,30,40,25,7,0
1064 DATA 55,15,55,25,65,25,65,15,7,0, 60,20,60,30,70,30,70,20,7,0
1065 DATA 55,15,55,25,60,30,60,20,7,0, 65,15,65,25,70,30,70,20,7,0
1066 DATA 75,15,75,25,90,25,90,15,7,0, 75,25,82,35,87,35,90,25,7,0
1067 INK#ch,5:CURSOR#ch,2,138:PRINT#ch,'FILL  frame'
1067 INK#ch,5:CURSOR#ch,2,138:PRINT#ch,'FILL  frame'
1068 ch=1:SCALE#ch,200,-143,-100:CSIZE#ch,0,0:INK#ch,4:rx=0:ry=0:rz=0:kch=0
1069 END DEFine
```



Note: Data for building the Graphics

```
1071 DEFine PROCedure QQuit
1072 INK#0,7:CURSOR#0,480,8:PRINT#0,'Y/N':PAUSE:IF KEYROW(5)=64:STOP [LRUN dn$]
1073 END DEFine
```

```
1100 DEFine PROCedure Menu_3DCommands
1101 REPeat Com_lp
1102   SELect ON k
1103     =27:
1104     =81,113:QQuit:BLOCK#0,20,10,480,8,0                 :REMark (Q)uit
1105     = 66,98 :bg1=0:bg2=7:PAPER#1,0:CLS#1                :REMark (B)lack background
1106     =87,119:bg1=7:bg2=0:PAPER#1,7:CLS#1                 :REMark (W)hite background
1107     =49 TO 55,71,103,80,112:Obj_Shape                   :REMark Load Object DATA
1108     =  32    :IF aset=-1:aset=5:ELSE aset=-1            :REMark Toggle animation
1109     =102     :IF cset= 1 OR cset=3:cset=2:ELSE cset=1   :REMark (f)rame On/Off
1110     =  70    :IF cset= 1 OR cset=2:cset=3:ELSE cset=1   :REMark (F)ILL  On/Off
1111     =78,110:IF nset= 1 :nset=2:ELSE nset=1              :REMark (N)ode ID On/Off
1112     = 43,61 :vs=vs+.1  :IF vs>=2.5 :vs=2.5              :REMark (+)Increase Vector size
1113     = 45     :vs=vs- .1  :IF vs<= .5 :vs= .5            :REMark (-)Decrease Vector size
1114     = 62     :fs=fs+50   :IF fs>7500 :fs=7500           :REMark (>)Increase Focal Scale
1115     = 60     :fs=fs -50   :IF fs<  250 :fs=  250        :REMark (<)Decrease Focal Scale
1116     =192    :wx=wx -10 :IF wx<=  10 :wx=  10            :REMark ← move left
1117     =200    :wx=wx+10 :IF wx>=280 :wx=280               :REMark → move right
1118     =208    :wy=wy+10 :IF wy>=190 :wy=190               :REMark ↑ move up
1119     =216    :wy=wy -10 :IF wy<=  10 :wy=  10            :REMark ↓ move down
1120     = 88     :Obj_Ang:rx=rx -5:IF rx<    0:rx=rx+360    :REMark (X)Clockwise Loop
1121     =120    :Obj_Ang:rx=rx+5:IF rx>360:rx=rx -360       :REMark (x)Anti- Loop
1122     = 89     :Obj_Ang:ry=ry -5:IF ry<    0:ry=ry+360    :REMark (Y)Clockwise Spin
1123     =121    :Obj_Ang:ry=ry+5:IF ry>360:ry=ry -360       :REMark (y)Anti- Spin
1124     = 90     :Obj_Ang:rz=rz -5:IF rz<    0:rz=rz+360    :REMark (Z)Clockwise Roll
1125    =122    :Obj_Ang:rz=rz+5:IF rz>360:rz=rz -360        :REMark (z)Anti- Roll
1126   END SELect
1127   CLS
1128   IF k1 AND k2
1129     :nres=2099:sn=23:mn=38:vres=2133:vo=10:Obj_Draw
1130     nres=2075:sn=1:mn=21:vres=2115:vo=15
1131   END IF
1132   Obj_Pos:Obj_Draw:INK bg2:k=CODE(INKEY$(#4,aset))
1133   IF aset=5:Obj_Auto:Obj_Ang:PAUSE 5:ELSE Obj_Ang
1134   SELect ON k=49 TO 54,71,103,80,112:BLOCK#2,150,10,280,14,0
1135 END REPeat Com_lp
1136 END DEFine
```

Note: Updates the various Position variables

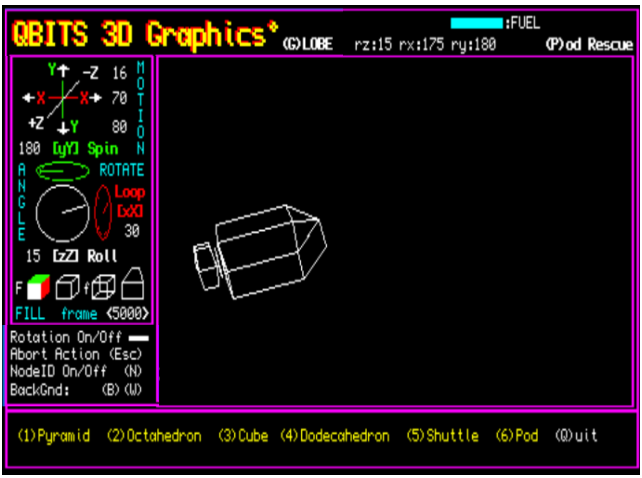
```
1112 DEFine PROCedure Obj_Pos
1139  ch=3:INK#ch,7
1140  CURSOR#ch,  4,106:PRINT#ch,FILL$(' ',3-LEN(rz))&rz
1141  CURSOR#ch,  4,  46:PRINT#ch,FILL$(' ',3-LEN(ry))&ry
1142  CURSOR#ch,84,  92:PRINT#ch,FILL$(' ',3-LEN(rx))&rx
1143  CURSOR#ch,72,  18:PRINT#ch,FILL$(' ',4-LEN(wx))&wx
1144  CURSOR#ch,78,  34:PRINT#ch,FILL$(' ',3-LEN(wy))&wy
1145  CURSOR#ch,84,    4:PRINT#ch,FILL$(' ',2-LEN(vs*20))&vs*20
1146  CURSOR#ch,80,138:PRINT#ch,FILL$(' ',4-LEN(fs))&fs
1147 END DEFine
```

```
1149 DEFine PROCedure Obj_Ang                Note: Displays the Angle changes
1150 ch=3:INK#ch,0
1151 FILL#ch,1:CIRCLE#ch,34,70,17,1,0:FILL#ch,0
1152 FILL#ch,1:CIRCLE#ch,34,97,15,.32,PI/2  :FILL#ch,0
1153 FILL#ch,1:CIRCLE#ch,62,70,14,.26,0      :FILL#ch,0
1154 INK#ch,7:LINE#ch,34,70 TO 34+17*COS(RAD(rz)),70+ 18*SIN(RAD(rz))
1155 INK#ch,4:LINE#ch,34,97 TO 34+16*COS(RAD(ry)),97+4.5*SIN(RAD(ry))
1156 INK#ch,2:LINE#ch,62,70 TO 62+  4*COS(RAD(rx)),70+ 15*SIN(RAD(rx)):ch=1
1157 END DEFine

1159 DEFine PROCedure Obj_Auto               Note: Random Change of Rotation Angles
1160 rx=rx+5*RND(1 TO 5):IF rx>=360:rx=0
1161 ry=ry+5*RND(1 TO 5):IF ry>=360:ry=0
1162 rz=rz+5*RND(1 TO 5):IF rz>=360:rz=0
1163 END DEFine

1165 DEFine PROCedure Obj_Node               Note: Load Node coordinates
1166 LOCal lp,a,b,c:RESTORE nres
1167 FOR lp=sn TO mn
1168     READ a,b,c:x(lp)=a*vs:y(lp)=b*vs:z(lp)=c*vs
1169 END FOR lp
1170 END DEFine

1172 DEFine PROCedure Obj_Calc               Note: Calculate the Vectors
1173 cx=COS(RAD(rx)):sx=SIN(RAD(rx))
1174 cy=COS(RAD(ry)):sy=SIN(RAD(ry))
1175 cz=COS(RAD(rz)):sz=SIN(RAD(rz))
1176 FOR np=sn TO mn
1177    yt=y(np):y(np)=cx*yt-sx*z(np):z(np)=sx*yt+cx*z(np)
1178    xt=x(np):x(np)=cy*xt+sy*z(np):z(np)=sy*xt+cy*z(np)
1179    xt=x(np):x(np)=cz*xt-sz*y(np):y(np)=sz*xt+cz*y(np)
1180    vx(np)=wx+(x(np)*fs)/(z(np)+fs)
1181    vy(np)=wy+(y(np)*fs)/(z(np)+fs)
1182 END FOR np
1183 END DEFine

1185 DEFine PROCedure Obj_Draw
1186 LOCal lp,v,a,b,c,d,I : Obj_Node:RESTORE vres : iset=2:Obj_Calc
1187 FOR lp=1 TO vo
1188   READ a,b,c,d,i:IF cset=1:INK bg2:FILL 0:END IF
1189   IF cset=2:Obj_Cull:IF c1>0:GO TO 1193:END IF :INK bg2:FILL 0:END IF
1190   IF cset=3:Obj_Cull:IF c1>0:GO TO 1193:END IF :INK i     :FILL 1:END IF
1191   LINE vx(a),vy(a) TO vx(b),vy(b) TO vx(c),vy(c) TO vx(d),vy(d)
1192   LINE TO vx(a),vy(a):FILL 0
1193 END FOR lp
1194 IF nset=2:FOR n=sn TO mn:CURSOR vx(n),vy(n),-2,2:PRINT n       Note: sn start node mn max node
1195 END DEFine

1197 DEFine PROCedure Obj_Cull
1198 c1=(x(b)-x(a))*(y(c)-y(a))-(x(c)-x(a))*(y(b)-y(a))      Note: Check Frame surface facing view point
1199 END DEFine
```

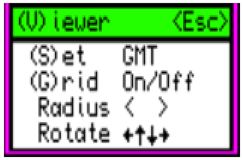
```
1201 REMark QBITS Pod Rescue

1203 DEFine PROCedure Pod_Rescue
1204 DIM Pod(6,5):RESTORE 1206:sk=2          Note: fu = sk [skill level - limits number Fuel use times]
1205 FOR a=1 TO 6:FOR b=1 TO 5:READ num%:Pod(a,b)=num%:END FOR b:END FOR a
1206 DATA 0,0,60,75,45,  30,30,15,-50,45,  15,15,90,-50,80
1207 DATA 15,60,180,-50,-25,  300,90,330,65,-30, 215,330,30,-85,-20
1208 a=RND(1 TO 6):cset=2:vs=.5:CLS:Pod_Draw
1209 CURSOR#2,400, 2:PRINT#2,':FUEL':BLOCK#2,120,6,280,4,5
1210 rzp=Pod(a,1):rxp=Pod(a,2):ryp=Pod(a,3):wxp=Pod(a,4):wyp=Pod(a,5)
1211 rx=0:ry=0:rz=0:wx=0:wy=0:vres=2115:vo=16:Obj_Draw
1212 xx=4:yy=6:ax=-5:ay=5:az=-10:ix=2:iy=2:ia=5:fu=sk:Gch=0
1213 REPeat Main_lp
1214   IF wy>=wyp-2 AND wy<=wyp+2             Note: IF check statements – alignment of Shuttle with Pod
1215    IF wx>=wxp-2 AND wx<=wxp+2
1216     IF ry>=ryp-5 AND ry<=ryp+5
1217      IF rx>=rxp-20 AND rx<=rxp+20
1218       IF rz>=rzp-5 AND rz<=rzp+5 AND ax=5:Gch=1: EXIT Main_lp     Note: Docking successful
1219      END IF
1220     END IF
1221    END IF
1222   END IF
1223   BLOCK#2,fu,10,280,4,0:IF fu>=120:Gch=0: EXIT Main_lp            Note: Docking Unsuccessful
1224   Obj_Ang:Obj_Pos:Get_Keys
1225   IF wx+xx<-125 OR wx+xx>125:xx=-xx
1226   IF wy+yy< -90 OR wy+yy> 90:yy=-yy
1227   wx=wx+xx:wy=wy+yy:rx=rx+ax:ry=ry+ay:rz=rz+az      Note: Shuttle Motion variables updated
1228   IF rx+ax<=0:rx=360+ax
1229   IF rx+ax>=360:rx=0+ax
1230   IF ry+ay<=0:ry=360+ay                   Note: List of Angle of Rotation checks and setting
1231   IF ry+ay>=360:ry=0+ay
1232   IF rz+az<=0:rz=360+az
1233   IF rz+az>=360:rz=0+az
1234   CLS:Pod_Draw:Obj_Draw:IF k=27:Gch=0:EXIT Main_lp
1235 END REPeat Main_lp
1236 BLOCK#2,150,20,280,2,0:k1=1:k2=1
1237 vs=1:wx=0:wy=0:rx=15:ry=30:rz=10:cset=3:aset=-1
1238 IF Gch=0:CURSOR#2,300,14:PRINT#2,'Hard Luck Try Again':Beeps 3
1239 IF Gch=1:CURSOR#2,300,14:PRINT#2,'Successful Docking' :Beeps 2
1240 PAUSE 50:PAPER 248:CLS:PAUSE 50:PAPER bg1:CLS
1241 END DEFine

1243 DEFine PROCedure Pod_Draw
1244 ow=wx:od=wy:ox=rx:oy=ry:oz=rz:ov=vs
1245 rz=rzp:rxp=rxp+5:ry=ryp:wx=wxp:wy=wyp:IF rxp>=360:rxp=0
1246 rx=rxp:CURSOR#2,284,14:PRINT#2,'rz:';rz;' rx:';rx;' ry:';ry;'  '
1247 nres=2099:sn=23:mn=38:vres=2133:vo=11:Obj_Draw
1248 wx=ow:wy=od:rx=ox:ry=oy:rz=oz:nres=2075:sn=1:mn=22:vres=2115:vo=16
1249 END DEFine
```

Note: Pod_Draw saves current Position, Rotation, then Sets for Rescue Pod. Pod **wx wy** are not given and has to be judged. Pod's **rz** & **ry** Angles of Rotation are fixed, but **rx** continues to Loop. All three must be matched for a **Successful Docking**.

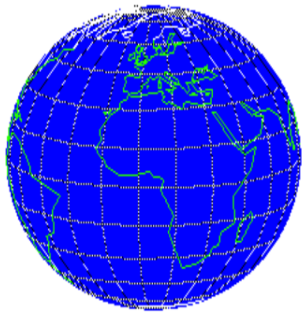
```
1251 DEFine PROCedure Get_Keys
1252 k=CODE(INKEY$(10)):tx=xx:ty=yy:bx=ax:by=ay:bz=az
1253 SELect ON k                     Note: Continuous Motion created by variables xx, yy, ix : ax, ay, az, ai
1254   =192:xx=xx -ix                :REMark Left
1255   =200:xx=xx+ix                 :REMark Right
1256   =208:yy=yy+iy                 :REMark Up
1257   =216:yy=yy -iy                :REMark Down
1258   =  88:ax=ax+ia                :REMark Loop Clockwise
1259   =120:ax=ax -ia                :REMark Loop Anti-clockwise
1260   =  89:ay=ay+ia                :REMark Spin Clockwise
1261   =121:ay=ay -ia                :REMark Spin Anti-clockwise
1262   =  90:az=az+ia                :REMark Spin Clockwise
1263   =122:az=az -ia                :REMark Spin Anti-clockwise
1264   =43,61:vs=vs+.1:IF vs>=.8:vs=.8  :REMark (+)Increase Vector size
1265   =45    :vs=vs -.1:IF vs<=.4:vs=.4  :REMark (-)Decrease Vector size
1266 END SELect
1267 IF xx>11 OR xx<-11:xx=tx        :REMark Move Limits
1268 IF yy> 8 OR yy<- 8:yy=ty
1269 IF ax>15 OR ax<-15:ax=bx        :REMark Angle Limits
1270 IF ay>15 OR ay<-15:ay=by
1271 IF az>15 OR az<-15:az=bz
1272 SELect ON k=88,89,90,120,121,122,192,200,208,216:fu=fu+sk:Beeps 1
1273 END DEFine

1275 DEFine PROCedure Beeps(b)
1276 SELect ON b
1277  =1:BEEP 5000,0,500,6,1,2,0,0
1278  =2:BEEP 9500,0,200,6,2,1,0,0
1279  =3:BEEP 30000,1,9,200,-5,8,0,0
1280  =4:BEEP 25000,0,200,8,1,2,0,0
1281  =5:BEEP 3000,0,400,2,1,0,0,0
1282 END SELect
1283 END DEFine
```



```
1300 REMark QBITS Globe WorldMap

1302 DEFine PROCedure Globe3D
1303 PAPER 0:CLS:ch=3:CLS#ch:INK#ch,0:RESTORE 1305:Beeps 4
1304 FOR i=1 TO 15:PAUSE 2:READ a,b,str$:GTitle a,b,str$
1305 DATA 4,2,'(C)ontinents (Esc)',7,14,' (1)Europe',7,23,' (2)Africa'
1306 DATA 7,32,' (3)Asia',7,41,' (4)America Nth',7,50,' (5)America Sth'
1307 DATA 7,59,' (6)Australasia',7,68,' (7)Arctic',7,77,' (8)Antarctic'
1308 DATA 7,86,' (Z)oom',4,98,'(V)iewer    <Esc>',7,110,' (S)et  GMT'
1309 DATA 7,119,' (G)rid  On/Off',7,128,'  Radius <  >',7,137,'  Rotate ¼¾¿½'
1310 BEEP
1311 R=90:wrx=0:wry=12:zm=12      :REMark Radius Pixels: x,y Angle coordinates
1312 S=0 :M=0 :P=15 :O=0          :REMark Spin/Meridian/Parallel - Rotation
1313 Gcol=248 :Ccol=4 :Acol=7     :REMark Grid/Coastline - Colours
1314 vh=1                         :REMark vh=0 make visible view hidden
1315 REPeat G_lp
1316   IF KEYROW(1)= 8:PAPER bg1:INK bg2:CLS:EXIT G_lp
1317   IF KEYROW(1)=64:IF aset=-1:aset=5:ELSE aset=-1
1318   IF KEYROW(2)= 8:Continents
1319   IF KEYROW(7)=16:Viewer
1320   S=S+3:World:Calc_ang:Grid:RESTORE 2500:Maps:PAUSE aset
1321 END REPeat G_lp
1322 END DEFine

1324 DEFine PROCedure GTitle(icol,ypos,str$)
1325 STRIP#ch,icol:CURSOR#ch,2,ypos:PRINT#ch,str$;FILL$(' ',18-LEN(str$))
1326 END DEFine

1328 DEFine PROCedure World
1329  CLS:INK 1:FILL 1:CIRCLE 0,0,R:FILL 0    :REMark gcol globe colour
1330 END DEFine

1332 DEFine PROCedure Continents
1333 GTitle 6,2,'(C)ontinents (Esc)':Beeps 5
1334  REPeat Choice
1335    k=CODE(INKEY$(-1))
1336    SELect ON k
1337      =27:R=90:GTitle 4,2,'(C)ontinents (Esc)':EXIT Choice
1338      =49:R=90:wrx= 0:wry= 45:S=  15:zm=12 :REMark (1) Europe
1339      =50:R=90:wrx= 0:wry=   6:S=  18:zm=  6 :REMark (2) Africa
1340      =51:R=90:wrx= 0:wry= 45:S=  80:zm=  3 :REMark (3) Asia
1341      =52:R=90:wrx= 0:wry= 50:S= -99:zm=  6 :REMark (4) America Nth
1342      =53:R=90:wrx= 0:wry=-20:S= -60:zm=  6 :REMark (5) America Sth
1343      =54:R=90:wrx= 0:wry=-18:S=134:zm=  6 :REMark (6) Australasia
1344      =55:R=90:wrx= 0:wry= 90:S=  15:zm=12 :REMark (7) Arctic
1345      =56:R=90:wrx= 0:wry=-90:S=    0:zm=  6 :REMark (8) Antarctic
1346      =90,122  :IF R<90+zm*10:R=R+zm        :REMark (Z)oom
1347    END SELect
1348    World:Calc_ang:Grid:RESTORE 2500:Maps:INK 0
1349 END REPeat Choice
1350 END DEFine
```



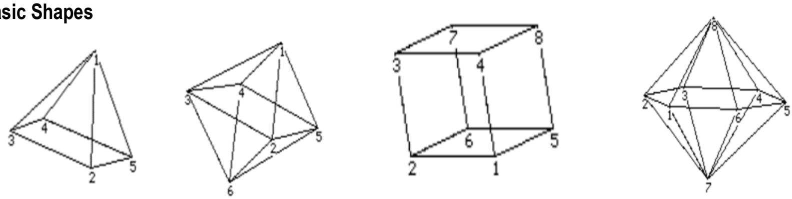

```
1352 DEFine PROCedure Viewer
1353 GTitle 6,98,'(V)iewer     (Esc)':Beeps 5
1354 REPeat View_lp
1355   k=CODE(INKEY$(-1))
1356   SELect ON k
1357      =27:GTitle 4,98,'(V)iewer     (Esc)':EXIT View_lp
1358      =115,83:wrx=0:wry=0:S=0           :REMark (S)et
1359      =103,71:IF M=0:M=15:ELSE M=0      :REMark (G)rid
1360      =  46,60:IF R<93:R=R+3            :REMark > Radius
1361      =  44,62:IF R>33:R=R-3            :REMark < Radius
1362      =192      :wrx=wrx+6              :REMark Left   world radial axis X
1363      =200      :wrx=wrx-6              :REMark Right
1364      =208      :wry=wry+6              :REMark Up     world radial axis Y
1365      =216      :wry=wry-6              :REMark Down
1366   END SELect
1367   World:Calc_ang:Grid:RESTORE 2500:Maps:INK 0
1368 END REPeat View_lp
1369 END DEFine

1371 DEFine PROCedure Grid
1372  INK Gcol:IF M=0 :RETurn
1373  FOR O=M TO 360 STEP M
1374      T=0:FOR L=90 TO -90 STEP -P:Calc_posn
1375  END FOR O
1376  FOR L=-90+g TO 90-g STEP M
1377      T=0:FOR O= 0 TO 360 STEP  P:Calc_posn
1378  END FOR L
1379 END DEFine

1381 DEFine PROCedure Maps
1382  REPeat Loop
1383   READ num,col:INK col:T=0   :IF num=9999:EXIT Loop
1384   READ L,O:Calc_posn                        :REMark Longitude & Latitude Offset
1385   FOR i=2 TO num:READ L,O:T=1:Calc_posn:END FOR i
1386  END REPeat Loop
1387 END DEFine

1389 DEFine PROCedure Calc_ang
1390  sx=SIN(RAD(wrx)):cx=COS(RAD(wrx)) :REMark x coordinates
1391  sy=SIN(RAD(wry)):cy=COS(RAD(wry)) :REMark y coordinates
1392 END DEFine

1394 DEFine PROCedure Calc_posn
1395  Ms=SIN(RAD(O-S)):Mc=COS(RAD(O-S))              :REMark O Longitude S Rotation
1396  Pc=COS(RAD(L))  :Ps=SIN(RAD(L))                :REMark L Latitude
1397  wvz=R*(Ps*sy*cx-Pc*Ms*sx+Pc*Mc*cy*cx)          :REMark Z axis
1398  wvx=R*(Pc*Ms*cx+Ps*sy*sx+Pc*Mc*cy*sx)          :REMark X axis
1399  wvy=R*(Ps*cy-Pc*Mc*sy)                         :REMark Y axis
1400  IF vh=1 AND wvz<0:T=0                          :REMark vh=0 view hidden plane
1401  IF T=0:px=wvx:py=wvy:T=1:RETurn                :REMark T=0 Set px,py
1402  IF T=1:LINE px,py TO wvx,wvy:px=wvx:py=wvy     :REMark T=1 Draw & Set px,py
1403 END DEFine
```



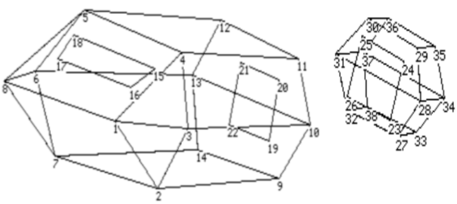
```
1950 REMark QBITS Wireframe Data

1952 DEFine PROCedure Obj_Name
1953 OVER#2,1:CURSOR#2,0,0:CSIZE#2,0,0:INK#2,7
1954 FOR i=0 TO 1:CURSOR#2,220+i,14:PRINT#2,'(G)GLOBE'
1955 FOR i=0 TO 1:CURSOR#2,432+i,14:PRINT#2,'(P)od Rescue'
1956 OVER#2,0:CSIZE#0,0,0:INK#0,6
1957 CURSOR#0,  6,8:PRINT#0,'(1)Pyramid  (2)Octahedron  (3)Cube'
1958 CURSOR#0,218,8:PRINT#0,'(4)Dodecahedron  (5)Shuttle  (6)Pod'
1959 END DEFine
```


```
1961 DEFine PROCedure Obj_Shape
1962 DIM x(40),y(40),z(40),vx(40),vy(40),fr(16,6)
1963 iset=1:Obj_Ang
1964 IF k=49:nres=2001:sn=1:mn= 5:vres=2008:vo= 5:rx=60:ry=30:rz=0
1965 IF k=50:nres=2015:sn=1:mn= 6:vres=2023:vo= 8:rx=15:ry=30:rz=0
1966 IF k=51:nres=2033:sn=1:mn= 8:vres=2043:vo= 6:rx=15:ry=30:rz=0
1967 IF k=52:nres=2051:sn=1:mn= 8:vres=2061:vo=12:rx=15:ry= 0:rz=0
1968 SELect ON k=49 TO 52:k1=0:k2=0
1969 IF k=53:nres=2075:sn= 1:mn=22:vres=2115:vo=16:rx= 0:ry=60:rz=0:k1=1
1970 IF k=54:nres=2099:sn=23:mn=38:vres=2133:vo=11:rx= 0:ry=60:rz=0:k2=1
1971 REMark WARNING maintain correct nres:vres:RESTORE DATA Lines
1972 IF k=71 OR k=103:Globe3D:CLS#3:STRIP#3,0:Init_QB3D
1973 IF k=80 OR k=112:Pod_Rescue
1974 END DEFine
```

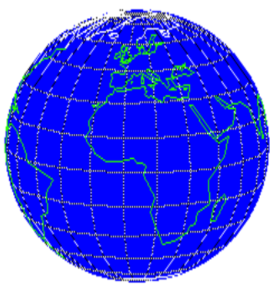
Basic Shapes

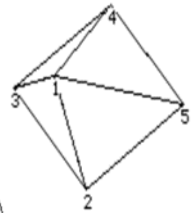


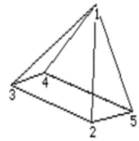
Shuttle & Pod



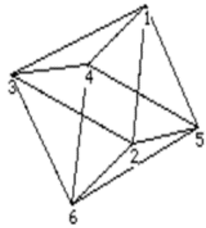
World Maps



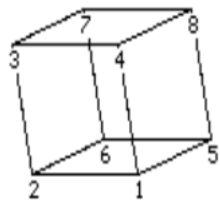




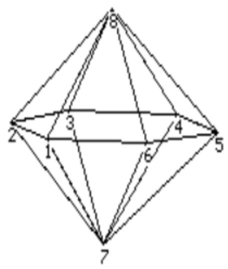

```
2000 REMark Pyramid  5 Nodes
2001 DATA    0,   0 ,-20              :REMark Node 1 nearest to view point
2002 DATA  20, 20,  20
2003 DATA  20,-20,  20
2004 DATA -20,-20,  20
2005 DATA -20, 20,  20

2007 REMark Pyramid 5 Frames
2008 DATA 1,2,3,1,2
2009 DATA 1,3,4,1,4
2010 DATA 1,4,5,1,3
2011 DATA 1,5,2,1,5
2012 DATA 5,4,3,2,bg2

2014 REMark Diamond 6 Nodes
2015 DATA    0, 30,   0               :REMark Node 1
2016 DATA  20,   0,-20
2017 DATA  20,   0, 20
2018 DATA -20,   0, 20
2019 DATA -20,   0,-20
2020 DATA    0,-30,   0               :REMark Node 6

2022 REMark Diamond 8 Frames
2023 DATA 1,2,3,1,5                   :REMark Frame 1
2024 DATA 6,3,2,6,3
2025 DATA 1,3,4,1,2
2026 DATA 6,4,3,6,4
2027 DATA 1,4,5,1,5
2028 DATA 6,5,4,6,3
2029 DATA 1,5,2,1,2
2030 DATA 6,2,5,6,4                   :REMark Frame 8
```




```
2032 REMark Cube  8 Nodes
2033 DATA  20,-20,-20                 :REMark Node 1
2034 DATA -20,-20,-20
2035 DATA -20,-20, 20
2036 DATA  20,-20, 20                 :REMark Node 4
2037 DATA  20, 20,-20                 :REMark Node 5
2038 DATA -20, 20,-20
2039 DATA -20, 20, 20
2040 DATA  20, 20, 20                 :REMark Node 8

2042 REMark Cube  6 Frames
2043 DATA 8,7,6,5,bg2                 :REMark back Frame
2044 DATA 2,6,7,3,2
2045 DATA 4,3,7,8,4
2046 DATA 5,1,4,8,3
2047 DATA 5,6,2,1,5
2048 DATA 1,2,3,4,bg2                 :REMark front Frame
```

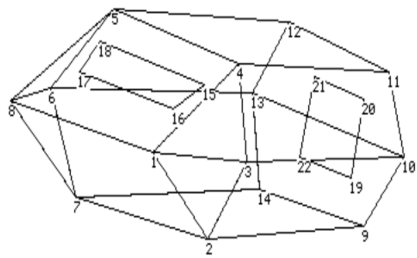


```
2050 REMark Polygon 8 Nodes
2051 DATA  32,  0,  0                 :REMark Node 1
2052 DATA  16, 24,  0
2053 DATA -16, 24,  0
2054 DATA -32,  0,  0
2055 DATA -16,-24,  0
2056 DATA  16,-24,  0
2057 DATA   0,  0,-32
2058 DATA   0,  0, 32                 :REMark Node 8
```




```
2060 REMark Polygon 12 Frames
2061 DATA 1,2,7,7,6                   :REMark 1
2062 DATA 2,3,7,7,5
2063 DATA 3,4,7,7,4
2064 DATA 4,5,7,7,3
2065 DATA 5,6,7,7,2
2066 DATA 6,1,7,7,bg2                 :REMark 6
2067 DATA 2,1,8,8,2
2068 DATA 3,2,8,8,3
2069 DATA 4,3,8,8,bg2
2070 DATA 5,4,8,8,5
2071 DATA 6,5,8,8,6
2072 DATA 1,6,8,8,4                   :REMark 12
```

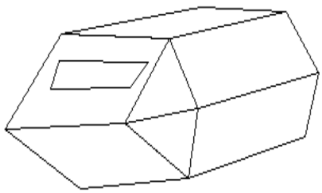
```
2074 REMark Space Shuttle 22 Nodes
2075 DATA -40,  0, 20                 :REMark Node 1
2076 DATA -18,-20, 20                 :REMark Node 2
2077 DATA -20,  0, 30
2078 DATA -18, 20, 20
2079 DATA -18, 20,-20
2080 DATA -20,  0,-30
2081 DATA -18,-20,-20                 :REMark Node 7
2082 DATA -40,  0,-20                 :REMark Node 8
2083 DATA  38,-20, 20                 :REMark Node 9
2084 DATA  40,  0, 30
2085 DATA  38, 20, 20
2086 DATA  38, 20,-20
2087 DATA  40,  0,-30
2088 DATA  38,-20,-20                 :REMark Node 14
2089 DATA -24, 14, 16                 :REMark Node 15
2090 DATA -30,  8, 14
2091 DATA -30,  8,-14
2092 DATA -24, 14,-16                 :REMark Node 18
2093 DATA  40,-10, 10                 :REMark Node 19
2094 DATA  40, 10, 10
2095 DATA  40, 10,-10
2096 DATA  40,-10,-10                 :REMark Node 22
```



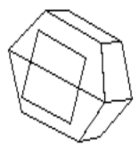
Note: Node **xyz** coordinates for Shuttle and Pod are set as part of the same group so they can become one when joined. This is handled by the DATA line references for both Nodes and Frames.

```
2098 REMark Rescue Pod 16 Nodes
2099 DATA  43,-10, 10               :REMark Node 23 1 Hatch
2100 DATA  43, 10, 10
2101 DATA  43, 10,-10
2102 DATA  43,-10,-10               :REMark Node 26 4
2103 DATA  47,-15, 12               :REMark Node 27 5 Front
2104 DATA  45,  0, 20
2105 DATA  47, 15, 12
2106 DATA  47, 15,-12
2107 DATA  45,  0,-20
2108 DATA  47,-15,-12               :REMark Node 32  10
2109 DATA  58,-15, 12               :REMark Node 33  11 Rear
2110 DATA  60,  0, 20
2111 DATA  58, 15, 12
2112 DATA  58, 15,-12
2113 DATA  60,  0,-20
2114 DATA  58,-15,-12               :REMark Node 38 16
```


```
2116 REMark Space Shuttle 16 Frames
2117 DATA 9,10,13,14,5              :REMark Rear Frames
2118 DATA 10,11,12,13,240
2119 DATA 2,9,14,7,5               :REMark Side Frames
2120 DATA 6,7,14,13,5
2121 DATA 5,6,13,12,240
2122 DATA 5,12,11,4,240
2123 DATA 4,11,10,3,240
2124 DATA 3,10,9,2,5
2125 DATA 3,2,1,3,5                :REMark Front Frames
2126 DATA 1,2,7,8,5
2127 DATA 7,6,8,7,5
2128 DATA 8,6,5,8,240
2129 DATA 4,1,8,5,240
2130 DATA 1,4,3,1,240
2131 DATA 15,16,17,18,0            :REMark Pilot Window 15
2132 DATA 19,20,21,22,191          :REMark Rear Door    16
```



```
2134 REMark Rescue  Pod 11 Frames
2135 DATA 33,34,37,38,5            :REMark Rear Frame   17
2136 DATA 34,35,36,37,240
2137 DATA 32,31,28,27,5            :REMark Front Frame   20
2138 DATA 31,30,29,28,240
2139 DATA 27,28,34,33,5            :REMark Side Frames   23
2140 DATA 28,29,35,34,240
2141 DATA 29,30,36,35,240
2142 DATA 37,36,30,31,240
2143 DATA 38,37,31,32,5
2144 DATA 32,27,33,38,5
2145 DATA 26,25,24,23,191          :REMark Pod Hatch    28
```

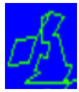


2150 REMark Globe Data for World Maps

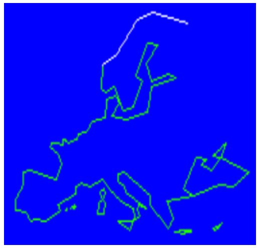
Note: Each block of Code begins with a **FOR** loop number of entries followed by an **INK** Colour.

```
2500 DATA  7,Acol        :REMark Iceland
2501 DATA  66.5,-22.5,65.4,-24.5,66.6,-16,65,-13.5,63,-19,64,-22,66.5,-22.5
```

```
2502 DATA 24,Ccol        :REMark UK & Ireland
2503 DATA  58.5,-5,58.2,-1.8,56,-3.3,56,-2,53,.5
2504 DATA  53,1.6,52.2,1.7,51.3,.8,51.3,1.5,50.9,1
2505 DATA  50,-5.8,51.4,-3.7,51.7,-5,53.3,-4.5,53.3,-3
2506 DATA  55,-3.5,54.7,-5,57.5,-6.5,58.5,-5
2507 DATA  55.3,-6.5,54.3,-10,51.4,-10,52.2,-6.3,55.3,-6.5     :REMark  48
```



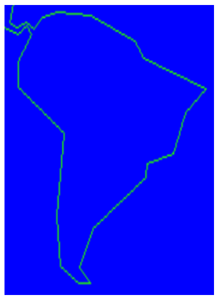
```
2508 DATA 89,Ccol        :REMark EUROPE
2509 DATA  41,29,42,35,41,38,42.5,42.3,46,37                   :REMark  10
2510 DATA  48,39,46.5,35,46,37,44.3,34,45.5,32
2511 DATA  46.2,33.5,47,31,42.5,27,41,29,40.8,23
2512 DATA  38,24,36.5,22.8,40.5,19.5,42,19.5,45.7,13.7
2513 DATA  45.5,12.3,44.4,12.3,43.6,13.6,42.5,14.1,40,18.5 :REMark  50
2514 DATA  40.5,17,39.7,16.5,39,17.2,38,15.6,38,12.5
2515 DATA  36.6,15,38.9,16.1,40,15.7,41.3,13,43,10.5
2516 DATA  44.3,8.9,43.2,6.2,43.5,4,42.7,3,41.8,3.3
2517 DATA  39.5,-.4,38.7,.3,36.6,-2.1,36.5,-4.8,36,-5.4
2518 DATA  37.1,-6.7,37,-8.8,38.6,-9.4,41.2,-8.6,43.1,-9.3     :REMark 100
2519 DATA  43.7,-7.7,43.3,-1.5,46.1,-1.2,47.3,-2.5,48,-4.7
2520 DATA  48.6,-4.7,48.8,-3.1,48.7,-1.7,49.8,-2,49.8,-1.3
2521 DATA  49.4,-1.1,49.3,-.1,49.7,.2,50.2,1.5,50.9,1.6
2522 DATA  51.4,3.6,53.3,4.7,54,8.3,57,8.1,57.6,10.7
2523 DATA  56.4,11.9,54.5,10,54,14.2,55,20,59,22                :REMark 150
2524 DATA  60,30,60.6,28,60,22,63,21,65.6,26
2525 DATA  66,22,61,17,60,19,56,16,55.4,13
2526 DATA  59,10.3,58,7.6,58.5,6,62.5,5.5                       :REMark 176
```

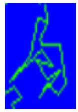


```
2527 DATA  5,Acol,62.5,5.5,64,10,70.3,19,71.2,27,67.8,41.5 :REMark Artic area

2528 DATA 10,Ccol        :REMark Corsica & Sardinia
2529 DATA  43,9.4,42.4,8.5,41.5,8.8,40.9,9.8,39.1,9.7
2530 DATA  38.9,8.4,40.8,8.4,41.3,9.2,42.1,9.6,43,9.4          :REMark  20

2531 DATA 11,Ccol        :REMark Balearic Isles
2532 DATA  40,3.1,39.9,3.1,39.8,3.2,39.9,3.3,39.8,3.5          :REMark  10
2533 DATA  39.3,3.1,39.4,2.9,39.6,2.8,39.5,2.7,39.4,2.6,40,3.1

2534 DATA  5,Ccol,39.1,1.7,39,1.8,38.9,1.6,39,1.5,39.1,1.7     :REMark  10

2535 DATA  6,Ccol        :REMark Cyprus
2536 DATA  35.5,32,35.6,33,35.9,34,35.2,33,35.2,32,35.6,32

2537 DATA  7,Ccol        :REMark Crete
2538 DATA  35.8,24,35.9,26,35.7,27.5,35.5,27.5,35.6,26,35.5,26,35.8,24
```

```
2539 DATA 61,Ccol       :REMark AFRICA
2540 DATA  28,35,28,33,15,40,10.5,45,12,51.4                :REMark  10
2541 DATA  4,47.7,-5,39,-16,41,-20,35,-25,35
2542 DATA  -26,33,-29,32,-34,26,-35,20,-18,12
2543 DATA  -11,14,-1,9,3,10,4.6,8.4,4.3,5.9
2544 DATA  6.5,4.3,4.8,-2,4.6,-7.7,7.8,-12.9,9.6,-13.4      :REMark  50
2545 DATA  12.4,-16.7,14.9,-17.6,17.2,-16.1,21.3,-17.2,28,-12.9
2546 DATA  30.3,-9.5,31,-9.8,32,-9.8,33.3,-8.3,33.9,-6.9
2547 DATA  35.8,-6,35.9,-5.4,35.2,-4.7,35,-2,36.4,1
2548 DATA  37.3,10.2,36.7,10.4,37,11,36.1,10.5,35.2,11.1
2549 DATA  34,10,32.8,12.5,32.94,13.2,32.4,15.3,31.5,15.6
2550 DATA  30,19,31,20,32,19.7,33,22,31,29
2551 DATA  31.6,31,31.2,33.5,37,36,37,28,40,26,41,29       :REMark 122

2552 DATA  4,Ccol,28.6,-16.1,28,-16.7,28.4,-17,28.6,-16.1   :REMark Canary Isles

2553 DATA  7,Ccol,29.5,-13.3,29,-13.3,28.8,-14,28,-14.5
2554 DATA       28.3,-13.8,29,-13.7,29.5,-13.4              :REMark  14

2555 DATA  6,Ccol,28.2,-15.6,28.2,-15.4,27.8,-15.3,27.6,-15.7
2556 DATA       27.9,-15.8,28.2,-15.6

2557 DATA  6,Ccol,-13,49,-17,44,-25,44,-25,47,-15,50.5,-13,49:REMark Madagascar

2558 DATA 13,Acol       :REMark ASIA
2559 DATA  66.5,39,67.2,33,64.5,35,64,40,68.2,44
2560 DATA  69,67,72,70,77,112,74,110,72,130
2561 DATA  70,175,67,190,66,177                             :REMark  26
2562 DATA 52,Ccol,66,177,63,180,60,170                      :REMark Leave Arctic
2563 DATA  60,163,55,162,51,157,57,156,62,163
2564 DATA  62,157,59,153,59,143,55,135,54,141
2565 DATA  48,140,39,128,35,129.5,34,126,39,125.5
2566 DATA  41,121,38.5,118,30,122,23,117,21,110
2567 DATA  22,108,19,105.5,14.5,109,11.5,109,8,105
2568 DATA  13,100.5,9,99,5,103.5,1,104,4,101
2569 DATA  9,98,17,97,23,92,15,80,10,80
2570 DATA  8,77,12,74.5,21,72,25,67,25,56
2571 DATA  30,50,29.5,49,24,53,25,56,24,56
2572 DATA  23,60,17,56,12.5,44,28,35                        :REMark 104

2573 DATA  7,Acol,77,70,76,60,71,50,70,51,75,60,76,70,77,70:REMark Novaja

2574 DATA  7,Ccol       :REMark Sri Lanka
2575 DATA  9.7,80,7,82,6.5,81.8,6.3,80.5,6.4,80,8,79.7,9.7,80
```

```
2576 DATA 74,Ccol      :REMark AMERICA
2577 DATA  52,-56,50,-65,46,-64,43.7,-70.4
2578 DATA  41.5,-70.7,40.6,-74,37,-76,35.2,-75.7,31,-81.6
2579 DATA  27,-80,25,-80.5,28,-82.7,29,-82.5,30,-84
2580 DATA  30.3,-89,29,-90,29.7,-94,27,-97.5,22,-97.7
2581 DATA  19,-96,18.4,-94,19,-91,21,-90,21.6,-87            :REMark  50
2582 DATA  16,-89,15.6,-83,10.5,-83.5,9,-81.5,9.7,-79
2583 DATA  8,-77,11,-75,12,-71,10.6,-63,4,-52
2584 DATA  0,-50,-6,-34,-12,-39,-22,-41,-25,-48
2585 DATA  -28,-48,-41,-63,-51,-69,-55,-65,-55,-70
2586 DATA  -50,-76,-37,-74,-18,-70,-6,-81,0,-81              :REMark 100
2587 DATA  6.6,-77.5,9,-79,7,-81,9.5,-85,13,-88
2588 DATA  14,-91.5,16.2,-95,15.7,-96.6,19.6,-106,22,-105.7
2589 DATA  29,-112.4,31.3,-113,31.6,-115,30,-114.6,23,-109.5
2590 DATA  25,-112.3,30,-115.9,34,-118.5,34.5,-120.7,39,-124
2591 DATA  43,-124.5,48.5,-124.5,59,-138,61,-148,54,-165  :REMark 148

2592 DATA 11,Acol,54,-165,59,-158,62,-166,68,-167,71,-157 :REMark Arctic
2593 DATA       68,-110,70,-82,60,-95,54,-80,63,-77,52,-56

2594 DATA  5,Acol,75,-105,73,-90,70,-105,73,-120,75,-105   :REMark Victoria

2595 DATA  5,Acol,83,-45,81,-120,78,-105,81,-75,83,-45     :REMark Elizabeth

2596 DATA  6,Acol,78,-75,67,-60,60,-60,64,-75,75,-90,78,-75 :REMark Baffin

2597 DATA 12,Acol,60,-44,65,-40,70,-22,82                   :REMark Greenland
2598 DATA       -15,83.6,-30,78.5,-73,76,-68,75.6,-59,70
2599 DATA       -51,66,-53.5,61,-48,60,-44

2600 DATA 15,Acol      :REMark Arctic Ice sheet
2601 DATA 77,-114,73,-124,74,-132,76,-130,79,-160
2602 DATA 76,-170,74,176,78,160,83,140,81,110
2603 DATA 82,70,84,30,82,10,76,-10,74,-18                   :REMark  30

2604 DATA 18,Ccol      :REMark Caribean
2605 DATA 22,-85,23,-83,23,-80.5,20,-74,20,-70
2606 DATA 18.5,-68,18.5,-71,17.5,-71.5,18,-72,18.5,-74.5
2607 DATA 19,-74.5,19,-72.5,20,-74,20,-77.5,20.5,-77
2608 DATA 22.5,-81.5,22,-84,22,-85                          :REMark  36

2609 DATA  5,Ccol,18.2,-78.2,18.4,-78,18,-76.2,17.9,-77.8,18.2,-78.2

2610 DATA  5,Ccol,18.5,-67,18.5,-65.5,18,-65,18,-67,18.5,-67

2611 DATA 22,Ccol      :REMark Japan
2612 DATA 45.5,141.8,43.3,145.7,42,143,42.6,141.6,40.6,140
2613 DATA 38.2,139.6,37,136.9,35.6,135.7,35.6,133,34,130.9
2614 DATA 32.9,132,31.4,131.3,31.2,130.2,33.3,129.7,34,130.9
2615 DATA 34.5,135,33.5,135.7,36,140.6,39.8,142,42.5,139.7
2616 DATA 43.5,141.4,45.5,141.8                             :REMark  44

2617 DATA  5,Ccol      :REMark Taiwan
2618 DATA 25.5,121.5,23.5,120,22,121,25,122,25.5,121.5     :REMark  10

2619 DATA  6,Ccol      :REMark Hainan
2620 DATA 20,108.6,20,110.3,19.8,110.3,18.3,109.9,18.8,108,20,108.6
```



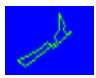


```
2621 DATA 19,Ccol       :REMark Philippines
2622 DATA 21,122,18,122.5,16.5,122.5,15,121.5,14,122
2623 DATA 13.5,125,7,126,5,125,7,123,5,122
2624 DATA 9,125,8,123,11,121,10,124,13,122
2625 DATA 8,117,12,120,18.5,121,18,122.5                :REMark 38
```

```
2626 DATA 11,Ccol       :REMark Indonesia
2627 DATA 6,95,1.7,98.8,-3.2,101.6,-5.9,105.7,-6.6,114.2,-8.6,127
2628 DATA -7.1,105.6,-2.9,105.9,.4,103.6,5,97.5,6,95    :REMark 22

2629 DATA 4,Ccol,2,128,1.5,129,-1,128,2,128

2630 DATA 6,Ccol,-3,126,-4,131,-3,130.5,-3,128,-4,126.5,-3,126
```

```
2631 DATA 13,Ccol       :REMark Borneo
2632 DATA 7,117.5,2.5,111,1.5,111,2,109.5,1,109
2633 DATA -3,110,-4,114.5,-4,116,1,117.5,1,119
2634 DATA 4,117.5,5,119,7,117.5                          :REMark 26

2635 DATA 17,Ccol       :REMark
2636 DATA 1,125,1,124,1.5,121,0,119.5,-3,118.5
2637 DATA -6,119,-6,120.5,-3,120.5,-5.5,122,-5.5,123
2638 DATA -4,123,-2,121.5,-.5,123.5,-1,121,.5,120.5
2639 DATA .5,124.5,2,125                                 :REMark 34

2640 DATA 12,Ccol       :REMark
2641 DATA 0,130,-2,134,-2.5,141,-6.5,148,-6.8,146.8
2642 DATA -10.7,151,-7.7,144.3,-9.3,143,-8,138.4,-5.4,138.1
2643 DATA -4,133.1,0,130                                 :REMark 24
```

```
2644 DATA 34,Ccol       :REMark Australia
2645 DATA -10.5,142.4,-17.5,141,-15,135.5,-12,137,-11,132
2646 DATA -15,129,-14,127,-17.5,122,-19,122,-20,120
2647 DATA -22,114,-26,113,-32,116,-34.5,115,-35.2,118
2648 DATA -31.5,130,-32.5,133.5,-35,135.5,-33,137.8,-35.2,137.5
2649 DATA -38,140.4,-39,143.4,-37.8,145,-39.2,146,-37.5,150
2650 DATA -34,151,-32.7,152.7,-29,153.6,-25.6,153,-20,148.4
2651 DATA -18.8,146.3,-14.5,144.7,-14.7,144,-10.5,142.4  :REMark 68

2652 DATA 4,Ccol,-42,144.9,-42,148,-44,146.5,-42,144.9   :REMark Tasmaina
```



```
2653 DATA 14,Ccol       :REMark New Zeeland
2654 DATA -34.5,172.7,-36.7,175.9,-37.5,176,-38,177.3,-37.4
2655 DATA 178.5,-41.6,175.5,-40.6,172.5,-42.8,171,-46,166.2,-46.7
2656 DATA 169.4,-40.2,175.3,-39.3,174,-37.7,174.8,-34.5,172.7
```

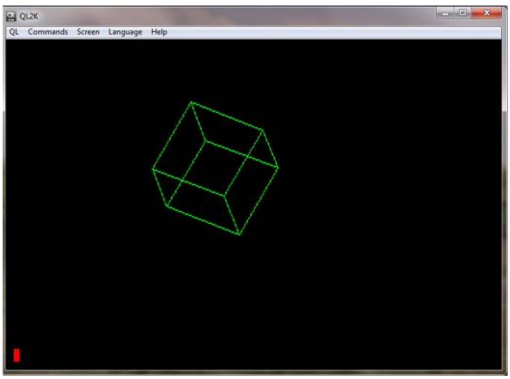
```
2657 DATA 29,Acol       :REMark Antarctica
2658 DATA -63,-56,-64,-60,-66,-65,-73,-75,-73,-85
2659 DATA -73,-100,-75,-100,-73,-125,-75,-137,-78,-165
2660 DATA -77.6,164,-72,170,-68,155,-66,135,-66,115
2661 DATA -66,90,-69.5,75,-68,70,-66,55,-69,40
2662 DATA -70,20,-70,0,-71,-10,-74,-20,-78,-35
2663 DATA -75,-60,-67,-61,-64.3,-69,-63,-55              :REMark 58
2664 DATA 9999                                           :REMark End check
```

QBITS 3DGraphics Procedures

Init_win	
Init_QB3D ``	Sets screen layout and KEY information. `
Menu_3DCommands	Menu loop to access key commands
Obj_Pos	Update and display Rotation and Motion variables
Obj_Ang	Update Rotation Graphics for Roll, Loop & Spin
Obj_Auto	Sets Auto Roll, Loop/ Spin of Object
Obj_Node	Loads Node xyz of Object
Obj_Calc	Calculates new vx vy coordinates of Object
Obj_Draw	Draws Object to screen
Obj_Cull	Identifies hidden frames
Pod_Rescue	Menu for Pod Rescue Simulation
Pod_Draw	Draws Pod and Shuttle for simulation
Get_keys	Keys set Rotation and Movement of Shuttle
Beeps(b)	Sounds for Shuttle Jets and Docking or Alarm for failure
Globe3D	Initiates changes to screen layout set revolving Globe.
GTitle	Menu Writes Str$ to screen
Wold	Draws World Circle
Continents	Menu to Select and display Continent
Viewer	Sets Map display to GMT
Grid	Draws Longitude & latitude Grid
Maps	Reads DATA and sets angle and position
Calc_ang	calculates start angle
Calc_posn	Calculates and draws map lines
Obj_Name	Displays Object Names and Action tn Screen
Obj_Shape	Sets DATA RESTORE references etc for Object.
DATA Lines	Basic Shapes, Shuttle, Rescue Pod, World Map

The basic Code for Rotation

```
100 REMark 3D_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=.1                    :REMark Vector length : Focal Point: Rotation angle

112 CLS
114 x(1)=-vl:y(1)=-vl:z(1)=-vl                  :REMark Nodes
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+.1:c=COS(ra):s=SIN(ra)

136 FOR np=1 TO 8
138   REMark Rotation on X Axis
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 to Draw 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
```

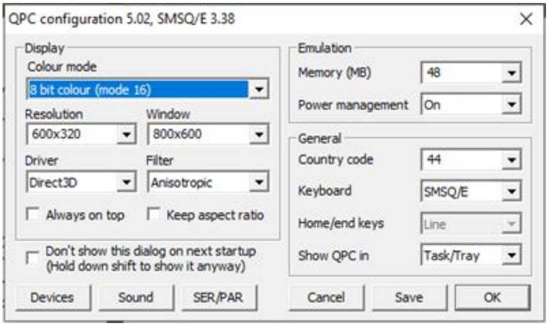


QBITS Progs Development

Most of the QBITS Progs were first envisaged back in the nineteen-eighties, a few were written and released between 1987 and 1992. Those released did undergo updates, but it wasn't until the 2002's when I downloaded a copy of QL2K emulator that I renewed my interest in QL SuperBASIC.

The QBITS Programs exploring 3D Graphics have been developed using various QL Platforms, the latest being the QPC2 v5 Emulator. The QBITS Progs form a group selected from Menu file QBITSProgs where on LRUN they import common variables from a QBITSConfig file. QBITSProgs are by default loaded from QPC2 Device Dos1_.

QPC2 Setting



Background Notes

My aspiration was an efficient code for Rotation of a Wireframe object with Motion to move about the screen altering its global x y position and being able to zoom in and out. Then add Perspective which has something to do focal scale. As a finishing touch FILL the visible surfaces so as the Wireframe Rotates it creates the illusion of a solid object.

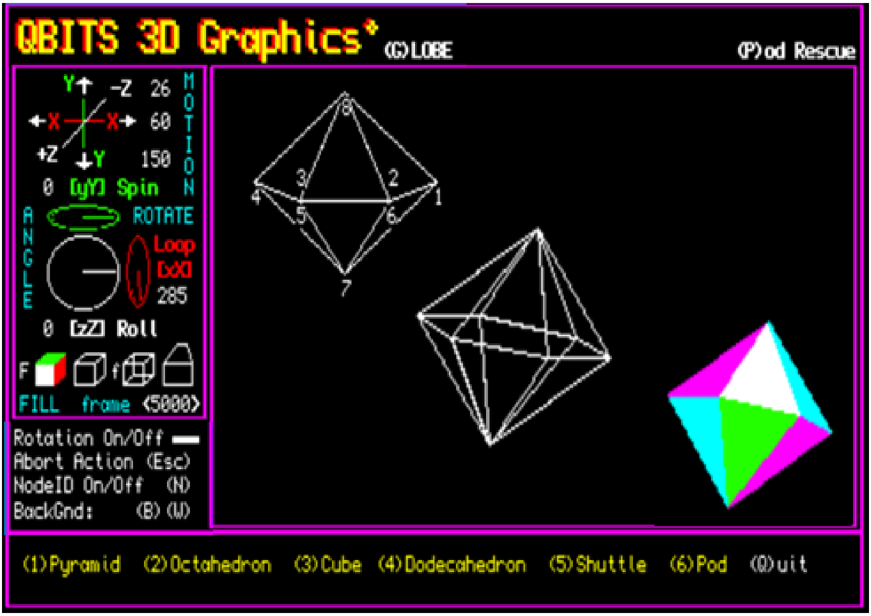
Early attempt to code for 3D Graphics began with QB3D_Wire512 which only displayed the Wireframe of a Cube. Further development added controls to alter the screen position horizontally and vertically. Later came the initial trial of **xyz** rotation of the Wireframe. Commands were added to allow manipulation of parameters controlling various aspects of movement, Size and Perspective. The next venture was to create Node and Frame Data to configure four Objects a Pyramid, Cube, Hexagon and a simple Space Shuttle.

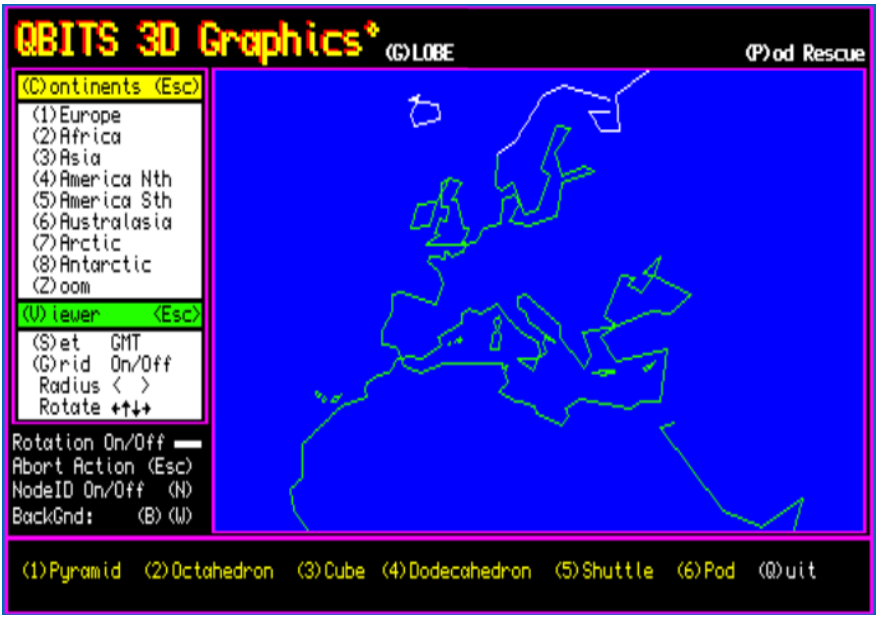
While deciding on useful things for the program it occurred to me that a user might prefer a White to a Black background. The Program includes pressing **B** or **W** to change the colour of PAPER (bg1) and INK. (bg2) either a Black background with White INK, or White background with Black INK.

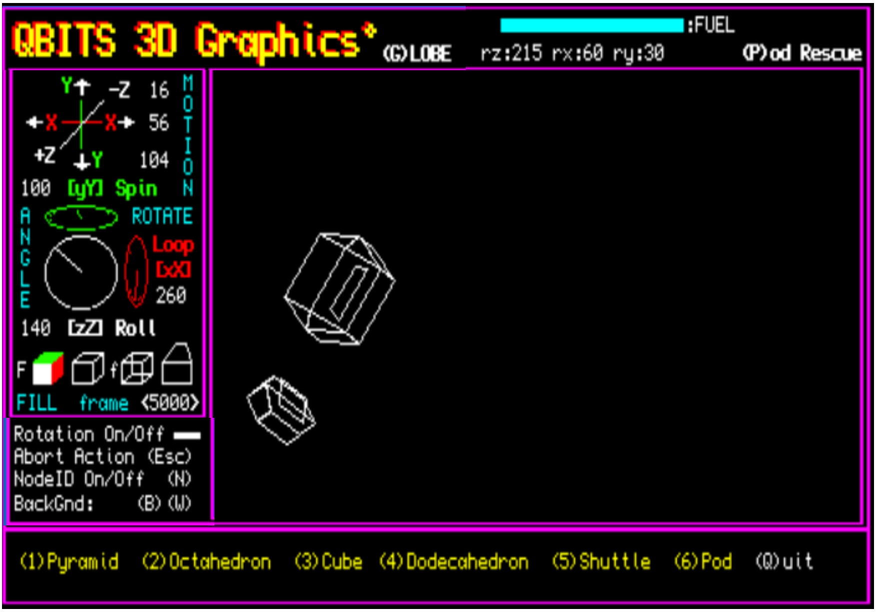
Future Challenges:

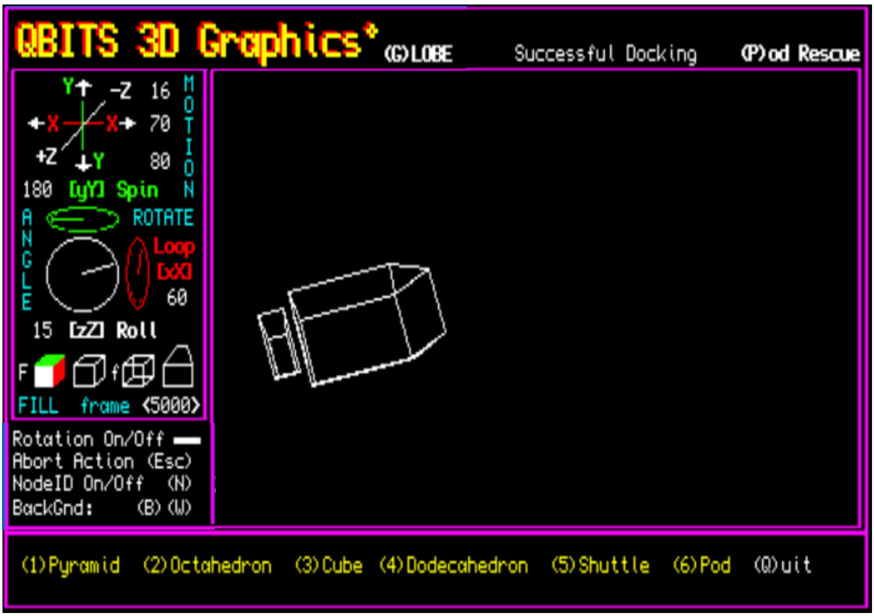
To restructure the WorldMap so there are no recursive shapes and FILL used to colour continents. Possibly enhance the Map with more land detail i.e mountain ranges etc.

Write a Companion Program to construct Wireframe Objects and generate their Node **xyz** coordinates and Frame DATA lists would be useful. [03 2023]


QBITS 3D Graphics*
(G)LOBE
(P)od Rescue
MOTION
ANGLE
0 [yY] Spin
ROTATE
Loop
[xX]
285
0 [zZ] Roll
FILL frame <5000>
Rotation On/Off
Abort Action (Esc)
NodeID On/Off (N)
BackGnd: (B) (W)
(1)Pyramid (2)Octahedron (3)Cube (4)Dodecahedron (5)Shuttle (6)Pod (Q)uit


QBITS 3D Graphics*
(G)LOBE
(P)od Rescue
(C)ontinents (Esc)
(1)Europe
(2)Africa
(3)Asia
(4)America Nth
(5)America Sth
(6)Australasia
(7)Arctic
(8)Antarctic
(Z)oom
(V)iewer (Esc)
(S)et GMT
(G)rid On/Off
Radius < >
Rotate
Rotation On/Off
Abort Action (Esc)
NodeID On/Off (N)
BackGnd: (B) (W)
(1)Pyramid (2)Octahedron (3)Cube (4)Dodecahedron (5)Shuttle (6)Pod (Q)uit


QBITS 3D Graphics*
(G)LOBE
:FUEL
rz:215 rx:60 ry:30
(P)od Rescue
Y -Z 16
X X 56
+Z Y 104
MOTION
100 [yY] Spin
ANGLE
ROTATE
Loop
[xX]
260
140 [zZ] Roll
F f
FILL frame <5000>
Rotation On/Off
Abort Action (Esc)
NodeID On/Off (N)
BackGnd: (B) (W)
(1)Pyramid (2)Octahedron (3)Cube (4)Dodecahedron (5)Shuttle (6)Pod (Q)uit


QBITS 3D Graphics*
(G)LOBE
Successful Docking
(P)od Rescue
Y -Z 16
X X 70
+Z Y 80
MOTION
180 [yY] Spin
ANGLE
ROTATE
Loop
[xX]
60
15 [zZ] Roll
F f
FILL frame <5000>
Rotation On/Off
Abort Action (Esc)
NodeID On/Off (N)
BackGnd: (B) (W)
(1)Pyramid (2)Octahedron (3)Cube (4)Dodecahedron (5)Shuttle (6)Pod (Q)uit