



Introduction

My growing interest with Computer Graphics back in the 1980's was in the manipulation of 3D images. Early Coding in QL SuperBASIC began with a Rotating Cube. However, the QLs Interpreter Performance did not inspire me. As for coding in machine code or assembly it was a little beyond my accomplishments at the time. However, I did jot down some notes for future reference.

QBITS 3DGraphics -Rotation

Returning to SuperBASIC with a range of new QL Platforms with improved performance I was again spurred on to exploring the possibilities of 3D Graphics. The 2023 **Special Edition** led to a number of improvements to earlier versions. Firstly, the presentation of the control keys and 3D Graphic display with Black or White backgrounds. The Polygon Wireframes viewed in full or as solid objects with or without coloured Frame surfaces. Added was a Revolving **Globe** and **POD Rescue**.

In this **Extended Edition** the **Globe Continents** and **Viewer** menus are combined giving more flexibility. (S)et to GMT remains with (G)rid On/Off Longitude/Latitude lines as does Rotation of Globe with cursor keys $\leftarrow \uparrow \downarrow \rightarrow$ in any direction. The Zoom option is now covered by resizing with +/- Radius.

POD Rescue was seen as a fun way to explore working with Multiple Objects. This is further explored with **ZZ** Plane values. Two methods are reviewed, the **Dependant** version works with objects linked to a common **xyz** position. The Shuttle and POD use this method and looking down on the **ZZ** Plane the **y** Rotation Angle can be used to determine in which order to Draw the Shuttle and Pod.

The **Independent** Method requires Objects along the **ZZ** Axis being Drawn in Sequence, the most distant (smallest) first then progress to the nearest (largest) as seen from the Viewpoint. If any Object changes its **ZZ** Plane status, then the Sequencing will need to be updated before the next Objects screen Draw...

QBITS 3DGraphics – Extended Edition

Images convey more than words they say, Exploring Three-Dimensional Images Motion and Angle of Rotation are represented more Graphically in a side Panel.



MOTION For the **Z** Axis ‘-’ reduces object size ‘+’ enlarges, brings object nearer as seen from Viewpoint. For **X** use Left $\leftarrow \rightarrow$ Right for **Y** use Up $\uparrow \downarrow$ Down Cursor keys. The **Z X Y (n)** values are shown in vertical column right of Graphics representation.

ANGLE - ROTATE The [yY]Spin about the YY axis is shown as horizontal ellipse above a Circle depicting the [zZ]Roll about the ZZ Axis. Displayed to the right is a vertical ellipse showing the [xX]Loop about the XX axis. The values (n) update as degrees of Rotation change.

Upper case ‘F’ Fills the viewable Frames with Ink Colour. Lower case ‘f’ Toggles Wireframe between Full/Solid. Use Left/Right ‘<’5000>’ Chevron keys to Decrease Increase the Focal Scale range from 100 to 5000.

Multiple Objects - see next page for details

[#] Speed AUTO Toggle On/Off with Spacebar

Node ID's Toggle On/Off with ‘N’

(R)eset BackGnd: ‘B’ Black or ‘W’ White

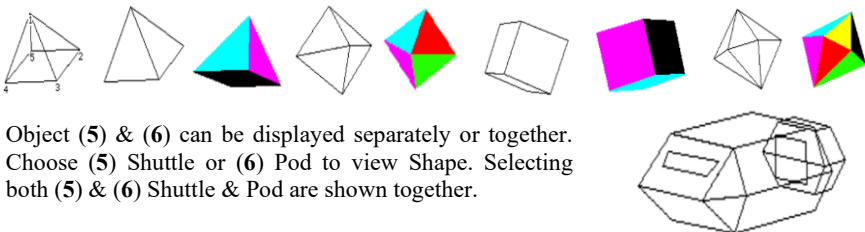
Object Shapes Select [1] [2] [3] [4] [5] [6]

Select ‘Q’ to (Q)uit



QBITS 3DGraphics – Wireframe Objects

Upon Selecting one of the Objects (1)(2)(3)(4)(5)(6) the Program first READs and sets up the **Node xyz** coordinates and then **Frame** sequences. The 2D Conversion of Vectors are then calculated to display the Wireframe to screen. You can check the (N)ode ID's and change Object's Orientation with changes to + Z - Visual Scale $\leftarrow X \rightarrow \uparrow Y \downarrow$ Screen position, and with zZxXyY the Angles of Rotation. ‘F’ FILL shows the changing coloured Frames as Object is Rotated.



Object (5) & (6) can be displayed separately or together. Choose (5) Shuttle or (6) Pod to view Shape. Selecting both (5) & (6) Shuttle & Pod are shown together.



QBITS 3DGraphics - Multiple Objects

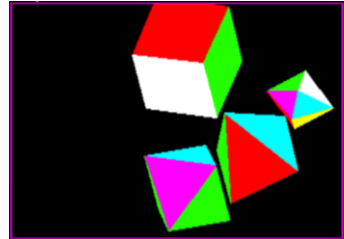
Object Shapes 1234 can be used in different combinations or all can be displayed together on Screen. Multiple Objects are Drawn in Sequence the default on Start Up or following a (R)eset is [1234]. These can be manually swapped with CTRL 1234.



F5 Toggles Lock/Unlock of Object to screen
[1] Shows current Object selected

QBITS 3DGraphics - Draw Sequence

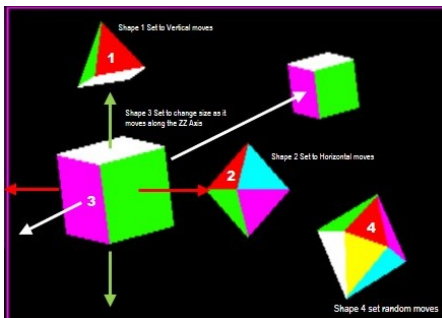
To manually change the Draw Sequence,  Select an Object 



QBITS 3DGraphics - AUTO

Press Spacebar to toggle On/Off AUTO where Screen Objects displayed are set in motion. Press [#] key to change speed of 'aset' [auto set : # increments 2...9 then back to 2].

While in AUTO Select an individual Object [1234] then use + ZZ - :< XX> : ↑ YY ↓ for MOTION and zZ xX yY to change r ANGLE of ROTATION.



For Example:

Objects [124] are set to 0 values on ZZ [1] is set for Vertical [2] for Horizontal moves with [4] left as Random.

Object [3] is Static on XX YY Axis but moves Backwards and Forwards along ZZ Axis, where size changes illude to a far or near distance from Viewpoint.



QBITS 3DGraphics - Dynamics

When Multiple Objects generate variable ZZ Axis values, how are Objects to be drawn in front or behind as they pass each other. The Draw Sequence can be set manually but is only OK for Static arrangements. In an AUTO environment a more dynamic arrangement is required to re-sequence after any Objects ZZ Axis value changes.

QBITS 3DGraphics - Object Sort

If AUTO is On Program checks for any ZZ value changes. If true the Objects [1234] 'vs' values are passed to a Simple Sort code Algorithm to determine the new sequence. This is then used in the next loop of Objects Drawn to screen.

To End Program Press (Q)uit, which if set up takes user back to QBITSProgs Menu.

QBITS 3DGraphics - Pod Rescue



Shuttle – Directional Jests

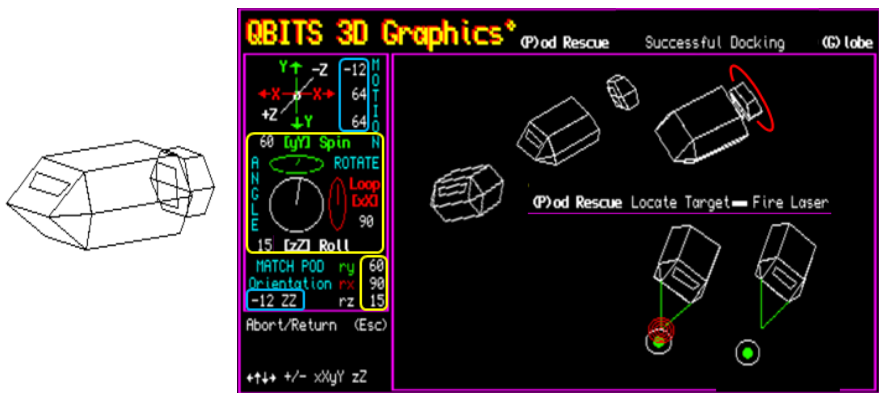
To start you have to bring the Shuttle under control as it Rotates and Moves randomly across the screen Use the Shuttles directional Jets [**Cursor keys**]. However, too long a burst can lead to uncontrollable Motion and Rotation and quickly deplete the fuel supply.

Targets - Space Debris

Once you have control of the Shuttle you need to locate and use its Lasers to Destroy five randomly positioned pieces of Space debris.

Shuttle Pod - Alignment

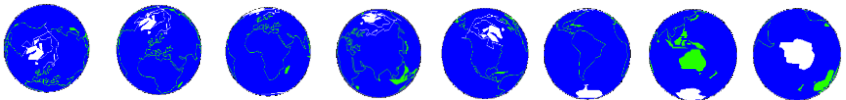
To Dock with the Rescue Pod the Shuttle must MATCH Pod Orientation on ZXY Axis and the Angles of Rotation.



To Abort/Return Press (**Esc**) key.

QBITS 3DGraphicsEE - Globe

Press 'G' and main window changes to show a World Map of Planet Earth. The Side Panel clears then scrolls down to reveal a Continents Menu from which you Select different Map areas or Options to change aspects of display.

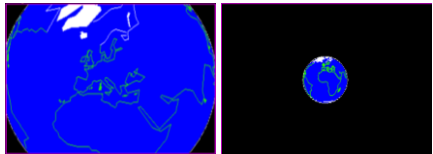


Continents

Select an area by pressing (1) to (8).

Use +/- Radius to Resize,

Enlarge or Reduce the display.



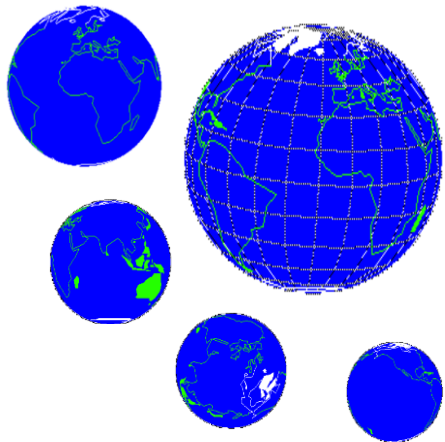
Greenwich Mean Time

Pressing 'S' returns Map to the Prime Meridian at Greenwich [GMT].

Longitude/Latitude Lines

Pressing 'G' toggles Grid On/Off.

Rotate Globe in different directions, with
Left ← → Right Up ↑ ↓ Down Cursor keys.



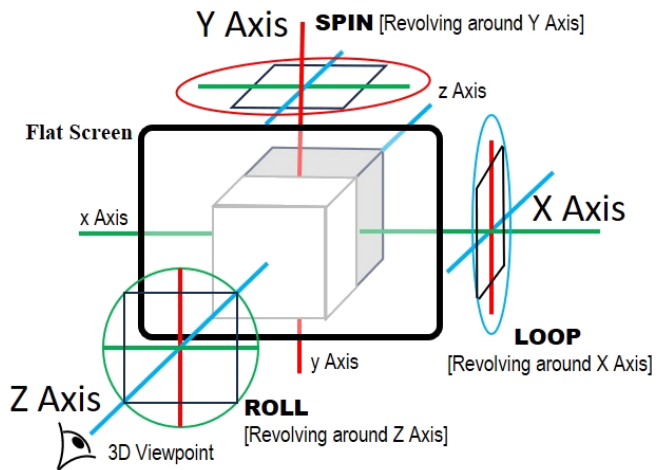
To Abort/Return Press (Esc) key.

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 **ZZ:XX:YY** 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 **ROLL zZ LOOP xX SPIN yY** It is no coincidence they all happen to be four letter words, a little conformity not a bad thing when computer coding.



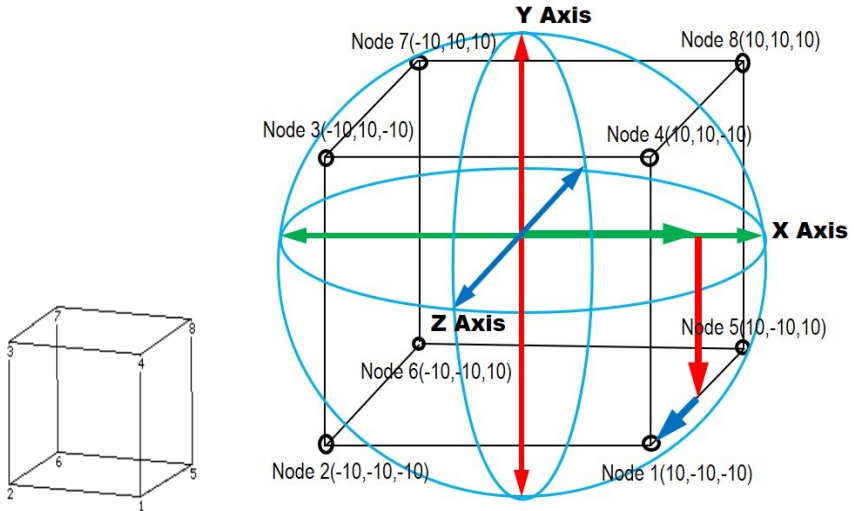
Imaginary Eye View

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 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 central **Z X Y** position. Following the Arrows [see below] **Node (1)** is shown on the **X** axis as **+x** units from $[x=0]$. On the **Y** axis it drops below $[y=0]$ 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 it can be given a value of **-z**.



Node (1) xyz
Node (2) xyz
Node (3) xyz
Node (4) xyz
Node (5) xyz
Node (6) xyz
Node (7) xyz
Node (8) xyz

DATA 8
DATA 10,-10,-10
DATA -10,-10,-10
DATA -10,-10, 10
DATA 10,-10, 10
DATA 10, 10,-10
DATA -10, 10,-10
DATA -10, 10, 10
DATA 10, 10, 10

:REMark Number of Nodes

The **Node coordinates** can now be written as a set of DATA lines, which can be used as the basic configuration information. This will apply to not only 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 our Viewpoint. If we **ROLL** the cube on its **ZZ Axis** then we see a square surface turning through 360 degrees. If we **LOOP** around the **XX Axis** then the surface presented changes to show two changing rectangular surfaces before returning to a square. A similar view is presented looking down when we **SPIN** the cube around the **YY Axis**. Turning the Cube through **ZZ XX YY Axis** the number of surfaces and their shapes change again.

Vector Calculations

Vector Coordinates are used to represent a **3D Object** on a **2D Screen**. These are the calculated **x y** two-dimensional screen positions derived from central **x y** position of the Object and correlates to each of the individual **Node z x y** 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**).

$$x = r \times \text{COS}(a)$$

$$y = r \times \text{SIN}(a)$$

If we then rotate further the angle to b:

$$x' = r \times \text{COS}(a + b)$$

$$y' = r \times \text{SIN}(a + b)$$

By using trigonometric addition of each equation:

$$x' = r \times \text{COS}(a) \text{COS}(b) - r \times \text{SIN}(a) \text{SIN}(b)$$

$$y' = r \times \text{SIN}(a) \text{COS}(b) + r \times \text{COS}(a) \text{SIN}(b)$$

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:

$$x' = x \times \text{COS}(b) - y \times \text{SIN}(b)$$

$$y' = y \times \text{COS}(b) + x \times \text{SIN}(b)$$

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:

$$ra = +.5 : c = \text{COS}(ra) : s = \text{SIN}(ra)$$

Then the code for position in each plane is as follows:

$$yt = y : y = c_x yt - s_x z : z = s_x yt + c_x z \quad \text{X axis (y, z planes)}$$

$$xt = x : x = c_x xt + s_x z : z = s_x xt + c_x z \quad \text{Y axis (x, z planes)}$$

$$xt = x : x = c_x xt - s_x y : y = s_x xt + c_x y \quad \text{Z axis (x, y planes)}$$

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 window **wx** horizontal and **wy** vertical positioning and the objects distance as viewed from the Vviewpoint. This is conveyed by reducing the Object **vs** vector size as it moves away and increasing to making it appear bigger as it moves towards the view point.

QBITS 3D Movement & Conversion

Movement is the Third step 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 central repositioning of the Object the **Cursor Left Right Up Down** keys are used to move to new **wx wy** window 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 decrease an Objects size in a uniform manner. The vector size '**vs**' is simply that with a range 0.4 to 2.4 in 0.05 increments 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 100 and 5000. 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, **Central 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:

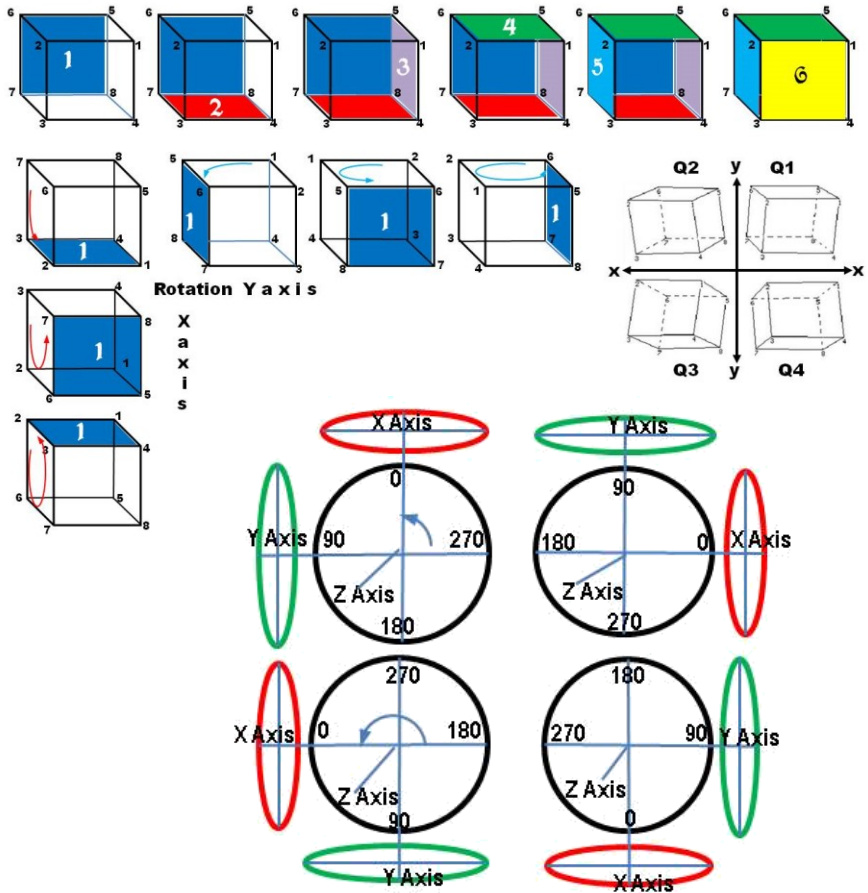
IF nset=2 : FOR n=sn TO mn : CURSOR vx(n),vy(n),-2,2 : PRINTn

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

Note: When using the **zZxXyY** keys to **ROLL/LOOP/SPIN** 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 having been changed to Rotate in altered planes. An example of this is where the actions of **xx** (Loop) and **yy** (Spin) or **xx** and **zz** (Roll) act in reverse to each other's original action.

QBITS Notes on ZXY Rotation

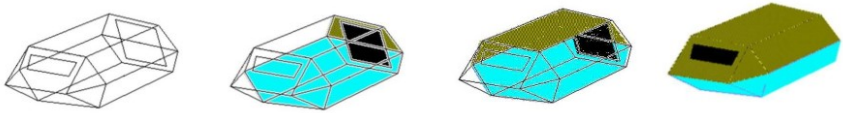
The **Frame sequence** as hinted before loads those **Frames hidden** from view first with the ones covering the viewed surfaces last. The problem is as an Object is rotated away from initial settings in any of its three axes then the sequence of Frames hidden from view and those that come into view 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 **zZxXyY** keys. In the example shown Rotation is around the **Z** axis. The actions of **LOOP** and **SPIN** change as it moves through each quadrant. Hopefully my diagrams above explain this better than I can put it 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 **Viewpoint**.

The two basic types of hidden surface removal are Object-space for Three-Dimensional processing and Image-space used in Two-Dimensional processing. This uses an algorithm that identifies those Frame surfaces of an object that are not seen from the view point. The most common method used 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 $C \leq 0$ is a surface facing towards the viewer.

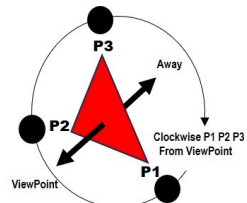
Obtaining the Vector Normal we use an equation based on the plane passing through three points: $P1=(x1, y1, z1)$, $P2=(x2, y2, z2)$, $P3=(x3, y3, z3)$:

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

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

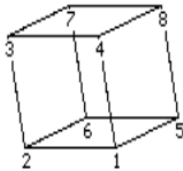
where $C = (x2 - x1)*(y3 - y1) - (x3 - x1)*(y2 - y1)$

C is the value we are interested in to determine the outward facing direction of the **Frame surface** and whether it is **Away** or **Towards** the **Viewpoint**.

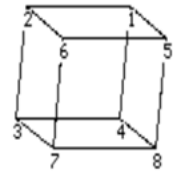


QBITS Anti Clockwise Method

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 is 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
 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
 [bg2 = Frame surface Colour]



QBITS 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** construction 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,ic : IF cset=1:INK bg2:FILL 0:END IF
  IF cset=2:Obj_Cull:IF c1>0:GOTO A:END IF :INK bg2:FILL 0:END IF
  IF cset=3:Obj_Cull:IF c1>0:GOTO A:END IF :INK ic :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
GOTO - A
END FOR lp
IF nset=2:FOR n=sn TO mn:CORSOR vx(n),vy(n),-2,2:PRINT n
END DEFine
```

QBITS Obj_Cull

To calculate the Vector Normal of a Frame's 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

For **cull set** where **cset=1** the Wireframe outlines of all the **Frames** of an Object are shown. For **cset=2** the procedure **Obj_Cull** is used to eliminate **hidden Frames** and display a Solid Object. For **cset=3** **FILL** is used to colour a **Frame** surface, which is stored as the Fifth value entered on a **Frame** DATA Line (see Cube DATA table above).

Note: 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 3DGraphics - Multiple Objects

Use of **Nodes**, **Vectors**, **Frames** and **cull** of non-viewed **Frames** allows the impression of a 3D Object in a two-dimensional flat screen. To display Multiple Objects requires further manipulation so as to hide objects or parts of objects as they cross each other's paths. In the methods previously mentioned both require access to individual Object Data to process their screen displays. For the Dependant method as in Pod Rescue, both Objects revolve around a common Axis so only **Node** and **Frame** Data pointers etc. are needed and this information is held by **Obj_Dat** [set as part of Program Startup].

```
1100 DEFine PROCedure Obj_Dat(n)
```

Note: Infor for Objects 1 to 4 is also held by **Obj_Dat**

```
1105 IF n=5 :nres=2077:sn= 1:mn=22 :vres=2119:vo=16
```

```
1106 IF n=6 :nres=2101:sn=23:mn=38:vres=2137:vo=11
```

```
1107 END DEFine
```

Obj_Get and **Obj_Put** handle variables for Screen Positioning, Angles of Rotation etc.

```
1109 DEFine PROCedure Obj_Put(n)
```

Note: Write Object Data

```
1110 Obj(n,4)=nres:Obj(n,5)=sn:Obj(n,6)=mn:Obj(n,7)=vres:Obj(n,8)=vo
```

```
1111 Obj(n,9)=rx:Obj(n,10)=ry:Obj(n,11)=rz:Obj(n,12)=px:Obj(n,13)=py:Obj(n,14)=vs
```

```
1112 Obj(n,15)=xx:Obj(n,16)=yy:Obj(n,17)=zz:Obj(n,18)=ax:Obj(n,19)=ay:Obj(n,20)=az
```

```
1113 END DEFine
```

```
1115 DEFine PROCedure Obj_Get(n)
```

Note: Read Object Data

```
1116 nres=Obj(n,4):sn=Obj(n,5):mn=Obj(n,6):vres=Obj(n,7):vo=Obj(n,8)
```

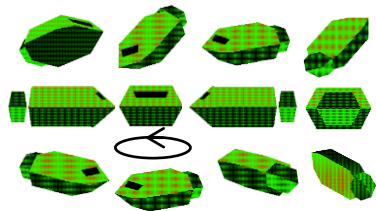
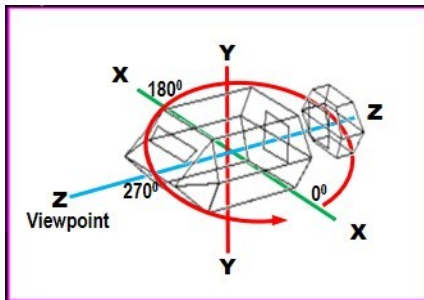
```
1117 rx=Obj(n,9):ry=Obj(n,10):rz=Obj(n,11):px=Obj(n,12):py=Obj(n,13):vs=Obj(n,14)
```

```
1118 xx=Obj(n,15):yy=Obj(n,16):zz=Obj(n,17):ax=Obj(n,18):ay=Obj(n,19):Obj(n,20)=az
```

```
1119 END DEFine
```

QBITS 3DGraphics - Dependant Method

When both Objects hold the same central **zxy** coordinates as in **Pod Rescue**, the Pod lying behind the Shuttle occupies **Y Angle** between **0°** and **180°** and the **Shuttle** is Drawn last. If the Pod is in front ie. **Angle >180°** - the **Pod** is Drawn last.



As the Shuttle Rotates the Pod is either seen or obscured from Viewpoint. This depends on the **Y Angle** looking down on the **ZZ** Plane.

```
1224 Obj_Get n
```

```
1225 IF k1 AND k2 Note: If both K1 and K2 are true then both Shuttle and Pod are Displayed.
```

```
1226 IF ry>180:Obj_Dat 5:Obj_Draw:Obj_Dat 6:ELSE Obj_Dat 6:Obj_Draw:Obj_Dat 5
```

```
1227 END IF
```

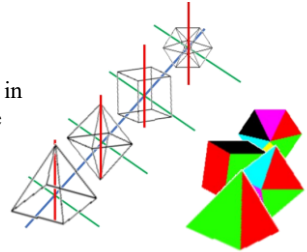
```
1228 IF n>4:Obj_Draw
```

QBITS 3DGraphics - Independent Method

The **ZZ** Plane extends front and back of a two-dimensional flat screen, position along this can be represented by increasing or decreasing the size of a displayed Object. Visual Scale '**vs**' has been used to Up/Down size an Objects screen image and thereby can illude to a position along the **ZZ** Plane.

QBITS 3DGraphics - Manual Sequencing

When displaying Multiple Objects the Shapes are drawn in Sequence starting with lowest '**vs**' value. If the sequence is manually constructed this remains static to the overlapping of objects as they cross each other's paths.



QBITS 3DGraphics - Dynamic Sequencing

If an object is constantly changing its **ZZ** position **vs** value with respect to others, then a more Dynamic Sequencing method is required especially in AUTO mode [**aset**<-1].

```
1256 DEFINE PROCEDURE Obj_Group
```

```
1257 FOR i=1 TO 4
```

```
1258 IF Obj(vp(i),1)=1 OR vp(i)=n
```

```
1259 Obj_Get vp(i)
```

```
1260 IF aset<-1 Note: AUTO On
```

```
1261 px=px+xx:py=py+yy:ovs=vs:vs=vs+zz:
```

```
1262 rx=rx+ax:ry=ry+ay:rz=rz+az:
```

```
1263 Obj_Space Note: Check on Boundaries
```

```
1264 IF px<=-120 OR px>=120:xx=-xx
```

```
1265 IF py<=-90 OR py>=90:yy=-yy
```

```
1266 IF vs<-.4 OR vs>=2.4:zz=-zz
```

```
1267 END IF
```

```
1268 Obj_Put vp(i):Obj_Get vp(i):Obj_Draw
```

```
1269 END IF
```

```
1270 END FOR i
```

```
1271 IF vck=1:Obj_Sort:vck=0
```

```
1272 END DEFINE
```

Note: Objects Sequence **vp(1 to 4)** [view point]

Checks Object Lock On or if Current Object

IF ovs<>vs:**vck**=1 Check for **ZZ** Axis changes

Note: AUTO Updates & Checks

Note: Save Update :Draw Object to Screen

Note: **ZZ** Axis Changes [visual check]

The Sort code compares '**vs**' values held by **Obj(n,14)** and sets sequence order in **vp(n)**

```
1274 DEFINE PROCEDURE Obj_Sort
```

```
1275 FOR v=1 TO 4
```

```
1276 FOR j=1 TO 4-v+1
```

```
1277 IF Obj(vp(j),14) > Obj(vp(j+1),14)
```

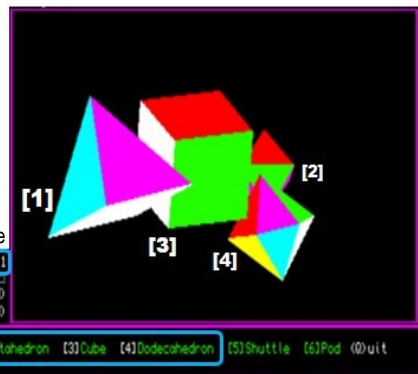
```
1278 temp=vp(j):vp(j)=vp(j+1):vp(j+1)=temp
```

```
1279 END IF
```

```
1280 END FOR j
```

```
1281 END FOR v
```

```
1282 END DEFINE
```



Sequence

FS [22] CTS [2 4 3 1]

[1] AUTO Off/On

NodeID Off/On (N)

(R) Back/End: (B) (L)

[1]Pyramid [2]Octahedron [3]Cube [4]Dodecahedron [5]Shuttle [6]Pod @Quit

QBITS 3DGraphics Procedures

Init_win	Sets Screen layout and Title
QB3D_Aid	Sets Side Panel for 3D Graphics
QB3D_key	Sets Lower Side Panel Action keys and 3D Objects
Obj_Shape	Sets DATA references etc for Object
Obj_Dat(n)	Sets DATA RESTORE for Nodes and Frames
Obj_Put(n)	Motion Rotation Write Object Data
Obj_Get(n)	Motion Rotation Fetch Object Data
Obj_keys(key,vm)	Motion Rotation Increase/Decrease Variables
Obj_Space	Motion Rotation Boundaries Check
Beeps(b)	Sounds for Shuttle Jets and Docking or Alarm failure
QExit	End Program

Menu_3DCommands	Menu loop to access key commands
Obj_Sel	Sets Status for Object in use
Obj_Lock(n)	F5 Lock/Unlock Object 1234 to screen
Obj_View	CTRL 1234 Manual setting of Objects sequence
Obj_Group	Multiple Objects 1 to 4 Static & AUTO Orientation
Obj_Sort	Sets Draw Sequence for Objects 1 to 4 vs value
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
Obj_Pos	Update and display Rotation and Motion variables
Obj_Ang	Update Rotation Graphics for Roll, Loop & Spin

Pod_Rescue	Menu for Pod Rescue Simulation
Pod_Draw	Draws Pod and Shuttle for simulation
Pod_Fires	Targets and Shuttle Lasers
Pod_Set	Set up different Rescue Pod Locations
Pod_Chk	Development checks for Shuttle Pod Alignment

Globe3D	Initiates changes to screen layout set revolving Globe.
GTitle	Menu Writes Str\$ to screen
Wold	Draws World Circle
Continents	Menu to Select Continent and display options
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

DATA Lines	Nodes, Frames of basic Objects, Shuttle, Rescue Pod, World Map
-------------------	--

QBITS 3DGraphicsSE Code

1000 REMark **QBITS_3DGraphicsEE_bas** [3D Graphics SE 2024 QL40th – QPC2] vM30

1002 dev\$='win1_' :MODE 4:gx=0:gy=0 :REMark Basic Settings

1004 **WHEN ERROr** : CONTINUE : **END WHEN**

1006 REMark **Import QBITSConfig Settings - QPC2**

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

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

1009 DIM x(50),y(50),z(50),vx(50),vy(50),Obj(6,20),pl(4),vp(4)

1010 bg1=0:bg2=7:k=49 :REMark Paper/Ink Settings

1011 fs=5000 :REMark focal scale

1012 aset=-1:cset=1:nset=1 :REMark Action switches

1013 px=0:ix=0:py=0:iy=0:vs=1.4:vm=2.4:iz=5E-2 :REMark xX,yY,zZ Axis [iz=0.05]

1014 rx=0:ax=0:ry=0:ay=0:az=0:iz=5E-2:ia=5 :REMark Incruments iz=0.05

1016 REMark 3D Revolving Graphics :SE Globe and Pod Rescue added

1017 REMark EE Extends Globe Menu+ more difficulties to Pod Rescue

1018 REMark Multiple Objects - XYZ Axis & Angle Rotation Intergation

1020 **Obj_Shape:Init_win:QB3D_Aid:QB3D_Key:Menu_3DCommands**

1022 **DEFinE PROCEDURE Init_win**

1023 OPEN#4,con_10x10a10x10_4

1024 OPEN#3,sch_:WINDOW#3,116,150,4+gx,26+gy

1025 WINDOW#2,512,224,gx,gy :BORDER#2,1,3:PAPER#2,0:CLS#2

1026 WINDOW#1,386,196,122+gx,26+gy:BORDER#1,1,3:PAPER#1,0:INK#1,7

1027 WINDOW#0,512,32,gx,224+gy :BORDER#0,1,3:PAPER#0,0:INK#0,7:CLS#0

1028 ch=2:CUSOR#ch,0,0:OVER#ch,1

1029 CSIZE#ch,2,1:str\$='QBITS 3D Graphics'

1030 INK#ch,2:FOR i=0 TO 1:CUSOR#ch,2+i,3:PRINT#ch,str\$

1031 INK#ch,6:FOR i=0 TO 1:CUSOR#ch,4+i,4:PRINT#ch,str\$

1032 CSIZE#ch,0,0 :INK#ch,7

1033 FOR i=0 TO 1:CUSOR#2,220+i,14:PRINT#2,'(P)od Rescue'

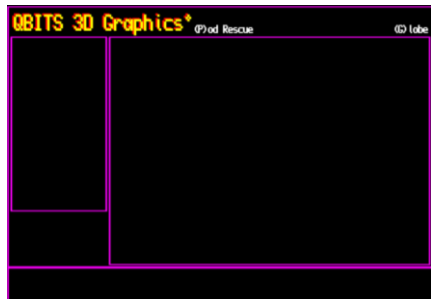
1034 FOR i=0 TO 1:CUSOR#2,460+i,14:PRINT#2,'(G)lobe'

1035 OVER#ch,0:ch=3:SCALE#ch,170,0,0:BORDER#ch,1,3

1036 **END DEFinE**

QBITS 3D Graphics*

Note: Window Setu



```

1038 DEFINE PROCEDURE QB3D_Aid
1039 LOCAL a,b,c,d,e,f,g,h,j,k:mx=34:my=70
1040 OVER#ch,1:INK#ch,7:CSIZE#ch,2,0:RESTORE 1036
1041 FOR i=1 TO 4:READ a,b,STR$:CURSOR#ch,a,b:PRINT#ch,STR$
1042 DATA 7,17,'←',60,17,'→',32,2,'↑',34,34,'↓'
1043 OVER#ch,1:CSIZE#ch,0,0:INK#ch,7
1044 FOR i=1 TO 11
1045 READ a,b,c,STR$:INK#ch,c
1046 CURSOR#ch,a,b:PRINT#ch,STR$
1047 CURSOR#ch,a,b:PRINT#ch,STR$
1048 END FOR i
1049 DATA 26,2,4,'Y',46,34,4,'Y',18,18,2,'X',54,18,2,'X',12,30,7,'+Z'
1050 DATA 52,4,7,'-Z',32,106,7,'[zZ] Roll',30,46,4,'[yY] Spin'
1051 DATA 82,80,2,'[xX]',82,70,2,'Loop'
1052 OVER#ch,0:INK#ch,5:CURSOR#ch,70,58:PRINT#ch,'ROTATE'
1053 INK#ch,2:LINE#ch,mx-9,my+74 TO mx+11,my+74 :REMark XX
1054 INK#ch,4:LINE#ch,mx,my+62 TO mx,my+86 :REMark YY
1055 INK#ch,7:LINE#ch,mx-10,my+62 TO mx+12,my+86 :REMark ZZ
1056 INK#ch,7:CIRCLE#ch,mx,my,18,1,0 :REMark Roll
1057 INK#ch,4:CIRCLE#ch,mx,my+27,18,.32,PI/2: :REMark Spin
1058 INK#ch,2:CIRCLE#ch,mx+28,my,17,.32,0:INK#ch,5 :REMark Loop
1059 STR$='MOTION':FOR i=1 TO 6:CURSOR#ch,100,-8+i*9:PRINT#ch,STR$(i)
1060 STR$='ANGLE':FOR i=1 TO 1055:CURSOR#ch,4,49+i*9:PRINT#ch,STR$(i)
1061 INK#ch,7:CURSOR#ch,2,124:PRINT#ch,'F f'
1062 FOR i=1 TO 12
1063 READ a,b,c,d,e,f,g,h,j,k : INK#ch,j :FILL#ch,k
1064 LINE#ch,a,b TO c,d TO e,f TO g,h TO a,b:FILL#ch,0
1065 END FOR i
1066 DATA 10,15,10,25,20,25,20,15,7,1,20,15,20,25,25,30,25,20,2,1
1067 DATA 10,25,15,30,25,30,20,25,4,1,30,15,30,25,40,25,40,15,7,0
1068 DATA 40,15,40,25,45,30,45,20,7,0,30,25,35,30,45,30,40,25,7,0
1069 DATA 55,15,55,25,65,25,65,15,7,0,60,20,60,30,70,30,70,20,7,0
1070 DATA 55,15,55,25,60,30,60,20,7,0,65,15,65,25,70,30,70,20,7,0
1071 DATA 75,15,75,25,90,25,90,15,7,0,75,25,82,35,87,35,90,25,7,0
1072 INK#ch,5:CURSOR#ch,2,138:PRINT#ch,'FILL frame'
1073 ch=1:SCALE#ch,200,-143,-100:CSIZE#ch,0,0:INK#ch,4:rx=0:ry=0:rz=0:kch=0
1074 END DEFINE

```

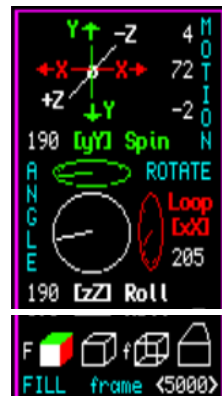
Note: 3D Graphics Side Panel

Note: Data to Build Menu Graphics

Note:

Side Panel Displays **MOTION** - movement values of Object along each of the Axis **ZZ** [-20 to +20] : **XX** [-120 TO +120] : **YY** [-90 to +90]

The **ROTATE** Spin Loop Roll about each of the Axis is shown Graphically to represent the changing **ANGLE** and by each display there is also a numerical value giving the degrees of Rotation [0 to 360].



The bottom displays represent various **FRAME** Modes **[F]** For Frame **FILL**. For Wireframe **[f]** toggles between Full and Solid view. Focal Scale **[fs]** <100 to 5000> use Chevrons to Increase/Decrease.

```

1076 DEFINE PROCEDURE QB3D_Key
1077 ch=2:BLOCK#2,120,42,0,178,0:OVER#ch,1:CUSOR#ch,0,0:CSIZE#ch,0,0:INK#ch,7
1078 CUSOR#ch,14,178:PRINT#ch,'F5 [ ] CTRL':FOR i=1 TO 4:vp(i)=i
1079 LINE#2,1,17 TO 1,19 TO 3,19:LINE#2,10,17 TO 10,19 TO 12,19 TO 12,17
1080 BLOCK#2,8,4,2,182,4:BLOCK#2,9,4,29,182,7
1081 CUSOR#ch,2,188:PRINT#ch,'[#] AUTO Off/On <----->'
1082 CUSOR#ch,2,198:PRINT#ch,'NodeID Off/On (N)'
1083 CUSOR#ch,2,209:PRINT#ch,'(R) BackGnd: (B)(W)'
1084 OVER#ch,0:ch=0:CSIZE#ch,0,0:INK#ch,4:CLS#ch
1085 CUSOR#ch, 6,8:PRINT#ch,'[1]Pyramid [2]Octahedron [3]Cube '
1086 PRINT#ch,'[4]Dodecahedron [5]Shuttle [6]Pod'
1087 INK#0,7:CUSOR#0,440,8:PRINT#ch,'(Q)uit'
1088 END DEFINE

```

```

F5 [ ] CTRL 1 2 3 4
[#] AUTO Off/On <----->
NodeID Off/On (N)
(R) BackGnd: (B)(W)

```

```

[1]Pyramid [2]Octahedron [3]Cube [4]Dodecahedron [5]Shuttle [6]Pod (Q)uit

```

Note: Obj_Shape with Obj_Dat provides start up values and Pointers for Node and Frame DATA Tables and loads Object array Obj(n,20). They are used by (R)eset if actioned.

```

1090 DEFINE PROCEDURE Obj_Shape
1091 px=0:py=0:vs=1.4:xx=4:yy=4 :REMARK WARNING RESTORE nres vres DATA Lines
1092 Obj_Dat 1 :rx= 60 :ry= 30 :rz= 0:Obj_Put 1 :pl(1)=6
1093 Obj_Dat 2 :rx= 15 :ry= 30 :rz= 0:Obj_Put 2 :pl(2)=78
1094 Obj_Dat 3 :rx= 15 :ry= 30 :rz= 0:Obj_Put 3 :pl(3)=168
1095 Obj_Dat 4 :rx= 15 :ry= 0 :rz= 0:Obj_Put 4 :pl(4)=222
1096 Obj_Dat 5 :rx= 15 :ry= 30 :rz=10:Obj_Put 5
1097 Obj_Dat 6 :rx= 15 :ry= 30 :rz=10:Obj_Put 6
1098 END DEFINE

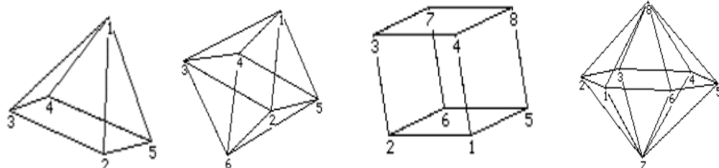
```

```

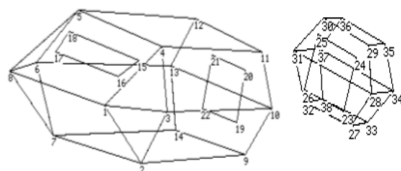
1100 DEFINE PROCEDURE Obj_Dat(n)
1101 IF n=1 :nres=2002:sn= 1:mn= 5 :vres=2010:vo= 5
1102 IF n=2 :nres=2017:sn= 1:mn= 6 :vres=2025:vo= 8
1103 IF n=3 :nres=2035:sn= 1:mn= 8 :vres=2045:vo= 6
1104 IF n=4 :nres=2053:sn= 1:mn= 8 :vres=2063:vo=12
1105 IF n=5 :nres=2077:sn= 1:mn=22 :vres=2119:vo=16
1106 IF n=6 :nres=2101:sn=23:mn=38:vres=2137:vo=11
1107 END DEFINE

```

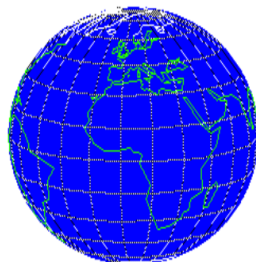
Basic Shapes



Shuttle & Pod



World Maps

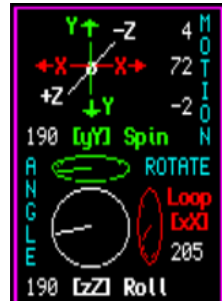


1109 **DEFine PROCEDURE** Obj_Put(n) Note: Write Object Shape Data
 1110 Obj(n,4)=nres:Obj(n,5)=sn:Obj(n,6)=mn:Obj(n,7)=vres:Obj(n,8)=vo
 1111 Obj(n,9)=rx:Obj(n,10)=ry:Obj(n,11)=rz:Obj(n,12)=px:Obj(n,13)=py:Obj(n,14)=vs
 1112 Obj(n,15)=xx:Obj(n,16)=yy:Obj(n,17)=zz:Obj(n,18)=ax:Obj(n,19)=ay:Obj(n,20)=az
 1113 **END DEFine**

1115 **DEFine PROCEDURE** Obj_Get(n) Note: Read Object Shape Data
 1116 nres=Obj(n,4):sn=Obj(n,5):mn=Obj(n,6):vres=Obj(n,7):vo=Obj(n,8)
 1117 rx=Obj(n,9):ry=Obj(n,10):rz=Obj(n,11):px=Obj(n,12):py=Obj(n,13):vs=Obj(n,14)
 1118 xx=Obj(n,15):yy=Obj(n,16):zz=Obj(n,17):ax=Obj(n,18):ay=Obj(n,19):Obj(n,20)=az
 1119 **END DEFine**

1121 **DEFine PROCEDURE** Obj_Keys(key,vm) Note: Change Object Shape Data

1122 **SElect ON** key
 1123 =43,61:IF zz< 0.1:zz=zz+iz:vs=vs+zz } :REMark (+)Increase Vector size
 1124 =45 :IF zz> -0.1:zz=zz-iz:vs=vs-zz } :REMark (-)Decrease Vector size
 1125 =192 :IF xx> -12 :xx=xx-ix:px=px-xx } :REMark Left ←
 1126 =200 :IF xx< 12 :xx=xx+ix:px=px+xx } :REMark Right →
 1127 =208 :IF yy< 12 :yy=yy+iy:py=py+yy } :REMark Up ↑
 1128 =216 :IF yy> -12 :yy=yy-iy:py=py-yy } :REMark Down ↓
 1129 =121 :IF ay< 15 :ay=ay+ia:ry=ry+ay } :REMark (y)Spin Clockwise
 1130 = 89 :IF ay> -15:ay=ay-ia:ry=ry-ay } :REMark (Y)Spin Anti-CW
 1131 =120 :IF ax< 15 :ax=ax+ia:rx=rx+ax } :REMark (x)Loop ClockWise
 1132 = 88 :IF ax> -15:ax=ax-ia:rx=rx-ax } :REMark (X)Loop Anti-CW
 1133 =122 :IF az< 15 :az=az+ia:rz=rz+az } :REMark (z)Roll Clockwise
 1134 = 90 :IF az> -15:az=az-ia:rz=rz-az } :REMark (Z)Roll Anti-CW
 1135 **END SElect** :Obj_Space
 1136 **END DEFine**



1138 **DEFine PROCEDURE** Obj_Space Note: Angle of Rotation Checks and Settings

1139 IF px<=-120:px=-120:END IF :IF px>=120:px=120:END IF
 1140 IF py<=-90:py=-90:END IF :IF py>= 90:py= 90:END IF
 1141 IF vs<= .4:vs= .4:END IF :IF vs>= 2.4:vs=2.4:END IF
 1142 IF rx>360 :rx=0 :END IF :IF rx<0 :rx=360 :END IF
 1143 IF ry>360 :ry=0 :END IF :IF ry<0 :ry=360 :END IF
 1144 IF rz>360 :rz=0 :END IF :IF rz<0 :rz=360 :END IF
 1145 **END DEFine**

1147 **DEFine PROCEDURE** Beeps(b) Note: Action Sounds

1148 **SElect ON** b
 1149 =1:BEEP 5000,0,500,6,8,2,0,0
 1150 =2:BEEP 9500,0,200,6,2,1,0,0
 1151 =3:BEEP 30000,1,200,6,-5,8,0,0
 1152 =4:BEEP 25000,0,200,8,1,2,0,0
 1153 =4:BEEP 25000,0,200,8,1,2,0,0
 1154 =5:BEEP 3000,0,400,2,1,0,0,0
 1155 **END SElect**
 1156 **END DEFine**

1158 **DEFine PROCEDURE** QExit Note: STOP/ Return to QBITSProgs Menu

1159 INK#0,7:CURSOR#0,480,8:PRINT#0,'Y/N':PAUSE:IF KEYROW(5)=64:LRUN dn\$:STOP
 1160 **END DEFine**

1200 REMark 3D Graphics Main Menu

1202 DEFine PROCEDURE Menu_3DCommands

```

1203 k=0:fsch=0:Obj_View:n=1:Obj_Sel
1204 REPEAT Com_lp
1205 SElect ON k
1206 =81,113:QExit:BLOCK#0,20,10,480,8,0 :REMark (Q)uit
1207 =66, 98:bg1=0:bg2=7:PAPER#1,0:CLS#1 :REMark (B)lack background
1208 =87,119:bg1=7:bg2=0:PAPER#1,7:CLS#1 :REMark (W)hite background
1209 =71,103:Globe3D :REMark (G)lobe3D
1210 =80,112:fsck=1:Obj_Shape:Pod_Rescue :fsck=0 :REMark (P)od Rescue
1211 =49 TO 55:n=k-48:Obj_Sel :REMark Select Object 1-6
1212 =32 :IF aset=-1:aset=5:ELSE aset=-1 :REMark Toggle animation
1213 =102 :IF cset=1 OR cset=3:cset=2:ELSE cset=1 :REMark (f)rame On/Off
1214 =70 :IF cset=1 OR cset=2:cset=3:ELSE cset=1 :REMark (F)ILL On/Off
1215 =78,110:IF nset= 1:nset=2:ELSE nset=1 :REMark (N)ode ID On/Off
1216 =62 :fs=fs+50 :IF fs>5000 :fs=5000 :REMark (>)Increase Focal Scale
1217 =60 :fs=fs-50 :IF fs< 100 :fs= 100 :REMark (<)Decrease Focal Scale
1218 =43,45,61,192,200,208,216,88 TO 122:Obj_Keys k,2,4:Obj_Put n
1219 =145 TO 148:IF n<5:Obj_View :REMark Set Viewpoint Position
1220 =248 :IF n<5:Obj_Lock n :REMark Lock Object to Screen
1221 =82,114:Obj_Shape:FOR i=1 TO 4:vp(i)=i :REMark Reset Objects
1222 END SElect
1223 CLS:IF n<5:Obj_Group Note: Group Mode
1224 Obj_Get n:CUSOR#2,44,178:PRINT#2,n
1225 IF k1 AND k2 Note: Shuttle+Pod
1226 IF ry>180:Obj_Dat 5:Obj_Draw:Obj_Dat 6:ELSE Obj_Dat 6:Obj_Draw:Obj_Dat 5
1227 END IF
1228 IF n>4:Obj_Draw:END IF :IF k=35:aset=aset+1:IF aset>=9:aset=2:END IF :END IF
1229 IF aset=-1 Note: Static Mode
1230 xx=4:yy=4:zz=5E-2:ix=0:iy=0:iz=0:ax=5:ay=5:az=5:ia=0
1231 Obj_Put n:BLOCK#2,16,3,98,192,0
1232 ELSE Note: Animation On
1233 ix=2:iy=2:iz=5E-2:ia=5:BLOCK#2,16,3,98,192,7
1234 END IF
1235 Obj_Ang:Obj_Pos:k=CODE(INKEY$(#4,aset)):INK bg2 Note: Screen Info Updates
1236 END REPEAT Com_lp
1237 END DEFine

```

1239 DEFine PROCEDURE Obj_Sel

```

1240 IF n<5:k1=0:k2=0:END IF Note: Object Load Checks
1241 IF n>4:aset=-1:FOR i=1 TO 4:Obj(i,1)=1:Obj_Lock i Note: Reset Unlink Shuttle+Pod
1242 IF n=5:k1=1:Obj_Shape:END IF Note: Unlock Object 1 to 4
1243 IF n=6:k2=1:Obj_Shape:END IF Note: Link Shuttle
1244 END DEFine Note: Link Pod

```

[1]Pyramid [2]Octahedron [3]Cube [4]Dodecahedron [5]Shuttle [6]Pod [0]uit

1246 DEFine PROCEDURE Obj_Lock(n)

```

1247 IF Obj(n,1)=0:Obj(n,1)=1:INK#0,7:ELSE Obj(n,1)=0:INK#0,4 Note:F5 Toggle Object Lock/Unlock
1248 CURSOR#0,pl(n),8:IF Obj(n,1)=1:PRINT#0,'[',n,']':ELSE PRINT#0,'[',n,']'
1249 END DEFine

```

F5 [4]CTRL 3 4 1 2

1251 DEFine PROCEDURE Obj_View

```

1252 IF k>144 AND k<149:vp1=vp(n):vp2=vp(k-144):vp(k-144)=vp1:vp(n)=vp2 Note: Use CTRL 1234 to Swap Objects
1253 FOR i=1 TO 4:CUSOR#2,74+i*9,178:PRINT#2,vp(i) Note: Printout of current status
1254 END DEFine

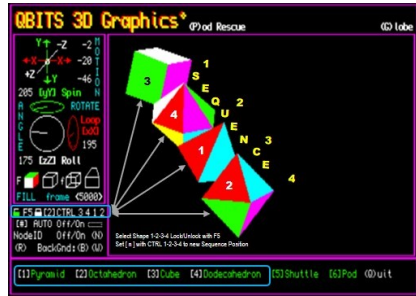
```

Note: For Group displays Objects are set to be Draw in sequence ie 1234. When an Object encounters another dependant on the sequence setting, they will pass in front or behind.

```

1256 DEFINE PROCEDURE Obj_Group
1257 FOR i=1 TO 4
1258   IF Obj(vp(i),1)=1 OR vp(i)=n
1259     Obj_Get vp(i)
1260     IF aset<>-1 Note: AUTO Mode

```



```

1261   px=px+xx:py=py+yy:ovs=vs:vs=vs+zz:IF ovs<>vs:vck=1
1262   rx=rx+ax:ry=ry+ay:rz=rz+az
1263   Obj_Space :REMark Check Boundaries
1264   IF px<=-120 OR px>=120:xx=-xx
1265   IF py<=-90 OR py>= 90:yy=-yy
1266   IF vs<=.4 OR vs>=2.4:zz=-zz
1267   END IF

```

Check for any ZZ Axis changes

```

1268   Obj_Put vp(i):Obj_Get vp(i):Obj_Draw
1269   END IF
1270 END FOR i
1271 IF vck=1:Obj_Sort:vck=0:Obj_View Note: If any ZZ Axis Changes
1272 END DEFINE

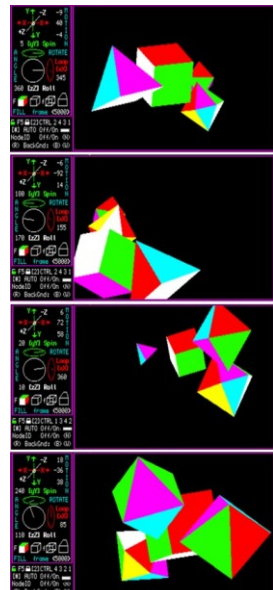
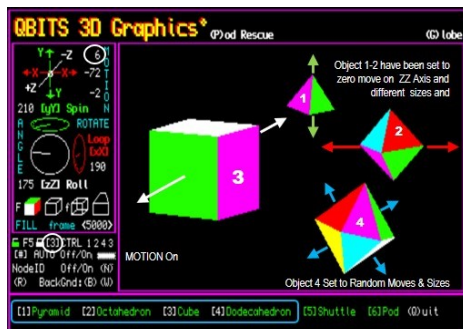
```

Note: When Group Objects change their Visual Scale vs their relative ZZ Axis position to the Viewpoint can change the Sequence of their being Drawn. Therefore, after each Group Draw if any Object changes their vs value, then the whole Group needs to be re-sequenced for the next display to screen. As there are only four Objects involved a simple Bubble Sort is used.

```

1274 DEFINE PROCEDURE Obj_Sort
1275 FOR v=1 TO 4
1276   FOR j=1 TO 4-v+1
1277     IF Obj(vp(j),14)>Obj(vp(j+1),14)
1278       temp=vp(j):vp(j)=vp(j+1):vp(j+1)=temp
1279     END IF
1280   END FOR j
1281 END FOR v
1282 END DEFINE

```



```

1284 DEFine PROCEDURE Obj_Node
1285 LOCAL np,a,b,c:RESTORE nres
1286 FOR np=sn TO mn
1287   READ a,b,c:x(np)=a*vs:y(np)=b*vs:z(np)=c*vs
1288 END FOR np
1289 END DEFine

```

Note: Load Node xyz coordinates

```

1291 DEFine PROCEDURE Obj_Calc
1292 cx=COS(RAD(rx)):sx=SIN(RAD(rx))
1293 cy=COS(RAD(ry)):sy=SIN(RAD(ry))
1294 cz=COS(RAD(rz)):sz=SIN(RAD(rz))
1295 FOR np=sn TO mn
1296   yt=y(np):y(np)=cx*yt-sx*z(np):z(np)=sx*yt+cx*z(np)
1297   xt=x(np):x(np)=cy*xt+sy*z(np):z(np)=sy*xt+cy*z(np)
1298   xt=x(np):x(np)=cz*xt-sz*y(np):y(np)=sz*xt+cz*y(np)
1299   vx(np)=px+(x(np)*fs)/(z(np)+fs)
1300   vy(np)=py+(y(np)*fs)/(z(np)+fs)
1301 END FOR np
1302 END DEFine

```

Note: Calculate Vectors

```

1304 DEFine PROCEDURE Obj_Draw
1305 LOCAL np,v,a,b,c,d,i:Obj_Node:RESTORE vres:iset=2:Obj_Calc
1306 FOR np=1 TO vo
1307   READ a,b,c,d,ic:IF cset=1:INK bg2:FILL 0:END IF
1308   IF cset=2:Obj_Cull:IF c1>0:GO TO 1310:END IF :INK bg2:FILL 0:END IF
1309   IF cset=3:Obj_Cull:IF c1>0:GO TO 1310:END IF :INK ic :FILL 1:END IF
1310   LINE vx(a),vy(a) TO vx(b),vy(b) TO vx(c),vy(c) TO vx(d),vy(d)
1311   LINE TO vx(a),vy(a):FILL 0
1312 END FOR np
1313 IF nset=2:FOR np=sn TO mn:CURSOR vx(np),vy(np),-2,2:PRINT np
1314 END DEFine

```

Note: Draws Objects Frames

Note: Obj_Calc

Note: Node point & number of Frames

Note:Draw Frame

Note: sn start Node mn max node

```

1316 DEFine PROCEDURE Obj_Cull
1317 c1=(x(b)-x(a))*(y(c)-y(a))-(x(c)-x(a))*(y(b)-y(a))
1318 END DEFine

```

Note: Check Frames Facing away from Viewpoint

```

1320 DEFine PROCEDURE Obj_Pos
1321 ch=3:INK#ch,7 :pz$=vs*20:pz=pz$-28
1322 CURSOR#ch, 4,106:PRINT#ch,FILL$(' ',3-LEN(rz))&rz
1323 CURSOR#ch, 4, 46:PRINT#ch,FILL$(' ',3-LEN(ry))&ry
1324 CURSOR#ch,84, 92:PRINT#ch,FILL$(' ',3-LEN(rx))&rx
1325 CURSOR#ch,72, 18:PRINT#ch,FILL$(' ',4-LEN(px))&px
1326 CURSOR#ch,78, 34:PRINT#ch,FILL$(' ',3-LEN(py))&py
1327 CURSOR#ch,78, 4:PRINT#ch,FILL$(' ',3-LEN(pz))&pz
1328 IF fsck=0:CURSOR#ch,80,138:PRINT#ch,FILL$(' ',4-LEN(fs))&fs
1329 END DEFine

```

Note: Updates Display of Position changes

```

1331 DEFine PROCEDURE Obj_Ang
1332 ch=3:INK#ch,0
1333 FILL#ch,1:CIRCLE#ch,34,70,17,1,0 :FILL#ch,0
1334 FILL#ch,1:CIRCLE#ch,34,97,15,.32,PI/2 :FILL#ch,0
1335 FILL#ch,1:CIRCLE#ch,62,70,14,.26,0 :FILL#ch,0
1336 INK#ch,7:LINE#ch,34,70 TO 34+17*COS(RAD(rz)),70+ 18*SIN(RAD(rz))
1337 INK#ch,4:LINE#ch,34,97 TO 34+16*COS(RAD(ry)),97+4.5*SIN(RAD(ry))
1338 INK#ch,2:LINE#ch,62,70 TO 62+ 4*COS(RAD(rx)),70+ 15*SIN(RAD(rx)):ch=1
1339 END DEFine

```

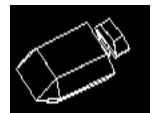
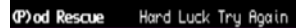
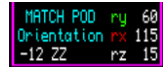
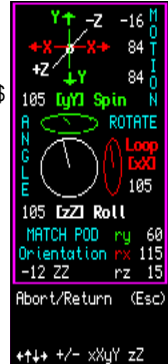
Note: Updates Displays the Angle changes

1400 REMark QBITS Pod Rescue

```

1402 DEFINE PROCEDURE Pod_Rescue
1403 RESTORE 1406:PAPER#1,0:CLS#0:Pod_Set
1404 BLOCK#2,120,44,0,176,0:BLOCK#3,112,32,0,116,0
1405 FOR i=1 TO 11:READ ch,ci,x1,y1,STR$;INK#ch,ci:CURSOR#ch,x1,y1:PRINT#ch,STR$
1406 DATA 3,5,8,118,'MATCH POD',3,5,2,128,'Orientation',3,7,4,138,pvs$&' ZZ'
1407 DATA 3,4,74,118,'y'&pry$,3,2,74,128,'rx',3,7,74,138,'rz'&prz$
1408 DATA 2,7,300,14,'Locate Target Fire Laser',2,7,4,178,'Abort/Return (Esc)'
1409 DATA 2,7,4,208,'↔↕↔↔ ↗↘ xXyY zZ',0,7,212,8,'Targets',0,7,348,8,'FUEL'
1410 BLOCK#2,12,3,380,18,7
1411 BLOCK#0,124,9,376,8,7:BLOCK#0,122,7,377,9,0:BLOCK#0,120,5,378,10,5
1412 FOR i=1 TO 5
1413 INK#0,7 :CIRCLE#0,290+i*44,60,18:INK#0,4
1414 FILL#0,1:CIRCLE#0,290+i*44,60,8:FILL#0,0
1415 END FOR i
1416 FOR a=41 TO 45:vx(a)=RND(-6 TO 6)*20:vy(a)=RND(-5 TO 5)*10
1417 tc=1:tn=40+tc:tf=0:sk=1:cset=2:nset=1 :REMark Set Wire Frame/Node ID Off
1418 ix=2:iy=2:ax=5:ay=5:az=5:ia=5:Gch=0 :REMark Set Motion Variables
1419 Obj_Get 5:px=0:py=0:vs=6:rx=0:ry=0:rz=0:Obj_Put 5 :REMark Update Shuttle
1420 : Note: For Test purposes set target count tc=5
1421 REPEAT Chk_lp
1422 IF tc>5
1423 IF py>=pwy-2 AND py<=pwy+2
1424 IF px>=pwx-2 AND px<=pwx+
1425 IF rz>=prz-5 AND rz<=prz+5 Note: Check px, py, vs to pwx, pwt, pvs Motion
1426 IF ry>=pry-5 AND ry<=pry+5
1427 IF rx>=prx-10 AND rx<=prx+10 AND ax=5 AND vs=pvs:Gch=1:EXIT Chk_lp
1428 END IF
1429 END IF Note Check rx,ry,rz to prx,pry,prz Rotation Angle
1430 END IF
1431 END IF
1432 END IF Note: For test purposes add Pod_Chk
1433 k=CODE(INKEY$(10)):Obj_Keys k,95 :IF k=27:Gch=0:EXIT Chk_lp
1434 SELECT ON k=88,89,90,120,121,122,192,200,208,216:fu=fu+sk:Beeps 5
1435 BLOCK#0,fu,5,498-fu,10,0 :IF fu>=120:Gch=0:EXIT Chk_lp
1436 IF k=32 AND tc<=5:fu=fu+sk:Pod_Fire:Obj_Get 5
1437 px=px+xx:py=py+yy:rx=rx+ax:ry=ry+ay:rz=rz+az:Obj_Space
1438 IF px<=-120 OR px>=120:xx=-xx:END IF :IF py<=-90 OR py>=90:yy=-yy:END IF
1439 CLS:Obj_Put 5:Pod_Draw:Obj_Get 5:Obj_Draw:Obj_Ang:Obj_Pos
1440 END REPEAT Chk_lp
1441 :
1442 BLOCK#2,160,10,300,14,0
1443 IF Gch=0:CURSOR#2,320,14:PRINT#2,'Hard Luck Try Again':Beeps 3
1444 IF Gch=1:CURSOR#2,320,14:PRINT#2,'Successful Docking':Beeps 2
1445 FOR i=1 TO 20:PAUSE 5:END FOR i
1446 xx=4:ix=0:yy=4:iy=0:ax=5:ay=5:az=5:ia=0:k1=1:k2=1
1447 px=0:py=0:rx=15:ry=30:rz=10:vs=1.4:cset=3:aset=-1:Obj_Put 5:n=5
1448 BLOCK#2,120,42,0,178,0:BLOCK#2,160,10,300,14,0:QB3D_Aid:QB3D_Key
1449 END DEFINE

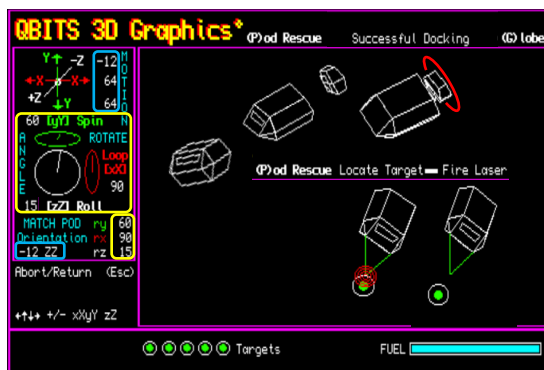
```




```

1451 DEFine PROCEDURE Pod_Draw
1452 Obj_Get 6:rx=rx+5:prx=rx:Obj_Space:Obj_Put 6:Obj_Draw
1453 CURSOR#3,92,128:PRINT#3,FILL$(' ',3-LEN(rx))&rx
1454 END DEFine

```



```

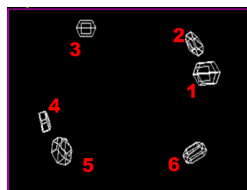
1456 DEFine PROCEDURE Pod_Fire
1457 x(40)=-60:y(40)=0:z(40)=0:sn=40:mn=40:Obj_Calc:INK 4:Beeps 1
1458 LINE vx(1),vy(1) TO vx(40),vy(40):LINE vx(8),vy(8) TO vx(40),vy(40)
1459 INK 4:FILL 1:CIRCLE vx(tn),vy(tn),3:FILL 0
1460 INK 7:CIRCLE vx(tn),vy(tn),8:PAUSE 5:INK 2
1461 IF vx(40)>=vx(tn)-8 AND vx(40)<=vx(tn)+8
1462   IF vy(40)>=vy(tn)-8 AND vy(40)<=vy(tn)+8
1463     FOR i=2 TO 8 STEP 2:CIRCLE vx(40),vy(40),i:PAUSE 2:END FOR i:PAUSE 20
1464     FILL#0,1:INK#0,0:CIRCLE#0,290+tc*44,60,20:FILL#0,0:tc=tc+1:tn=40+tc
1465   END IF
1466 END IF
1467 IF tc>5:CURSOR#2,300,14:PRINT#2,' Proceed to Docking '
1468 END DEFine

```

```

1470 DEFine PROCEDURE Pod_Set
1471 Obj_Get 6:sp=RND(1 TO 6)
1472 IF sp=1: rx= 10: ry= 60: rz= 15:px= 75:py= 25:vs=.8
1473 IF sp=2: rx= 30: ry= 15: rz= 30:px= 50:py= 45:vs=.7
1474 IF sp=3: rx= 15: ry= 90: rz= 15:px=-50:py= 80:vs=.6
1475 IF sp=4: rx= 60: ry=180: rz= 15:px=-70:py=-15:vs=.6
1476 IF sp=5: rx= 90: ry=330: rz=300:px= 65:py=-30:vs=.7
1477 IF sp=6: rx=330: ry= 30: rz=215:px=-50:py=-35:vs=.8
1478 Obj_Put 6:z$=vs:pvs$=z$*20-28
1479 pry$=FILL$(' ',3-LEN(pry))&pry:prz$=FILL$(' ',3-LEN(prz))&prz
1480 END DEFine

```



```

1482 DEFine PROCEDURE Pod_Chk
1483 CURSOR 0,10:PRINT 'vs 'vs,pvs\px 'px,pwx\py 'py,pwy\Pod Check'
1484 CURSOR 0,70:PRINT 'ry 'ry,pry\rx 'rx,prx\rz 'rz,prz
1485 END DEFine

```

Note: For Testing Shuttle Alignment

1500 REMark QBITS Globe WorldMap

1502 DEFINE PROCEDURE Globe3D

```

1503 BLOCK#2,120,42,0,178,0:PAPER 0:CLS:CLS#0:ch=3:CLS#ch:INK#ch,0:Beeps 4
1504 RESTORE 1505:FOR i=1 TO 16:PAUSE 2:READ a,b,STR$:GTitle a,b,STR$
1505 DATA 4,4,' Continents',7,16,' ',7,18,' (1)Arctic',7,27,' (2)Europe'
1506 DATA 7,36,' (3)Africa',7,45,' (4)Asia',7,54,' (5)America Nth'
1507 DATA 7,63,' (6)America Sth',7,72,' (7)Australasia',7,81,' (8)Antarctic'
1508 DATA 7,90,' ',7,99,' (S)et GMT',7,109,' (G)rid On/OFF'
1509 DATA 7,118,' %3/4%1/2 Rotate',7,127,' -/+ Radius',7,136,' '
1510 CURSOR#2,0,178:PRINT#2,' Abort/Return (Esc)\| Select Continent\|,or Action'
1511 R=90:wrX=0:wry=12:zm=12 :REMark Radius Pixels: x,y Angle coordinates
1512 S=0 :M=0 :P=15 :O=0 :g=0 :REMark Spin/Meridian/Parallel/rOtation/grid
1513 Gcol=248 :Ccol=4 :Acol=7 :REMark Grid/Coastline - Colours
1514 vh=1 :REMark BLOCK#2,112,32,2,188,0:vh=1 :REMark vh=0 make visible view hidden
1515 BEEP:S=S+3:World:Calc_ang:RESTORE 2500:Maps:Continents
1516 END DEFINE

```

1518 DEFINE PROCEDURE GTitle(ic,ypos,STR\$)

```

1519 STRIP#ch,ic:CURSOR#ch,2,ypos:PRINT#ch,STR$:FILL$(' ',18-LEN(STR$))
1520 END DEFINE

```

1522 DEFINE PROCEDURE World

```

1523 CLS:INK 63:CIRCLE 0,0,R+1:INK 1:FILL 1:CIRCLE 0,0,R:FILL 0 :REMark gcol
1524 END DEFINE

```

1526 DEFINE PROCEDURE Continents

```

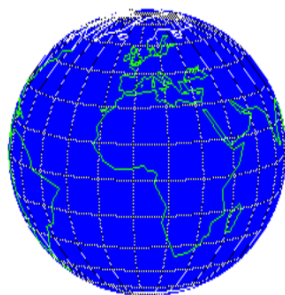
1527 REPEAT Choice
1528 k=CODE(INKEY$(-1))
1529 SELECT ON k
1530 =27:PAPER bg1:INK bg2:CLS:QB3D_Aid:QB3D_Key:EXIT Choice
1531 =50:R=90:wrX= 0:wry= 45:S= 15:zm=12 :REMark (2) Europe
1532 =51:R=90:wrX= 0:wry= 6:S= 18:zm= 6 :REMark (3) Africa
1533 =52:R=90:wrX= 0:wry= 45:S= 80:zm= 3 :REMark (4) Asia
1534 =53:R=90:wrX= 0:wry= 50:S= 99:zm= 6 :REMark (5) America Nth
1535 =54:R=90:wrX= 0:wry=-20:S=- 60:zm= 6 :REMark (6) America Sth
1536 =55:R=90:wrX= 0:wry=-18:S=134:zm= 6 :REMark (7) Australasia
1537 =49:R=90:wrX= 0:wry= 90:S= 15:zm=12 :REMark (1) Arctic
1538 =56:R=90:wrX= 0:wry=-90:S= 0:zm= 6 :REMark (8) Antarctic
1539 =115,83 :wrX=0 :wry=0:S=0 :REMark (S)et
1540 =103,71 :IF M=0 :M=15:ELSE M=0 :REMark (G)rid
1541 = 43,61 :IF R<150:R=R+3 :REMark + Radius
1542 = 45 :IF R> 30:R=R-3 :REMark - Radius
1543 =192:wrX=wrX+6 :REMark Left World Radial X Axis
1544 =200:wrX=wrX-6 :REMark Right
1545 =208:wry=wry+6 :REMark Up World Radial Y Axis
1546 =216:wry=wry-6 :REMark Down
1547 END SELECT
1548 World:Calc_ang:Grid:RESTORE 2500:Maps:INK 0
1549 END REPEAT Choice :REMark View_lp
1550 END DEFINE

```

```

1552 DEFine PROCEDURE Grid
1553 INK Gcol:IF M=0 THEN RETURN
1554 FOR O=M TO 360 STEP M
1555   T=0:FOR L=90 TO -90 STEP -P:Calc_posn
1556   END FOR O
1557   FOR L=-90+g TO 90-g STEP M
1558     T=0:FOR O= 0 TO 360 STEP P:Calc_posn
1559     END FOR L
1560   END DEFine

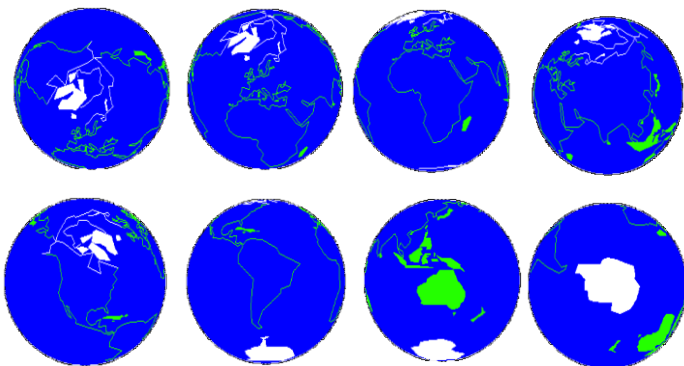
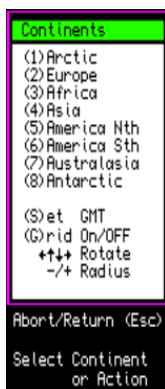
```



```

1562 DEFine PROCEDURE Maps
1563 REPEAT Loop
1564   READ num,col,fil:INK col:T=0 :IF num=9999:EXIT Loop
1565   READ L,O:Calc_posn :REMark Longitude & Latitude Offset
1566   FILL fil:FOR i=2 TO num:READ L,O:T=1:Calc_posn:END FOR i:FILL 0
1567 END REPEAT Loop
1568 END DEFine

```



```

1570 DEFine PROCEDURE Calc_ang
1571 sx=SIN(RAD(wrx)):cx=COS(RAD(wrx)) :REMark x coordinates
1572 sy=SIN(RAD(wry)):cy=COS(RAD(wry)) :REMark y coordinates
1573 END DEFine

```

```

1575 DEFine PROCEDURE Calc_posn
1576 Ms=SIN(RAD(O-S)):Mc=COS(RAD(O-S)) :REMark O Longitude S Rotation
1577 Pc=COS(RAD(L)) :Ps=SIN(RAD(L)) :REMark L Latitude
1578 wvz=R*(Ps*sy*cx-Pc*Ms*sx+Pc*Mc*cy*cx) :REMark Z axis
1579 wvx=R*(Pc*Ms*cx+Ps*sy*sx+Pc*Mc*cy*sx) :REMark X axis
1580 wvy=R*(Ps*cy-Pc*Mc*sy) :REMark Y axis
1581 IF vh=1 AND wvz<0 THEN T=0 :REMark vh=0 view hidden plane
1582 IF T=0:tx=wx:ty=wvy:T=1:RETURN :REMark T=0 Set tx,ty
1583 IF T=1:LINE tx,ty TO wx,wvy:tx=wx:ty=wvy :REMark T=1 Draw & Set px,py
1584 END DEFine

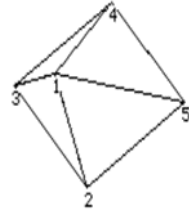
```

2000 REMark **Objects Data Nodes Frames etc.**

2002 REMark **Pyramid 5 Nodes**

2003 DATA 0, 0, -20
2004 DATA 20, 20, 20
2005 DATA 20, -20, 20
2006 DATA -20, -20, 20
2007 DATA -20, 20, 20

:REMark Node 1 nearest to view point



2009 REMark **Pyramid 5 Frames**

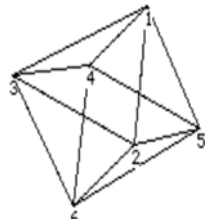
2010 DATA 1,2,3,1,2
2011 DATA 1,3,4,1,4
2012 DATA 1,4,5,1,5
2013 DATA 1,5,2,1,5
2014 DATA 5,4,3,2,bg2



2016 REMark **Diamond 6 Nodes**

2017 DATA 0, 30, 0
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

:REMark Node 1



:REMark Node 6

2024 REMark **Diamond 8 Frames**

2025 DATA 1,2,3,1,5
2026 DATA 6,3,2,6,3
2027 DATA 1,3,4,1,2
2028 DATA 6,4,3,6,4
2029 DATA 1,4,5,1,5
2030 DATA 6,5,4,6,3
2031 DATA 1,5,2,1,2
2032 DATA 6,2,5,6,4

:REMark Frame 1



:REMark Frame 8

2034 REMark **Cube 8 Nodes**

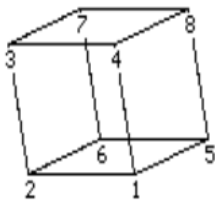
2035 DATA 20, -20, -20
2036 DATA -20, -20, -20
2037 DATA -20, -20, 20
2038 DATA 20, -20, 20
2039 DATA 20, 20, -20
2040 DATA -20, 20, -20
2041 DATA -20, 20, 20
2042 DATA 20, 20, 20

:REMark Node 1

:REMark Node 4

:REMark Node 5

:REMark Node 8

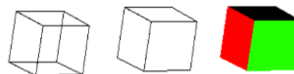


2044 REMark **Cube 6 Frames**

2045 DATA 8,7,6,5,bg2
2046 DATA 2,6,7,3,2
2047 DATA 4,3,7,8,4
2048 DATA 5,1,4,8,3
2049 DATA 5,6,2,1,5
2050 DATA 1,2,3,4,bg2

:REMark back Frame

:REMark front Frame



2152 REMark Polygon 8 Nodes

2153 DATA 32, 0, 0

:REMark Node 1

2154 DATA 16, 24, 0

2155 DATA -16, 24, 0

2156 DATA -32, 0, 0

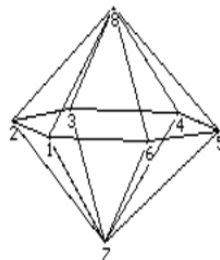
2157 DATA -16, -24, 0

2158 DATA 16, -24, 0

2159 DATA 0, 0, -32

2160 DATA 0, 0, 32

:REMark Node 8



2062 REMark Polygon 12 Frames

2063 DATA 1,2,7,7,6

:REMark 1

2064 DATA 2,3,7,7,5

2065 DATA 3,4,7,7,4

2066 DATA 4,5,7,7,3

2067 DATA 5,6,7,7,2

2068 DATA 6,1,7,7,bg2

:REMark 6

2069 DATA 2,1,8,8,2

2070 DATA 3,2,8,8,3

2071 DATA 4,3,8,8,bg2

2072 DATA 5,4,8,8,5

2073 DATA 6,5,8,8,6

2074 DATA 1,6,8,8,4

:REMark 12



2076 REMark Space Shuttle 22 Nodes

2077 DATA -40, 0, 20

:REMark Node 1

2078 DATA -18, -20, 20

:REMark Node 2

2079 DATA -20, 0, 30

2080 DATA -18, 20, 20

2081 DATA -18, 20, -20

2082 DATA -20, 0, -30

2083 DATA -18, -20, -20

2084 DATA -40, 0, -20

2085 DATA 38, -20, 20

2086 DATA 40, 0, 30

2087 DATA 38, 20, 20

2088 DATA 38, 20, -20

2089 DATA 40, 0, -30

2090 DATA 38, -20, -20

2091 DATA -24, 14, 16

:REMark Node 14

2092 DATA -30, 8, 14

:REMark Node 15

2093 DATA -30, 8, -14

2094 DATA -24, 14, -16

:REMark Node 18

2095 DATA 40, -10, 10

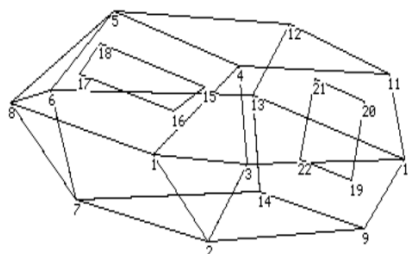
:REMark Node 19

2096 DATA 40, 10, 10

2097 DATA 40, 10, -10

2098 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.

2100 REMark **Rescue Pod 16 Nodes**

2101 DATA 43,-10, 10 :REMark Node 23 1 Hatch
2102 DATA 43, 10, 10
2103 DATA 43, 10,-10
2104 DATA 43,-10,-10 :REMark Node 26 4
2105 DATA 47,-15, 12 :REMark Node 27 5 Front
2106 DATA 45, 0, 20
2107 DATA 47, 15, 12
2108 DATA 47, 15,-12
2109 DATA 45, 0,-20
2110 DATA 47,-15,-12 :REMark Node 32 10
2111 DATA 58,-15, 12 :REMark Node 33 11 Rear
2112 DATA 60, 0, 20
2113 DATA 58, 15, 12
2114 DATA 58, 15,-12
2115 DATA 60, 0,-20
2116 DATA 58,-15,-12 :REMark Node 38 16



2118 REMark **Space Shuttle 16 Frames**

2119 DATA 9,10,13,14,5 :REMark Rear Frames 1
2120 DATA 10,11,12,13,240
2121 DATA 2,9,14,7,5 :REMark Side Frames 3
2122 DATA 6,7,14,13,5
2123 DATA 5,6,13,12,240
2124 DATA 5,12,11,4,240
2125 DATA 4,11,10,3,240
2126 DATA 3,10,9,2,5
2127 DATA 3,2,1,3,5 :REMark Front Frames 9
2128 DATA 1,2,7,8,5
2129 DATA 7,6,8,7,5
2130 DATA 8,6,5,8,240
2131 DATA 4,1,8,5,240
2132 DATA 1,4,3,1,240
2133 DATA 15,16,17,18,0 :REMark Pilot Window 15
2134 DATA 19,20,21,22,191 :REMark Rear Door 16



2136 REMark **Rescue Pod 11 Frames**

2137 DATA 33,34,37,38,5 :REMark Rear Frame 17
2138 DATA 34,35,36,37,240
2139 DATA 32,31,28,27,5 :REMark Front Frame 20
2140 DATA 31,30,29,28,240
2141 DATA 27,28,34,33,5 :REMark Side Frames 23
2142 DATA 28,29,35,34,240
2143 DATA 29,30,36,35,240
2144 DATA 37,36,30,31,240
2145 DATA 38,37,31,32,5
2146 DATA 32,27,33,38,5
2147 DATA 26,25,24,23,191 :REMark Pod Hatch 28



2500 REMark **Globe Data for World Maps**

Note: Each block of Code has a **FOR** loop number of entries followed by **INK** Colour and **FILL** 1=On / 0=Off.

2502 DATA 7,Acol,1 :REMark **Iceland**
2503 DATA 66.5,-22.5,65.4,-24.5,66.6,-16.65,-13.5,63,-19,64,-22,66.5,-22.5

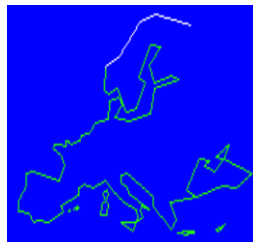
Acol =7 Artic ; Ccol = 4 Continent



2504 DATA 24,Ccol,0 :REMark **UK & Ireland**
2505 DATA 58.5,-5, 58.2,-1.8, 56,-3.3, 56,-2, 53,.5
2506 DATA 53,1.6, 52.2,1.7, 51.3,.8, 51.3,1.5, 50.9,1
2507 DATA 50,-5.8, 51.4,-3.7, 51.7,-5, 53.3,-4.5, 53.3,-3
2508 DATA 55,-3.5, 54.7,-5, 57.5,-6.5, 58.5,-5
2509 DATA 55.3,-6.5, 54.3,-10, 51.4,-10, 52.2,-6.3, 55.3,-6.5 :REMark 50



2510 DATA 89,Ccol,0 :REMark **EUROPE**
2511 DATA 41,29, 42,35, 41,38, 42.5, 42.3, 46,37 :REMark 10
2512 DATA 48,39, 46.5,35, 46,37, 44.3,34, 45.5,32
2513 DATA 46.2,33.5,47,31,42.5,27,41,29,40.8,23
2514 DATA 38,24,36.5,22.8,40.5,19.5,42,19.5,45.7,13.7
2515 DATA 45.5,12.3,44.4,12.3,43.6,13.6,42.5,14.1,40,18.5 :REMark 50
2516 DATA 40.5,17,39.7,16.5,39,17.2,38,15.6,38,12.5
2517 DATA 36.6,15,38.9,16.1,40,15.7,41.3,13,43,10.5
2518 DATA 44.3,8.9,43.2,6.2,43.5,4,42.7,3,41.8,3.3
2519 DATA 39.5,-4.38.7,-3,36.6,-2.1,36.5,-4.8,36,-5.4
2520 DATA 37.1,-6.7,37,-8.8,38.6,-9.4,41.2,-8.6,43.1,-9.3 :REMark 100
2521 DATA 43.7,-7.7,43.3,-1.5,46.1,-1.2,47.3,-2.5,48,-4.7
2522 DATA 48.6,-4.7,48.8,-3.1,48.7,-1.7,49.8,-2.49.8,-1.3
2523 DATA 49.4,-1.1,49.3,-1.49.7,-2.50.2,1.5,50.9,1.6
2524 DATA 51.4,3.6,53.3,4.7,54.8,3.57,8.1,57.6,10.7
2525 DATA 56.4,11.9,54.5,10.54,14.2,55,20,59,22 :REMark 150
2526 DATA 60,30,60.6,28,60,22,63,21,65.6,26
2527 DATA 66,22,61,17,60,19,56,16,55.4,13
2528 DATA 59,10.3,58.7,6,58.5,6,62.5,5.5 :REMark 176



2529 DATA 5,Acol,0,62.5,5.5,64,10,70.3,19,71.2,27,67.8,41.5 :REMark Artic area

2530 DATA 10,Ccol,1 :REMark **Corsica & Sardinia**
2531 DATA 43.9,4,42.4,8.5,41.5,8.8,40.9,8,39.1,9.7
2532 DATA 38.9,8.4,40.8,8.4,41.3,9.2,42.1,9.6,43,9.4 :REMark 20

2533 DATA 11,Ccol,1 :REMark **Balearic Isles**
2534 DATA 40,3.1,39.9,3.1,39.8,3.2,39.9,3.3,39.8,3.5 :REMark 10
2535 DATA 39.3,3.1,39.4,2.9,39.6,2.8,39.5,2.7,39.4,2.6,40,3.1

2536 DATA 5,Ccol,1,39.1,1.7,39,1.8,38.9,1.6,39,1.5,39.1,1.7 :REMark 10

2537 DATA 6,Ccol,1 :REMark **Cyprus**
2538 DATA 35.5,32,35.6,33,35.9,34,35.2,33,35.2,32,35.6,32

2539 DATA 7,Ccol,1 :REMark **Crete**
2540 DATA 35.8,24,35.9,26,35.7,27.5,35.5,27.5,35.6,26,35.5,26,35.8,24

2541 DATA 61,Ccol,0 :REMark **AFRICA**
 2542 DATA 28,35,28,33,15,40,10.5,45,12,51.4 :REMark 10
 2543 DATA 4,47.7,-5,39,-16,41,-20,35,-25,35
 2544 DATA -26,33,-29,32,-34,26,-35,20,-18,12
 2545 DATA -11,14,-1,9,3,10,4,6,8,4,4,3,5,9
 2546 DATA 6,5,4,3,4,8,-2,4,6,-7,7,7,8,-12,9,9,6,-13,4 :REMark 50
 2547 DATA 12,4,-16,7,14,9,-17,6,17,2,-16,1,21,3,-17,2,28,-12,9
 2548 DATA 30,3,-9,5,31,-9,8,32,-9,8,33,3,-8,3,33,9,-6,9
 2549 DATA 35,8,-6,35,9,-5,4,35,2,-4,7,35,-2,36,4,1
 2550 DATA 37,3,10,2,36,7,10,4,37,11,36,1,10,5,35,2,11,1
 2551 DATA 34,10,32,8,12,5,32,94,13,2,32,4,15,3,31,5,15,6
 2552 DATA 30,19,31,20,32,19,7,33,22,31,29
 2553 DATA 31,6,31,31,2,33,5,37,36,37,28,40,26,41,29 :REMark 122



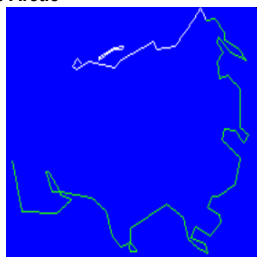
2554 DATA 4,Ccol,1,28,6,-16,1,28,-16,7,28,4,-17,28,6,-16,1 :REMark **Canary Isles**

2555 DATA 7,Ccol,29,5,-13,3,29,-13,3,28,8,-14,28,-14,5
 2556 DATA 28,3,-13,8,29,-13,7,29,5,-13,4 :REMark 14

2557 DATA 6,Ccol,28,2,-15,6,28,2,-15,4,27,8,-15,3,27,6,-15,7
 2558 DATA 27,9,-15,8,28,2,-15,6

2559 DATA 6,Ccol,1,-13,49,-17,44,-25,44,-25,47,-15,50,5,-13,49:REMark **Madagascar**

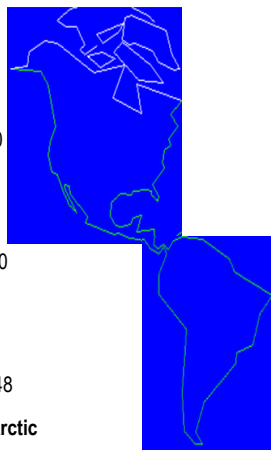
2560 DATA 13,Acol,0 :REMark **ASIA**
 2561 DATA 66,5,39,67,2,33,64,5,35,64,40,68,2,44
 2562 DATA 69,67,72,70,77,112,74,110,72,130
 2563 DATA 70,175,67,190,66,177 :REMark 26
 2564 DATA 52,Ccol,0,66,177,63,180,60,170 :REMark **Leave Arctic**
 2565 DATA 60,163,55,162,51,157,57,156,62,163
 2566 DATA 62,157,59,153,59,143,55,135,54,141
 2567 DATA 48,140,39,128,35,129,5,34,126,39,125,5
 2568 DATA 41,121,38,5,118,30,122,23,117,21,110
 2569 DATA 22,108,19,105,5,14,5,109,11,5,109,8,105
 2570 DATA 13,100,5,9,99,5,103,5,1,104,4,101
 2571 DATA 9,98,17,97,23,92,15,80,10,80
 2572 DATA 8,77,12,74,5,21,72,25,67,25,56
 2573 DATA 30,50,29,5,49,24,53,25,56,24,56
 2574 DATA 23,60,17,56,12,5,44,28,35 :REMark 104



2575 DATA 7,Acol,1,77,70,76,60,71,50,70,51,75,60,76,70,77,70:REMark **Novaja**

2576 DATA 7,Ccol,1 :REMark **Sri Lanka**
 2577 DATA 9,7,80,7,82,6,5,81,8,6,3,80,5,6,4,80,8,79,7,9,7,80

2578 DATA 74,Ccol,0 :REMark **AMERICA**
 2579 DATA 52,-56,50,-65,46,-64,43.7,-70.4
 2580 DATA 41.5,-70.7,40.6,-74,37,-76,35.2,-75.7,31,-81.6
 2581 DATA 27,-80,25,-80.5,28,-82.7,29,-82.5,30,-84
 2582 DATA 30.3,-89,29,-90,29.7,-94,27,-97.5,22,-97.7
 2583 DATA 19,-96,18.4,-94,19,-91,21,-90,21.6,-87 :REMark 50
 2584 DATA 16,-89,15.6,-83,10.5,-83.5,9,-81.5,9.7,-79
 2585 DATA 8,-77,11,-75,12,-71,10.6,-63,4,-52
 2586 DATA 0,-50,-6,-34,-12,-39,-22,-41,-25,-48
 2587 DATA -28,-48,-41,-63,-51,-69,-55,-65,-55,-70 :REMark 100
 2588 DATA -50,-76,-37,-74,-18,-70,-6,-81,0,-81
 2589 DATA 6.6,-77.5,9,-79.7,-81,9.5,-85,13,-88
 2590 DATA 14,-91.5,16.2,-95,15.7,-96.6,19.6,-106,22,-105.7
 2591 DATA 29,-112.4,31.3,-113,31.6,-115,30,-114.6,23,-109.5
 2592 DATA 25,-112.3,30,-115.9,34,-118.5,34.5,-120.7,39,-124
 2593 DATA 43,-124.5,48.5,-124.5,59,-138,61,-148,54,-165 :REMark 148



2594 DATA 11,Acol,0,54,-165,59,-158,62,-166,68,-167,71,-157 :REMark **Arctic**
 2595 DATA 68,-110,70,-82,60,-95,54,-80,63,-77,52,-56

2596 DATA 5,Acol,1,75,-105,73,-90,70,-105,73,-120,75,-105 :REMark **Victoria**

2597 DATA 5,Acol,1,83,-45,81,-120,78,-105,81,-75,83,-45 :REMark **Elizabeth**

2598 DATA 6,Acol,1,78,-75,67,-60,60,-60,64,-75,75,-90,78,-75 :REMark **Baffin**

2599 DATA 12,Acol,1,60,-44,65,-40,70,-22,82 :REMark **Greenland**

2600 DATA -15,83.6,-30,78.5,-73,76,-68,75.6,-59,70

2601 DATA -51,66,-53.5,61,-48,60,-44

2602 DATA 15,Acol,0 :REMark **Arctic Ice sheet**

2603 DATA 77,-114,73,-124,74,-132,76,-130,79,-160

2604 DATA 76,-170,74,176,78,160,83,140,81,110

2605 DATA 82,70,84,30,82,10,76,-10,74,-18 :REMark 30

2606 DATA 18,Ccol,1 :REMark **Caribbean**

2607 DATA 22,-85,23,-83,23,-80.5,20,-74,20,-70

2608 DATA 18.5,-68,18.5,-71,17.5,-71.5,18,-72,18.5,-74.5

2609 DATA 19,-74.5,19,-72.5,20,-74,20,-77.5,20.5,-77

2610 DATA 22.5,-81.5,22,-84,22,-85 :REMark 36

2611 DATA 5,Ccol,1,18.2,-78.2,18.4,-78,18,-76.2,17.9,-77.8,18.2,-78.2

2612 DATA 5,Ccol,1,18.5,-67,18.5,-65.5,18,-65,18,-67,18.5,-67

2613 DATA 22,Ccol,1 :REMark **Japan**

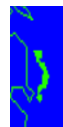
2614 DATA 45.5,141.8,43.3,145.7,42,143,42.6,141.6,40.6,140

2615 DATA 38.2,139.6,37,136.9,35.6,135.7,35.6,133,34,130.9

2616 DATA 32.9,132,31.4,131.3,31.2,130.2,33.3,129.7,34,130.9

2617 DATA 34.5,135,33.5,135.7,36,140.6,39.8,142,42.5,139.7

2618 DATA 43.5,141.4,45.5,141.8 :REMark 44



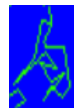
2619 DATA 5,Ccol,1 :REMark **Taiwan**

2620 DATA 25.5,121.5,23.5,120,22,121,25,122,25.5,121.5 :REMark 10

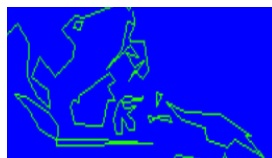
2621 DATA 6,Ccol,1 :REMark **Hainan**

2622 DATA 20,108.6,20,110.3,19.8,110.3,18.3,109.9,18.8,108,20,108.6

2623 DATA 19,Ccol,1 :REMark **Philippines**
 2624 DATA 21,122,18,122.5,16.5,122.5,15,121.5,14,122
 2625 DATA 13.5,125.7,126.5,125.7,123.5,122
 2626 DATA 9,125.8,123,11,121,10,124,13,122
 2627 DATA 8,117,12,120,18.5,121,18,122.5 :REMark 38



2628 DATA 11,Ccol,1 :REMark **Indonesia**
 2629 DATA 6.95,1.7,98.8,-3.2,101.6,-5.9,105.7,-6.6,114.2,-8.6,127
 2630 DATA -7.1,105.6,-2.9,105.9,4,103.6,5,97.5,6,95 :REMark 22



2631 DATA 4,Ccol,1,2,128,1.5,129,-1,128,2,128
 2632 DATA 6,Ccol,1,-3,126,-4,131,-3,130.5,-3,128,-4,126.5,-3,126

2633 DATA 13,Ccol,1 :REMark **Borneo**
 2634 DATA 7,117.5,2.5,111,1.5,111.2,109.5,1,109
 2635 DATA -3,110,-4,114.5,-4,116,1,117.5,1,119
 2636 DATA 4,117.5,5,119,7,117.5 :REMark 26



2637 DATA 17,Ccol,1 :REMark
 2638 DATA 1,125,1,124,1.5,121,0,119.5,-3,118.5
 2639 DATA -6,119,-6,120.5,-3,120.5,-5.5,122,-5.5,123
 2640 DATA -4,123,-2,121.5,-5,123.5,-1,121,5,120.5
 2641 DATA .5,124.5,2,125 :REMark 34

2642 DATA 12,Ccol,1 :REMark
 2643 DATA 0,130,-2,134,-2.5,141,-6.5,148,-6.8,146.8
 2644 DATA -10.7,151,-7.7,144.3,-9.3,143,-8,138.4,-5.4,138.1
 2645 DATA -4,133.1,0,130 :REMark 246

2646 DATA 34,Ccol,1 :REMark **Australia**
 2647 DATA -10.5,142.4,-17.5,141,-15,135.5,-12,137,-11,132
 2648 DATA -15,129,-14,127,-17.5,122,-19,122,-20,120
 2649 DATA -22,114,-26,113,-32,116,-34.5,115,-35.2,118
 2650 DATA -31.5,130,-32.5,133.5,-35,135.5,-33,137.8,-35.2,137.5
 2651 DATA -38,140.4,-39,143.4,-37.8,145,-39.2,146,-37.5,150
 2652 DATA -34,151,-32.7,152.7,-29,153.6,-25.6,153,-20,148.4
 2653 DATA -18.8,146.3,-14.5,144.7,-14.7,144,-10.5,142.4 :REMark 68

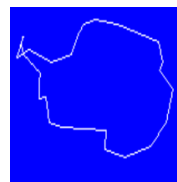


2654 DATA 4,Ccol,1,-42,144.9,-42,148,-44,146.5,-42,144.9 :REMark Tasmania

2655 DATA 14,Ccol,1 :REMark **New Zealand**
 2656 DATA -34.5,172.7,-36.7,175.9,-37.5,176,-38,177.3,-37.4
 2657 DATA 178.5,-41.6,175.5,-40.6,172.5,-42.8,171,-46,166.2,-46.7
 2658 DATA 169.4,-40.2,175.3,-39.3,174,-37.7,174.8,-34.5,172.7



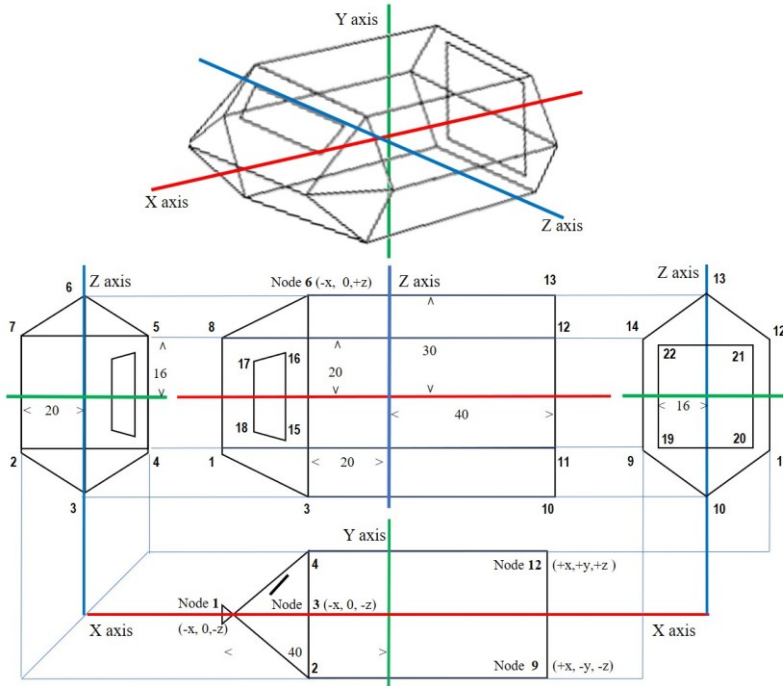
2659 DATA 29,Acol,1 :REMark **Antarctica**
 2660 DATA -63,-56,-64,-60,-66,-65,-73,-75,-73,-85
 2661 DATA -73,-100,-75,-100,-73,-125,-75,-137,-78,-165
 2662 DATA -77.6,164,-72,170,-68,155,-66,135,-66,115
 2663 DATA -66,90,-69.5,75,-68,70,-66.55,-69,40
 2664 DATA -70,20,-70,0,-71,-10,-74,-20,-78,-35
 2665 DATA -75,-60,-67,-61,-64.3,-69,-63,-55 :REMark 58
 2666 DATA 9999,0,0 :REMark End check



QBITS 3D Wireframe Design

To expand on the simple wireframe objects of Pyramid, Diamond, Cube, I include a simple Space Shuttle design. First the object is drawn schematically shown with front and side elevations. This is then Mapped to the XX YY ZZ Planes, the **Nodes** (xyz) values in relevant units of distance +/- values used to create the **Node** and **Frame** tables.

Here's the Layouts for the Space Shuttle showing values in the **XX YY ZZ** planes.



Data List One is for each of the **Node xyz** values. **DATA List Two** is the Wireframe that links the Nodes to form **Frames**. These are **READ** and used by the Plane Equation of **Obj_Cull** to determine if the outward facing surface of the polygon is towards or away from Viewpoint of observation. It is therefore important they are ordered correctly, that is Counter or Anticlockwise to the Viewpoint if not to be seen.

Note:

Any created list for new Objects can be added to the **3D DATA Lists** following the Format presented in Program Lines 2000 onwards. The **RESTORE** pointers **nres**, **vres** need to be added to **Obj_Dat(n)** and start up positions to **Obj_Shape**. The number group will need to be extended ie 1-6 to -7-8-9 on Line 1211 of **Menu_3DCommnad**.

