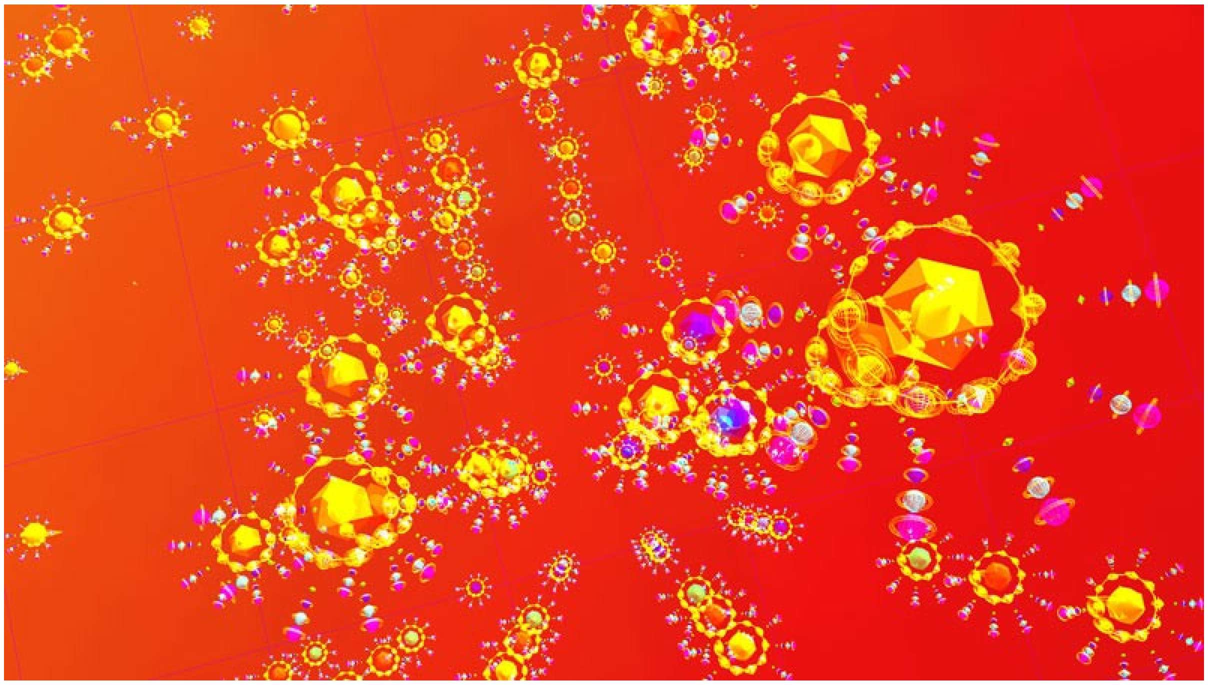
Antz User Manual

2012, January 23rd

by: Shane Saxon



Acknowledgements

Synergist:
Dave Warner

Coders:
Shane Saxon
Lucas Erickson
Mark Buchanan

Data Applications:
Allan B Scrivner
Jeff Sale

Table of Contents

Section 1 - Introduction
1.1   Quick Start
1.2   What is Antz?
1.3   Developer

Section 2 - Installation

2.1   MSW
2.1.1   Windows XP and 2003
2.1.2   Windows 7 (32 bit)
2.1.3   Windows 7 x64 (64 bit)
2.2   OSX
2.3   Linux

***Section 3 - Input & Output**

3.1   Keyboard & Mouse
3.2   Joystick & 3D Controllers
3.3   Command Line
3.4   File
3.5   *Database
3.6   *Live Channels
3.6.1   *TNG3B
3.6.2   *Audio
3.6.3   *Video
3.6.4   *Network
3.7   Display Systems
3.7.1   *3D Stereoscopic
3.7.2   *Cluster
3.7.3   *zSpace - infinitez

Section 4 - Graphing

4.1   Topology & Coordinates
4.1.1   Default Topo - Mixed Tree
4.1.2   Cube Topo - Cube Tree
4.1.3   Sphere Topo - Polar Tree
4.1.4   Torus Topo - Orbital Tree
4.1.5   Pin Topo - Stacked Tree
4.1.6   *nD Tree, Fractal Tree...
4.2   Geometry
4.2.1   Primitives
4.2.2   Color & Texture Maps
4.2.3   *Points & Lines
4.2.4   *Surface - FFT
4.2.5   *3D Model
4.3   Lighting
4.4   Cameras
4.5   Animation & Physics

***Section 5 - *HUD**

5.1   *Compass & Axes
5.2   *Console
5.3   *Labels
5.4   *Legend

Section 6 - Commands

6.1 Mouse
6.1.1 Selection
6.1.2 Navigation
6.1.3 Object Manipulation
6.2 Keyboard
6.2.1 Fullscreen
6.2.2 Active Axes
6.2.3 Creating & Deleting Objects
6.2.4 Selection
6.2.5 Position & Orientation
6.2.6 Scale
6.2.7 Animation
6.2.8 Topology & Facets
6.2.9 Geometry
6.2.10 Ratio - Torus Inner Radius
6.2.11 Grid Segments
6.2.12 Color & Transparency
6.2.13 Texture Maps
6.2.14 Hide
6.2.15 Freeze
6.2.16 Set Points
6.2.17 Scene Presets
6.2.18 State File
6.2.19 Channel File
6.2.20 Channel Assignment
6.2.21 URL recordID Retrieval
6.3 Command Line

Section 7 - File Types

7.1 State - .CSV
7.2 Channel - .CSV
7.3 *Event Log - .CSV
7.4 Texture Map - .JPG, .TGA...
7.5 *3D Model - .3DS

***Section 8 - *DataBase**

8.1 *MySQL Tables
8.2 *Storing Scene State
8.3 *Retrieving Scene State
8.4 *Labels and Conversion Factors
8.5 *External Files - Video, Audio, Channel Data

Appendix A - Command Cheat Sheet

A.1   Mouse
A.2   Keyboard

Appendix B - State File - .CSV

B.1   Overview
B.1.1 Tree Hierarchy
B.1.2 Position, Topology & Coordinates
B.1.3 Texture Maps & Color
B.2   Field Description Table
B.3   Key Tables
B.3.1   type (node type)
B.3.2   topo
B.3.3   geometry
B.3.4   cube_facet_map
B.3.5   shader
B.3.6   index_color

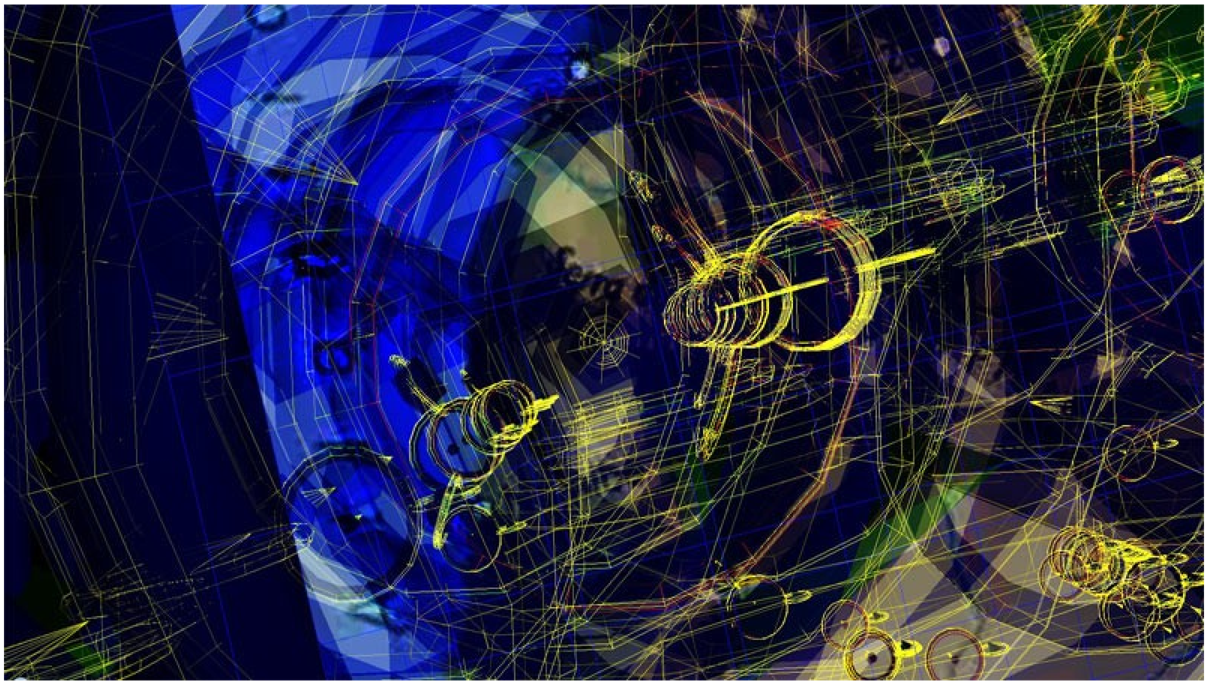
Appendix C - Channel File - .CSV

***Appendix D - Database Tables**

Appendix E - Glossary

*indicates not yet implemented, but planned for future version.

Section 1   Introduction



1.1   Quick Start

If you just want to get started, read the installation instructions in the next section and then jump to Appendix A - 'Command Cheat Sheet'.

Use 'N' to create new objects and the mouse to move objects or navigate the scene. Click-hold on the background with the Left, Right or BOTH buttons for different navigation modes.

1.2   What is Antz?

Antz is a 3D data viewer for interaction with complex datasets. Spatial cues based on shape, color, size and texture enable the user to identify key patterns.

Think of it as 'spreadsheet meets cyberspace.' An immersive multi-dimensional spatially based environment for realtime analysis and decision making. The visual environment enables a more efficient mechanism for identifying patterns and relationships. Capable of combining stored data with live sources for time sensitive situations. It is good for finding the needle in the haystack.

For example, you may have a table of data representing academic test scores displayed as a series of objects distributed over a map. Data fields are mapped to parameters such as geometry, transparency, velocity and position. At each location a pin with tori would represent components of the students demographics and scores over a period of time.

The ultimate goal is to close the cognitive loop between users and powerful logic engines known as computers.

Antz is an open-source project that is free to the world for any purpose, free as in free. Source code and compiled apps are provided for all major desktop platforms, including Linux, MSW and OSX.

The user can move and otherwise interact with the objects. Physics such as angular velocity and multiple tracks of time based channel data allow for animating the scene.

*Realtime IO channels can work with a host of inputs such as audio, video, GPS or EEG brainwave sensors. The system is well suited for real-time machine control of robotic systems and scientific instruments where low-latency physics based feedback algorithms are necessary to interact in the real world.

1.3   Developer

*'Developer Guide' is under construction, to download the source code and development installation instructions. goto:

http://temporalzone.com/code/docs/antz_install.txt

If you would like to contribute or have questions, feel free to get in touch via the contact page on the website. The project is hosted at:

http://temporalzone.com

Section 2   Installation

Application installation is specific for each operating system. In general, no installer package is used, but rather just copy the application with sub-directories and manually install any required system files as specified below. Currently the MSW version is most up-to-date with the OSX and Linux versions being updated later.

Optional sub-directories include sample CSV state files and texture maps.
maps/
*data/

2.1   MSW

The freeglut.dll and glut32.dll must be copied to the appropriate system folder, (location depends on the specific flavor of MSW.) In addition, Windows XP requires the 2008 version of the Visual C redistributable to be installed, unless you have already done so, such as when you have already installed VS2008. Windows 7 does not require this step.

Note that we have not tested the application under Vista nor versions older then XP. Win98 may work using the XP instructions, Vista is likely the same as W7 (64bit.)

2.1.1 Windows XP (32 Pro) and Windows 2003 (Enterprise R2 SP1)

```
freeglut.dll -> Windows\System32\
glut32.dll -> Windows\System32\


run vcredist_x86_2008.exe

run the app!
```

2.1.2 Windows 7 (32 bit)

```
freeglut.dll -> Windows\System32\
glut32.dll -> Windows\System32\
```

run the app!

2.1.3 Windows 7 x64 (64 bit)

```
freeglut.dll -> Windows\SysWOW64\
glut32.dll -> Windows\SysWOW64\
```

run the app!

2.2   OSX

Tested to work with 10.6 and 10.7
(likely works with 10.5 and possibly 10.4)

Uses the built in GLUT library by Apple, no installation required.

Simply download and run the application!

2.3   Linux

Tested to work with CentOS 5.5 (32-bit), Ubuntu (32-bit)

Note that there is a make file for compiling.

Prerequisites:

- Internet Connection

Step 1 (install freeglut libraries)

- open terminal
- sudo yum install freeglut-devel

Step 2

- Run Application
-

***Section 3 Input & Output Overview**

*Significant updates are planned for handling multiple types of real-time IO.

3.1   Keyboard & Mouse

The primary method of input is the keyboard and mouse. Generally all operations can be performed through use of the keyboard, where as the mouse provides an easier method for selecting and moving objects. For more detail see the 'Command' section of this document.

3.2 *Joystick & 3D Control Interfaces

*Auxiliary input devices such as joysticks or 3D mice allow for more intuitive navigation and object manipulation. Planned support for interfaces include:

HID library - Supports a variety of joysticks, mice, game controllers, etc.

*SpaceNavigator   - www.3dconnexion.com
A 3D mouse that offers a full 6 degrees of freedom with good sensitivity.
This allows for simultaneous x/y/z translation and rotation.

*zSpace - www.infinitez.com
A desktop VR system with 3D head tracking and stylus pen that can be used to manipulate objects in space.

3.3 Command Line

The command line will accept parameters upon launch such as CSV files to load and the URL to use for retrieving complete records based on ID.
See 'Command' section for detail.

3.4 File

The state of the entire scene can be saved and retrieved by the State File which is in CSV format. In addition, a separate Channels File is used for time based track data that can be played back to animate objects position, color and scale. For details see the 'File' section of this document.

3.5 *Database

*DB support is currently underway to support MySQL and is designed to be directly compatible with the applications CSV files that store the state and time based channel data.

3.6 *Live Channels

*Currently no live channels are supported, only file based channel data.
Future support for a variety of live input and output devices will add significant capability to combine real-time real-world data with stored data.
Also will allow for combining and analyzing multiple channel types and sources in a unified environment. Applications include realtime EEG, visual processing and machine control for operation of scientific instruments.

3.6.1 *TNG3B

*A simple RS232 based input device that provides 8 analog channels and 8 digital switches that stream at 2kHz sampling rate, can modulate parameters in realtime by assigning channels. TNG3B can connect to a myriad of inexpensive sensors that capture everything from EEG signals to data-gloves.

3.6.2 *Audio

*In addition to the built-in audio capability of most computers, a large variety of affordable audio IO solutions exist that offer high quality IO connections. The low-cost professional devices typically support 192kHz at

24bit with lots of channels (16,32,64....) Audio inputs can be configured to have a large variety of sensors (similar to TNG3B), and have very good low noise ADC and DAC circuits. Live capture can be analyzed and compared with other stored data sources.

3.6.3 *Video

*Video support will include multiple channels of un-compressed HD for realtime image processing utilizing CPU and GPU based algorithms. Low-latency will be achieved through direct support for hardware from Aja, Blackmagic and Matrox. QuickTime is being considered, but cross-platform compatibility and high-latency are both issues. File based video will be supported (perhaps with VLC) for film/video standards such as AVI, MOV, JPEG2000, DCP, DnX....

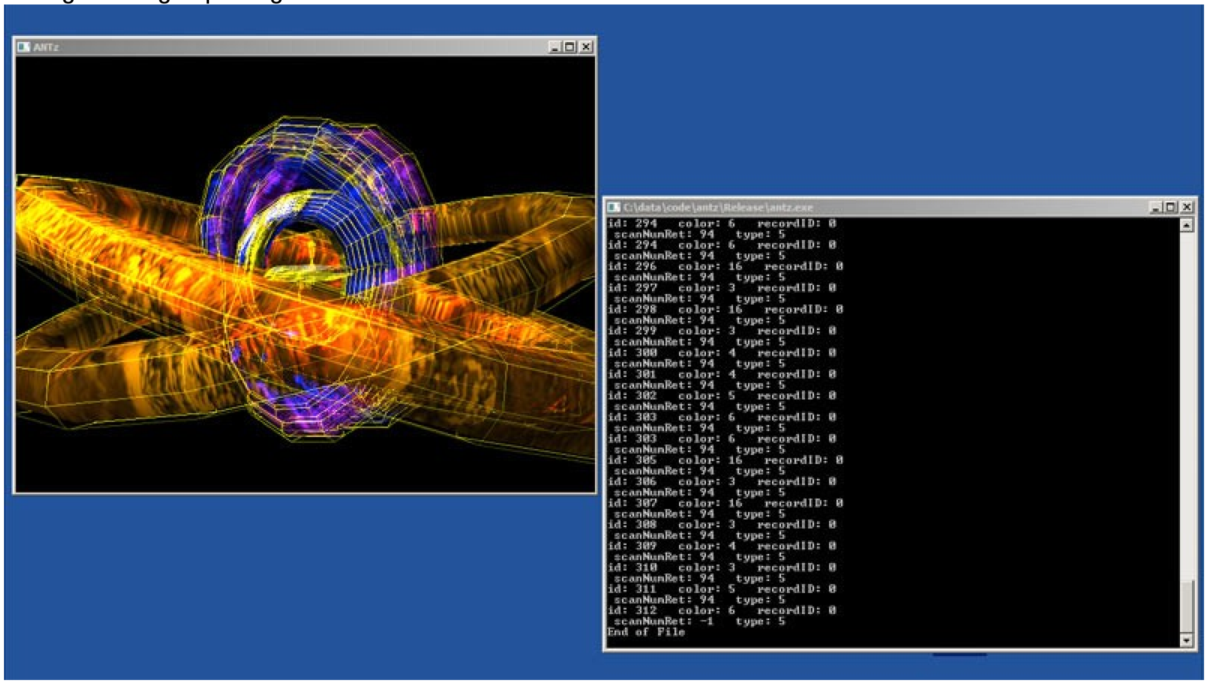
3.6.4 *Network

*Network support for clustering and storage with realtime data updates from external sources. Support for clustering allows the application to run across multiple systems and enables multi-user environments. Support for IP and UDP for streaming channel data. Use of SNMP is under consideration.

3.7   Displays

Fullscreen is entered at startup, press 'ESC' to exit and you will see the graping window and a console window. The console window outputs user command feedback and updates such as file loading information. The graphing window can be stretched across multiple displays. Fullscreen only supports single display.

**image of graphing and console windows.



```
id: 294   color: 6   recordID: 0
 scanNumRet: 94   type: 5
id: 294   color: 6   recordID: 0
 scanNumRet: 94   type: 5
id: 296   color: 16   recordID: 0
 scanNumRet: 94   type: 5
id: 297   color: 3   recordID: 0
 scanNumRet: 94   type: 5
id: 298   color: 16   recordID: 0
 scanNumRet: 94   type: 5
id: 299   color: 3   recordID: 0
 scanNumRet: 94   type: 5
id: 300   color: 4   recordID: 0
 scanNumRet: 94   type: 5
id: 301   color: 4   recordID: 0
 scanNumRet: 94   type: 5
id: 302   color: 5   recordID: 0
 scanNumRet: 94   type: 5
id: 303   color: 6   recordID: 0
 scanNumRet: 94   type: 5
id: 303   color: 6   recordID: 0
 scanNumRet: 94   type: 5
id: 305   color: 16   recordID: 0
 scanNumRet: 94   type: 5
id: 306   color: 3   recordID: 0
 scanNumRet: 94   type: 5
id: 307   color: 16   recordID: 0
 scanNumRet: 94   type: 5
id: 308   color: 3   recordID: 0
 scanNumRet: 94   type: 5
id: 309   color: 4   recordID: 0
 scanNumRet: 94   type: 5
id: 310   color: 3   recordID: 0
 scanNumRet: 94   type: 5
id: 311   color: 5   recordID: 0
 scanNumRet: 94   type: 5
id: 312   color: 6   recordID: 0
 scanNumRet: -1   type: 5
End of File
```

*Planned support for multiple windows and multiple displays at fullscreen. Also allow various viewport configurations and support for tiled stereoscopic 3D environments.

3.7.1 *3D Stereoscopic

*The entire scene is in 3D which allows for integration of stereoscopic displays. Potentially a large variety of display types may be supported from consumer laptops and TVs to custom solutions such as the zSpace or even a full cave environment.

3.7.2 *Cluster

*Currently the application is based on GLUT using OpenGL and includes a Linux version tested on CentOS. This means that in theory it should be relatively easy to create a version for the Star Cave at calit2 using CGLX. We also plan to use the equalizer library for cave environments.

3.7.3 *zSpace - inifinitez

*Currently in development to support the zSpace system by inifintez.com, a 3D stereoscopic display with head tracking and 3D stylus input.

Section 4: Graphing

The scene is comprised of 3D objects that represent data based on spatial parameters such as position, color, size and geometry type. The objects can be positioned anywhere in the scene and be represented by a single primitive or a collection of different objects in a tree topology. In addition to objects, the scene is made up of multiple cameras, grids, lights and the HUD.

The primitives and grid can be set to a specific color as well as texture mapped with any image such as a map or abstract pattern for additional dimensional data cues. The object properties can be animated or changed with time based on velocity rates (physics) and channel assignments. Channels may either be streamed from file or other IO devices. All of the parameters serve as spatial cues. So green may represent good, while red is bad. A large object might indicate a particular property, such as a high flow rate of a water source or lots of money in a fund, while a particular geometry type may be used to represent an ethnic group, age bracket or whatever the user defines.

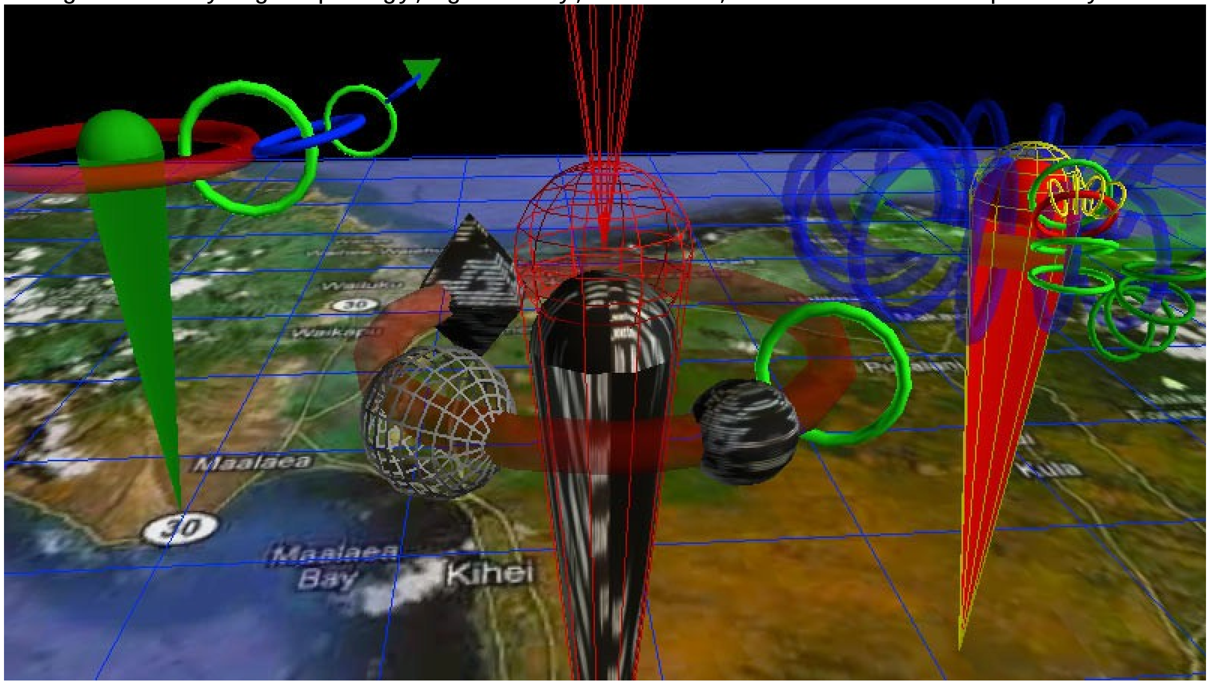
*HUD is under development. The HUD is the 2D overlay on top of the 3D scene used to display text and indicators that display the object labels, compass and legend. Some HUD objects (such as labels) track the position of 3D objects.

*Currently channels can only be streamed from file, future support for audio, video and other IO devices (EEG) will allow for combining real-time real-world data with that of files.

4.1 Topology & Coordinates

Each topology or 'topo' type is defined by a unique set of coordinate systems (basis.) Multiple topologies are supported that include the cube, sphere, torus, pin, and grid. A node can be of any topo type and the various types may be mixed within a tree hierarchy.

**image of varying topology, geometry, texture, color and transparency.



The scene is composed of a collection of root-pins which may have any number of child-nodes attached in a tree topology. Root pins are independent of each other and can be placed anywhere in the scene. The child-nodes are attached to the parent branch in the tree relative to the parents topo type, position, orientation and scale. The color, scale, geometry and other visual cues are used to represent different parameters of the data set being visualized.

The coordinate system of a node is based on both the nodes topo type and its parents topo type. For example a torus attached to a torus is positioned differently then a pin attached to a torus. However, for most topologies it is the parent topo type that determines the child-nodes coordinate system. Any topo type may be assigned to a node and types can be mixed throughout a tree structure.

In general, child nodes are typically scaled down by 25% or 50% relative to there parent. The main exception is children of pins which are not scaled down. The scale of any node may be changed.

When a parent node changes topo types all of its children are effectively re-mapped into the new coordinates. However the nodes underlying coordinates remain the same, only the graphed position appears differently. The range of different coordinate systems vary, so some re-mapping will wrap the new coordinates. For example an object placed on inside of a torus would have a 'translate_y' (latitude) that is in excess of +/-90 limit of a standard polar system for a sphere. The result is that the node is wrapped around the pole of the sphere and will appear upside down on the backside of the sphere.

order	field_name	kml_name	kml_lo	kml_hi	torus_lo	torus_hi
1	translate_x	longitude	-180	180	-180	180
2	translate_y	latitude	-90	90	-180	180
3	translate_z	altitude	0	none	none	none
4	rotate_y	heading	0	360	0	360
5	rotate_x	tilt	0	180	0	180
6	rotate_z	roll	-180	180	-180	180

*note that the only difference in kml vs torus is the latitude range.

Most topologies use the torus limits and allow for latitude of +/- 180 deg. This works well for the a cube which is square or a torus which has an outer side similar to the sphere but with an inner side for which there is no obvious analogy. So any point on a torus can be mapped to a point on the face of a cube and vice versa. However, the range of a sphere only represents half the latitude of most other topo types, hence the sphere wrapping method.

4.1.1 Default Topo - Mixed Tree

The 'default' coordinate system varies depending on the node type and branch level. Root-nodes, cameras and grids default to a Left-Handed Cartesian Coordinate System based on world coordinates. The primary grid is locked to the World Origin (except for scale.) Torus topo is the default for child-nodes of a parent that is topo type default, pin or torus.

4.1.2 Cube Topo - Cubic Tree

The 'cube' topology consists of six independent coordinate systems, one for each facet. Child-nodes are positioned relative to the center of a particular facet and use a Left-Handed Cartesian Coordinate System. Each node has a 'facet' number which keeps track of which side of the parent cube it belongs to. When the node position exceeds the boundaries of a facet it is then automatically re-mapped to the next corresponding facet.

A local facet coordinate system is centered at (0,0). The lower left corner (-X,-Y) is at negative (-180.0, -180.0) deg longitude and latitude, the upper right corner (+X,+Y) is at positive (+180.0, +180.0) deg latitude and longitude. Altitude (+Z) is relative to the cube face and units are the same as the x and y axis, (rather then feet or meters.)

The facet numbers with orientation are compliant with google KML, see the 'cube_facet_map' in the Appendix.

4.1.3 Sphere Topo - Polar Tree

The 'sphere' topology uses polar coordinates that are compatible with the google KML standard. However, the altitude (translate_z) unit length is not the same, the conversion is :

altitude = translate_z / (earth diameter * 4 * (360/2pi)
or
translate_z = altitude(feet) * 2.00875x10^-10

Longitude, latitude and orientation follow the KML spec for '<Model>' at:

http://code.google.com/apis/kml/documentation/kmlreference.html

----
4.1.4 Torus Topo - Orbital Tree

The 'torus' topology creates an Orbital Tree, similar to the way planets and moons orbit in our solar system, but with an orthogonal twist. That is each sub-orbit is orthogonal to its parents orbit. Unlike the Moon around the Earth and Earth around the Sun which are all roughly in the same plane. The torus orbital tree is similar, except that the moon would be orbiting around the poles instead of the equator.

Multiple children at the same level will have the same orbit orientation, but may have different positions and radius from the parent. The depth of the tree is not limited, (computer resources do impose a practical limit.)

*Future feature - additional topologies to include Polyhedrons, Fractal Tree and nD Tree. nD Tree has an arbitrary number of branches and can be used to represent a variety of dimensional basis. Currently the Sphere topo type can be used to build an nD tree, however it does not impose any special rules which define the particular basis.

4.1.5 Pin Topo - Stacked Tree

The 'pin' topology stacks objects vertically. The root-pins are 5 units high (z in world coordinates,) from the tip to the top, (center of ice-cream sphere.)

Use translate_x to changes height. A value of -180 will place a child object at the tip of the pin, zero is level with the ice-cream sphere and 0-180 will be above the pin.

Offset is 'translate_z' a value of 180 will offset a torus so that its ring intersects the pin.

4.1.6 *nD Tree, Fractal Tree

The 'n' topology is defined by the user. The coordinate system will default to an n-sphere basis where the user specifies 'n' dimensions. Methods are yet to be implemented and help from someone that has a strong background in related mathematics, computer science and/or spatial reasoning would be appreciated. The core concept is to support any type of space with an n-dimensional Neural Physics Engine. The premise is that we do not know how many dimensions exist nor do we know how they operate.

Fractal Tree is a type of nD Tree where the coordinates, distribution and other properties are defined using fractal or chaos based function.

4.2   Geometry

*A variety of geometric types are supported. These range from basic primitives and line graphs to complex fractal geometries, FFTs and custom 3D models. The line graphs can serve as a traditional plot or operate in a realtime oscilloscope fashion.

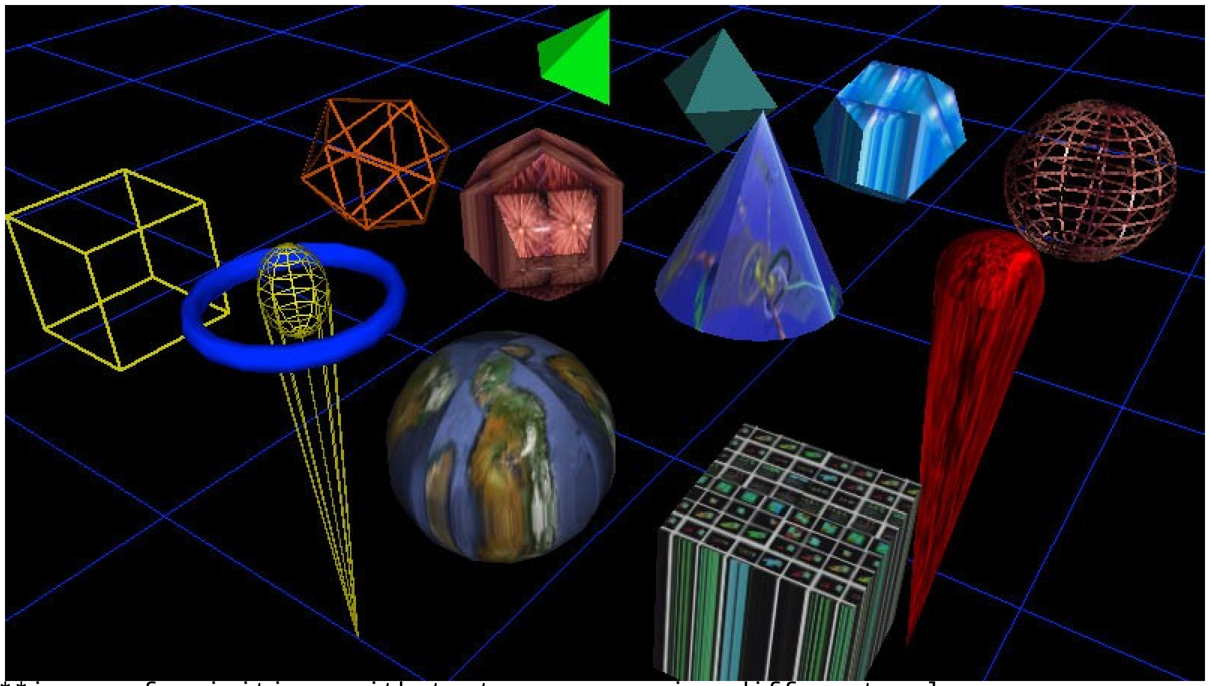


**image of primitives with texture maps using different colors.

4.2.1 Primitives

The default root pin type looks like an ice-cream cone and typically has at least one torus (ring shape) around it. The default child geometry is a torus. However, both the root-pin and its children can be represented by any number of geometric primitives such as a cone, cube, tetrahedron, sphere, etc... Primitives can be solid or wireframe. For a complete list of primitives please see the 'Geometry Table' in the appendix.

4.2.2 Color & Texture Maps

Optionally, a texture map image can be applied which is effected by the color assigned (including transparency.) For a complete list of geometric primitives see the appendix.

4.2.3 *Points & Lines

*Traditional looking graphs can be generated using Channels to plot lines and/or a series of points. However, these graphs are in 3D, though you can ignore the Z channel to make a 2D plot. Points and Lines graphs are capable of changing with time to create an oscilloscope type graph.

4.2.4 *Surface, FFT

*The surface object is similar to a Points & Lines graph in that it uses channels to map onto the surface of an object. A common example is an FFT surface. Other surface types include a sphere, torus and cylinder. These graphs are generally designed to be updated with time.

4.2.5 *Custom 3D Model

*Support for 3DS models will be added to allow for any type of geometry to be loaded to be used to represent a root pin or child node.

4.3 Grid

There are two types of grids the primary grid and secondary grids. In addition to the grid lines, a texture map may be applied to the grid.

The primary grid is locked to the origin and cannot be translated or rotated. This provides a fixed global reference point. The primary grid has a default texture map file loaded at application startup, "maps/map00001.jpg" which can be re-assigned at run-time in the same manner as other objects. See 'Commands – Texture Maps' section. The primary grid cannot be deleted but you can 'Hide' it and set the texture ID to zero and it will not be visible.

The grid parameters are modified in a similar manner as other objects and respond to changes in color, hide, scale and texture map assignment. There are some difference in behavior from other objects. Only the grid lines are effected by color and lighting, the texture map is always drawn in full white. Hide will turn off the grid lines, but not the texture map. To hide the texture you must assign it to texture zero which prevents any texture from being drawn. The scale and grid segment commands are specific to the active axes. Vertical stacks are added and subtracted with the grid segment command when the z axis is active, (see 'Command – Active Axes' section.)

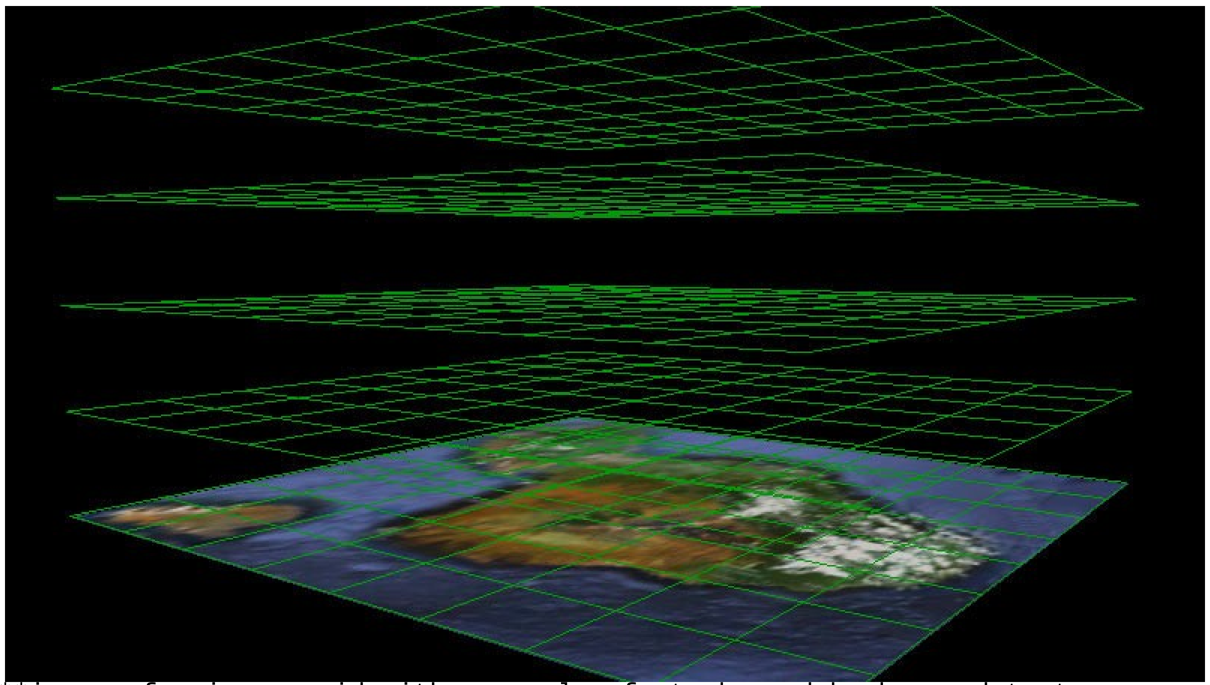


**image of primary grid with a couple of stacks and background texture.

4.4 Lighting

The scene has a pair of white lights designed to provide good front and back shading of objects. They are located in world coordinates at x: -1500, y: 1500, z: -1500 and x: 1500, y: -1500, z: 1500. Standing at the world origin, (center of the grid,) one light is over you right shoulder behind and the other is in front of you down to the left. Generally this will light most objects fairly well, though note that objects moved far from the origin may appear to have different shading.

*Planned update to allow user to specify the number of lights, position and color.

4.5 Cameras

The camera can be flown over the scene using either the keyboard or the mouse. The mouse has several distinct camera translation modes and the keyboard works in a standard flight mode. Also, the camera will auto-center actively chosen objects. By default there are 5 cameras in the scene which with positions stored in the State File. You may create or destroy child cameras, the root-camera is locked and cannot be delted.

Clipping planes are required by OpenGL for efficient 3D rendering. They define the volume in which objects are rendered, any objects outside this volume are clipped and therefore not drawn. You commonly observe this in 3D video games as the point at which distance objects disappear. The default near clipping plane has been set to 1 and far is set to a 1000. So objects more then a 1000 units from the camera (in world coordinates) are clipped, similar for anything closer then 1 unit from the camera.

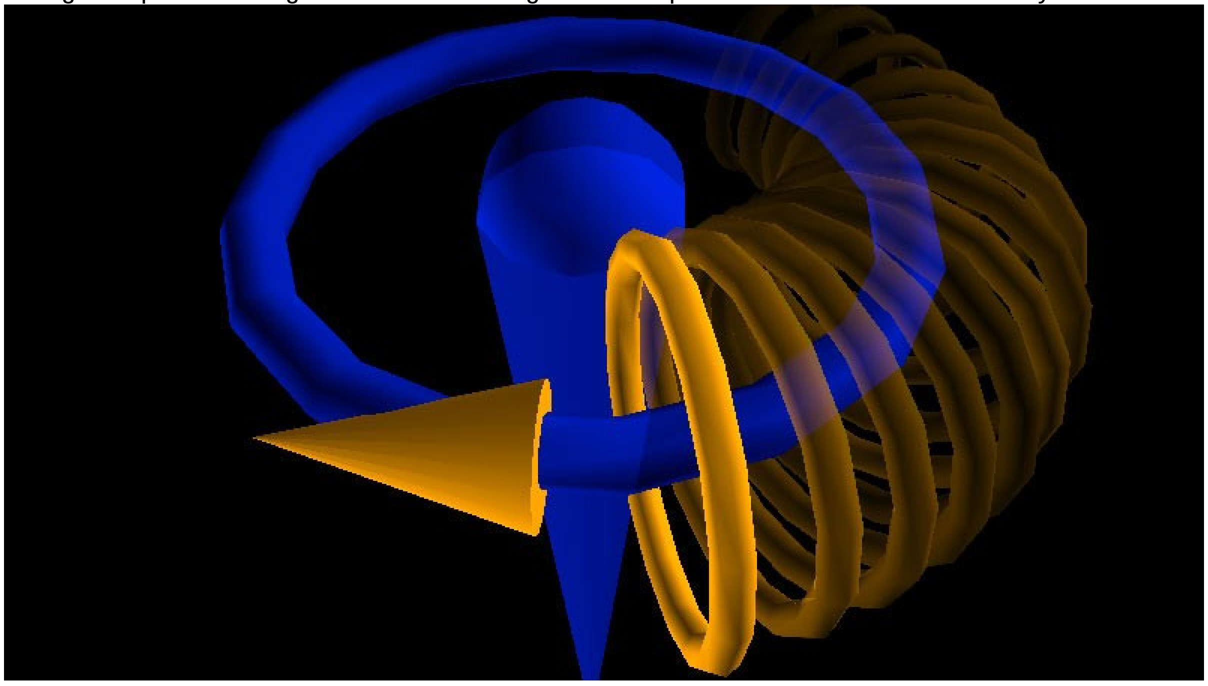
*Planned update will allow the near and far clip parameters to be set by the user.

4.6 Animation

Objects have velocity rates associated with them. This can be used to make objects rotate or orbit around their parent, or simply move across the screen. The other method for animation is using Channels that are keyed on a frame by frame basis. The Channels may be assigned to parameters such as position, rotation, scale and color offset, (see 'Commands – Channel Assignment' section.)

*Physics is currently limited to basic velocity rates, including rotation... significant enhancements are planned.... ranging from Newtonian mechanics to relativistic flight sim, QED and nD user defined basis.

**image representing torus rotating around parent based on velocity.



Section 5: *HUD

The Heads Up Display (HUD) provides feedback of the current state of the scene and command status. In general, it is a 2D overlay on top of the 3D scene which contains indicators such as text labels, compass and console output.

*The HUD console will implement a LISP interpreter or equivalent.

5.1   *Compass & Axes

The compass displays the position and orientation of the currently active object. If the camera is selected, it will display the altitude and coordinates of the camera, along with the direction (bearing) it is facing and vertical angle. Units are typically in degrees and global units, however it is possible to switch the units to custom types with offsets. If a pin is selected the coordinates of the pin will be displayed with the altitude and orientation. Indicators will include the active axes state.

5.2   *Console

The console serves as the primary message output for the user. Most commands return a result that is sent to the console to give the user feedback on the action taken and the resulting change of state.

5.3   *Labels

Pins can have a label with up to 3 lines associated with them. For example a name with latitude and longitude could be displayed. The 3 line label can display specific units and properties defined by the table of conversion factors. Please see the File Section of this document for information on how to setup unit conversion factors.

5.4   *Legend

A variety of icons are provided to aid in creating the legend. Icons are designed to refer to parameters such as scale, color range, geometry type, texture and orientation. The legend parameters are part of the conversion table (see the File Section of this document for additional info.)

Section 6: Commands

The primary method for control is through the keyboard and mouse. Additionally, parameters may be passed in from the command line upon application launch.

This section covers each command in detail, for a simple cheat sheet please see the appendix.

The mouse allows for navigating the scene, selecting items, moving objects and scaling child-nodes. The keyboard can do all the functions the mouse is capable of plus additional commands that change both object parameters and global settings. The keyboard responds to multiple simultaneous key presses and may be used simultaneously with the mouse. So you can do a rotation, scale and change color all at the same time while using the mouse to position the selected objects. The mouse can be a lot easier for navigating, selecting and positioning objects.

The typical limit for the maximum simultaneous key presses is usually 2-5 standard keys, plus the modifiers (SHIFT, CTRL, ALT, COMMAND.) The number of simultaneous keys supported depends on which specific keys are being pressed and the keyboard model, (some high-end keyboards feature N-key rollover (NKRO) which do not have this limitation.)

Objects in a selection set appear with a yellow wireframe around them. Note that in addition to selected objects, there is also the active object, which has a red wireframe around it. Most commands will operate on all the selected nodes at once when one of the selected items is also the active node. If the active node is not part of the selection set then only the active node will be effected.

Some commands ignore the selection set and only act on the active object regardless of selection states. Such commands include creating new pins and deleting them.

Some commands are axis specific and depend on the currently active axes ('X' key changes the active axes.) This allows for operations such as non-uniform scaling, setting individual limits and channel assignments.

The 'SHIFT' key reverses the action of several commands like scaling ('Z' key) and changes the speed of translation and rotation when using the keyboard.

A useful animation 'bug' is to change the selection to another object while performing a rotation, the original object will continue to rotate after you release the rotation key (arrows). To do this, while rotating, click with the mouse on another object, or any selection command that changes the active node; press TAB, Select Grid, Select Camera, etc... to initiate the animation state. Also applies to zoom and translate, though an object that is continuously scaling or translating will typically become unwieldily.

*Feature update for the animation 'bug' will provide a method for setting the motion without the need to select a different object.

Note that the 'number keys' apply to the number row across the top of the keyboard and NOT the number-pad.

6.1 Mouse

Left-Click
Depends on whether the node being picked is already selected. Selects a single node when clicking on a node that is NOT already selected. If the node IS already selected then will allow dragging all selected nodes. All other nodes are un-selected. Clicking on the background unselects all.

Right-Click
Toggles the selection status of the node picked, picking an already selected node will un-select it. You can select multiple objects by clicking on them.

The mouse behaves differently depending on whether an object is clicked or the background is clicked. Also the combination of Left and Right buttons being pressed determines the mode.

Navigation is performed by clicking on the scene background, (the grid and its texture map are considered part of the background.)

6.1.1 Selection - Mouse

L - Click on an object NOT selected will select it and unselect all others.
L - Click on an object that IS selected will result in no change.
R - Click on object toggles its selection state on/off.

Clicking on an object effects its selection state. The currently active object is drawn with a red wireframe around it. If it is part of the selection set then a yellow wireframe is drawn.

Left clicking on an object that is NOT selected will result in all objects being un-selected and only the picked object becoming selected. If the object is already selected then the selection set will remain the same. The keyboard may also be used to change the state of the active object.

6.1.2 Navigation - Mouse

R - Hold on background FLY's camera.
L - Hold on background orbits object in both axes of EXAMINER mode XY
L+R - Hold on background orbits and ZOOMS in and out Examiner mode XZ

Note that it is possible to switch directly between Examiner mode XZ and XY by releasing or pressing the right button while continuing to hold the left button.

6.1.3 Object Manipulation - Mouse

L - Hold on an object NOT selected will drag only the object, DRAGS in XY.
L - Hold on an object CURRENTLY selected will drag all, DRAGS in XY.
R - Hold on an object DRAGS all selected objects in XZ plane (L-R & Up-Down).

Objects can be moved (translated), rotated and scaled using the mouse. Child nodes behave differently from root pins since they are fixed to there parent. The child can be rotated around it's parent and scaled. Scaling is subject to the active axes ('X' key).

The mouse will effect the entire selection set simultaneously, this allows you move groups of objects.

Object manipulation occurs when then the mouse button is held while dragging.

6.2   Keyboard

The following list of keyboard commands gives the name of the function followed by a dash '-' then the default key(s) that are assigned to the function. For example: 'New Node – N' means pressing the 'N' key will create a new node.

6.2.1 Fullscreen - 'ESC'

The application starts up in Fullscreen mode. Pressing 'ESC' will exit Fullscreen and run the app in a window. The console window also becomes visible.

*Currently there is no method to re-enter Fullscreen, likely to change.

6.2.2 Active Axes - 'X'

The active axes determines which axes commands will be applied to. The active axes can either be a single axis or a combination, the default is all three (XYZ). Each press iterates through XY, X, Y, Z and then back to XYZ. SHIFT-X will iterate in the reverse direction.

The only commands that currently use the Active Axes are scaling, limits and channel assignments.

*Some of the new commands planned in the future will also be axes specific.

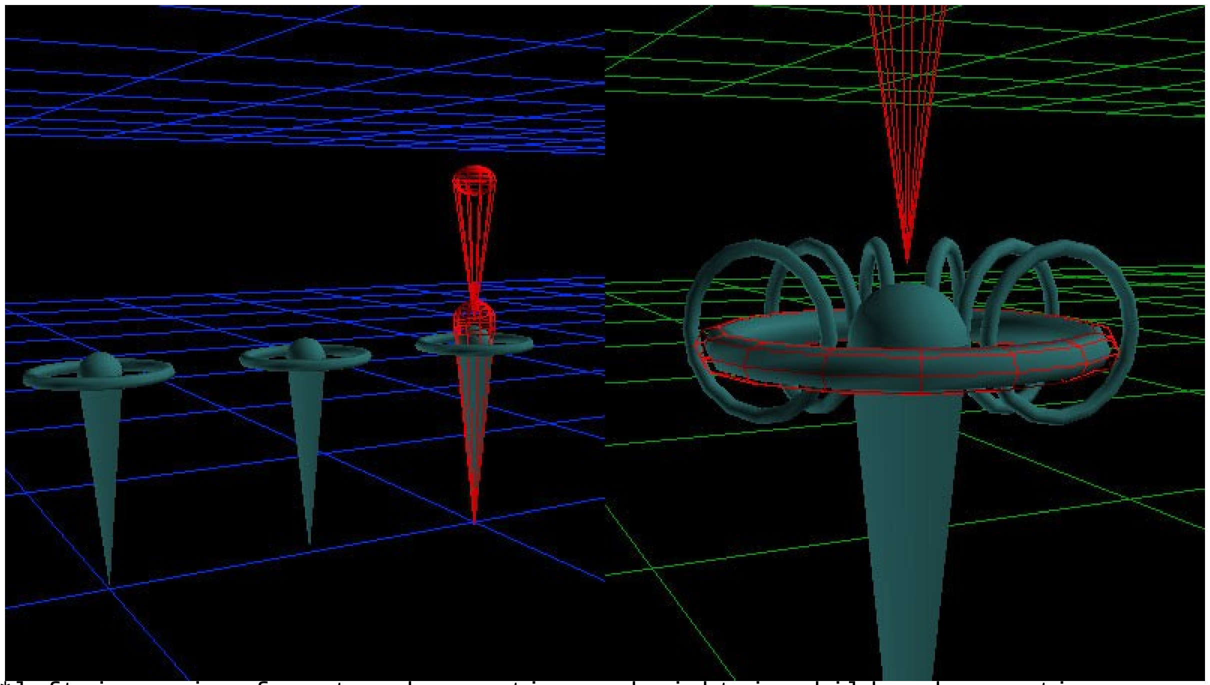
6.2.3 Creating & Deleting Objects - 'N' 'DEL'

Objects can be individually added or deleted from the scene using keyboard commands. (There are also presets that load sets of objects, see the 'Scene Presets' section.)

New Node - 'N'
'N' key creates a new object, if a root-pin is currently selected it will create another root-pin, the newly created root-pin will be made active. If a child-node is currently selected it will create another child-node attached to the active parent-node. The original child-node will remain selected for easy creation of additional children at the same branch level.

The default geometry for a root-pin is an ice-cream cone looking shape with a single torus around it. The default for child-nodes is a torus. In order to create a child-node you must select the parent you want it attached to. Additional child-nodes are created at the same level spaced around the parent.

Pressing SHIFT-N with a root-pin selected will create additional nodes at the first (primary) branch level. If a grid or camera is selected it will create sub-children, otherwise new cameras and grids are always attached directly to the root-camera or grid.

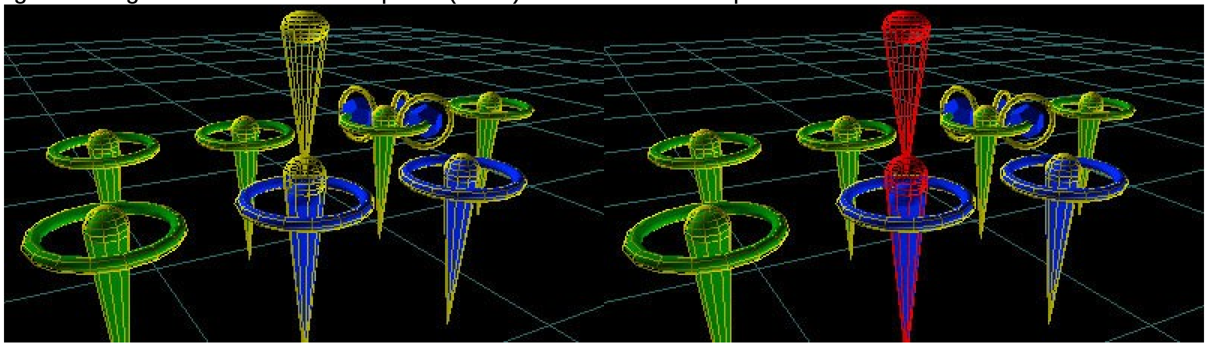

**left image is of root-node creation and right is child-node creation.

Delete Node - 'DEL'
'DEL' key deletes the active node and all children attached to it. Press and hold to delete multiple objects. Applies only to the active node and not the entire selection set. The root-camera and root-grid are locked and cannot be deleted

*May update delete to effect entire selection set.

6.2.4 Selection - Keyboard

If the active node is NOT part of the selection set (red wireframe) then the command will apply only to the active node. If the node is part of the selection set (yellow wireframe) then the command will apply to all of the selection set.

**Left image is an active object (yellow) that IS part of the selection set. Right image is an active pin (red) that is NOT part of the selection set.


6.2.4.a Object Selection - Keyboard

Essentially there are two types of selection, the active node (drawn with a red wireframe around it) and items that are part of the selections set (drawn with yellow wireframes.) The purpose of the active selection is to allow keyboard traversal of the objects without effecting the selection set. This is needed to add or subtract items from the selection set.

If the item is active but not part of the selection set it will have a red wireframe around it and any commands will only apply to the active object. However if the active object is part of the selection set it will have both the red and yellow wireframe drawn, in which case all items in the set will be effected at once.

The keyboard can be used to change the actively selected node by traversing through the objects. In addition the camera and grid can be selected. Each root-pin is at the same level and are considered siblings.... A pin's tree can be traversed by selecting the child, parent or sibling of the current node.

6.2.4.b Select All - Keyboard - '4'

'4' key toggles selection of all nodes. '~' (tilda) un-selects all nodes. A yellow wireframe will appear around the selected objects.

6.2.4.c Choose Sibling - Keyboard - 'Tab'

If the grid or camera is currently selected then the previous node will be re-selected. If a node is already selected then the next sibling will be selected, (if it exists.) If a root-pin is currently selected then the next root-pin will be made active, (displayed with a wireframe above and around it.) If a child node is selected then the child's sibling will be selected, (if it exists.) If no siblings then nothing will happen. Pressing SHIFT at the same time will iterate through siblings in the reverse order.

Note that if you are unable to choose a sibling root-pin it is likely because the currently active object is a child and not the root-pin. Pressing SHIFT-Enter multiple times will eventually select the root-node of the pin, then you can change to other pins.

**insert image of 3 selected pins, with a root-pin active (red).

6.2.4.d Choose Child or Parent Node - Keyboard - 'Enter'

Selects the child node if it exists, otherwise nothing will happen. Pressing SHIFT at the same time will select the parent node, if the root-pin is currently selected then selecting the parent will do nothing. If the camera or grid is selected then nothing will happen.

6.2.4.e Select/Deselect - Keyboard - 'Spacebar'

Toggles the active node selection set status. If the node is not part of the selection set it will be added and a yellow wireframe drawn around the node. If already part of the selection set then it will be removed from the set and the yellow wireframe will be replaced with a red one to indicate the node is the active object but not part of the selection set.

6.2.4.f Camera Selection - Keyboard - 'C'

If the camera is not currently selected it will select the previously active camera. If the camera is already selected it will select the next camera. The default scene is created with 5 cameras to choose from. Each camera can be positioned separately and the position is stored in the State File.

SHIFT-C will reset the camera position to its default. Each camera has it's own default position.

6.2.4.g Grid Selection - Keyboard - 'G'

If the grid is not currently selected it will select the previously active grid. If the grid is already selected it will select the next grid. The default scene is created with a single default Root-Grid that is centered on the global origin of 0,0,0. Pressing 'Y' or 'SHIFT-Y' will change the total number of X, Y and Z segments that can be added or subtracted.

Note that the root-grid can be scaled but is not allowed to rotate or translate. This restriction is to maintain a defined global coordinate origin.

The secondary grids can be scaled, translated and rotated. Secondary grids may be created by selecting a grid and pressing 'N' for new. Generally all grids are relative to world coordinates, however it is possible to create sub-grids (grids attached to grids) by pressing SHIFT-N.

Scaling and changing the grid segment count is specific to the active axes. With the Z axes active, changing the segment count will add stack layers in 3D.

The primary grid has the default textureID = 1 assigned which corresponds with 'map00001.jpg', this can be changed by pressing 'T'. Note that pressing 'SHIFT-T' will set the textureID = 0 resulting in no texture, the grid lines will remain. If a texture is desired without grid lines, then press 'H' to hide the grid lines, the lines will be hidden, but texture will remain visible. Grid lines also respond to changes in color and transparency, the grid texture is effected by transparency but not color.

6.2.5 Translation and Rotation - (see below)

Translate:

W	Forward
S	Back
A	Left
D	Right
E	Up
Q	Down

Rotate with ARROW keys:

Left	rotate left	
Right	rotate right	
Up	rotate up	(effects rotation of child nodes only)
Down	rotate down	(effects rotation of child nodes only)

In addition to the mouse, objects and cameras can be positioned and rotated using the keyboard. You first must select the item(s) you wish to manipulate then you can use the keys to translate (WASD gaming standard) and rotate with the arrow keys.

Will apply to all selected objects at once. Press 'C' to select camera or TAB to select pins, also can use the mouse for selecting objects, (for more information see the 'Selection' section of this guide.)

The pins will move in world coordinates, where as the camera moves relative to the direction it is facing.

Torus child nodes can be offset from the parent ring using 'Q' and 'E' keys.

Pressing SHIFT at the same time will increase the speed of an operation.

6.2.6 Scale - 'Z'

Z	Scale objects up (SHIFT+Z to scale down)
Mouse	Scales child nodes only (mouse vertical axis)

Objects can be scaled up or down using the 'Z' key or the mouse. The mouse will scale a child object but not the root pin, (for more detail see the 'Mouse' section of this document.) Scaling effects the active axes only ('X' key), this allows for both uniform and non-uniform scaling. So if XYZ axes is active then scaling will be uniform or symmetric in all directions. If you scale a pin with only the XY axes active then the pin will get wide or narrow (SHIFT-Z) but stay the same height. If the only Z axis is active then the pin would become taller or shorter (SHIFT+Z) while staying the same width. The active axes effects the uniform or non-uniform scaling of both the root pin and child nodes.

*6.2.6.a inner & outer radius
**images showing 3 pins with non-uniform scaling

6.2.7 Animation

A useful 'bug' is to change selection to another object (TAB, New, etc..) while performing a rotation, the object will continue to rotate indefinitely. The same methods also apply to zoom and translate. This is due to the fact that rotations, translations and scale commands set a rate that updates the parameter each cycle. Normally releasing the key sets the rate back to zero. However, if a different object is selected while the key is still down, then the previously selected objects never receives the key-up command. This results in the rate being kept indefinitely.

For example, press '4' to select all, then press the up or left arrow to start rotating, then while holding the arrow key press 'G' to select the grid... now all the objects will continue to rotate, even after releasing the arrow key. To stop an object from rotating, re-select it and press the same key and release.

*The 'bug' that results in animation will be made into a feature providing a specific method to animate without changing selections.

6.2.8 Topology & Facets - 'J'

Iterates through the list of topo types and also changes the geometric primitive to match the topology. You may also change the geometry separately from the topo type. The default topo type for a root-pin is a Pin which stacks its child nodes vertically. Children of either Cube, Sphere or Grid topo types also default to Pins. However, the default topo type for a child of a Pin is a Torus, this is also true for child-nodes of Tori.

Each topo type has its own unique set of coordinate systems, this is what makes it a topology type.

SHIFT-J will change the 'facet' number, this applies to objects such as cubes to assign the node to a particular face.

6.2.9 Object Geometry - 'O' (alpha)

Iterates through the list of geometric primitives. The default pin is typically an ice cream cone and the default child nodes are tori. Primitives include both solid and wireframe objects. Standard objects include a cube, sphere, tetrahedron, torus, and several others, see the Appendix for a complete list. Changing object geometry does not effect the topo type. However, changing topo types will change the geometry type.

Pressing SHIFT iterates in the reverse order.

*Planned support for adding geometric primitives that include FFT surfaces and external 3D models.

**image with variety of primitives

6.2.10 Ratio - 'R'

Iterates through the list of geometric primitives. The default pin is typicall

**image with variety of tori with differing inner radius values.

6.2.11 Grid Segments - 'Y'

Changes the number of grid segments, both 2D and 3D stacks (layers) are effected by the active axes. If all three axes (XYZ) are active then will add grid segments in both directions on XY plane and add stacks in the Z direction. If you wish to add segments without effecting the number of stacks set the Active Axes to XY. If you want to add just stacks, set the Active Axes to Z. Similar for changing the individual segment count for X and Y directions independently. Using SHIFT will subtract segments.

**images showing different amounts of grid segments

6.2.12 Color & Transparency

- (minus)	previous color
+ (plus)	next color
9	more translucent (less opaque)
0 (zero)	less translucent (more opaque)

Global Settings:

8	Transparency mode (3 alpha modes + none)
B	Background between black and white
R	Rescale normals toggle

6.2.10.a Index Color - '-' '+'

Object color can be selected by index from a palette of 20 preset colors by pressing the next or previous keys.

**image with different colors

6.2.12.b Color Offset

The color offset shifts the hue by up to one component. The internal values range from 0.0f to 1.0f where a value of 0 is the primary color and 1 would be a full shift by one RGB component. For example a value of 1.0 will result in red becoming green, or green becoming blue, or blue becoming red. This value can be modulated using the Channels CSV file by assigning the Z channel of the object, (see 'Channel Assignment' section for detail.)

*Currently the Color Offset is accessible only with data loaded in from a CSV file. The State File retains the current color offset as well as the channel assignment which can be modulated from the Channels data.

6.2.12.c Transparency - '9' '0'

By default objects are 100% opaque. To make the selected objects transparent press '9' to decrease opacity. Pressing '0' (zero) will increase opacity (less translucent.) Note that there is also a global alpha mode for the entire scene that effects how transparency is calculated.

6.2.12.d Alpha Mode - '8'

Changes the transparency (alpha) mode of the entire scene. The default is a rather standard subtractive transparency. The other modes include a 'Dark', 'Additive' and 'None'. 'Dark' results in an overall darker looking scene that appears highly saturated. 'Additive' is akin to a flame or light beam where each transparent object adds color to whatever is behind it, when several objects stack up they appear brighter. This can be quite useful for data-sets where multiple data-points pile up. Instead of the objects being obscured they will appear to be a bright spot where the data-points/objects overlap. You can also select 'None' to turn transparency off.

Note that in 'Standard' transparency mode objects may range from completely transparent to fully opaque. However, in other modes such as 'Additive' and 'Dark' the objects may not be able to be either fully opaque or fully transparent. This is fundamental to the OpenGL methods used to calculate the various transparency modes.

**images showing the different alpha modes, 3 or 4 with none

6.2.12.e Rescale Normals - 'R'

By default normals are always rescaled, this results in uniform shading of objects regardless of the scaled size. Scaled objects must have their normals (lighting vector for polygons) rescaled in order to have 'proper' lighting. However, not rescaling the normals has the interesting effect of making small objects appear brighter and large ones darker. This may be a desired 'look' since it has the benefit of making it easier to spot small nested groups. This is particularly true when using additive transparency. So with normal rescaling ON everything appears as you would expect it to. With rescaling OFF small objects appear brighter and large ones darker.

Note that re-scaling normals does cause a minor performance loss, so turning it OFF may speed things up a little in situations where the GPU is not very fast.

**image of Rescale Normals ON & OFF

6.2.12.f Background Color - 'B'

Toggle the background color between black and white. Useful for saving ink when printing.

*mouse-background camera selection only works with a black background.

6.2.13 Texture Maps - 'T'

Iterates through the list of textures for the currently active objects, (SHIFT-T for reverse order.) The default for an object is no texture assigned, except for the grid which defaults to textureID = 1, (map00001.jpg). If you do not want a texture on the grid, then select the grid ('G') and press SHIFT-T once to decrement to textureID = 0. This will result in only the grid lines being drawn.

Object textures are also effected by color and lighting parameters, the exception is the grid texture which is always lit 100% white. Textures are loaded at application launch (see 'File Types – Texture Map' section for more detail.) SHIFT reverses the iteration order.

*Planned future support for choosing a texture during runtime.

**image of texture maps on grid and objects

6.2.14 Freeze - 'F'

Toggles freeze state. By default all nodes are not frozen and they can be positioned, scaled and rotated. If frozen then they will stop being updated and not be modifiable. Includes keyboard and mouse controls for position, scale, and rotation. Also stops animation (physics) and live channel data. When selected, frozen nodes appear with a wireframe colored ice-blue.

6.2.15 Hide - 'H'

If hidden, the object will no longer be drawn. However if it is the active object or part of a selection set then the red or yellow wireframe will be drawn around the invisible object. To unhide you can use the keyboard selection methods to navigate to the hidden object and then press the hide key again. Note that hidden objects are not frozen, so animation and live channel data will continue to be updated. If you wish you can freeze and then hide to prevent updates. Child nodes only hide themselves, where as hiding a root-node hides the entire tree.

*add feature to unhide all by selecting all and then pressing hide.

**images of hidden object with wireframe around it, plus un-hidden image

----
6.2.16 Set Points - '[' ']'

The set point is typically used to restrict an objects movement. Root pins have a default low set point of Z = 0.0 which prevents them from being moved below the surface of the primary grid. This restriction can be turned off by selecting the Z axes (using the 'X' key) and then pressing '[' key (left bracket) will turn off the Z axis low set-point for the current selection. The set point is applied to all currently active axes. For example, setting an objects low set point with Z axis active will prevent the object from being moved lower then its current position. Similar for the high set point, as a ceiling. Set points for root pins apply to global coordinates. Set points for child nodes are relative to the child coordinate system.

*Currently the set-points are toggled (using the left and right brackets,) this will be updated so that simultaneously pressing SHIFT will clear the set point. The toggle can be confusing, for example if you press the key while XYZ axes are all active, whichever axes has a set point currently ON will be turned OFF, while those with no set point will be turned ON (set.)

*Update set-point for children to restrict rotation around the parent.

6.2.17 Scene Presets - '5' '6' '7'

Loads a preset scene. There are 3 hard-coded preset scenes that can be loaded by pressing the corresponding number keys 5, 6, and 7. Note that you can load the presets repeatedly and the additional set of objects will be positioned relative to the currently active node.

6.2.18 State File - Load & Save

L	Load ANTZ0001.CSV (Open)
K	Keep ANTZ0001.CSV (Save)

1	Load ANTZ0001.CSV (SHIFT+1 to Save)
2	Load ANTZ0002.CSV (SHIFT+2 to Save)
3	Load ANTZ0003.CSV (SHIFT+3 to Save)

The State File contains the entire scene, including all object parameters and camera locations. Some global parameters are not stored such as Alpha Mode, Grid Segments, Background Color....

It is possible to merge scenes by loading multiple files into the same scene. Camera and grid parameters will be what the last file loaded sets them to.

*future support for storing global parameters.

6.2.18.a Keep State – 'K'

Saves the scene to the CSV file 'ANTZ0001.CSV', more detail is provided in the 'File' section of this document.

6.2.18.b Load State – 'L'

Loads the scene from the CSV file 'ANTZ0001.CSV'

*Keep and Load will be updated to allow choosing a file using the standard OS file dialog box.

6.2.18.c Quick State - '1' '2' '3'

Loads or Saves the State File to a preset filename. Pressing either 1, 2 or 3 will Load the corresponding state file 'ANTZ0001.CSV', 'ANTZ0002.CSV' or 'ANTZ0003.CSV'. To Save the state file press SHIFT-1, SHIFT-2, or SHIFT-3.

6.2.19 Load Channel File - 'P'

Loads the Channel File into memory and starts playing the file. The default name is "TNG00001.CSV", see the 'File - Channels' section and appendix for more info.

*Will add a file dialog box for the user to select a specific file.

6.2.20 Channel Assignment - '<' '>'

The Channel Assignments allow for objects to be animated based on the data contained in the Channels File. Channels may be mapped to position, scale and color offset, depending on the object type. This allows for time based data to move a pin along a path or a child node to change position and scale. There is also a Color Offset channel ('Z') that allows for a hue shift that can be animated with time.

Normally all objects have their XYZ channels set to zero which is equivalent to no channel assigned. If you want to animate the Color Offset then first select the object(s) then make 'Z' the Active Axes. Now press either '<' or '>' keys to change the channel assigned. This may be done either before or after loading the Channels File, though nothing will change until the file is loaded. The Color Offset shifts the hue of the current color so that red becomes green, green becomes blue, and blue becomes red.

To animate position of a selected root pin set the Active Axes to XY and then change the channel. Child nodes are different in that the X axis is mapped to the position around the parent and the Y is scale, Z remains Color Offset.

See the 'File Types – Channels' section for additional info.

*Future support will include live IO channels such as audio, EEG, light, temperature and video.

6.2.21 URL recordID retrieval - 'U'

Opens the default system browser with the currently active objects recordID appended to the specified URL. This can be used to retrieve and display additional information about a particular object using a standard browser.

The default URL is hard-coded and appears as:

'http://temporalzone.com/id.html?id=0'

To set the URL you may pass it in as a command line parameter, see the 'Command Line' section for more info.

6.3 Command Line

The application excepts 2 types of command line parameters, files to load at startup and the URL used for recordID retrieval through the system browser.

You may pass multiple files:

```
C:\apps>antz.exe antz0001.CSV antz0002.CSV
```

This can also be combined with the URL, which must start with 'http'.

```
C:\apps>antz.exe antz0001.CSV antz0002.CSV http://myDomain.com/recordID.html
```

This will open both CSV files and set the recordID URL.

Section 7: File Types

This section is designed to be an overview of the file types supported by the application. Please see the appendix on information on how CSV files are formatted. The 'Commands' section explains how to Load and Save the CSV files. Texture Maps are loaded at application launch.

The primary file type is the State File, it is used to store and retrieve the entire state of the current scene.

The secondary data file type is the Channels file which allows for data that changes over time, such as audio, EEG, position, etc.... It is used to animate object parameters.

Additionally, various image formats are supported for texture mapping.

File Name Conventions:

State	antz000x.CSV
Channels	TNG00001.CSV
*Event Log	log0000x.CSV
Texture Maps	map0000x.JPG
*3DS	geo0000x.3DS

*An Event Log file is planned.

*Support for 3DS files to load custom 3D models is planned.

7.1 State - .CSV

**image of dataset loaded in from a CSV file.

The antz000x.CSV is a snapshot of the entire scene. It contains all information needed to represent the spatial characteristics of the scene at the specific moment the file is generated. However, some global parameters are not stored in the file, such as window position, Alpha Mode (transparency type), and grid spacing....

The State File also stores velocity rates that result in animation. However, the data that changes over time (per cycle) is stored in the Channels file.

It is possible to merge scenes from multiple files. This can be done by the user at runtime or at launch by passing in multiple file names.

See the 'Commands – State File' section on how to Load and Save files. Also 'Commands – Command Line' on how to load files upon application launch.

See the Appendix for file formatting and parameter descriptions.

7.2 Channels - .CSV

The 'antzch000x.CSV' contains time based channel information. The channels can be used to modulate (animate) objects position, scale and color. Each object has up to 3 channels assigned to it. To change the assigned channel use the '<' and '>' keys, applies the currently active axes, ('X' key.)

Channels effect different parameters based on the object type. For root pins the X and Y channels effect the X and Y position and Z effects the color. For children, X is rotation around the parent, Y is scale, and Z is color.

See the 'Command' section on how to Load the Channels file or the appendix on how to format the file.

*Future support to include mapping channels to more parameters.

7.3 *Event Log

*Event log records all user commands and other specified triggers.

7.4 Texture Map - .JPG, .TGA...

Up to 256 textures can be loaded for use in the scene. The textures are located in a sub-folder of the main application named "maps\". The texture maps need to be sequentially numbered to be properly organized, starting with 'map00001.jpg' on up to 'map00256.jpg'.

By default the grid uses 'map00001.jpg', this can be changed. See the 'Commands - Keyboard - Texture Maps' section for information on how to assign the textures to objects.

Be aware that non-sequential texture names can be used, but there numbers will not correspond to the textureID stored in the CSV State file. (Also they must be named with the ".JPG" extension, but do NOT need to be a JPEG format... A variety of image types can be read in, but they must have the names specified. Yes it's strange to rename a PNG image to 'map00008.jpg', but it will work.)

The textures are automatically loaded at launch and may cause a significant delay if they are numerous and large. Note that the application will appear to not be responding until all textures are loaded (can be minutes.)

The GPU memory determines the total size of the textures loaded. Maximum texture size depends on GPU hardware, however the user does not need to be concerned as the loading routine will down-convert large textures (if required by the hardware.) A typical GPU today will support maximum size of either 1024x1024 or 2048x2048, some support 4096x4096. Power of 2 sizes are also no longer required in most hardware, the texture loader will handle any necessary scaling.

All textures are converted to 8-bits per channel. Both RGB and RGBA with alpha channel is supported.

Readable Image Formats:

```
      BMP - non-1bpp, non-RLE (from stb_image documentation)
      PNG - non-interlaced (from stb_image documentation)
      JPG - JPEG baseline (from stb_image documentation)
      TGA - greyscale or RGB or RGBA or indexed, uncompressed or RLE
      DDS - DXT1/2/3/4/5, uncompressed, cubemaps (can't read 3D DDS)
      PSD - (from stb_image documentation)
      HDR - converted to LDR, unless loaded with *HDR* functions (RGBE or
            RGBdivA or RGBdivA2)
```

*Future Feature - Allow for selecting a specific texture map with any name and file path. The path will be stored in the CSV Table Map file and/or DB.

*Future Feature – Use an image sequence for video playback.

7.5 *3DS Models

*Future support for 3DS models will allow for adding custom geometry to the scene for use as root pin, child node objects, or grid geometry.

Section 8: *DataBase (MySQL)

*DB support is currently in development. The DB structure mimics that of the CSV state and channels files. It is possible to import and export between the DB and the CSV files. Certain data such as textures maps and 3DS models are referenced using file paths in the DB, but the data itself is stored externally.

Appendix A Command Cheat Sheet

A.1   Mouse

R - Click on object toggles its selection on/off
R - Hold on object DRAGS selected objects in XZ plane (L-R & Up-Down)
R - Hold on background FLY's camera

L - Click on an object NOT selected will select it and unselect all others
L - Hold on an object NOT selected will drag only the object, DRAGS in XY
L - Hold on an object CURRENTLY selected will drag all, DRAGS in XY
L - Hold on background orbits object in both axes of EXAMINER mode XY

L+R - Hold on background orbits and ZOOMS in and out Examiner mode XZ

Note that it is possible to switch directly between Examiner mode XZ and XY by releasing the right button and continuing to hold the left button.

A.2   Keyboard

Numbers, -, =, etc apply to the main keyboard, not the number pad.

A useful 'bug' is to change selection to another object (TAB, New, etc..) and do this while performing a rotation, it will continue to rotate, also applies to zoom and translate…

It is possible to press multiple keys at once (3-5 typical depending on the keyboard and key combo…) So for example, you can do a rotation and zoom while simultaneously changing the color.

SHIFT	Reverses some functions, speeds up rotation and translation
ESC	Fullscreen Exit
X	change active aXes, for non-uniform scaling, channels, segments

--- Node Creation ---

N	New node, creates nodes, (SHIFT-N to create a new primary torus)
Del	Delete node, deletes active node and all its child branches

--- Selection ---

Tab	select sibling node, (SHIFT+Tab for previous)
Enter	select child node (SHIFT+Enter for parent)
C	select Camera, repeat to select next camera
G	select Grid, repeat to iterate through grids
4	select All Pins toggle
~ (tilda)	un-select All
spacebar	select or unselect current object

--- Translate Camera or Objects ---

A	X decrease
D	X increase;
W	Y increase;
S	Y decrease;
E	Z increase;
Q	Z decrease;

```
--- Rotate Arrows ---
Left        rotate left
Right       rotate right
Up          rotate up
Down        rotate down

--- Scale ---
Z           Scale objects up (SHIFT+Z for down) based on active aXes 'X'

--- Topology & Geometry ---
J           Change 'topo' type, also changes the geometry primitive
            SHIFT-J will change the facet (on a cube...)

O (alpha)   Object geometry, does not change the 'topo' type.
R           Ratio sets inner radius of a torus, SHIFT-R to reduce.

Y           Grid Segments, adds (or SHIFT-Y to subtract) 2D and 3D layers
                  effected by the active aXes (Z axis creates 3D grid layers)

--- Color & Transparency ---

+ (plus)    next color
- (minus)   previous color

9           less opaque (more translucent)
0 (zero)    more opaque (less translucent)

B           Background color, toggle between black and white
R           Rescale normals toggle, effects lighting of scaled objects
8           change transparency mode (3 alpha modes + none)

T           Texture Map selection (SHIFT-T for previous map)

F           Freeze;
H           Hide;

[           Low set point (effected by active aXes)
]           High set point

--- Scene Presets ---

5           Preset 1
6           Preset 2
7           Preset 3

--- State Files ---

K           Keep ANTZ0001.CSV (SAVE)
L           Load ANTZ0001.CSV

1           Load ANTZ0001.CSV (SHIFT + 1 to save)
2           Load ANTZ0002.CSV (SHIFT + 2 to save)
3           Load ANTZ0003.CSV (SHIFT + 3 to save)

--- Channels File and Assignment ---

P           Load channels file (TNG00001.CSV) for animation

>           Channel Up (applies to currently Active aXes)
<           Channel Down

--- URL recordID Retrieval ---
U           open URL with recordID in a browser
```

Appendix B State File - .CSV

B.1 State File Overview

The scene 'State' file is written in CSV format (Comma Separated Values). The first line contains the field names for the columns. Individual node records start on the second line and continue with one node per line until the last node is reached. The file is terminated with an extra line return at the end, (the last line is blank.) All values are either 32bit signed integers or 32bit signed floats.

B.1.1 Tree Hierarchy

CSV files are written such that the nodes are grouped by root tree, child nodes are sorted using a recursive algorithm. It is not usually necessary to understand (or retain) the node ordering in the CSV file. However, for those who want to know.... The file write process iterates through all root nodes sequentially. First writes the root-node, then recursively calls itself for all child nodes. The parent node is always written before its child nodes. After the last child branch is traversed, the process starts over on the next root-node.

The CSV Read process is agnostic to node ordering within the file, the only parameter that defines the hierarchy is the 'parent_id'. Orphan nodes that are out of order in the file are sorted and attached to there parents after reading the entire file.

---
id (node ID)

The 'id' field is used to build the topo tree hierarchy and link to additional node type specific data. The node ID is only unique within the file. Whenever a file is opened (or imported,) a new set of IDs are generated. This allows for merging multiple files that contain overlapping node IDs without any contention.

An exception is the primary Root-Camera and Root-Grid which are always overwritten. This is also true for the first 4 child cams at branch level 1. All others nodes are created as 'New' and are assigned session unique IDs.

Note that if you save a file, then read it back in, delete and/or add some nodes, then re-save, you will get a different set of IDs. Use the 'record_id' to keep track of nodes over time (throughout multiple sessions.)

---
parent_id

The parent ID determines the tree hierarchy. Root-Nodes are set to zero, all child nodes must specify the 'parent_id', which is the only parameter used to build the hierarchy.

---
branch_level

Root-Nodes have 'branch_level' = 0. The first child, such as a primary torus is set to 1, each additional sub-child adds one. So a child of a primary torus is 'branch_level' = 2, and so on.... The 'branch_level' is automatically set during the node creation process and the CSV value ignored. It is included in the CSV file for convenience of 3rd party analysis.

---
record_id

Because the node 'id' is subject to automatic re-assignment by the app, the 'record_id' is provided for the user to keep track of the record that the node represents. It is retained by the node and is not modified by the application. This may be used to reference the original DB record that the node was based on.

---
child_count - the number of child nodes directly attached to the node at the current 'branch_level'. Does not count the sub-children of the attached child nodes. The 'child_count' is automatically set during node creation and is provided for convenience of 3rd party analysis.

---
child_index - Specifies which child node that is actively selected, used mostly for keyboard navigation of the tree topology. Basically this allows the application to remember which child branch was last active.

B.1.2 Position & Coordinates

Position is based on translation, rotation and scale (of the parent.) Root nodes are relative to the global origin. Child node coordinates are relative to (or inherent) there parents position and topo type.

'topo' types include default, cube, sphere, torus, surface, pin and grid. Each defines a coordinate system with a unique set of operations that transform the position, orientation and scale of any attached child nodes.

Some topo types, (such as a 'cube') have multiple local coordinate systems. The child node 'facet' value tracks which system, (or cube side to use.) See 'Cube - Key Table' appendix and 'Coordinates & Topologies' section.

Radial offset is 'translate_z' for a torus.

Translational velocity is set by 'translate_rate_x/y/z' and is the delta distance applied per cycle, (typically 60 cycles per second.) Nodes may be restricted (or wrapped) by both local node limits imposed and/or the parent topo type limits.

Typically you do not assign a 'translate_rate' to a root-node as it will continue to move until it disappears by leaving the scene boundaries, (at the clipping planes.) However, child nodes will typically limit or wrap there coordinates when boundaries are hit, (depends on parent topo type.)

The following table describes the coordinate system for a KML based Sphere in comparison to a Torus. The range for the two are the same, except for translate_y where the Torus has a range of -180 to 180 latitude vs KML -90 to +90 deg. Cube and Grid topo coordinates are the same as a Torus. Latitude values greater then +/- 90 degrees are wrapped onto the Sphere, (causes upside down object.) For detail see 'Topologies & Coordinates' section.

order	field_name	kml_name	kml_lo	kml_hi	torus_lo	torus_hi
1	translate_x	longitude	-180	180	-180	180
2	translate_y	latitude	-90	90	-180	180
3	translate_z	altitude	0	none	none	none
4	rotate_y	heading	0	360	0	360
5	rotate_x	tilt	0	180	0	180
6	rotate_z	roll	-180	180	-180	180

Rotation

The 'rotate_x/y/z' sets the node orientation with units in degrees. Order of operation is:

1st rotate about z-axis by the Heading, rotate_y
2nd rotate about x-axis by the Tilt, rotate_x
3rd rotate about z-axis(again) Roll, rotate_z

Camera 'Tilt' differs from all other objects in that the 'rotate_x' axis is inverted. A value of 0 results in a view looking straight down the -z axis, basically looking straight down on a map. At x = 90 deg the cam will rotate CCW about the x-axis and look at the horizon towards North (with Heading rotate_y = 0.) Effectively this is a 'Right-Handed' coordinate system for rotation, (with an inverted up vector.) Translation is still a 'Left-Handed' coordinate system. Compatible with the google KML '<Camera>' standard.

Objects (other then the camera) use a 'Left-Handed' coordinate system for rotation (and translation.) An object with zero Tilt will be upright. At rotate_x = 45 deg the object will lean towards North, (with Heading = 0).

rotate_y Heading range is 0 to 360 deg, North = 0, E=90, S=180, W=270.
rotate_z Roll range is -180 to +180 deg.

'x' axis of a child node can be used to set the spacing of toroids around there parent. 'y' axis will spin the torus, not very noticeable unless it has a texture map or is assigned a different primitive.

Rotational velocity is set by 'rotate_rate_x/y/z' and is applied (added) to the 'rotate' orientation every cycle.

Orientation and position of nodes, (other then the camera,) are compatible with the google KML '<Model>' specification. The altitude units are not the same. Instead of feet (or meters) altitude is specified relative to the circumference, altitude units are 1 to 1 with translate_x at the equator.

Cameras use the '<Camera>' KML coordinate system which differs in that the 'Tilt' x-axis is inverted, (and camera up vector.) See 'Camera Position” in this appendix for further detail.

Camera Position

Position is set using 'translate_x/y/z' and 'rotate_x/y/z' for orientation.

Tilt is 'rotate_x', straight down is 0.0 deg, horizon is 90 deg, up is 180.
Heading is 'rotate_y', North is 0.0 deg, E 90, S 180, W 270.
Roll is 'rotate_z', CW is 0 to 180 deg and CCW is 0 to -180.0 deg.

Note that the x-axis 'tilt' is essentially a flipped axis, and is therefore a right-handed coordinate system. This is an exception, the rest of the coordinates are all left-handed systems.

Note that the 'rotateVec_x/y/z/s' parameter should be ignored, it is a unit vector calculated from 'rotate' and is automatically updated.

There are multiple cameras available to the user. There is a single root camera that all other cameras are attached to as child nodes. New child camera's may be created or existing child cams may be deleted. The Root-Camera is locked and cannot be deleted. By default there are four child cams attached to the Root-Camera.

The coordinate system is designed to be compatible with the Google KML standard for a '<Camera>', not to be confused with '<LookAt>'. For detail see 'Topology & Coordinates' section in this document.

Also see the '<Camera>' specification in the Google KML docs:

http://code.google.com/apis/kml/documentation/kmlreference.html

*At this time, all cameras coordinates are relative to the World Origin, which is flat cartesian space, (non-spherical coordinates.) Future updates will allow for attaching a camera to any node. Attaching a camera to a parent with a Sphere topo type would result in polar coordinates.

Scaling

Object size is set by 'scale_x/y/z'. A scale value of 0.5 is equivalent to a 50% reduction, default is 1.0 which results in no change in size. Valid scale values can be very small or large. A negative value results in an inversion of the geometry, including all child nodes of the current node. For uniform scaling the x, y and z values must be identical. Individual axes may have different scale values. This results in non-uniform scaling which causes an object to appear stretched or squished.

In general 'scale_rate_x/y/z' is used internally by the application but is not practical if set to a value other then zero. Otherwise continuos scaling will occur and an object will typically become far too large.

Ratio (torus inner radius)

The 'ratio' sets the inner radius of a torus relative to its size. Range is between 0.01 and 1.0, where 1.0 is a donut with no hole and 0.01 is a very thin donut. The default ratio is 0.1 which is equivalent to 10% of the outer radius. Nodes attached to a parent of Torus topo type are positioned relative to the surface, so scale and ratio (inner radius) effect the position.

*Ratio may effect other characteristics depending on geometry and topo type.

B.1.3 Texture Maps & Color

'texture_id' specifies which texture map to apply. When texture_id = 0 the texture map is disabled, zero is the default for most objects. The root-grid has a default textureID = 1 that is loaded from 'map00001.jpg'. For detail see 'File - Texture Maps' section.

'color_r/g/b/a' sets the RGBA of a node and is usually based on the 'color_index' that maps to a palette of colors, default palette has 20 colors, larger index numbers are mapped into the palette range. T

'color_a' sets the alpha transparency and is not set by the 'color_index'. Texture maps can be RGBA with per pixel alpha values.

Note that you may set the color to any custom RGBA you choose, but if the user changes the color, the custom color will be lost. In general, match the RGB values to the 'color_index - key table' in the appendix.

B.2 State File - Field Descriptions

field_name	type	description
id	int	node ID used for pin tree relationship graph
type	int	node type - camera, grid, pin, etc... (see key table)
data	int	additional node specific data, defined by the node type
selected	int	selection set status, 1 if part of active set, 0 if not
parent_id	int	ID of parent node
branch_level	int	root node is 0, each sub-level is 1, 2, 3, 4...n
child_list_id	int	same as nodeID
child_index	int	currently selected child node
child_count	int	number of child nodes attached (current max is 266)
channel x	int	channel number, mapped to translate x
channel y	int	channel number, mapped to translate y or scale
channel z	int	channel number, mapped to color offset
channel_index_x	int	previous data update time-stamp (last read)
channel_index_y	int	previous data update time-stamp (last read)
channel_index_z	int	previous data update time-stamp (last read)
average_x	int	*type of averaging applied to channel data
average_y	int	*type of averaging applied to channel data
average_z	int	*type of averaging applied to channel data
interval_x	int	*number of samples to average
interval_y	int	*number of samples to average
interval_z	int	*number of samples to average
rotate_vec_x	float	reserved - unit vector calculated from rotate_x/y/z
rotate_vec_y	float	reserved - unit vector calculated from rotate_x/y/z
rotate_vec_z	float	reserved - unit vector calculated from rotate_x/y/z
rotate_vec_s	float	reserved - unit vector calculated from rotate_x/y/z
scale_x	float	1.0 for no scaling, negative value inverts geometry
scale_y	float	1.0 for no scaling, negative value inverts geometry
scale_z	float	1.0 for no scaling, negative value inverts geometry
translate_x	float	longitude, relative to parent coordinate system
translate_y	float	latitude, relative to parent coordinate system
translate_z	float	altitude, relative to parent coordinate system
origin_x	float	reserved - local origin offset, for centering
origin_y	float	reserved - local origin offset, for centering
origin_z	float	reserved - local origin offset, for centering
rotate_rate_x	float	angular velocity rate applied per cycle
rotate_rate_y	float	angular velocity rate applied per cycle
rotate_rate_z	float	angular velocity rate applied per cycle
rotate_x	float	heading, 0 to 360 deg
rotate_y	float	tilt, 0 to 180 deg
rotate_z	float	roll, -180 to 180
scale_rate_x	float	scaling rate applied per cycle
scale_rate_y	float	scaling rate applied per cycle
scale_rate_z	float	scaling rate applied per cycle
translate_rate_x	float	velocity rate applied per cycle
translate_rate_y	float	velocity rate applied per cycle
translate_rate_z	float	velocity rate applied per cycle
translate_vec_x	float	reserved
translate_vec_y	float	reserved
translate_vec_z	float	reserved

State File - Field Descriptions... continued

shader	int	*shader type, flat, phong... (see key table)
geometry	int	primitive type used (see key table)
line_width	float	line width used for wireframes and line plots
point_size	float	vertex point size used for plots
ratio	float	ratio effects geometry such as inner radius of a torus
color_index	int	color index from color palette (see key table)
color_r	int	8bit RGBA color value (typically set by color_index)
color_g	int	8bit RGBA color value (typically set by color_index)
color_b	int	8bit RGBA color value (typically set by color_index)
color_a	int	8bit RGBA alpha transparency of node (not set by index)
color_fade	int	*fades older data points over time
texture_id	int	texture map ID, none = 0, starts at 1, 2, 3...
hide	int	hides the plot if set to 1
freeze	int	freezes the plot if set to 1
topo	int	topology type, cube, sphere, torus... KML coordinates
facet	int	facet node belongs to, such as which side of a cube
autozoom_x	int	*auto-zooms plots to keep in bounds of the screen
autozoom_y	int	*auto-zooms plots to keep in bounds of the screen
autozoom_z	int	*auto-zooms plots to keep in bounds of the screen
trigger_hi_x	int	if 1 then turn on upper limit
trigger_hi_y	int	if 1 then turn on upper limit
trigger_hi_z	int	if 1 then turn on upper limit
trigger_lo_x	int	if 1 then turn on lower limit
trigger_lo_y	int	if 1 then turn on lower limit
trigger_lo_z	int	if 1 then turn on lower limit
limit_hi_x	float	upper limit
limit_hi_y	float	upper limit
limit_hi_z	float	upper limit
limit_lo_x	float	lower limit
limit_lo_y	float	lower limit
limit_lo_z	float	lower limit
proximity_x	float	*reserved for future proximity and collision detection
proximity_y	float	*reserved for future proximity and collision detection
proximity_z	float	*reserved for future proximity and collision detection
proximity_mode_x	float	*reserved for future proximity and collision detection
proximity_mode_y	float	*reserved for future proximity and collision detection
proximity_mode_z	float	*reserved for future proximity and collision detection
segments_x	int	number of segments, 0 for none, grid default is 12
segments_y	int	number of segments, 0 for none, grid default is 12
segments_z	int	number of segments, 0 for none, grid default is 1
value	int	user defined value, variable can be whatever you want..
format_id	int	*draw the label by id
table_id	int	*table id maps external DB used by record id and format
record_id	int	record ID is the external source DB record key
size	int	size in bytes of memory used per node

B.3 Key Tables

B.3.1 type (node type)

type - key table	
value	desc
1	kNodeCamera
2	kNodeVideo
3	kNodeSurface
4	kNodePoints
5	kNodePin
6	kNodeGrid

B.3.2 topo

topo - key table	
value	desc
0	default
1	cube
2	sphere
3	torus
4	surface
5	pin

B.3.3 geometry

geometry - key table	
value	desc
0	kNPglutWireCube
1	kNPglutSolidCube
2	kNPglutWireSphere
3	kNPglutSolidSphere
4	kNPglutWireCone
5	kNPglutSolidCone
6	kNPglutWireTorus
7	kNPglutSolidTorus
8	kNPglutWireDodecahedron
9	kNPglutSolidDodecahedron
10	kNPglutWireOctahedron
11	kNPglutSolidOctahedron
12	kNPglutWireTetrahedron
13	kNPglutSolidTetrahedron
14	kNPglutWireIcosahedron
15	kNPglutSolidIcosahedron
16	kNPprimitivePin
17	kNPprimitiveWirePin
18	kNPglutWireTeapot
19	kNPglutSolidTeapot

B.3.4 cube_facet_map

cube_facet_map - key table	
value	desc
1	pos X
2	neg X
3	pos Y
4	neg Y
5	pos Z
6	neg Z

B.3.5 shader

shader - key table	
value	desc
1	kShadingWire
2	kShadingFlat
3	kShadingGouraud
4	kShadingPhong
5	kShadingReflection
6	kShadingRaytrace

B.3.6 indexColor

index_color - key table		
value	RGB value	color desc
0	50, 101, 101	steel grey
1	0, 255, 0	green
2	255, 0, 0	red
3	0, 0, 255	blue
4	255, 255, 0	yellow
5	152, 0, 255	purple
6	255, 168, 0	gold
7	0, 255, 255	cyan
8	255, 0, 255	magenta
9	0, 153, 0	dark green
10	185, 153, 102	tan
11	255, 180, 255	pink
12	0, 152, 255	sky blue
13	185, 255, 0	light green
14	152, 0, 0	dark red
15	127, 127, 127	grey
16	127, 127, 255	light purple
17	197, 82, 0	rust
18	0, 0, 0	black
19	255, 255, 255	white

Appendix C Channels File - .CSV

The channels file allows for animating the position, scale (of child-nodes) and color offset of pins. It is formatted as a single table with the field names at the top. Currently the file must be a total of 30 columns.

It increments one row each program cycle (once per video frame, typical rate is 60Hz) and when it reaches the end it loops back to the start.

The first column is the cycle count, followed by the channels columns, (29 of them.) Upon loading, the entire file is read into a buffer and begins to play (incrementing the current row.)

Each node has 3 parameters that can be mapped to any of the 30 channels.

For root-pins:

```
'channel x' = 'translate x'
'channel y' = 'translate y'
'channel z' = Color Offset
```

For child-nodes:

```
'channel x' = 'rotate x'
'channel y' = 'scale x/y/z'
'channel z' = Color Offset
```

You can animate the XY position of a root-pin and its color offset. Child-nodes allow you to rotate around the parent torus, scale (size) and shift the hue with the color offset.

The color offset has a value range of 0 to 255. Zero does nothing, while 255 shifts the hue by one full color component, that is red becomes green.

'P' key will load the file TNG00001.CSV

'<' and '>' keys will change the channels assigned to the currently active/selected objects based on the current Active Axes.

The console will display the assigned channels.

See the 'Command - Channels' and 'File - Channels' section for more info.

The Channels File is hard-coded to be 'TNG00001.CSV' and is located at the same level as the application.

A crude tutorial, but works...

To test the sample file 'TNG00001.CSV' follow these steps:

```
      Run the app
      Rotate the camera to view the grid corner to the right (NE corner.)
      press 'P' - loads the time based channel CSV file.
      press 'N' - create a new pin.
      press '+' - change pin color to green.
      press '<' - assigns channels to the pin.
```

The PIN will zoom in from the NE corner and color shift from red to green.

Channels are set (using < or >) based on the active axes XYZ / XY / Z
X & Y set the position
Z is color shift
color channel should be between 0-255
color shift works with most of the pallete but not all
green -> red (other colors to different colors...)
certain colors (like grey) do not have a noticeable shift
akin to changing the HUE in photoshop (or an old TV)
channels are in CSV columns 0-29
you can assign any channel to a particular axes of any object
ch0 is typically reserved for a cycle count

*Future updates to the Channels will allow for a variety of sample rates and depths, varying from 8bit low-sample rate single channels to high frequency audio, or even HD video.

***Appendix D Database Tables**

Appendix E - Glossary

Channel - Represents data that changes over a period of time, typically changes every cycle. Though some channel types like sound may have multiple samples per cycle. Each sample can be a single numeric value (like pressure) or have multiple values per sample, such as the X/Y/Z position of an object, multiple audio channels, or image raster composed of millions of pixels.

Child-Node - A node attached at any branch level other then zero. Not a root node.

Node - A single object in the scene. Can be a root-node or child-node. A child is considered a different node then its parent object. The pins, grids, lights, and cameras are all different types of nodes.

Orthogonal - Similar to perpendicular but in 3D or higher dimensions. For example, two planes that intersect at 90 degrees are orthogonal. For more info search 'orthogonal basis' in context to linear algebra.

Parent-Node - All child nodes are attached to a parent, which is the next branch level down the tree, root nodes are branch level zero.

Perpendicular - Two lines are perpendicular if they cross at 90 degree right angles, similar to orthogonal but only in 2D.

Pin - Tree graph data structure with root-pin and all its Child-Nodes.

Primitive - Geometric objects such as cube, sphere, cone, torus and pin.

Root-Node - The base object (or trunk) of of a tree (graph) of objects.

Root-Default - the first object in the node array, unused.

Root-Camera - all cameras are attached to the root camera.

Root-Grid - all grids are attached to the root grid. *may change this to allow standard nodes to mix topo types with grids.

Root-HUD - *contains all HUD elements with node mapping.

Root-Light - all lights are attached, lights can track objects.

Root-Pin - Any base object of a pin, by default looks like an ice-cream cone, but can be changed to other geometric primitives. Is a specific type of root-node.

Topo - Short for 'topology' which is used to determine the coordinate system and is typically associated with a geometry such as cube, sphere, torus, etc... May represent a set of any dimensional basis with associated dot and cross products.

Topology - see 'Topo' for definition in this context.

Tori - Multiple toroids.

Toroid - Same as torus.

Torus - A geometric object in the shape of a donut or ring.

*additional terms to be added.