Antz User Manual

2011, November 5th

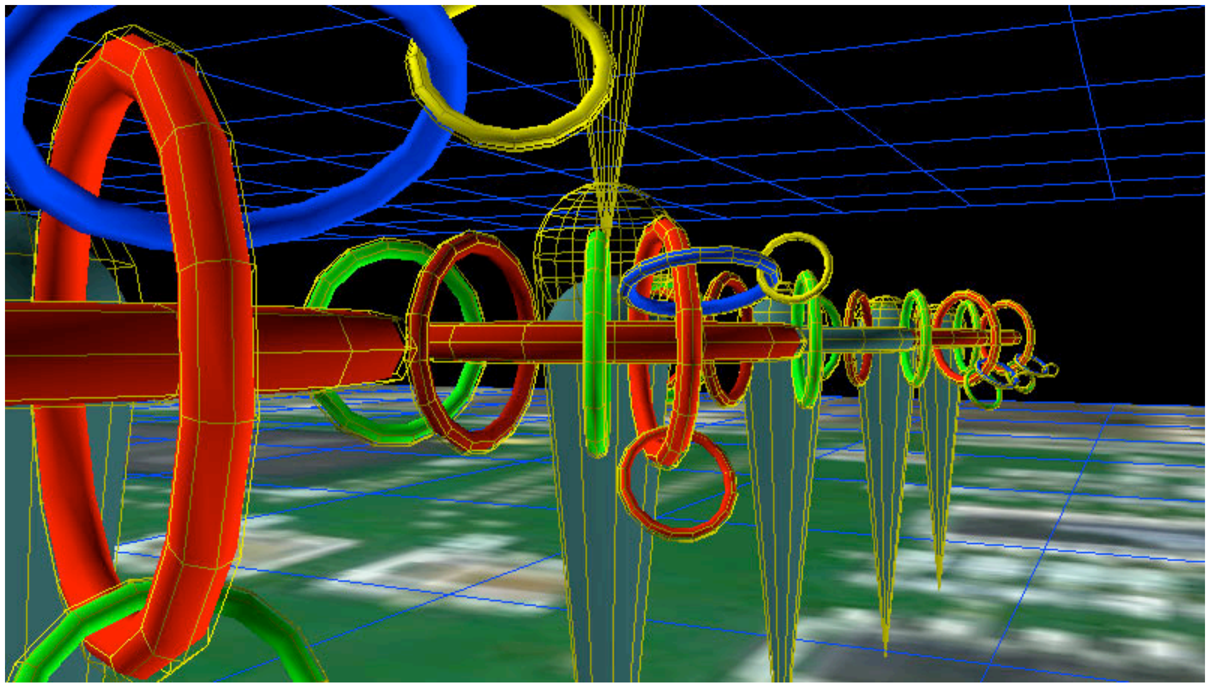
by: Shane Saxon

Draft

Table of Contents

*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 that facilitates user interaction with complex datasets through use of intuitive spatial cues. The purpose is to provide a more efficient mechanism for identifying patterns and relationships in vast quantities of information. It is an interactive multi-dimensional spatially based environment, or rather a unique type of cyberspace aimed at data visualization. Think of it as 'spreadsheet meets cyberspace.' Capable of displaying any dataset and combining it with real-time real-world inputs so that the user may control the virtual representation on the fly.
The ultimate goal of the application is to close the cognitive loop between users and the powerful logic engines known as computers.

For example, you may take a table of data representing academic test scores and display them as a series of tori that are distributed over a map. At each location various rings would represent components of the students demographics and various test types over a period of time. Parameters such as shape, color, transparency, size and position provide for a rich set of dimensional depth cues that can be mapped to the data. The visual representation enables one to view an entire region at once and quickly identify how specific factors effect performance.

In addition to static scenes, animated information aids in the pattern recognition process. Movement can either be based on rotational velocities (driven by physics) or externally defined time-based data channels such as position of a moving object.

*Realtime IO channels can work with a host of sensors such as audio, video, GPS or even EEG brainwaves. 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 with the real world.

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.

1.3   Developer

Please see the 'Developer Guide' for information on the source code and development installation instructions. 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, however it may work.

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)

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

*Secondary input devices such as Joysticks allow for more intuitive navigation and object manipulation. 3D control interfaces of interest include:

SpaceNavigator - www.3dconnexion.com
A 3D mouse that is a full 6-axes device that allows for simultaneous translation and rotation input.

InfiniteZ - www.infinitez.com
An affordable desktop VR system that includes 3D stereoscopic glasses with head tracking and a 3D stylus that can be used to manipulate objects.

3.3 Command Line

The command line will accept parameters upon launch, please see the 'Command' section of this document for more detail.

3.4 File

The state of the entire scene can be saved and retrieved using the State File in CSV format. In addition, a separate Channels File is used for time based data that can be recorded and played back for animating objects. See the 'File' section of this document for more detail.

3.5 *Database

*DB support is currently underway to support MySQL and is designed to be directly compatible with the application data files that store a the state and time based channels 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 static file based data. Also will allow for combining and analyzing multiple channel types and sources in a unified environment. Applications include realtime EEG, machine control and operation of scientific instruments.

3.6.1 *TNG3B

A simple RS232 based input device that provides 8 analog channels and 8 digital switches to be monitored in real-time. A myriad of inexpensive sensors can be used to capture everything from EEG signals to temperature and position, just to name a few....

3.6.2 *Audio

*In addition to the built-in audio capability of nearly all computers a large variety of affordable audio IO solutions exist that offer both in and out connections at up to 192kHz 24bit with lots of channels (16+). Audio can be recorded and analyzed in real-time while comparing to other data sources. Also, it is relatively easy using various simple circuits to connect all sorts of other sensors up to an audio input and actuators to an output.

3.6.3 *Video

*Video support will include multiple channels of real-time un-compressed HD video for use in everything from machine control to complex analysis through various CPU and GPU based algorithms.

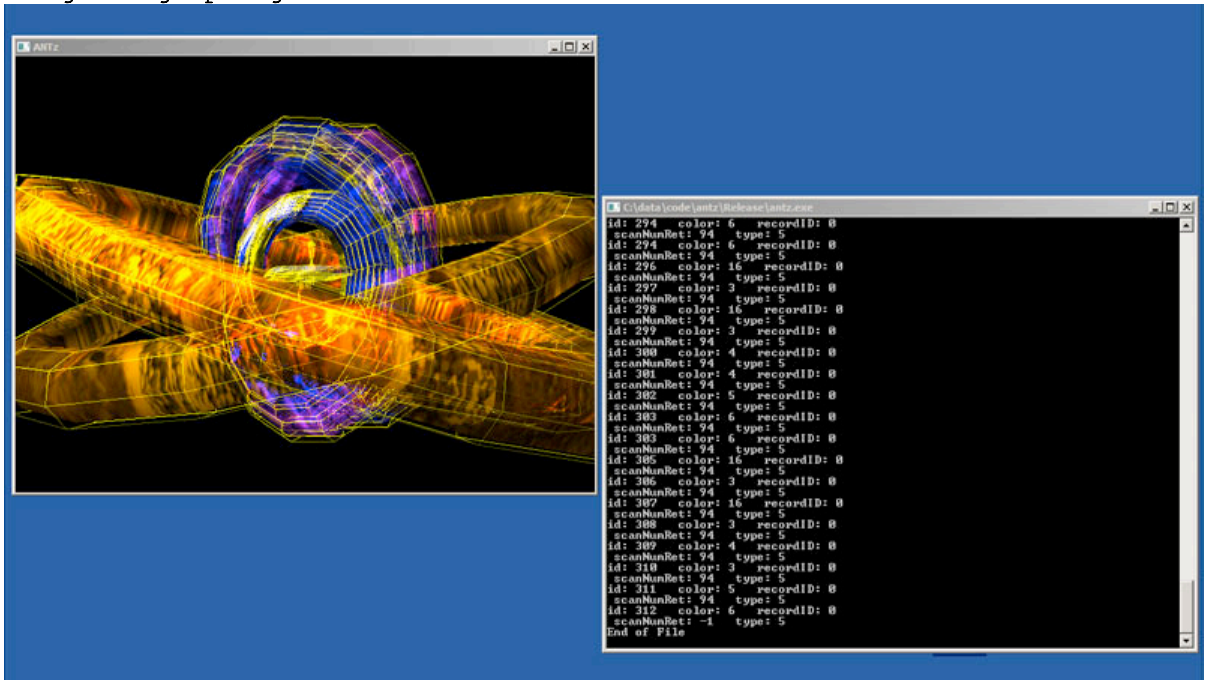
3.6.4 *Network

*Network support will allow retrieving real-time data from remote sources and support for clustering of the application across multiple systems to enable multi-user environments.

3.7 Displays

Currently only one graphing window is supported and a console window that outputs information for debugging. The graphing window can be stretched across multiple displays. In Fullscreen mode only the primary display is used.

**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 in Fullscreen mode will allow for both a single camera view and multiple camera views of the scene.

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 more custom solutions such as the InfiniteZ 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 make a version that utilizes the equalizer library that will work for a variety of cluster based cave environments.

Section 4: Graphing Overview

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.

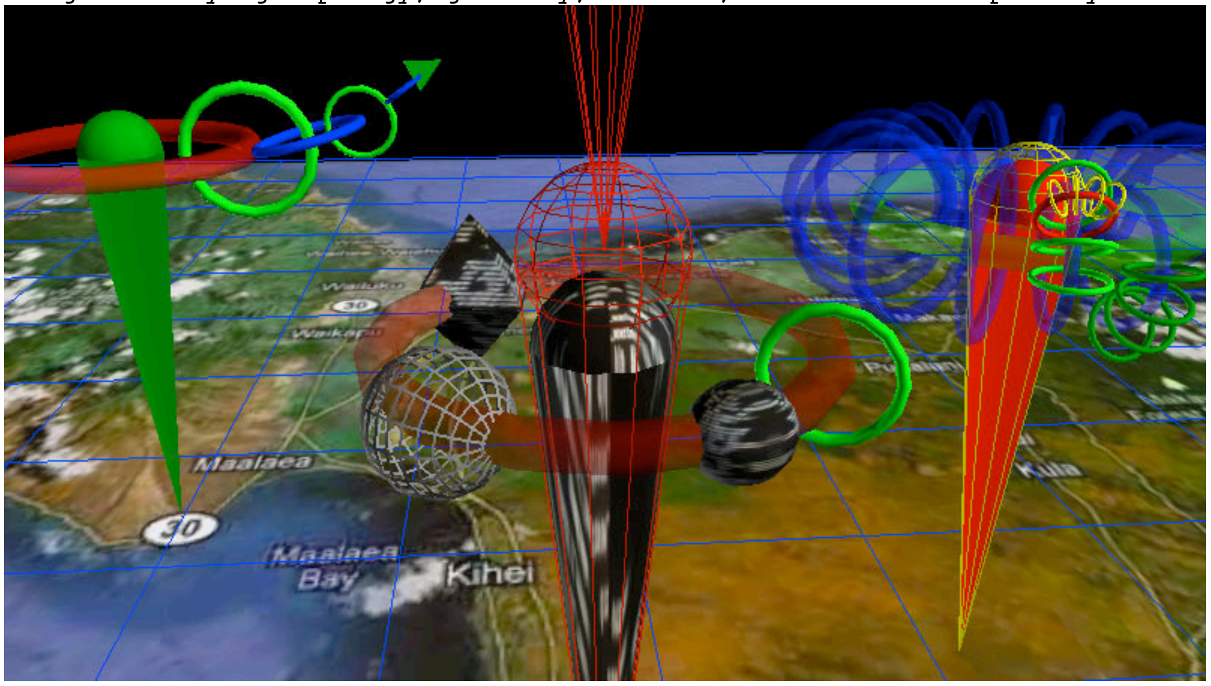
The primitives and grid can be set to a specific color as well as texture mapped with any image such as a map or abstract pattern for additional dimensional data cues. The object properties can be animated or changed with time based on velocity rates (physics) and channel assignments. Channels may either 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

**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 as sub-objects 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 their parent branch in the tree at a radius that is relative to parents scale. The color, scale, geometry type and position around the parent represent different parameters of the data set being visualized.

4.1.1 Orbital Tree

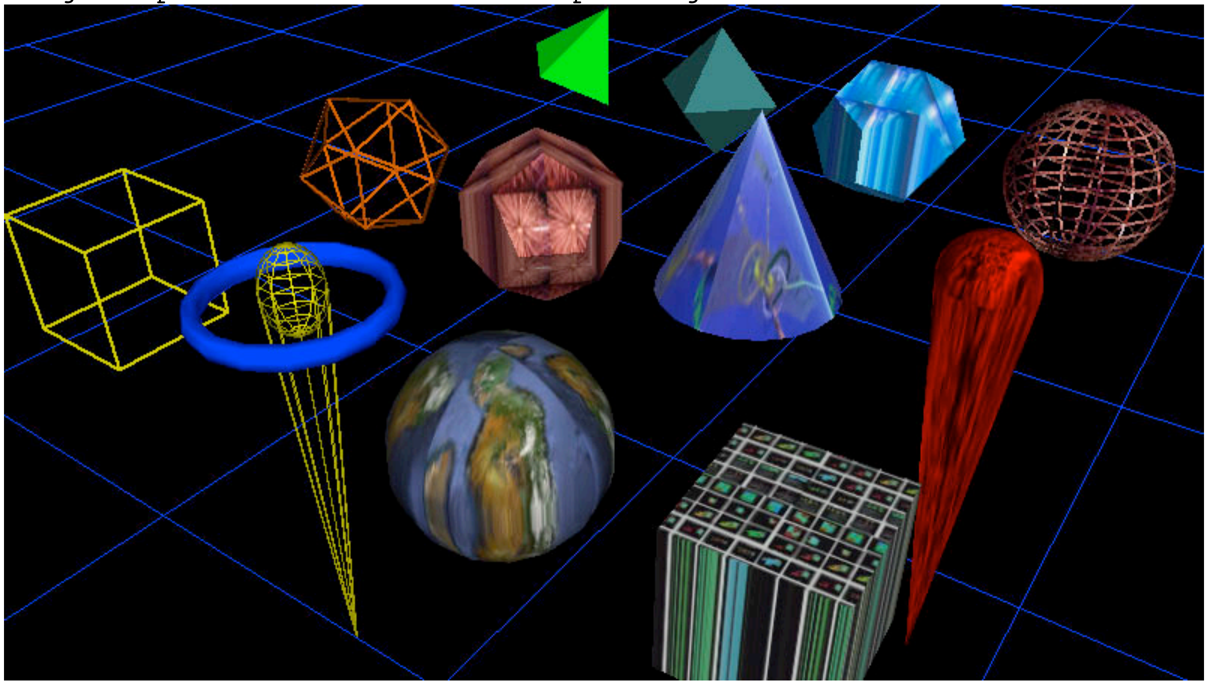
The topology is similar to the way planets and moons orbit in our solar system. The root pin is akin to the Sun and first order child nodes are like planets. Multiple children at the same level typically occupy the same orbit at different positions around their parent. The positions can be modified and set to either stationary or moving. A second order child node orbits its parent in an orthogonal orbit. Similar to the moon around Earth, except that it would be orbiting around the poles instead of the equator. Third order child nodes are similar to satellites around the Moon, but once again in an orthogonal orbit to that of parent. The depth of the tree is unlimited, however computer resources do impose a practical limit.

*Future feature - additional topologies to include Fractal Tree, Cubic Tree and nD Tree. The cubic tree has a fixed maximum number of 5 branches (+1 branch for the connection to the parent node,) at each level where as an nD Tree has an arbitrary number of branches.

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

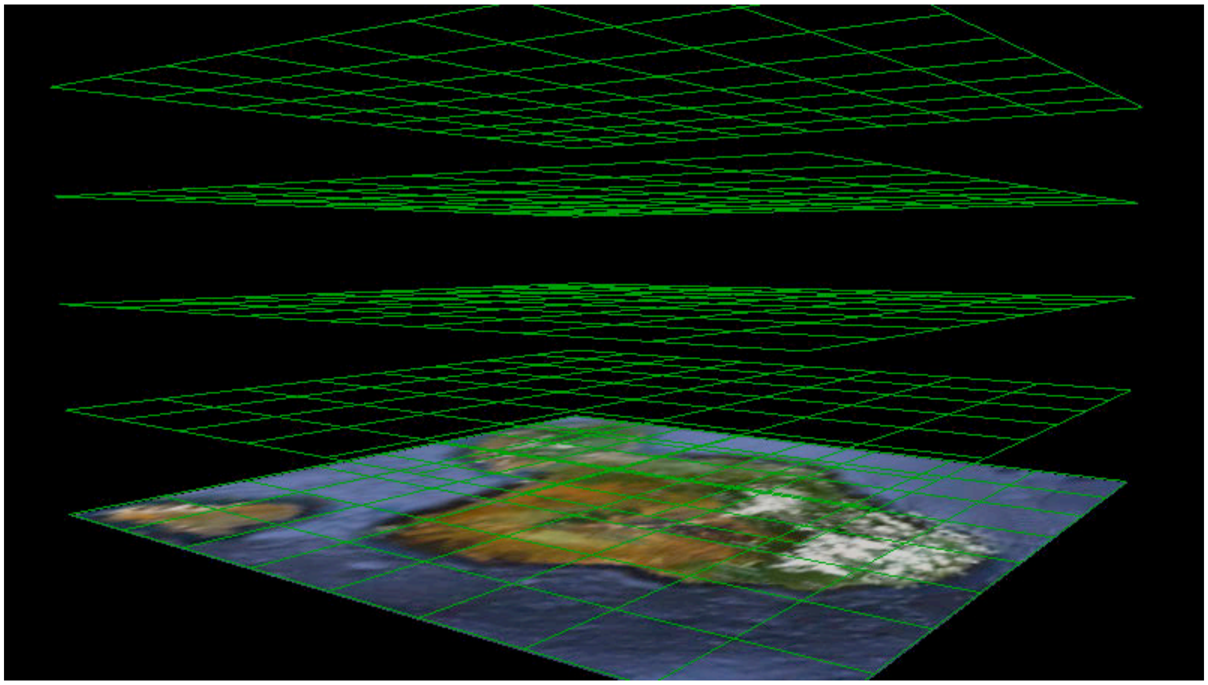
There are two types of grids the primary grid and *secondary grids. In addition to the grid lines, a texture map may be applied to the grid.

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 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.)

*Currently there is only the primary grid.

**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. There are 4 cameras in the scene which may be selected from and the positions of which are stored in the S.

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.

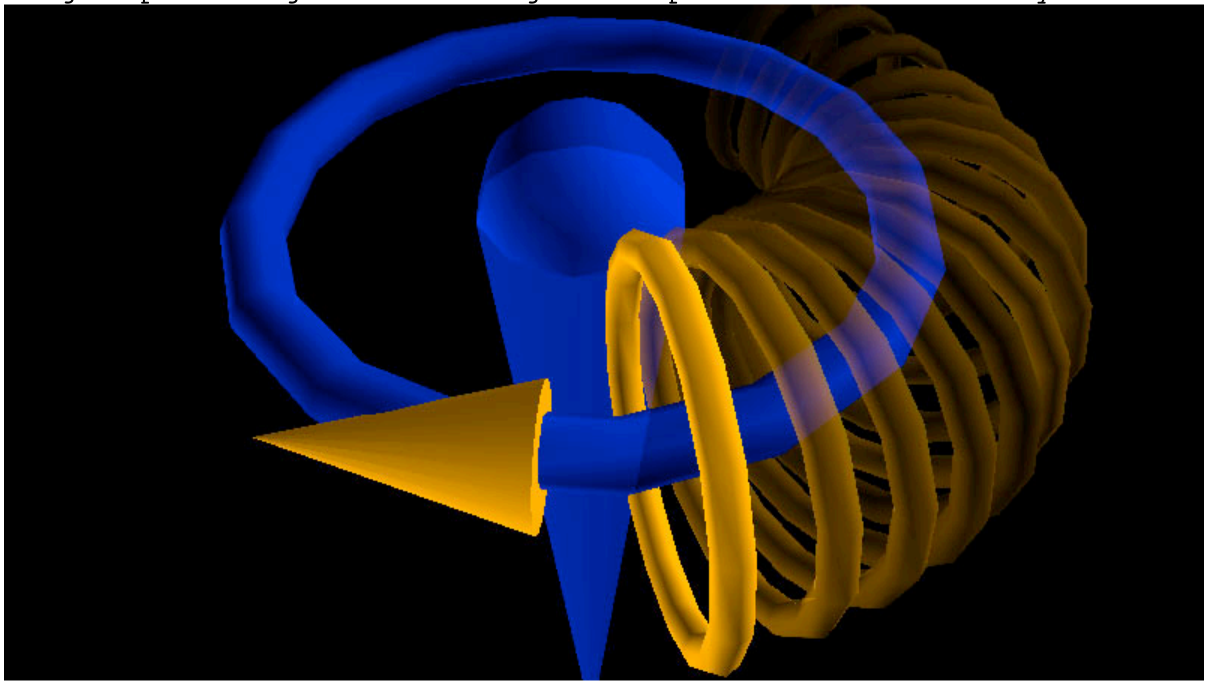
*Planned update will allow the near and far clip parameters to be set by thes user. Also the ability to add additional cameras.

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 Exit Fullscreen - 'ESC'

The application starts up in Fullscreen mode. This function allows Displays the scene in a window, there is also a system console window that becomes visible.

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.

*There will likely be new commands added in the future that will be axes specific.

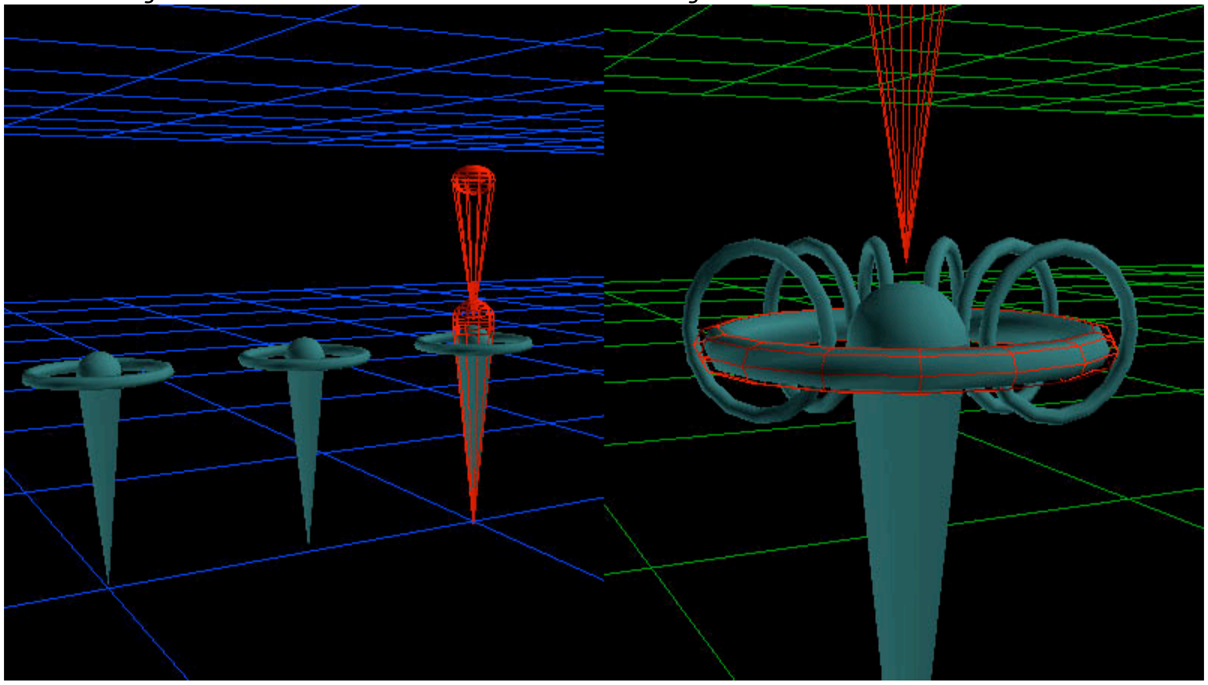
6.2.3 Creating & Deleting Objects - 'N' 'DEL'

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 (torus) you want it attached to. Additional child-nodes are created at the same level spaced around the parent.

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



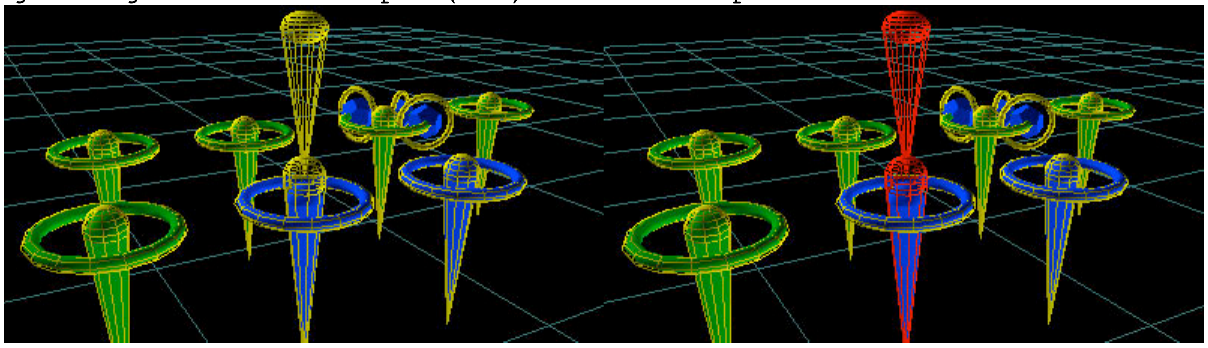
Delete Node - 'DEL'
'DEL' key deletes the active node and all children attached to it. Press and hold to delete multiple objects. Applies only to the active node and not the entire selection set.

*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'
Selects all nodes. A yellow wireframe will appear around the objects. If all nodes are already selected, then it will deselect all nodes.

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 4 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 grid with its center 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 primary 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 can be created by selecting the primary grid and pressing 'N' for new. 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 texture will remain visible. Grid lines also respond to changes in color and transparency, the grid texture will not be effected by changes in color or transparency.

*Secondary grids not yet supported.

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.

Child node radius can be offset using 'A' & 'D' keys and rotated with arrow keys.

Using the SHIFT key 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 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 a torus. 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.) Pressing SHIFT will iterate in the reverse order.

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

**image with variety of primitives

6.2.9 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

6.2.10 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

6.2.10.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.

6.2.10.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.

6.2.10.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

6.2.10.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

6.2.10.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.

6.2.11 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

6.2.12.a 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.

6.2.12.a 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.

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

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

----
6.2.13 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.

6.2.14 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.

6.2.15 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.

6.2.15.a Keep State – 'K'

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

6.2.15.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.

6.2.15.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.

6.2.16 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.

6.2.17 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.

6.2.18 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 segment count....

The State File will store velocity rates that result in rotational 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 or the appendix on how the files are formatted. Also 'Commands – Command Line' on how to load files upon application launch.

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 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 numberpad

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 (used by Z, Y, <, >, etc..)

--- Object Creation ---

N	New object, root-pin or child node
Del	Delete object, root-pin or child node

--- Selection ---

Tab	select sibling node, (SHIFT+Tab for previous)
Enter	select child node (SHIFT+Enter for parent)
4	select All Pins toggle
spacebar	select or unselect current object
C	select Camera (if selected, selects next camera, 4 total)
G	select Grid

--- 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	(effects position of child nodes)
Down	rotate down	

--- Scale & Geometry ---

Z	Scale objects up (SHIFT+Z for down) based on active aXes 'X'
O (alpha)	change object geometry type

--- Color & Transparency ---

+ (plus)	**next color**
- (minus)	previous color
9	less opaque (more translucent)
0 (zero)	more opaque (less translucent)
B	toggle Background between white and black
R	Rescale normals toggle -
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
Y	Grid Segments, adds (or SHIFT-Y to subtract) 2D and 3D layers effected by the active aXes (Z axis creates 3D grid layers)

--- 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 file is in CSV format (Comma Separated Values). The first line is blank, the 2nd line contains the field names for the columns. Individual nodes start with the 3rd line and continue to the last node. parameters are either 32bit integers or 32bit signed floats.

The current software version writes out 3 tables into a single file. (*This may change in the future such that only 1 table per file exists....) At this time you may ignore all but the first table, as the software only reads in the first table. (If your curious, the 2nd table is for auxiliary data specific to the node type, and the 3rd table is a list of child nodes attached to each node referenced by its nodeID.)

Tree Topology

Perhaps the most difficult aspect of formating the file is properly defining the hierarchy of nodes in the tree topology. The file is sequentially processed such that the order of the nodes in the file coupled with the branchLevel determines the hierarchy. At this time, the order is defined by a recursive algorithm that traverses the tree. Starts with the root pin then adds the first branch (the primary toroid.) The first child node of the primary toroid is next, followed by its first child and so on until it reaches the end of the branch (leaf.) It then goes backs up as many levels required to find a sibling and then down the sibling branch following its child nodes to the leaf. This pattern repeats until all nodes in the tree are created.

*An update is planned (very soon!) to make the file read function agnostic to the ordering of nodes in the file, so that one must only properly define the 'parent' node ID and the software will re-order as necessary.

Currently a root-pin, (branchLevel = 0) automatically creates a first level child torus, regardless if it is in the file.

*Planned future support for pin's with no primary toroid.

---
node ID
The 'id' field is used to build the pin tree hierarchy and link to additional node type specific data. The node ID is only unique within the file. When a file is read in a new set of IDs are assigned. This allows for merging multiple files with overlapping IDs without any contention. Note that if you save out a file, read it in, delete and/or add some nodes, then save, you will get a different set of IDs.

*currently the nodeID is ignored and the order of nodes in the file coupled with the branchLevel determine the tree hierarchy of a pin.

'parent' - The 'id' of the parent, except root-pins which are set to zero.

'branchLevel' starts at zero for the root-pin, 1 for the primary toroid, 2 for its child nodes, and so on....

---
recordID

Because the node 'id' is subject to change, the 'recordID' is provided for your use. It is retained by the node and is not modified by the application. This can be used to reference the original DB record that the node represents.

'childCount' - the number of child nodes attached to the node at the current level. Does not include sub-children of the attached child nodes.

---
childIndex - Specifies which child node is actively selected, used mostly for keyboard navigation of the tree topology.

B.2.2 Position

Translation is determined relative to the global origin for root-pins and relative to the parent node for all child nodes.

The child node radial offset from it's parent node is set by 'translate x'.

Typically you only set velocity rates for rotation as an object with a translational rate will continue to move until it disappears by leaving the scene boundaries/clipping-plane.

Root-pin global position is determined by 'translate x/y/z' and 'rotate_x/y/z'. The translational velocity is set by 'translateRate x/y/z' and is the delta distance applied per cycle (typically runs at 60 cycles per second.)

Rotation

The 'rotate x/y/z' sets the rotational position of the node in degrees (other then the camera.) '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 'rotateRate x/y/z', but at this time root-pins only rotate using the 'x' axis. Child nodes rotate about the 'x' and 'y' axis. Note that the rate is applied per cycle (60Hz) and is in radians, so a value of 0.01 is equivalent to about 6 rpm.

Scaling

The size of an object set by its 'scale x/y/z'. A scale value of 1.0 is equivalent to 100% and results in no change in size. Valid scale values can be negative or positive, negative values result 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 can have different scale values which result in a non-uniform scaling, or a stretched looking object.

Ratio (inner radius of a torus)

*The 'ratio' sets the inner radius of the torus as a factor of its size. Generally between 0.01 and 1.0, where 1.0 is a donut with no hole and 0.01 is a very thin donut... will be implemented shortly. Currently the default ratio is 0.1 equivalent to 10% of the outer radius.

Texture Maps

The 'textureID' specifies the texture map of an object. A textureID = 0 results in no texture map, and is the default for most objects. The primary grid has a default textureID = 1 which corresponds to the file 'map00001.jpg'. For more detail see the 'File - Texture Maps' section.

Color

The 'color r/g/b' is set by the colorIndex from a palette of 20 colors. Additionally the alpha transparency is set by 'color a'. Note that you may set the color to any custom RGB and alpha you choose, but if the user changes the color, the custom color will be lost. In general, match the RGB values to the colorIndex using the key table (at the end of this appendix section.)

B.2.2 Camera Position

There are four cameras available to the user, nodes 3, 4, 5 & 6. They are all attached as child nodes to a root-camera which is node 2. Effectively all the root camera does is determine which of the child node cameras is currently active using it childIndex.

You can set a cameras position in world coordinates using the 'translate x/y/z' and point them using 'rotateRad x/y/z'. The 'rotate x/y/z/s' vector can be ignored since it is calculated from 'rotateRad' and is automatically updated.

'rotateRad' is in polar coordinates using radians. The 'x' axis will tilt the camera up and down, the 'z' axis will rotate right and left. The 'y' axis is unused at this time.

The vertical tilt 'rotateRad x' is restricted from -1.57 (-pi/2 is straight up) to 1.57 (pi/2 straight down,) x = 0.0 is level.

Horizontal, left right rotation 'rotateRad z' is restricted from 0.0 to 6.28 (2pi). Looking straight down rotation is clockwise.

B.3 State File - Field Descriptions

Field name	type	description
id	int	nodeID 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 set, 0 if not
parent	int	ID of parent node
branchLevel	int	root node is 0, each sub-level is 1, 2, 3, 4...n
childArray	int	same as nodeID
childIndex	int	index of the currently selected child node
childCount	int	number of child nodes attached (max of 16)
channel x	int	channel number
channel y	int	channel number
channel z	int	channel number
channelIndex x	int	the cycle update index (current time-stamp)
channelIndex y	int	the cycle update index (current time-stamp)
channelIndex z	int	the cycle update index (current time-stamp)
averageType x	int	*type of averaging used for channel data
averageType y	int	*type of averaging used for channel data
averageType z	int	*type of averaging used for channel data
sampleInterval x	int	*number of samples per average
sampleInterval y	int	*number of samples per average
sampleInterval z	int	*number of samples per average
rotate x	float	rotation vector
rotate y	float	rotation vector
rotate z	float	rotation vector
rotate s	float	rotation vector
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	position, root pin world coordinates, child node offset
translate y	float	position, root pin world coordinates
translate z	float	position, root pin world coordinates
origin x	float	reserved
origin y	float	reserved
origin z	float	reserved
rotateRate x	float	rotational velocity rate
rotateRate y	float	rotational velocity rate
rotateRate z	float	rotational velocity rate
rotateRad x	float	polar coordinates used to calculate the rotate vector
rotateRad y	float	polar coordinates used to calculate the rotate vector
rotateRad z	float	polar coordinates used to calculate the rotate vector
scaleRate x	float	rate of scaling applied per cycle
scaleRate y	float	rate of scaling applied per cycle
scaleRate z	float	rate of scaling applied per cycle
translateRate x	float	translational velocity rate applied per cycle
translateRate y	float	translational velocity rate applied per cycle
translateRate z	float	translational velocity rate applied per cycle
translateVect x	float	reserved
translateVect y	float	reserved
translateVect 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)
lineWidth	float	line width used for wireframes and line plots
pointSize	float	point size used for point plots
ratio	float	geometry ratio, such as innerRadius of a torus
colorIndex	int	color index from color palette (see key table)
color r	int	8bit RGB color value (typically set by the index color)
color g	int	8bit RGB color value (typically set by the index color)
color b	int	8bit RGB color value (typically set by the index color)
color a	int	8bit alpha transparency of node
colorFade	int	*fades older data points over time
textureID	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
center	int	*centers plot on the current data point
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
scroll	int	*scrolls plots in an oscilloscope fashion
triggerOnHi x	int	if 1 then use the triggerLevel, typically for limits
triggerOnHi y	int	if 1 then use the triggerLevel, typically for limits
triggerOnHi z	int	if 1 then use the triggerLevel, typically for limits
triggerOnLow x	int	if 1 then use the triggerLevel, typically for limits
triggerOnLow y	int	if 1 then use the triggerLevel, typically for limits
triggerOnLow z	int	if 1 then use the triggerLevel, typically for limits
triggerLevelHi x	float	typically the upper limit
triggerLevelHi y	float	typically the upper limit
triggerLevelHi z	float	typically the upper limit
triggerLevelLow x	float	typically the lower limit
triggerLevelLow y	float	typically the lower limit
triggerLevelLow z	float	typically the lower limit
recordID	int	recordID of the external source DB record
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 radius x	float	*reserved for future proximity and collision detection
proximity radius y	float	*reserved for future proximity and collision detection
proximity radius z	float	*reserved for future proximity and collision detection
draw value x	int	*draw the label
draw value y	int	*draw the label
draw value z	int	*draw the label
selection set x	int	*reserved for future support of multiple selection sets
selection set y	int	*reserved for future support of multiple selection sets
selection set z	int	*reserved for future support of multiple selection sets
size	int	size in bytes of memory used per node

B.4 Key Tables

B.4.1 type (node type)

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

B.4.2 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.4.3 shader

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

B.4.3 indexColor

indexColor - key table		
value	RGB value	color desc
0	255, 255, 255	white
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	173, 255, 202	light aqua
11	255, 180, 255	pink
12	0, 152, 255	sky blue
13	185, 255, 0	light green
14	152, 0, 0	dark red
15	50, 101, 101	dark grey
16	127, 127, 255	light purple
17	185, 153, 102	tan
18	197, 82, 0	rust
19	127, 127, 127	grey

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 demo, but works....

To test the sample file follow these steps:

Launch the application.

Rotate the camera so that it is pointing to the grid corner to the right.

press - 'P' - (loads the time based channel CSV file).
press - 'N' - to create a pin.
press - '+' - to change color to green.
press - '<' - this will assign the channels to the object.

The PIN will zoom-in from the distance and change color 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

Glossary

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 different types of nodes.

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

Pin - The entire tree structure that represents a root-pin and all its child-nodes.

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

Child-Node - If attached to a pin then the default geometry is a torus, this can be changed to other geometry types.

Channels - Represent 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.

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

Toroid - Same as torus.

Tori - Multiple toroids.

*additional terms will be added.

Acknowledgements

Synergist:
Dave Warner

Demo Applications:
Allan B Scrivner
Jeff Sale

Coders:
Shane Saxon
Lucas Erickson
Mark Buchanan