The God Core
A Science Fiction Video Game Developed in C++

Author: Jeremy Greenburg
Mentor: Dr. Joshua Guerin
Second Reader: Bob Bradley

February 22, 2017

Contents

1 Preamble

2 Programming

2.1 The Language

2.2 APIs

2.2.1 OpenGL

OpenGL, or the Open Graphics Library, is one of the most popular graphics libraries out there. It gives access to linear algebra functions for matrix manipulation (which is important, as 3D graphics relies heavily upon matrix transformations), keyboard and mouse input, and windowing. I chose to use OpenGL over a different graphics library, such as Microsoft's DirectX, because it is open source and cross platform, which would make porting my game to a different Operating System a much easier task if I ever decide to in the future.

2.2.2 SOIL

SOIL, or the Simple OpenGL Interface Library, is a small extension to OpenGL that I picked up along the way. It is a texture library that can load .jpg and .png images and bind them to an OpenGL texture, making it very simple to incorporate such images into my game.

2.2.3 FMOD

I chose FMOD as the base for my game's audio as it is a simple, lightweight, and free to use sound API, and most other audio APIs that I looked at lacked support for MP3 files.

2.2.4 SQlite

I decided to use SQLite for my database because it is a lightweight simplified version of a SQL database, allowing the game data to be stored and embedded in the application without taking much room or take a great deal of time to perform a query.

2.3 Game Engine

I crafted the engine of my game in C++ over two years starting in my second semester sophomore year and ending the first semester of my senior year. It consists of 53 C++ files and was developed in Microsoft Visual Studio. The code can be found in the Appendix of this writeup or it can be located on GitHub at *https://github.com/Jerrgree/The-God-Core-Source*. The game can also be installed at GitHub.

2.3.1 Rectangles and Triangles

Rectangles and triangles are the two fundamental polygons that build up my game. Rectangles in particular make up the walls, floors, ceilings, doors, terminals, and most of the HUD and menu. They started as simply two arrays- one that holds all 9 (for triangles) or 12 (for rectangles) values describing the coordinates in the game that they inhabit, as well as a 4 value vector containing the objects RGBA values.

For collision purposes, when a rectangle class is expanded with the ability to calculate and store its norm and Plane equation (Form $ax + by + cz + d = 0$).

This equation is calculated using the any three corners of the rectangle (Calling them A, B, and C) as follows:

$$\vec{AB} = \begin{vmatrix} Bx - Ax \\ By - Ay \\ Bz - Az \end{vmatrix} \vec{AC} = \begin{vmatrix} Cx - Ax \\ Cy - Ay \\ Cz - Az \end{vmatrix}$$

$a = \vec{AB}_2 * \vec{AC}_3 - \vec{AB}_3 * \vec{AC}_2$

$b = \vec{AB}_3 * \vec{AC}_1 - \vec{AB}_1 * \vec{AC}_3$

$c = \vec{AB}_1 * \vec{AC}_2 - \vec{AB}_2 * \vec{AC}_1$

$d = aAx + bAy + cAz$

Formula obtained from http://keisan.casio.com/exec/system/1223596129

The norm of the plane can then be derived using the equation $\sqrt{a^2 + b^2 + c^2}$

2.3.2 In Game Terminals

In game terminals are each bound to a *terminal file*, a unique file that contains the contents of its respective terminal.

The terminal file is divided into two sections: the file names and the file contents. This is an example terminal file:

```
1   <FILES>
2   [01] Name1 -- TAG
3   [02] Name2 -- TAG2
4   [03] Name3 -- TAG3
5
6   <TAGS>
7   $HELP$
8   Type Read <num> to read the corresponding file
9   Type Clear to clear a file from the screen
10  Type Exit to exit the terminal
11  Type Help to see this message again
12  $END$
13
14  $TAG$
15  Content 1
16  $END$
17
18  $TAG2$
19  Content 2
20  $END$
21
22  $TAG3$
23  Content 3
24  $END$
```

The program parses the file by first separating the in game content (the bracketed number and name) that should be displayed to the user from it's tag. The tags are stored in an array, where its index is equal to the bracketed number. The help display is always stored at the 0th index.

Then, whenever the player types in a read command (E.G. Read 1), the program will send the terminal file and the correct tag to the text engine for the content to be displayed to the screen.

2.3.3 Triggers and Switches

Triggers are a more sophisticated way to implement interaction between two different objects. The implementation was designed to be abstracted away from object types so that, in theory, any arbitrary object could activate another, but in practice due to the few classes of objects in my game, it served as a way for terminals to power switches on.

The trigger class works by holding two void pointers, one for a triggering object and one for the target object. It also holds the object types (defined in GCTypes.h) of each object. Whenever an object is interacted with, every trigger in the game is attempted to be triggered **(trying to find better phrasing here)** and if the object is the same as the trigger pointer (no referencing needed as the pointers will always be equal), the target is dereferenced according to the appropriate type and activated.

It holds very similar function to a switch, the primary difference being that the switch is a tangible object in the game with the triggering object being itself, and the triggers being an intangible association between two objects. If I had more time in development, I would have liked to refactor much of the switch's internal functionality so that it is simply an object, with the actual interaction between the switch and it's target taking place as a trigger, but the conception and implementation of triggers came too late in development and implementing the switch change would require a good deal of code and data rewrites.

2.3.4   Camera Control

The CameraControl class is designed to control and manipulate the player's perspective as they navigate through the game. It contains two ordered triples of floating point numbers: The xyz location of the player, and the rotation along the x axis (looking left/right), the y axis (up/down), and the z axis (barrel roll). It also contains two additional floating point values, the movement speed and the turning speed.

The player can move forwards and backwards, as well as strafe left and right. To correctly formulate the player's movement, I had to envision a circle centered on the player with a radius of the player's movement speed. Based on the angle from the x and z rotation, the next place that the player move is simply a spot on the circumference of the circle based on the rotation angle, and moving forward can be derived from this formula:

z := z ± moveSpeed * cos(radian(x_angle))

x := x ∓ moveSpeed * sin(radian(x_angle))

Formula obtained from Robert De Yoso

Following that formula, it's simple to implement movement to the left, right by adding or subtracting $90°$, and backwards movement by adding $180°$.

Whenever OpenGL renders a new frame, the 'camera' is always returned to the origin of the map, so after drawing the level and before flushing the buffer, the Camera Control calls glTranslate to move the camera to the correct location, and then calls glRotate 3 times, once for each axis, to orient the camera in the correct direction.

2.3.5   Heads Up Display

The Heads Up Display is drawn after the level is draw, so that it overlays information to the player. It primarily is used to add a bit of flavor to the game by drawing the helmet for the player, but it also serves to display the developer console when activated.

The display also delivers a prompt to the user whenever they are in range of an object that can be interacted with.

2.3.6   2D

As multiple different objects required the ability to **2D IT UP CHANGE THIS SOON**, I extracted the ability to draw in 2D into it's own class and it was inherited whenever it was needed.

To convert OpenGL into 2D frame, I needed to first disable lighting, depth testing and depth masking. Next I pushed an *orthogonal* matrix onto OpenGL's matrix stack using the length and width of the screen so that all matrix transformations corresponded to a pixel on the screen. Re-enabling 3D is as simple as popping the orthogonal matrix from the stack and re-enabling depth testing and masking.

2.3.7   Collision Engine

This determines when the player has collided with an object in the world. There are two types of collisions: player-object collisions and player-wall collisions.

Player object collisions are simple to detect, as both the player and the object can be placed within imaginary "bounding spheres" that extend around the player and object. Collision can be detected with this formula:

$\sqrt{(x_2 - x_1) + (y_2 - y_1) + (z_2 - z_1)} < r_2 + r_1$ If the distance between the two spheres is less than the sum of the radii of the two spheres, the they must be colliding.

Player-wall collisions were much harder to reconcile. Because walls tend to be long and thin, you can't simply place one within a bounding sphere, the resulting sphere would simply be too massive.

To rectify that, the collision is split into two phases: broad and narrow.

In the broad phase, we use the plane equation $ax + by + cz + d$ that is derived in the Rectangle section. We use the formula $\frac{ax+by+cz+d}{\sqrt{a^2+b^2+c^2}}$, where x, y, and z are the player's x, y, and z coordinates. If the resulting value is less than the radius of the player's bounding sphere, the player has hit that plane and we move onto the narrow phase.

In the narrow phase, each wall is aligned on an axis: x, y, or z. We simply take the largest and smallest values of the coordinates on that axis (for instance, if the wall is x aligned, we take the largest and smallest x value). If the sphere is in between the two values, the player has hit the wall. Otherwise, they hit the plane but not the wall.

2.3.8 MusicManager

To play background music, I used the FMOD low level API for C++. FMOD can dynamically load and play as multiple sounds, which can either be set to loop (such as background music) or not to loop (for sound effects). Proper memory management is important, as the individual sounds are dynamically created outside of the Music Manager class and must be allocated and deallocated properly to avoid memory leaks.

2.3.9 TextEngine

The Text Engine was constructed to handle displaying all text to the screen. It uses OpenGL's glutBitmapCharacter function to display clear, concise text.

Every function to display text takes two coordinates (the x,y coordinates on the screen to start displaying the text), and the RGB color values for the text. There are two functions for displaying text, the simpler one merely takes in a string and prints it on the corresponding location on the screen. The more complex function takes in a file and a content tag. The files are structured like so:

```
1  $TAG 1$
2  Content 1
3  $END$
4
5  $Tag 2$
6  Content 2
7  $END$
```

The Text Engine searches through the designated file line by line until it discovers the line containing the proper tag. Then, until it reaches the closing 'END' tag, it stores every line inside of a vector. Once it has retrieved all of the necessary content, it starts to display the text to the screen line by line, starting from the designated XY position and increasing the Y value for each line.

2.3.10 Game Saving and Loading

2.3.11 Keyboard

The Keyboard class primarily serves to encapsulate the OpenGL callbacks that receive keystrokes: the normal function that accepts all alphanumeric and punctuation, and the special function that handles function keys and escape. However, there is a minor bit of overhead that goes into deciding where the input goes.

Under normal circumstances, the only normal keystrokes accepted are the WASD keys for movement, the E key for interaction, and the ' ' key for toggling the development console.

When in either a terminal or the development console, all keys are immediately concatenated to an input string with the exception of the ' ' which will close the development console, or the enter key which will send the input string to it's appropriate destination to be parsed and interpreted, after which the input string is cleared so that a new command can be entered.

Also accepted are the up and down arrow keys, which will cycle through the console/terminals command history.

2.3.12 Level Loading and displaying

Loading each level involves a series of SQL queries through the SQLite API. Loading each level first involves opening a connection with the database, and retrieving all data from each table in the database in turn. All important data from the database is stored in a class of the appropriate type, unnecessary data is discarded, and in the end each class is pushed into a vector of the appropriate type.

The data is loaded in a strict order, due to some objects having dependencies on others (that is, some objects require other objects to already exist). Thus the first things that are loaded are purely independent objects, all doors, walls, and terminals. Next switches are loaded, because they require both doors and terminals to already exist. Finally, the triggers are loaded, because they require both switches and terminals.

When loading switches and triggers, the objects also need to be *bound* to their appropriate target. This is why doors, switches, and terminals all carry their ID's into the program with them, while triggers and walls discard their ID. Once all of the objects that need to be bound are loaded into the game, the game proceeds to bind them to their target. For each switch that needs to be bound, the game loops through either the list of terminals or the list of doors for the appropriate object and creates a pointer to that object inside of the switch, thus ensuring that the switch can toggle its target instantly without needing to search every time it is triggered. The triggers are bound similar, with the difference that each object must perform two searches, one for the triggering object and one for the target object.

If there is any data error in regards to binding — that is, and object attempts to bind to an object that does not exist, the error is considered fatal and the game immediately shuts down after logging the error.

The OpenGL display function calls upon the Level class to display all in game objects. This is a simple matter, because each object has it's own function to display itself. Thus it is a simple matter to loop through each vector and tell each object to display itself.

2.3.13 Console and Logging

3 Appendices

3.1 Source Code

3.1.1 main.cpp

```
1  /*************************************************************\
2   * main.cpp                                                 *
3   * This file was created by Jeremy Greenburg               *
4   * As part of The God Core game for the University of      *
5   * Tennessee at Martin's University Scholars Organization  *
6   *                                                          *
7   * This file creates an OpenGL window to display the game  *
8   * and promptly passes control over to the GameManager object*
9  \*************************************************************/
10
11 // Because doth openGL demandeth
12 #include <cstdlib>
13 // OpenGL API
14 #include <GL\glut.h>
15 // time
16 #include <ctime>
17
18 // The Game manger
19 #include "GameManager.h"
20 GameManager Overlord;
21 // Save manager
22 #include "SaveManager.h"
23 // Return codes
24 #include "Return.h"
25 // System log
```

```
26  #include "Logger.h"
27  // Global variables
28  #include "Globals.h"
29
30  // Normal key presses
31  void normal(unsigned char key, int x, int y);
32
33  // For key releases
34  void key_up(unsigned char key, int x, int y);
35
36  // For Special keys
37  void special(int key, int x, int y);
38
39  // Mouse clicks
40  void mouse(int button, int state, int x, int y);
41
42  // Mouse movement
43  void motionPassive(int x, int y);
44
45  // Changing Window size (Not exactly working as hoped...
46  void changeSize(int w, int h);
47
48  // Initializes GLUT callbacks and returns true if core.sav exists (false otherwise
        )
49  bool initGame(int argc, char **argv);
50
51  // Manages the game's scenes
52  void manageScenes();
53
54  GLfloat light_diffuse[] = { 0.3f, 0.3f, 0.3f, 0.3f };
55  GLfloat light_position[] = { 0.0f, 0.0f, 0.0f, 0.0f }; // Currently nonexistant
        until I can figure out how lighting works
56  GLfloat mat_specular[] = { 1.0f, 1.0f, 1.0f, 1.0f };
57  GLfloat mat_shininess[] = { 75 };
58  GLfloat lmodel_ambient[] = { 0.6f, 0.6f, 0.6f, 1.0f };
59
60  using namespace std;
61
62  //****** FUNCTION DEFINITIONS ******\\
63
64  int main(int argc, char **argv)
65  {
66          Overlord.canContinue = initGame(argc, argv);
67
68          // Begin the game
69          glutMainLoop();
70
71          // If we ever get here, something bad happened
72
73          Logger log;
74          log.logLine("ERROR: GlutMainLoop exited early");
75
76          return EXIT_EARLY;
77  }
78
79  bool initGame(int argc, char **argv)
```

```
80   {
81           // Obliderate log file
82           Logger log;
83           log.nuke();
84
85           // Initialize GLUT
86           glutInit(&argc, argv);
87
88           // Create window
89           glutInitDisplayMode(GLUT_DOUBLE | GLUT_RGBA);
90           glutInitWindowPosition(50, 50);
91           glutInitWindowSize(500, 500);
92           glutCreateWindow("The God Core");
93
94           // register callbacks
95           glutDisplayFunc(manageScenes);
96           glutReshapeFunc(changeSize);
97           glutIdleFunc(manageScenes);
98           glutPassiveMotionFunc(motionPassive);
99           glutMouseFunc(mouse);
100          glutKeyboardFunc(normal);
101          glutKeyboardUpFunc(key_up);
102          glutSpecialFunc(special);
103
104          // Prebuilt function that works transparency
105          glBlendFunc(GL_SRC_ALPHA, GL_ONE_MINUS_SRC_ALPHA);
106
107          // Enable transparency
108          glEnable(GL_BLEND);
109          // Enable depth buffer
110          glEnable(GL_DEPTH_TEST);
111          // Let there be light!
112          glEnable(GL_LIGHTING);
113          // First light source
114          glEnable(GL_LIGHT0);
115
116          // Light properties
117          glMaterialfv(GL_FRONT, GL_SPECULAR, mat_specular);
118          glMaterialfv(GL_FRONT, GL_SHININESS, mat_shininess);
119          glColorMaterial(GL_FRONT_AND_BACK, GL_AMBIENT_AND_DIFFUSE);
120
121          // Light doesnt turn everything grey
122          glEnable(GL_COLOR_MATERIAL);
123
124          glLightfv(GL_LIGHT0, GL_DIFFUSE, light_diffuse);
125          glLightfv(GL_LIGHT0, GL_POSITION, light_position);
126          glLightModelfv(GL_LIGHT_MODEL_AMBIENT, lmodel_ambient);
127
128          glutWarpPointer(300, 300);
129
130          // Start in Fullscreen
131          glutFullScreen();
132
133          srand(time(NULL));
134
135          HUD.setStatus("INFO-WELL");
```

```
136
137          SaveManager SaveSystem;
138          return SaveSystem.checkSave();
139  }
140
141  // Everything below here is just passed along to the overlord
142
143  void mouse(int button, int state, int x, int y)
144  {
145          Overlord.mouse(button, state, x, y);
146  }
147
148  void motionPassive(int x, int y)
149  {
150          Overlord.motionPassive(x, y);
151  }
152
153  void changeSize(int w, int h)
154  {
155          Overlord.changeSize(w, h);
156  }
157
158  void manageScenes()
159  {
160          Overlord.manageScenes();
161  }
162
163  void normal(unsigned char key, int x, int y)
164  {
165          Overlord.normal(key, x, y);
166  }
167
168  void key_up(unsigned char key, int x, int y)
169  {
170          Overlord.key_up(key, x, y);
171  }
172
173  void special(int key, int x, int y)
174  {
175          Overlord.special(key, x, y);
176  }
```

3.1.2 CameraControl.h

```
1   /***************************************************************\
2    * CameraControl.h                                            *
3    * This file was created by Jeremy Greenburg                  *
4    * As part of The God Core game for the University of         *
5    * Tennessee at Martin's University Scholars Organization     *
6    *                                                            *
7    * This file contains the declaration of the CameraControl    *
8    * Class, which stores:                                       *
9    *       The x, y, z ordered triple of the player's location  *
10   *       The degree to which the player is turned, along      *
11   *             the x, y, and z axes                           *
12   * And contains methods to translate the player along         *
13   * 3D space                                                   *
```

```
14  \*************************************************************/
15
16  #ifndef CAMERA_CONTROL_H
17  #define CAMERA_CONTROL_H
18
19  class CameraControl
20  {
21  private:
22          // Speeds for moving and rotating
23          double moveSpeed = 0.1f, turnSpeed = 0.5f;
24
25  public:
26          // Negatively adjusts angle and modifies lx
27          void lookLeft();
28          // Positively adjusts angle and modifies lx
29          void lookRight();
30          // Positively adjusts angle and modifies ly
31          void lookUp();
32          // Negatively adjusts angle and modifies ly
33          void lookDown();
34          // Translate the camera to the left
35          void strafeLeft();
36          // Translates the to the right
37          void strafeRight();
38          // Translates the camera forwards
39          void moveForward(int mod);
40          // Translate the camera backards
41          void moveBackward(int mod);
42          // Moves the camera positively along the Y axis
43          void moveUp();
44          // Moves the camera negatively along the Z axis
45          void moveDown();
46          // Flips the camera
47          void invertCam();
48          // If the player begins to run
49          void increaseSpeed();
50          // If the player begins to walk
51          void decreaseSpeed();
52          // Resets the camera to it's initial state
53          void resetCam();
54          // calls gluLookAt
55          void Display();
56
57          // Location of the camera
58          double x =0.0, y = 0.0, z = -1.0;
59          double prevx, prevz;
60          // Angles of rotation
61          double x_angle = 0.0, y_angle = 0.0, z_angle = -1.0;
62  };
63
64  #endif
```

3.1.3   CameraControl.cpp

```
1  /*************************************************************\
2   * CameraControl.cpp                                         *
3   * This file was created by Jeremy Greenburg                 *
```

```
 4   * As part of The God Core game for the University of        *
 5   * Tennessee at Martin's University Scholars Organization     *
 6   *                                                            *
 7   * This file contains the definition of the CameraControl     *
 8   * Class. For more information, see CameraControl.h           *
 9   \*************************************************************/
10
11  // Class definition
12  #include "CameraControl.h"
13
14  // For sin()
15  #include <cmath>
16
17  // glut is unhappy when cstdlib isn't here :/
18  #include <cstdlib>
19
20  // OpenGL API
21  #include <GL\glut.h>
22
23  // To display Suit Warnings
24  #include "TextEngine.h"
25
26  // To include Globals Variables
27  #include "Globals.h"
28
29  // For converting degrees to radians
30  const double PI = 3.14159;
31
32  // Takes in an angle, in degrees, and returns the angle in radians
33  double toRadian(double angle)
34  {
35          return angle * PI / 180;
36  }
37
38  void CameraControl::lookLeft()
39  {
40          x_angle -= 3 * turnSpeed;
41
42          // To avoid potential underflow errors
43          if (x_angle < 0)
44          {
45                  x_angle += 360;
46          }
47  }
48  void CameraControl::lookRight()
49  {
50          x_angle += 3 * turnSpeed;
51
52          // To avoid potential overflow errors
53          if (x_angle > 360)
54          {
55                  x_angle -= 360;
56          }
57  }
58
59  void CameraControl::lookUp()
```

```
60  {
61          y_angle -= 2 * turnSpeed;
62
63          // To avoid potential underflow errors
64          if (y_angle < 0)
65          {
66                  y_angle += 360;
67          }
68  }
69
70  void CameraControl::lookDown()
71  {
72          y_angle += 2 * turnSpeed;
73
74          // To avoid potential overflow errors
75          if (y_angle > 360)
76          {
77                  y_angle -= 360;
78          }
79  }
80
81  void CameraControl::strafeLeft()
82  {
83          prevz = z;
84          prevx = x;
85          // Angles + 90 degrees for an angle that is perpendicular to x_angle
86          z = z + moveSpeed * cos(toRadian(x_angle + 90));
87          x = x - moveSpeed * sin(toRadian(x_angle + 90));
88  }
89
90  void CameraControl::strafeRight()
91  {
92          prevz = z;
93          prevx = x;
94          // Angles - 90 degrees for an angle that is perpendicular to x_angle
95          z = z + moveSpeed * cos(toRadian(x_angle - 90));
96          x = x - moveSpeed * sin(toRadian(x_angle - 90));
97  }
98
99  void CameraControl::moveForward(int mod)
100 {
101         prevz = z;
102         prevx = x;
103         z = z + moveSpeed * mod * cos(toRadian(x_angle));
104         x = x - moveSpeed * mod * sin(toRadian(x_angle));
105 }
106
107 void CameraControl::moveBackward(int mod)
108 {
109         prevz = z;
110         prevx = x;
111         z = z - moveSpeed * mod * cos(toRadian(x_angle));
112         x = x + moveSpeed * mod * sin(toRadian(x_angle));
113 }
114
115 void CameraControl::moveUp()
```

```
116  {
117          y -= moveSpeed;
118  }
119
120  void CameraControl::moveDown()
121  {
122          y += moveSpeed;
123  }
124
125  void CameraControl::invertCam()
126  {
127          z_angle += 180;
128  }
129
130  void CameraControl::resetCam()
131  {
132          x = 0.0;
133          y = 0.0;
134          z = -1.0;
135          x_angle = 0.0;
136          y_angle = 0.0;
137          z_angle = 0.0;
138
139  }
140
141  void CameraControl::increaseSpeed()
142  {
143          moveSpeed *= 2;
144  }
145
146  void CameraControl::decreaseSpeed()
147  {
148          moveSpeed /= 2;
149  }
150
151  void CameraControl::Display()
152  {
153          // To stop eternal movement
154          glLoadIdentity();
155
156          // Rotate along proper axes
157          glRotatef(y_angle, 1, 0, 0);
158          glRotatef(x_angle, 0, 1, 0);
159          glRotatef(z_angle, 0, 0, 1);
160
161          // Translate along the Plane
162          glTranslatef(x, y, z);
163  }
```

3.1.4   CollisionEngine.h

```
1  /************************************************************\
2   * CollisionEngine.h                                        *
3   * This file was created by Jeremy Greenburg               *
4   * As part of The God Core game for the University of       *
5   * Tennessee at Martin's University Scholars Organization   *
6   *                                                          *
```

```
 7   * This file creates the decleration of the CollisionEngine  *
 8   * class, which uses sweet sweet math to determine how the    *
 9   * player interacts with his environment                      *
10   \**************************************************************/
11
12   #ifndef COLLISION_ENGINE_H
13   #define COLLISION_ENGINE_H
14
15   class CollisionEngine
16   {
17   private:
18           // Determines if wall/door collision occured
19           bool collideWalls();
20           // Determines if other collision occured
21           bool collideObjects();
22           // Determines if an object can be interacted with
23           void checkInteract();
24   public:
25           // Master function that calls others
26           bool collide();
27
28   };
29
30   #endif
```

3.1.5 CollisionEngine.cpp

```
 1   /**************************************************************\
 2    * CollisionEngine.h                                          *
 3    * This file was created by Jeremy Greenburg                  *
 4    * As part of The God Core game for the University of         *
 5    * Tennessee at Martin's University Scholars Organization     *
 6    *                                                            *
 7    * This file contains the definition of the CollisionEngine   *
 8    * class. For more information, see SaveManager.h             *
 9    \**************************************************************/
10
11   #include "CollisionEngine.h"
12
13   // For the Cam
14   #include "Globals.h"
15   // absolute value
16   #include <cmath>
17
18   // System Log
19   #include "Logger.h"
20
21   using namespace std;
22
23   const double PLAYER_RADIUS = 0.5;
24   const double INTERACT_RADIUS = 1; // Object interactivity radius
25   const double COLLIDE_RADIUS = 0.5;
26
27   void CollisionEngine::checkInteract()
28   {
29           activeSwitch = NULL;
30           activeTerminal = NULL;
```

```
31            // Auto don't work in these parts
32            for (unsigned int i = 0; i < switches.size(); i++)
33            {
34                    double distance = pow((switches[i].getX() + Cam.x), 2) + pow((
                          switches[i].getY() + Cam.y), 2) + pow((switches[i].getZ() + Cam
                          .z), 2);
35                    distance = sqrt(distance);
36                    double radii = (PLAYER_RADIUS + INTERACT_RADIUS);
37
38                    if (distance < radii && switches[i].checkIfOn())
39                    {
40                            interactivity = true;
41                            activeSwitch = &switches[i];
42                            return;
43                    }
44            }
45
46            for (unsigned int i = 0; i < terminals.size(); i++)
47            {
48                    double distance = pow((terminals[i].getX() + Cam.x), 2) + pow((
                          terminals[i].getY() + Cam.y), 2) + pow((terminals[i].getZ() +
                          Cam.z), 2);
49                    distance = sqrt(distance);
50                    double radii = (PLAYER_RADIUS + INTERACT_RADIUS);
51
52                    if (distance < radii && terminals[i].checkIfOn())
53                    {
54                            interactivity = true;
55                            activeTerminal = &terminals[i];
56                            return;
57                    }
58            }
59
60            interactivity = false;
61    }
62
63    bool CollisionEngine::collideObjects()
64    {
65            for (unsigned int i = 0; i < terminals.size(); i++)
66            {
67                    double distance = pow((terminals[i].getX() + Cam.x), 2) + pow((
                          terminals[i].getY() + Cam.y), 2) + pow((terminals[i].getZ() +
                          Cam.z), 2);
68                    distance = sqrt(distance);
69                    double radii = (PLAYER_RADIUS + COLLIDE_RADIUS);
70
71                    if (distance < radii && terminals[i].checkIfOn())
72                    {
73                            return true;
74                    }
75            }
76
77            return false;
78    }
79
80    bool CollisionEngine::collideWalls()
```

```
81  {
82          // Gotta check doors first
83          // And if you hit an open door
84          // You just ignore collision
85          // Because otherwise you can't fit
86          for (auto i : doors)
87          {
88                  double distance = fabs(Cam.x * i.a + Cam.y * i.b + Cam.z * i.c + i
                          .d); // Distance from door
89
90                  if ((distance / i.getNorm() < PLAYER_RADIUS) && i.isInBounds())
91                  {
92                          if (i.isOpen) return false;
93                          else return true;
94                  }
95          }
96
97          for (auto i : walls)
98          {
99                  double distance = fabs(Cam.x * i.a + Cam.y * i.b + Cam.z * i.c + i
                          .d); // Distance from wall
100
101                 if ((distance / i.getNorm() < PLAYER_RADIUS) && i.isInBounds())
                          return true;
102         }
103
104         return false;
105 }
106
107 bool CollisionEngine::collide()
108 {
109         if (!collision)
110         {
111                 return false;
112         }
113
114         checkInteract();
115         return (collideWalls() || collideObjects());
116 }
```

3.1.6   Console.h

```
1  /***********************************************************\
2   * Connsole.h                                              *
3   * This file was created by Jeremy Greenburg              *
4   * As part of The God Core game for the University of     *
5   * Tennessee at Martin's University Scholars Organization  *
6   *                                                         *
7   * This file contains the declaration of the Console Class,*
8   * As well as the Trip struct for holding three integers   *
9   * The Developer Console takes input from the user and     *
10  * Activates various effects based upon what the user has  *
11  * Typed in.                                               *
12  \***********************************************************/
13
14 #ifndef CONSOLE_H
15 #define CONSOLE_H
```

```cpp
16
17  // To act as a circular buffer for console history
18  #include <deque>
19  // Stores actual console input
20  #include <vector>
21  // std::string
22  #include <string>
23  // For processing text
24  #include "TextEngine.h"
25
26  // Windows API
27  #include <shlobj.h>
28
29
30  // To make rgb calues easier to store
31  #include "Triple.h"
32
33  class Console
34  {
35  private:
36          /***** Variables for the console itself ****/
37
38          // Triples for good color, bad color, and nuetral colors
39          Triple VALID_COLOR, INVALID_COLOR, NEUTRAL_COLOR;
40          // What the console "says" (aka what appears on screen)
41          std::deque<std::string> console_log;
42          // The colors of said strings
43          std::deque<Triple> console_color;
44          // Contains the actual player input
45          std::vector<std::string> console_input;
46          // The current (finished) input being processed
47          std::string currentInput;
48          // The current (unfinished) input being type
49          std::string currentText;
50          // Console History
51          TextEngine log;
52
53          // Path to core.sav
54          char CHAR_PATH[MAX_PATH];
55          std::string SAVE_PATH;
56
57          // Is the console active or not
58          bool isActive;
59
60          // The bottom of the console
61          const int SCREENBOTTOM = 500;
62
63          // Prints the current input and console_history
64          void printInput();
65          // Processes completed input
66          void processInput();
67
68          // Command functions
69
70          // Toggles collision on and off
71          void toggleCollision();
```

```
72
73        // Toggles godMode on and off
74        void toggleGod();
75
76        // Decrpyts the entry in core.sav
77        void decrpytSave();
78
79        // Shutdowns program
80        void halt();
81
82        // Clears the console log
83        void clear();
84
85        // Writes input to core.sav
86        void writeToSave(std::string input);
87
88        // Reads a bit from the file
89        void readFromFile(std::string input);
90
91        // Changes the currently played track
92        void playSong(std::string input);
93
94  public:
95        // Initializes VALID_COLOR, INVALID_COLOR, NEUTRAL_COLOR, and SAVE_PATH
96        Console();
97        // Manages console functions if input has been provided
98        void activate(std::string input, std::string text);
99        // Manages console function if input is still being provided
100        void activate(std::string text);
101        // Returns the console_input[count]
102        std::string getHist(int count);
103        // Returns console_input.size()
104        int getHistNum();
105
106 };
107
108 #endif
```

3.1.7 Console.cpp

```
1  /************************************************************\
2   * Console.cpp                                              *
3   * This file was created by Jeremy Greenburg               *
4   * As part of The God Core game for the University of       *
5   * Tennessee at Martin's University Scholars Organization   *
6   *                                                          *
7   * This file contains the definition of the Console class   *
8   * For more information, see Console.cpp                    *
9  \************************************************************/
10
11 // File I/O
12 #include <fstream>
13
14 // Class declaration
15 #include "Console.h"
16
17 // For saving and loading
```

```
18  #include "SaveManager.h"
19
20  // System log
21  #include "Logger.h"
22
23  // Contains global environment variables
24  #include "Globals.h"
25
26  // Return codes
27  #include "Return.h"
28
29  using namespace std;
30
31  Console::Console()
32  {
33          // Green!
34          VALID_COLOR = makeTrip(0, 1, 0);
35          // Red!
36          INVALID_COLOR = makeTrip(1, 0, 0);
37          // Gray!
38          NEUTRAL_COLOR = makeTrip(1, 1, 1);
39
40          // Get path to documents
41          HRESULT ret = SHGetFolderPath(NULL, CSIDL_PERSONAL, NULL,
                  SHGFP_TYPE_CURRENT, CHAR_PATH);
42          // Assign to SAVE_PATH
43          SAVE_PATH = CHAR_PATH;
44          // Concatenate save file
45          SAVE_PATH += "\\The God Core\\core.sav";
46  }
47
48  void Console::activate(string input, string text)
49  {
50          currentInput = input;
51          // This should be empty. But just incase.
52          currentText = text;
53
54          processInput();
55          printInput();
56  }
57
58  void Console::activate(string text)
59  {
60          currentText = text;
61
62          printInput();
63  }
64
65  void Console::printInput()
66  {
67          deque<string>::iterator it = console_log.begin();
68          deque<Triple>::iterator jt = console_color.begin();
69          // Iterates through the console's current log and prints it to the screen
70          for (it; it != console_log.end(); it++, jt++)
71          {
72                  //                                        Index of it
```

```
73                      log.printString(0, 10 + 10 * (it - console_log.begin()),
74                              jt->a, jt->b, jt->c, *it);
75              }
76
77              // Prints whatever the user is typing
78              log.printString(0, SCREENBOTTOM / 2.4, 1, 1, 1, currentText);
79  }
80
81  void Console::processInput()
82  {
83              // TODO: Break this behemoth up into little, managable functions
84
85              if (currentInput == "TogClip")
86                      toggleCollision();
87
88              else if (currentInput == "TogGod")
89                      toggleGod();
90
91              else if (currentInput.substr(0, 5) == "Save ")
92                      writeToSave(currentInput.substr(5)); // Save everything after "
                            Save "
93
94              else if (currentInput == "Decrypt")
95                      decrpytSave();
96
97              else if (currentInput.substr(0, 5) == "Read ")
98                      readFromFile(currentInput.substr(5)); // Read everything after "
                            Read "
99
100             else if (currentInput == "Halt")
101                     halt();
102
103             else if (currentInput == "Clear")
104                     clear();
105
106             else if (currentInput.substr(0, 5) == "Play ")
107                     playSong(currentInput.substr(5)); // Process everything after "
                            Play "
108
109             else if (currentInput == "Goto Main")
110             {
111                     isInMain = true;
112                     isInConsole = false;
113                     HUD.toggleConsole();
114             }
115
116             // Invalid command
117             else
118             {
119                     console_log.push_back("ERROR: Do not recognize \"" + currentInput
                            + '\"');
120                     console_color.push_back(INVALID_COLOR);
121             }
122
123             // Clears the top of the console if too much history is added
124             if (console_log.size() > 9)
```

```cpp
125              {
126                      console_log.pop_front();
127                      console_color.pop_front();
128              }
129
130              // Store the current input
131              console_input.push_back(currentInput);
132  }
133
134  void Console::writeToSave(string input)
135  {
136              // Writes whatever is in input to the save file.
137              // Probably not going to be good for loading purposes
138
139              SaveManager Jesus;
140
141              Jesus.saveLevel();
142
143              console_log.push_back("Saved: " + input);
144              console_color.push_back(VALID_COLOR);
145  }
146
147  void Console::readFromFile(string input)
148  {
149              // Syntax = Read core.sav
150              if (input == "core.sav")
151              {
152                      ifstream infile(SAVE_PATH);
153
154                      string text;
155
156                      // For now, core.sav only has one line. Hopefully I'll update this
157                              when I change that
158                      infile >> text;
159
160                      console_log.push_back(text);
161                      console_color.push_back(VALID_COLOR);
162              }
163
164              // Syntax = Read TAG FILE
165              else
166              {
167                      // There should be a space seperating the file and the tag. We
168                              find that space
169                      size_t pos = input.find(' ');
170
171                      // If there ain't no space
172                      if (pos == string::npos)
173                      {
174                              console_log.push_back("ERROR: No tag detected");
175                              console_color.push_back(INVALID_COLOR);
176                      }
177
178                      // Hooray! There's a space
179                      else
180                      {
```

```cpp
179                         string tag = input.substr(0, pos);
180                         string file = input.substr(pos + 1); // +1 to avoid the
                               space
181
182                         const char* TEXT_PATH = "Resources\\Text\\";
183                         string fullPath = TEXT_PATH + file;
184
185                         // Simply to test for the file's existence
186                         ifstream infile(fullPath);
187
188                         string text;
189                         getline(infile, text);
190
191                         // If there ain't no file
192                         if (!infile)
193                         {
194                                 console_log.push_back("ERROR: File \"" + file +
                                     "\" not found");
195                                 console_color.push_back(INVALID_COLOR);
196                         }
197
198                         // Hooray! There's a file
199                         else
200                         {
201                                 console_log.push_back("Reading \"" + file + "\"
                                     with tag \"" + tag + '\"');
202                                 console_color.push_back(VALID_COLOR);
203
204                                 vector<string> readText = log.getText(file, tag);
205
206                                 vector<string>::iterator it;
207
208                                 for (it = readText.begin(); it != readText.end();
                                     it++)
209                                 {
210                                         // Push everything we found into the log
211                                         console_log.push_back(*it);
212                                         console_color.push_back(NEUTRAL_COLOR);
213
214                                         // So we don't grow too much, keep bounds
                                             checking
215                                         if (console_log.size() > 9)
216                                         {
217                                                 console_log.pop_front();
218                                                 console_color.pop_front();
219                                         }
220                                 }
221                         }
222
223                         infile.close();
224                 }
225         }
226 }
227
228 void Console::toggleCollision()
229 {
```

```
230            console_log.push_back("Noclip toggled.");
231            console_color.push_back(VALID_COLOR);
232
233            collision = !collision;
234    }
235
236    void Console::toggleGod()
237    {
238            console_log.push_back("God Mode toggled.");
239            console_color.push_back(VALID_COLOR);
240
241            godMode = !godMode;
242    }
243
244    void Console::decrpytSave()
245    {
246            SaveManager Jesus;
247
248            console_log.push_back(Jesus.readSave());
249            console_color.push_back(VALID_COLOR);
250    }
251
252    void Console::halt()
253    {
254            Logger log;
255            log.logLine("Exiting via console");
256            exit(EXIT_OK);
257    }
258
259    void Console::clear()
260    {
261            console_log.clear();
262            console_color.clear();
263            console_input.clear();
264    }
265
266    void Console::playSong(string input)
267    {
268            int sNum = getSongNum(input);
269
270            if (sNum == -1) // Invalid input
271            {
272                    console_log.push_back("ERROR: " + input + " not a valid song file
                        ");
273                    console_color.push_back(INVALID_COLOR);
274            }
275
276            else // Valid input
277            {
278                    songNum = sNum;
279                    changeSong = true;
280                    string song = getSongName(sNum);
281                    console_log.push_back("Now playing " + song);
282                    console_color.push_back(VALID_COLOR);
283            }
284    }
```

```
285
286   string Console::getHist(int count)
287   {
288           int size = console_input.size();
289           if (console_input.empty())
290           {
291                   return "";
292           }
293
294           // If, somehow, a fool manages to get a variable that is out of bounds
295
296           else if (count >= size)
297           {
298                   return console_input.back();
299           }
300
301           else if (count < 0)
302           {
303                   return console_input.front();
304           }
305
306           else
307           {
308                   return console_input[size - count - 1];
309           }
310   }
311
312   int Console::getHistNum()
313   {
314           return console_input.size();
315   }
```

3.1.8   Cylinder.h

```
 1   /****************************************************************\
 2    * Cylinder.h                                                   *
 3    * This file was created by Jeremy Greenburg                    *
 4    * As part of The God Core game for the University of           *
 5    * Tennessee at Martin's University Scholars Organization       *
 6    *                                                              *
 7    * This file contains the declaration of the Cylinder Class,    *
 8    * Which contains the functionality to load and display a       *
 9    * Cylindrical object in game                                   *
10   \****************************************************************/
11
12   #ifndef CYLINDER_H
13   #define CYLINDER_H
14
15   #include <cstdlib>
16
17   #include <GL\glut.h>
18
19   class Cylinder
20   {
21   private:
22           // A few variables to control the shape of the cylinder
23           double baseRadius, topRadius, height;
```

```
24          int stacks , slices ;
25
26          // Arrays for the location , orientation , and color of the cylinder
27          double translate [3] , rotate [3] , color [4];
28          // A thingamajig for glut
29          GLUquadric *quad ;
30  public :
31          Cylinder ( double _baseRadius , double _topRadius , double _height , int
                  _stacks , int _slices ,
32                  const double (&_translate ) [3] , const double (&_rotate ) [3] , const
                        double (&_color ) [4]) ;
33
34          void Display () ;
35          ~ Cylinder () ;
36  };
37
38  # endif
```

3.1.9 Cylinder.cpp

```
 1  /************************************************************\
 2   * Cylinder . cpp                                           *
 3   * This file was created by Jeremy Greenburg               *
 4   * As part of The God Core game for the University of       *
 5   * Tennessee at Martin's University Scholars Organization   *
 6   *                                                         *
 7   * This file contains the defintion of the Cylinder class.  *
 8   * for more information , see Cylinder .h                   *
 9  \************************************************************/
10
11  # include " Cylinder .h"
12
13  // For copying
14  # include < iterator >
15  # include < utility >
16
17  using namespace std ;
18
19  Cylinder :: Cylinder ( double _baseRadius , double _topRadius , double _height , int
        _stacks , int _slices ,
20          const double (&_translate ) [3] , const double (&_rotate ) [3] , const double (&
                _color ) [4])
21  {
22          baseRadius = _baseRadius ;
23          topRadius = _topRadius ;
24          height = _height ;
25          stacks = _stacks ;
26          slices = _slices ;
27
28          copy ( begin ( _color ) , end ( _color ) , color ) ;
29          copy ( begin ( _translate ) , end ( _translate ) , translate ) ;
30          copy ( begin ( _rotate ) , end ( _rotate ) , rotate ) ;
31
32          quad = gluNewQuadric () ;
33  }
34
35  Cylinder ::~ Cylinder ()
```

```
36  {
37          //gluDeleteQuadric ( quad );
38  }
39
40  void Cylinder :: Display ()
41  {
42          glColor4d ( color [0] , color [1] , color [2] , color [3]);
43
44          glPushMatrix ();
45
46          glTranslated ( translate [0] , translate [1] , translate [2]);
47          glRotated ( rotate [0] , 1 , 0 , 0);
48          glRotated ( rotate [1] , 0 , 1 , 0);
49          glRotated ( rotate [2] , 0 , 0 , 1);
50
51          gluCylinder ( quad , baseRadius , topRadius , height , slices , stacks );
52
53          glPopMatrix ();
54  }
```

3.1.10   Door.h

```
 1  /**************************************************************\
 2   * Door.h                                                    *
 3   * This file was created by Jeremy Greenburg                 *
 4   * As part of The God Core game for the University of         *
 5   * Tennessee at Martin's University Scholars Organization    *
 6   *                                                            *
 7   * This file contains the declaration of the Door class       *
 8   * It's mostly a fancy wrapper for a Plane with a bit          *
 9   * Of added functionality                                     *
10  \**************************************************************/
11
12  #ifndef DOOR_H
13  #define DOOR_H
14
15  // Class decleration
16  #include "Plane.h"
17  // std::string
18  #include <string>
19
20  // Figure out a way to bind a controller to the door to activate it.
21  class Door
22  {
23  private:
24          // Name, so a switch can find it
25          std::string id;
26          // The physical door
27          Plane rect;
28  public:
29          // Is the door open?
30          bool isOpen;
31          // Plane's a, b, c, and d.
32          // For easier access
33          double a, b, c, d;
34
35          // Takes in the initial Plane and name
```

```
36          Door(Plane _rect, std::string _id);
37          // Calls rect.Display()
38          void Display();
39          // Returns rect.getNorm()
40          double getNorm();
41          // Returns id
42          std::string getID();
43          // Returns rect.isInBounds()
44          bool isInBounds();
45  };
46
47  #endif
```

3.1.11 Door.cpp

```
1   /**************************************************************\
2    * Door.cpp                                                   *
3    * This file was created by Jeremy Greenburg                  *
4    * As part of The God Core game for the University of         *
5    * Tennessee at Martin's University Scholars Organization     *
6    *                                                            *
7    * This file contains the defintion of the Door class.        *
8    * for more information, see Door.h                           *
9   \**************************************************************/
10
11  // Class declaration
12  #include "Door.h"
13
14  using namespace std;
15
16  Door::Door(Plane _rect, std::string _id) : rect(_rect), id(_id)
17  {
18          isOpen = false;
19          a = rect.a;
20          b = rect.b;
21          c = rect.c;
22          d = rect.d;
23  };
24
25  void Door::Display()
26  {
27          if (!isOpen) rect.Display();
28  }
29
30  double Door::getNorm()
31  {
32          return rect.getNorm();
33  }
34
35  string Door::getID()
36  {
37          return id;
38  }
39
40  bool Door::isInBounds()
41  {
42          return rect.isInBounds();
```

```
43  }
```

3.1.12 GameManager.h

```
1   /**************************************************************\
2    * GameManager.h                                              *
3    * This file was created by Jeremy Greenburg                  *
4    * As part of The God Core game for the University of         *
5    * Tennessee at Martin's University Scholars Organization     *
6    *                                                            *
7    * This file contains the declaration of the GameManger class*
8    * Which oversees and manages the flow of the game            *
9   \**************************************************************/
10
11  #ifndef GAMEMANAGER_H
12  #define GAMEMANAGER_H
13
14  //****** LIBRARIES AND CLASSES ******\\
15
16  // For the keyboard functionality
17  #include "Keyboard.h"
18
19  // glut really wants cstdlib here
20  #include <cstdlib>
21
22  // For arrays of strings
23  #include <string>
24  #include <vector>
25
26  // OpenGL API
27  #include <GL\glut.h>
28
29  // Standard I/O for debugging
30  #include <iostream>
31
32  // To manage background music
33  #include "MusicManager.h"
34
35  // To manage saving and loading
36  #include "SaveManager.h"
37
38  class GameManager
39  {
40  private:
41          // Variables
42
43          // Objects
44          MusicManager SoundSystem;
45          Keyboard board;
46
47          // Because the main menu is dumb, we have to know when to get a click
48          bool processClick = false;
49
50          // When in the main menu, mouse coords of a click
51          int mouse_x, mouse_y;
52
53          // Functions
```

```
54
55  public :
56
57          // Captures mouse clicks
58          void mouse(int button , int state , int x, int y);
59          // Captures mouse motion
60          void motionPassive(int x, int y);
61          // CHanges window size
62          void changeSize(int w, int h);
63          // Manages scene display
64          void manageScenes ();
65          // Displaying function
66          void draw ();
67          // Function to bring about game end on Level 4
68          void endGame ();
69          // Normal key presses
70          void normal(unsigned char key , int x, int y);
71          // Key releases
72          void key_up(unsigned char key , int x, int y);
73          // Special keys
74          void special(int key , int x, int y);
75          // To manage playing and releasing music
76          void manageMusic ();
77
78          // Wether or not core.sav exists
79          bool canContinue ;
80
81  };
82
83  #endif
```

3.1.13   GameManager.cpp

```
 1  /**************************************************************\
 2   * GameManager . cpp                                         *
 3   * This file was created by Jeremy Greenburg               *
 4   * As part of The God Core game for the University of       *
 5   * Tennessee at Martin 's University Scholars Organization  *
 6   *                                                          *
 7   * This file contains the defintion of the GameManager class.*
 8   * for more information , see GameManager .h                 *
 9  \**************************************************************/
10
11  // Class declaration
12  #include "GameManager .h"
13  // Globals
14  #include "Globals .h"
15  // Level
16  #include "Level .h"
17  // Main Menu
18  #include "MainMenu .h"
19
20  #include "Logger .h"
21
22  #include "Return .h"
23
24  using namespace std ;
```

```
25
26  void GameManager::mouse(int button, int state, int x, int y)
27  {
28          if (button == GLUT_RIGHT_BUTTON)
29          {
30                  // Jokes on my I never ended up using the right button
31                  if (state == GLUT_DOWN)
32                  {
33
34                  }
35
36                  else
37                  {
38
39                  }
40          }
41
42          else if (button == GLUT_LEFT_BUTTON)
43          {
44                  if (state == GLUT_DOWN)
45                  {
46                          if (isInMain)
47                          {
48                                  mouse_x = x;
49                                  mouse_y = y;
50                                  processClick = true;
51                          }
52
53                          Logger log;
54                          vector<string> output = { "X: ", to_string(x), " ", "Y:",
                                  to_string(y) };
55                          log.logLine(output);
56                  }
57
58                  else
59                  {
60
61                  }
62          }
63  }
64
65  void GameManager::motionPassive(int x, int y)
66  {
67          static int _x = 0, _y = 0;
68
69          // If nothing else is happening basically
70          if (!isInConsole && !isInTerminal && !isInMain)
71          {
72                  if (x > _x)
73                  {
74                          Cam.lookRight();
75                          _x = x;
76                  }
77
78                  else if (x < _x)
79                  {
```

```cpp
80                          Cam.lookLeft();
81                          _x = x;
82                  }
83
84                  if (y < _y)
85                  {
86                          Cam.lookUp();
87                          _y = y;
88                  }
89
90                  else if (y > _y)
91                  {
92                          Cam.lookDown();
93                          _y = y;
94                  }
95
96                  // Loop around to the other side of the screen
97
98                  bool updateMouse = false;
99                  int newY = y, newX = x;
100                 if (y == 0 || y > 700)
101                 {
102                         updateMouse = true;
103                         newY = 300;
104                         _y = 300;
105                 }
106
107                 if (x == 0 || x > 700)
108                 {
109                         updateMouse = true;
110                         newX = 300;
111                         _x = 300;
112                 }
113
114                 if (updateMouse)
115                 {
116                         glutWarpPointer(newX, newY);
117                 }
118         }
119 }
120
121 void GameManager::changeSize(int w, int h)
122 {
123         // Don't want to divide by zero
124         if (h == 0)
125                 h = 1;
126
127         double ratio = w * 1.0 / h;
128
129         // Use the Projection Matrix
130         glMatrixMode(GL_PROJECTION);
131
132         // Reset Matrix
133         glLoadIdentity();
134
135         // Set the viewport to be the entire window
```

```cpp
136            glViewport(0, 0, w, h);
137
138            // Set the correct perspective.
139            gluPerspective(45, ratio, 1, 100);
140
141            // Get Back to the Modelview
142            glMatrixMode(GL_MODELVIEW);
143    }
144
145    void GameManager::draw()
146    {
147            if (loading)
148            {
149                    lvl.loadLevel(curr_level);
150
151                    loading = false;
152
153                    // Save current progress after loading level
154                    SaveManager Jesus; // saves
155                    Jesus.saveLevel();
156            }
157
158            else
159            {
160                    lvl.displayLevel();
161            }
162    }
163
164    void GameManager::endGame()
165    {
166            if (loading)
167            {
168                    lvl.loadLevel(curr_level);
169
170                    loading = false;
171
172                    // Save current progress after loading level
173                    SaveManager Jesus; // saves
174                    Jesus.saveLevel();
175            }
176
177            else
178            {
179                    // The time left for each segment
180                    static int timeLeft = 1000;
181                    // The last level is divided into 3 segments
182                    static int segment = 1;
183                    // Wether the current segment has been initialized yet
184                    static bool initSegment = true;
185
186                    // The last portion of the game is divided into 3 segments
187                    if (segment == 1)
188                    {
189                            glClearColor(1, 1, 1, 1);
190                    }
191
```

```
192                    else if (segment == 2)
193                    {
194                            if (initSegment)
195                            {
196                                    HUD.displayWarning("");
197                                    HUD.displayWarning("QUANT");
198                                    initSegment = false;
199                            }
200
201                            for (unsigned i = 0; i < walls.size(); i++)
202                            {
203                                    walls[i].mutate();
204                            }
205                    }
206
207                    else if (segment == 3)
208                    {
209                            if (initSegment)
210                            {
211                                    HUD.goFade(15);
212                                    HUD.setStatus("INFO-UN");
213                                    initSegment = false;
214                            }
215
216                            for (unsigned i = 0; i < walls.size(); i++)
217                            {
218                                    walls[i].mutate();
219                            }
220                    }
221
222                    else
223                    {
224                            HUD.setStatus("INFO-WELL");
225                            isInMain = true;
226                            songNum = 0;
227                            changeSong = true;
228                    }
229
230                    // Switch segments
231                    if (timeLeft == 0)
232                    {
233                            timeLeft = 1000;
234                            segment++;
235                            initSegment = true;
236                    }
237                    timeLeft--;
238
239
240                    // Draw the titular object
241                    glPushMatrix();
242                    glTranslated(0, 0, -7);
243                    glColor4d(.9, .9, .9, 1);
244                    glutSolidSphere(3, 50, 50);
245                    glPopMatrix();
246
247                    lvl.displayLevel();
```

```
248                 }
249         }
250
251         void GameManager::manageScenes()
252         {
253                 // If we need to change the song, we can do it here
254                 if (changeSong)
255                 {
256                         manageMusic();
257                 }
258
259                 // Clears the previous drawing
260                 glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
261
262                 if (isInTerminal)
263                 {
264                         activeTerminal->DisplayScreen();
265                 }
266
267                 else if (isInMain)
268                 {
269                         // Enable using textures (pictures)
270                         glutSetCursor(GLUT_CURSOR_LEFT_ARROW);
271                         static MainMenu MM;
272
273                         // For some reason, MM breaks horribly when it's a global or class
274                              member
274                         // So we'll just handle mouse clicks in the display function
275                         // Rather than the mouse click function
276                         // Because I'm a competent programmer
277                         if (processClick)
278                         {
279                                 MM.getClick(mouse_x, mouse_y);
280                                 processClick = false;
281                         }
282
283                         MM.display();
284                 }
285
286                 // glutSetCursor(GLUT_CURSOR_LEFT_ARROW); Keypads maybe?
287
288                 else
289                 {
290                         // Enable using textures (pictures)
291                         glutSetCursor(GLUT_CURSOR_NONE);
292
293                         if (curr_level != "LEVELFOUR") draw();
294
295                         else endGame();
296
297                         // Moves the camera to the correct position
298                         Cam.Display();
299
300                         // Prompt the user to interact if we should
301                         if (interactivity) HUD.displayWarning("INTERACT");
302                         else if (curr_level != "LEVELFOUR") HUD.displayWarning("");
```

```
303
304                    // Prints the HUD
305                    HUD.DisplayHUD();
306            }
307
308            // Displays the current drawing
309            glutSwapBuffers();
310 }
311
312 void GameManager::manageMusic()
313 {
314            // All variables need to persist between frames
315            static SoundClass background;
316
317            SoundSystem.releaseSound(background);
318            changeSong = false;
319
320            // Because you can never have too much bounds checking
321            if (songNum >= 0 && songNum <= 9)
322            {
323                    std::string song = getSongName(songNum);
324                    SoundSystem.makeSound(&background, song.c_str());
325                    SoundSystem.playSound(background);
326            }
327 }
328
329 // Normal key presses
330 void GameManager::normal(unsigned char key, int x, int y)
331 {
332            board.normal(key, x, y);
333 }
334
335 // Key releases
336 void GameManager::key_up(unsigned char key, int x, int y)
337 {
338            board.key_up(key, x, y);
339 }
340
341 // Special keys
342 void GameManager::special(int key, int x, int y)
343 {
344            board.special(key, x, y);
345 }
```

3.1.14   GCTypes.h

```
 1  /***********************************************************\
 2   * GCTypes.h                                               *
 3   * This file was created by Jeremy Greenburg               *
 4   * As part of The God Core game for the University of      *
 5   * Tennessee at Martin's University Scholars Organization  *
 6   *                                                         *
 7   * This file contains integer types corresponding to various *
 8   * In game object types                                    *
 9  \***********************************************************/
10
11  #ifndef GC_TYPES_H
```

```
12  #define GC_TYPES_H
13
14  // Object Types
15
16  #define T_NULL 0                    // Nothing
17  #define T_DOOR 1                    // Door
18  #define T_TERMINAL 2    // Terminal
19  #define T_SWITCH 3                  // Switch
20  #define T_LEVEL_END 4   // Switch that ends level
21
22  typedef int GCtype;
23
24  #endif
```

3.1.15   Globals.h

```
 1  /************************************************************\
 2   * Globals.h                                               *
 3   * This file was created by Jeremy Greenburg               *
 4   * As part of The God Core game for the University of       *
 5   * Tennessee at Martin's University Scholars Organization   *
 6   *                                                          *
 7   * This file contains the declaration of the Globals        *
 8   * All of them.                                             *
 9   * Thers a lot of them                                      *
10  \************************************************************/
11
12  #ifndef GLOBALS_H
13  #define GLOBALS_H
14
15  // ALLLLLLL the classes
16  #include "HeadsUpDisplay.h"
17  #include "CameraControl.h"
18  #include "Level.h"
19  #include "Terminal.h"
20  #include "Door.h"
21  #include "Switch.h"
22  #include "Plane.h"
23  #include "Trigger.h"
24  #include "Cylinder.h"
25
26  // Remember that if you're doing anything else, globals are bad.
27  // But we're in the hellscape that is graphics
28  // There are no rules here
29  // Only madness dwells here
30
31  // Typedefs make life easy
32  typedef std::vector<Plane> vr;
33  typedef std::vector<Door> vd;
34  typedef std::vector<Switch> vs;
35  typedef std::vector<Terminal> vt;
36  typedef std::vector<Trigger> vtr;
37  typedef std::vector<Cylinder> vc;
38
39  // Pointers to various interactive objects
40  extern Switch *activeSwitch;
41  extern Terminal *activeTerminal;
```

```
42
43  // Vectors containing all of the level's assets
44  extern vr walls;
45  extern vd doors;
46  extern vs switches;
47  extern vt terminals;
48  extern vtr triggers;
49  extern vc cylinders;
50
51  extern bool
52          // Are we colliding / Can we die?
53          collision, godMode,
54          // Go dim or go dark?
55          goDim, goDark,
56          // Dunno if I actually need this one
57          loading,
58          // Is in varius different stages of non-normal play
59          isInConsole, isInTerminal, isInMain,
60          // Should we change the song?
61          changeSong,
62          // Is something in interaction range?
63          interactivity;
64
65  // Number of song to change to
66  extern int songNum;
67
68  // Current level (int and string)
69  extern int levelNum;
70  extern std::string curr_level;
71
72  // Constant strings of the song names
73  extern const char *SONG0, *SONG1, *SONG2, *SONG3, *SONG4, *SONG5,
74                                        *SONG6, *SONG7, *SONG8, *SONG9;
75
76  // A few global objects
77  extern HeadsUpDisplay HUD;
78  extern CameraControl Cam;
79  extern Level lvl;
80
81  // Converts a songname to an integer
82  int getSongNum(std::string input);
83  // Converts an integer to a songname
84  std::string getSongName(int input);
85  // Converts a level name to an integer
86  int getLevelNum(std::string input);
87  // Converts level_num to a string in curr_level
88  std::string getLevelString(int input);
89
90  #endif
```

3.1.16   Globals.cpp

```
1  /************************************************************\
2   * Globals.cpp                                              *
3   * This file was created by Jeremy Greenburg               *
4   * As part of The God Core game for the University of       *
5   * Tennessee at Martin's University Scholars Organization   *
```

```
 6   *                                                          *
 7   * This file instantiates the global variables              *
 8   \***********************************************************/
 9
10   #include "Globals.h"
11
12   vr walls;
13   vd doors;
14   vs switches;
15   vt terminals;
16   vtr triggers;
17   vc cylinders;
18
19   Switch *activeSwitch = NULL;
20   Terminal *activeTerminal = NULL;
21
22   bool collision = true;
23   bool godMode = false;
24   bool goDim = false;
25   bool goDark = false;
26   bool loading = true;
27   bool isInConsole = false;
28   bool isInTerminal = false;
29   bool isInMain = true;
30   bool changeSong = true;
31   bool interactivity = false;
32
33   int songNum = 0;
34
35   int levelNum = 0;
36   std::string curr_level = "LEVELZERO";
37
38   const char* SONG0 = "Dark Fog.mp3";
39   const char* SONG1 = "Mismer.mp3";
40   const char* SONG2 = "One Sly Move.mp3";
41   const char* SONG3 = "Hypnothis.mp3";
42   const char* SONG4 = "Cold Hope.mp3";
43   const char* SONG5 = "Spacial Harvest.mp3";
44
45   HeadsUpDisplay HUD;
46   CameraControl Cam;
47   Level lvl;
48
49   int getSongNum(std::string input)
50   {
51           if (input == SONG0 || input == "0")
52                   return 0;
53           if (input == SONG1 || input == "1")
54                   return 1;
55           if (input == SONG2 || input == "2")
56                   return 2;
57           if (input == SONG3 || input == "3")
58                   return 3;
59           if (input == SONG4 || input == "4")
60                   return 4;
61           if (input == SONG5 || input == "5")
```

```
62                  return 5;
63          return -1; // Invalid song
64  }
65
66  std::string getSongName(int input)
67  {
68          std::string ret;
69          switch (input)
70          {
71          case 0: ret = SONG0;
72                  break;
73          case 1: ret = SONG1;
74                  break;
75          case 2: ret = SONG2;
76                  break;
77          case 3: ret = SONG3;
78                  break;
79          case 4: ret = SONG4;
80                  break;
81          case 5: ret = SONG5;
82                  break;
83          default: ret = "\0";;
84                  break;
85          }
86
87          return ret;
88  }
89
90  int getLevelNum(std::string input)
91  {
92          if (input == "LEVELZERO" || input == "LEVELZERO\n")
93                  return 0;
94          if (input == "LEVELONE" || input == "LEVELONE\n")
95                  return 1;
96          if (input == "LEVELTWO")
97                  return 2;
98          if (input == "LEVELTHREE")
99                  return 3;
100         if (input == "LEVELFOUR")
101                 return 4;
102         return -1; // Invalid song
103 }
104
105 std::string getLevelString(int input)
106 {
107         std::string ret;
108         switch (input)
109         {
110         case 0: ret = "LEVELZERO";
111                 break;
112         case 1: ret = "LEVELONE";
113                 break;
114         case 2: ret = "LEVELTWO";
115                 break;
116         case 3: ret = "LEVELTHREE";
117                 break;
```

```
118          case 4: ret = "LEVELFOUR";
119                  break;
120          default: ret = "ERROR";;
121                  break;
122          }
123
124          return ret;
125  }
```

3.1.17 HeadsUpDisplay.h

```
 1  /***************************************************************\
 2   * HeadsUpDisplay.h                                           *
 3   * This file was created by Jeremy Greenburg                  *
 4   * As part of The God Core game for the University of         *
 5   * Tennessee at Martin's University Scholars Organization     *
 6   *                                                            *
 7   * This file contains the declaration of the HeadsUpDisplay   *
 8   * Class, which created an Orthoganl Matrix infront of the    *
 9   * Screen which allows for a 2D Heads Up Display to be        *
10   * Printed before the user at any time                        *
11   * It also passes input to the developer console             *
12  \***************************************************************/
13
14  #ifndef HEADSUPDISPLAY
15  #define HEADSUPDISPLAY
16
17  // Base class for 2D operations
18  #include "TwoD.h"
19
20  // For displaying text in the HUD
21  #include "TextEngine.h"
22  // The Developer Console
23  #include "Console.h"
24
25  class HeadsUpDisplay : public TwoD
26  {
27  private:
28          // Duration of time to dim screen (Goes from black to clear as time
                  progresses)
29          int dimTime = 0;
30          // Duration of time to go dark (completely black)
31          int darkTime = 0;
32          // Duration of the time to fade the screen (goes from clear to black as
                  time progresses)
33          int fadeTime = 0;
34          // Wether or not to dim
35          bool dimNow = false;
36          // Wether or not to darken
37          bool darkNow = false;
38          // Wether or not to fade
39          bool fadeNow = false;
40          // Wether or not we are in developer console
41          bool devConsole = false;
42
43          // Tag to current alert
44          std::string currentAlert;
```

```cpp
45          // Tag to current status
46          std::string currentStatus;
47          // Text to print to the screen
48          std::string currentText;
49          // What the user is typing
50          std::string currentInput;
51
52          // To Display text
53          TextEngine helmet;
54          // Dev Console
55          Console dev;
56
57          // Draws an info bar at the top of the screen
58          void drawHelmetBounds();
59          // Displays suit alerts
60          void DisplayAlerts();
61          // Draws the Heads Up Display
62          void drawHUD();
63          // Manages the dimming of the screen
64          void dim();
65          // Manages the darkening of the screen
66          void dark();
67          // Manages the fading of the screen
68          void fade();
69          // Draws the box which stores the info text
70          void drawInfoBox();
71          // Draws the developer console window
72          void drawConsole();
73          // Displays standard info in the top left corner
74          void displayInfo(std::string tag);
75
76
77  public:
78          // Manages the HUD
79          void DisplayHUD();
80
81          /****           ALTERATION FUNCTIONS          *****\
82          \**** Should always be called before DisplayHud *****/
83
84          // Tells the HUD how long to dim
85          void goDim(int time);
86
87          //Tells the HUD how long to go dark
88          void goDark(int time);
89          // Tels the HUD how long to fade for
90          void goFade(int time);
91          // Flips dev_console
92          void toggleConsole();
93
94          // Takes in a tag to print to screen
95          void displayWarning(std::string warning);
96
97          // Takes in a string to display in the status box
98          void setStatus(std::string status);
99
100         // Takes in a string to print to screen
```

```
101          void printToConsole ( std :: string text );
102
103          // Signifies a completed input to the console
104          void inputString ( std :: string text );
105
106          // Returns an item of the console's log
107          std :: string getHist ( int count );
108
109          // Returns the number of items in the console's log
110          int getHistNum ();
111   };
112
113   #endif
```

3.1.18 HeadsUpDiplay.cpp

```
 1   /************************************************************\
 2    * HeadsUpDisplay.cpp                                        *
 3    * This file was created by Jeremy Greenburg                 *
 4    * As part of The God Core game for the University of        *
 5    * Tennessee at Martin's University Scholars Organization    *
 6    *                                                           *
 7    * This file contains the definition of the HeadsUpDisplay   *
 8    * Class. For more information , see HeadsUpDisplay.h         *
 9   \************************************************************/
10
11   // Class Declaration
12   #include "HeadsUpDisplay.h"
13
14   // OpenGL API
15   #include <gl\glut.h>
16
17   // For counting seconds
18   #include <ctime>
19
20   // For displaying Planes
21   #include "Plane.h"
22
23   // For displaying triangles
24   #include "Triangle.h"
25
26   using namespace std;
27
28   void HeadsUpDisplay :: drawHelmetBounds ()
29   {
30          // Helmet bounds are black
31          double colors [4] = { 0, 0, 0, 1 };
32
33          // The top of the helmet
34          double top_vertices [9] =
35          {
36                  SCREENRIGHT , SCREENTOP , -1,
37                  SCREENLEFT , SCREENTOP , -1,
38                  SCREENRIGHT / 2.0 , SCREENBOTTOM / 20.0 , -1
39          };
40
41          // The left of the hemlet
```

```
42          double left_vertices[9] =
43          {
44                  SCREENLEFT, SCREENBOTTOM, -1,
45                  SCREENLEFT, SCREENTOP, -1,
46                  SCREENRIGHT / 20.0, 3 * SCREENBOTTOM / 5.0, -1
47          };
48
49          // The back of the helmet
50          double right_vertices[9] =
51          {
52                  SCREENRIGHT, SCREENBOTTOM, -1,
53                  SCREENRIGHT, SCREENTOP, -1,
54                  19 * SCREENRIGHT / 20.0, 3 * SCREENBOTTOM / 5.0, -1
55          };
56
57          Triangle top_helm{ top_vertices, colors };
58          Triangle left_helm{ left_vertices, colors };
59          Triangle right_helm{ right_vertices, colors };
60
61          top_helm.Display2D();
62          left_helm.Display2D();
63          right_helm.Display2D();
64  }
65
66  void HeadsUpDisplay::DisplayAlerts()
67  {
68          helmet.openFile(.5 * SCREENRIGHT, .5 * SCREENBOTTOM,
69                  1, 1, 1,
70                  "suitAlerts.log", currentAlert);
71  }
72
73  void HeadsUpDisplay::dim()
74  {
75          static int startTime;
76          static bool timeSet = false;
77          if (dimNow)
78          {
79                  if (!timeSet)
80                  {
81                          startTime = time(NULL);
82                          timeSet = true;
83                  }
84
85                  int currentTime = time(NULL);
86                  int timeElapsed = currentTime - startTime;
87                  if (timeElapsed < dimTime)
88                  {
89                          // A black square that grows more transparent as time
                               passes
90                          double colors[4] = { 0, 0, 0, (double)(dimTime -
                               timeElapsed) / dimTime };
91                          double dimVert[12] =
92                          {
93                                  SCREENLEFT, SCREENTOP, -1,
94                                  SCREENLEFT, SCREENBOTTOM, -1,
95                                  SCREENRIGHT, SCREENBOTTOM, -1,
```

```cpp
96                              SCREENRIGHT , SCREENTOP , -1
97                      };
98
99                      Plane black{ dimVert , colors };
100                     black.Display2D ();
101             }
102
103             else
104             {
105                     dimNow = false;
106                     timeSet = false;
107             }
108        }
109 }
110
111 void HeadsUpDisplay::fade()
112 {
113        static int startTime;
114        static bool timeSet = false;
115        if (fadeNow)
116        {
117                if (!timeSet)
118                {
119                        startTime = time(NULL);
120                        timeSet = true;
121                }
122
123                int currentTime = time(NULL);
124                int timeElapsed = currentTime - startTime;
125                if (timeElapsed < fadeTime)
126                {
127                        // A black square that grows more transparent as time
                           passes
128                        double colors[4] = { 0, 0, 0, 1 - ((double)(fadeTime -
                           timeElapsed) / fadeTime) };
129                        double dimVert[12] =
130                        {
131                                SCREENLEFT , SCREENTOP , -1,
132                                SCREENLEFT , SCREENBOTTOM , -1,
133                                SCREENRIGHT , SCREENBOTTOM , -1,
134                                SCREENRIGHT , SCREENTOP , -1
135                        };
136
137                        Plane black{ dimVert , colors };
138                        black.Display2D ();
139                }
140
141                else
142                {
143                        fadeNow = false;
144                        timeSet = false;
145
146
147                        // Go dark till game ends
148                        darkNow = true;
149                        darkTime = 1000;
```

```
150                        }
151                }
152 }
153
154 void HeadsUpDisplay::dark()
155 {
156        static int startTime;
157        static bool timeSet = false;
158        if (darkNow)
159        {
160                if (!timeSet)
161                {
162                        startTime = time(NULL);
163                        timeSet = true;
164                }
165
166                int currentTime = time(NULL);
167                int timeElapsed = currentTime - startTime;
168                if (timeElapsed < darkTime)
169                {
170                        // A black square that obscures vision
171                        double colors[4] = { 0, 0, 0, 1 };
172                        double dimVert[12] =
173                        {
174                                SCREENLEFT, SCREENTOP, -1,
175                                SCREENLEFT, SCREENBOTTOM, -1,
176                                SCREENRIGHT, SCREENBOTTOM, -1,
177                                SCREENRIGHT, SCREENTOP, -1
178                        };
179
180                        Plane black{ dimVert, colors };
181                        black.Display2D();
182                }
183
184                else
185                {
186                        darkNow = false;
187                        timeSet = false;
188                }
189        }
190 }
191
192 void HeadsUpDisplay::drawConsole()
193 {
194        double colors[4] = { .1, .1, .1, .9 };
195        double vertices[12] =
196        {
197                SCREENLEFT, SCREENTOP, -1,
198                SCREENLEFT, SCREENBOTTOM / 5, -1,
199                SCREENRIGHT, SCREENBOTTOM / 5, -1,
200                SCREENRIGHT, SCREENTOP, -1
201        };
202
203        Plane console_tab{ vertices, colors };
204        console_tab.Display2D();
205
```

```
206            if ( currentInput != "" )
207            {
208                    dev.activate ( currentInput , currentText );
209                    currentInput.clear ();
210            }
211
212            else
213            {
214                    dev.activate ( currentText );
215            }
216    }
217
218    void HeadsUpDisplay :: drawInfoBox ()
219    {
220            double colors [4] = { 0, 1, 1, .5 };
221            double vertices [12] =
222            {
223                    SCREENLEFT , SCREENTOP , -1,
224                    SCREENLEFT , SCREENBOTTOM / 10, -1,
225                    SCREENRIGHT / 10, SCREENBOTTOM / 10, -1,
226                    SCREENRIGHT / 10, SCREENTOP , -1
227            };
228
229            Plane info{ vertices , colors };
230            info.Display2D ();
231    }
232
233    void HeadsUpDisplay :: displayInfo ( string tag )
234    {
235            helmet.openFile ( SCREENLEFT , SCREENTOP +  20, 1, 1, 1,
236                    "suitAlerts.log", currentStatus );
237    }
238
239    void HeadsUpDisplay :: goDim ( int time )
240    {
241            dimTime = time;
242            dimNow = true;
243    }
244
245    void HeadsUpDisplay :: goDark ( int time )
246    {
247            darkTime = time;
248            darkNow = true;
249    }
250
251    void HeadsUpDisplay :: goFade ( int time )
252    {
253            fadeTime = time;
254            fadeNow = true;
255    }
256
257    void HeadsUpDisplay :: displayWarning ( std :: string warning )
258    {
259            currentAlert = warning;
260    }
261
```

```cpp
262  void HeadsUpDisplay::setStatus(std::string status)
263  {
264          currentStatus = status;
265  }
266
267  void HeadsUpDisplay::printToConsole(std::string text)
268  {
269          currentText = text;
270  }
271
272  void HeadsUpDisplay::inputString(std::string text)
273  {
274          currentInput = text;
275  }
276
277  void HeadsUpDisplay::toggleConsole()
278  {
279          devConsole = !devConsole;
280  }
281
282  void HeadsUpDisplay::drawHUD()
283  {
284          drawHelmetBounds();
285
286          if (dimNow)
287          {
288                  dim();
289          }
290
291          else if (fadeNow)
292          {
293                  fade();
294          }
295
296          // Not else if due to fade -> dark transition
297          if (darkNow)
298          {
299                  dark();
300          }
301
302          drawInfoBox();
303          displayInfo(currentStatus);
304
305          if (devConsole)
306          {
307                  drawConsole();
308          }
309
310          if (currentAlert != "")
311          {
312                  DisplayAlerts();
313          }
314  }
315
316  string HeadsUpDisplay::getHist(int count)
317  {
```

```
318          return dev.getHist(count);
319  }
320
321  int HeadsUpDisplay::getHistNum()
322  {
323          return dev.getHistNum();
324  }
325
326  void HeadsUpDisplay::DisplayHUD()
327  {
328          prepare2D();
329
330          drawHUD();
331
332          prepare3D();
333  }
```

3.1.19 Keyboard.h

```
 1  /***********************************************************\
 2   * Keyboard.h                                              *
 3   * This file was created by Jeremy Greenburg              *
 4   * As part of The God Core game for the University of     *
 5   * Tennessee at Martin's University Scholars Organization *
 6   *                                                         *
 7   * This file contains the declaration of the Keyboard class, *
 8   * which logs keypresses from the user and determines,    *
 9   * depending on the context, what action to take such.    *
10  \***********************************************************/
11
12  #ifndef KEYBOARD_H
13  #define KEYBOARD_H
14
15  // std::string
16  #include <string>
17
18  class Keyboard
19  {
20  private:
21          // Signals to recieve a part of the console's history
22          bool getPrev, getNext;
23
24  public:
25          // Normal keys
26          void normal(unsigned char key, int x, int y);
27          // To read console input
28          void inputConsole(unsigned char key, int x, int y);
29          // To read terminal input
30          void inputTerminal(unsigned char key, int x, int y);
31          // To interact with the world
32          void interact(unsigned char key, int x, int y);
33          // If a key is released
34          void key_up(unsigned char key, int x, int y);
35          // Special keys (functions, arrows, ect.)
36          void special(int key, int x, int y);
37          // Manages interactivity
38          void interact();
```

```
39  };
40
41  #endif
```

3.1.20 Keyboard.cpp

```
 1  /**************************************************************\
 2   * Keyboard.cpp                                               *
 3   * This file was created by Jeremy Greenburg                  *
 4   * As part of The God Core game for the University of         *
 5   * Tennessee at Martin's University Scholars Organization     *
 6   *                                                            *
 7   * This file contains the defintion of the Keyboard class.    *
 8   * for more information , see Keyboard.h                      *
 9  \**************************************************************/
10
11  // Class decleration
12  #include "Keyboard.h"
13
14  // std::string
15  #include <string>
16
17  // glut really wants cstdlib here
18  #include <cstdlib>
19
20  // OpenGL API
21  #include <GL\glut.h>
22
23  // To recieve and manage global variables
24  #include "Globals.h"
25  // Collision detection
26  #include "CollisionEngine.h"
27
28  // Return codes
29  #include "Return.h"
30  // System log
31  #include "Logger.h"
32
33  using namespace std;
34
35  void Keyboard::normal(unsigned char key, int x, int y)
36  {
37          // If we are currently capturing input
38          if (isInConsole)
39          {
40                  inputConsole(key, x, y);
41          }
42
43          // If we're in a computer
44          else if (isInTerminal)
45          {
46                  inputTerminal(key, x, y);
47          }
48
49          // Otherwise (as long we aren't in a menu)
50          else if (!isInMain)
51          {
```

```cpp
52                      interact(key, x, y);
53              }
54  }
55
56  void Keyboard::inputConsole(unsigned char key, int x, int y)
57  {
58          // User string input
59          static string input;
60          // Number in console history
61          static int count = 0;
62
63          // Up arrow, recieves the next older entry in the console's history
64          if (getPrev)
65          {
66                  input = HUD.getHist(count);
67
68                  if (count < HUD.getHistNum() - 1)
69                  {
70                          count++;
71                  }
72
73                  getPrev = false;
74          }
75
76          // Down arrow, recieves the next newer entry in the console's history
77          else if (getNext)
78          {
79                  input = HUD.getHist(count);
80
81                  if (count > 0)
82                  {
83                          count--;
84                  }
85
86                  getNext = false;
87          }
88
89          // Enter key, process and clear input
90          else if (key == 13)
91          {
92                  HUD.inputString(input);
93                  input.clear();
94                  count = 0;
95          }
96
97          // Tilda, close the console
98          else if (key == '~' || isInConsole == false)
99          {
100                  input.clear();
101                  isInConsole = false;
102                  HUD.toggleConsole();
103                  count = 0;
104          }
105
106          // Backspace. Self explanatory
107          else if (key == 8 && !input.empty())
```

```cpp
108                {
109                        input.pop_back();
110                }
111
112                // Otherwise, type normally
113                else
114                {
115                        input += key;
116                }
117
118                // Print what's been typed so far
119                HUD.printToConsole(input);
120 }
121
122 // Pretty much a copy pasta of inputConsole because I'm a terrible programmer
123 // I'll try to combine em in the future, I swear
124 // Just adjust all of these to do terminally stuff I guess
125 void Keyboard::inputTerminal(unsigned char key, int x, int y)
126 {
127                // TODO: Fix terminal input with active Terminal hijibis
128
129                // User string input
130                static string input;
131                // Number in console history
132                static int count = 0;
133
134                // Up arrow, recieves the next older entry in the console's history
135                if (getPrev)
136                {
137                        input = activeTerminal->getHist(count);
138
139                        if (count < activeTerminal->getHistNum() - 1)
140                        {
141                                count++;
142                        }
143
144                        getPrev = false;
145                }
146
147                // Down arrow, recieves the next newer entry in the console's history
148                else if (getNext)
149                {
150                        input = activeTerminal->getHist(count);
151
152                        if (count > 0)
153                        {
154                                count--;
155                        }
156
157                        getNext = false;
158                }
159
160                // Enter key, process and clear input
161                else if (key == 13)
162                {
163                        activeTerminal->getInput(input);
```

```
164                     input.clear();
165                     count = 0;
166             }
167
168             // Backspace. Self explanatory
169             else if (key == 8 && !input.empty())
170             {
171                     input.pop_back();
172             }
173
174             // Otherwise, type normally
175             else
176             {
177                     input += key;
178             }
179
180             // Print what's been typed so far
181             activeTerminal->getText(input); // Drawing handled elsewhere?
182 }
183
184 void Keyboard::interact(unsigned char key, int x, int y)
185 {
186             CollisionEngine col;
187             // Speed at which the player moves
188             int speedMod = 1;
189
190             int modKey = glutGetModifiers();
191
192             if (modKey == GLUT_ACTIVE_SHIFT)
193             {
194                     speedMod = 2;
195             }
196
197             else
198             {
199                     speedMod = 1;
200             }
201
202             switch (key)
203             {
204             case 'w':
205             case 'W':
206                     Cam.moveForward(speedMod);
207                     if (col.collide())
208                     {
209                             Cam.moveBackward(speedMod);
210                     }
211                     break;
212             case 'a':
213             case 'A':
214                     Cam.strafeRight();
215                     if (col.collide())
216                     {
217                             Cam.strafeLeft();
218                     }
219                     break;
```

```
220          case 's':
221          case 'S':
222                  Cam.moveBackward(speedMod);
223                  if (col.collide())
224                  {
225                          Cam.moveForward(speedMod);
226                  }
227                  break;
228          case 'd':
229          case 'D':
230                  Cam.strafeLeft();
231                  if (col.collide())
232                  {
233                          Cam.strafeRight();
234                  }
235                  break;
236          case 'e':
237          case 'E':
238                  interact();
239                  break;
240          case '~':
241                  isInConsole = true;
242                  HUD.toggleConsole();
243                  break;
244
245                  // Enter
246          case 13:
247                  //goDim = true;
248                  break;
249
250                  // Escape
251          case 27:
252                  isInMain = true;
253                  songNum = 0;
254                  changeSong = true;
255                  break;
256          }
257  }
258
259  void Keyboard::key_up(unsigned char key, int x, int y)
260  {
261          // I'm sure I'll do something smart here
262  }
263
264  void Keyboard::special(int key, int x, int y)
265  {
266          Logger log;
267          // We start in fullscreen
268          static bool fullScreen = true;
269          switch (key)
270          {
271          case GLUT_KEY_F1:
272                  fullScreen = !fullScreen;
273                  break;
274
275          case GLUT_KEY_F2:
```

```
276                    // Only way to exit main loop.
277                    log.logLine("Exiting via F2");
278                    exit(EXIT_OK);
279                    break;
280
281            case GLUT_KEY_F3:
282                    Cam.resetCam();
283                    break;
284
285            case GLUT_KEY_F4:
286                    isInMain = !isInMain;
287                    break;
288
289            case GLUT_KEY_F5:
290                    log.logCamCoords();
291                    break;
292
293            case GLUT_KEY_UP:
294                    if (isInConsole || isInTerminal)
295                    {
296                            getPrev = true;
297                            getNext = false;
298
299                            // To ensure that the input is updated BEFORE next key
                                  press
300                            normal(0, 0, 0);
301                    }
302                    break;
303
304            case GLUT_KEY_DOWN:
305                    if (isInConsole || isInTerminal)
306                    {
307                            getNext = true;
308                            getPrev = false;
309
310                            // To ensure that the input is updated BEFORE next key
                                  press
311                            normal(0, 0, 0);
312                    }
313                    break;
314        }
315
316        if (fullScreen)
317        {
318                glutFullScreen();
319        }
320
321        else
322        {
323                glutReshapeWindow(1367, 767);
324                glutPositionWindow(50, 50);
325        }
326 }
327
328 void Keyboard::interact()
329 {
```

```
330            // Only do things if we actually can
331            if (interactivity)
332            {
333                    if (activeSwitch != NULL)
334                    {
335                            activeSwitch->toggleTarget();
336
337                            for (unsigned int i = 0; i < triggers.size(); i++)
338                            {
339                                    triggers[i].tryToTrigger(activeSwitch, T_SWITCH);
340                            }
341                    }
342
343                    else if (activeTerminal != NULL)
344                    {
345                            isInTerminal = true;
346
347                            for (unsigned int i = 0; i < triggers.size(); i++)
348                            {
349                                    triggers[i].tryToTrigger(activeTerminal,
                                            T_TERMINAL);
350                            }
351                    }
352            }
353 }
```

3.1.21   Level.h

```
 1 /************************************************************\
 2  * Level.h                                                 *
 3  * This file was created by Jeremy Greenburg              *
 4  * As part of The God Core game for the University of     *
 5  * Tennessee at Martin's University Scholars Organization *
 6  *                                                         *
 7  * This file contains the declaration of the Level class  *
 8  * Which loads all level assets from a sqlite database    *
 9  * (data.db)                                               *
10 \************************************************************/
11
12 #ifndef LEVEL_H
13 #define LEVEL_H
14
15 // std;:string
16 #include <string>
17 // std::vector
18 #include <vector>
19 // Planes for walls/doors/such else
20 #include "Plane.h"
21
22 // SQLite API
23 #include "sqlite3.h"
24
25 // Glut API
26 #include <GL\glut.h>
27
28 class Level
29 {
```

```
30  private:
31          // Used to load cylinders
32          GLUquadricObj *quadratic;
33          // The current level being loaded
34          std::string currLevel;
35
36          // Look, the names are self-explanatory
37          void loadWalls(sqlite3 *db);
38          void loadDoors(sqlite3 *db);
39          void loadCylinders(sqlite3 *db);
40          void loadSwitches(sqlite3 *db);
41          void loadTerminals(sqlite3 *db);
42          void loadTriggers(sqlite3 *db);
43
44          // Binds the triggering object and target object to a single trigger
45          bool bindTrigger(std::string id, std::string trigger, std::string
                 triggerType);
46          bool bindTarget(std::string id, std::string target, std::string targetType
                 );
47
48  public:
49          // Manages the loading of the level
50          void loadLevel(std::string levelName);
51          // Draws the level
52          void displayLevel();
53  };
54
55  #endif
```

3.1.22   Level.cpp

```
 1  /************************************************************\
 2   * Level.cpp                                               *
 3   * This file was created by Jeremy Greenburg               *
 4   * As part of The God Core game for the University of       *
 5   * Tennessee at Martin's University Scholars Organization   *
 6   *                                                          *
 7   * This file contains the defintion of the Level class.     *
 8   * for more information, see Level.h                        *
 9  \************************************************************/
10
11  // Class declaration
12  #include "Level.h"
13  // To use Planes
14  #include "Plane.h"
15  // Vectors to plop stuff in
16  #include "Globals.h"
17  // Return codes
18  #include "Return.h"
19  // System log
20  #include "Logger.h"
21  // Oject Types
22  #include "GCTypes.h"
23
24  #include <iostream>
25
26  using namespace std;
```

```
27
28   void Level::loadWalls(sqlite3 *db)
29   {
30           walls.clear();
31           // Prepared Statement
32           sqlite3_stmt *stm;
33           // SQL command
34           string cmd;
35           // Connection Error Test
36           int err;
37           cmd = "SELECT * FROM walls WHERE LEVEL = \"" + currLevel + "\"";
38
39           err = sqlite3_prepare(db, cmd.c_str(), -1, &stm, 0);
40
41           if (err != SQLITE_OK)
42           {
43                   Logger log;
44                   vector<string> output = { "FATAL ERROR: failed to load walls from
                           ", currLevel };
45                   log.logLine(output);
46                   exit(STATEMENT_ERROR);
47           }
48
49           // While we still get rows of output
50           while (sqlite3_step(stm) == SQLITE_ROW)
51           {
52                   double x1, x2, x3, x4,
53                          y1, y2, y3, y4,
54                          z1, z2, z3, z4,
55                          r, g, b, a;
56                   string axis;
57
58                   x1 = sqlite3_column_double(stm, 2);
59                   x2 = sqlite3_column_double(stm, 3);
60                   x3 = sqlite3_column_double(stm, 4);
61                   x4 = sqlite3_column_double(stm, 5);
62
63                   y1 = sqlite3_column_double(stm, 6);
64                   y2 = sqlite3_column_double(stm, 7);
65                   y3 = sqlite3_column_double(stm, 8);
66                   y4 = sqlite3_column_double(stm, 9);
67
68                   z1 = sqlite3_column_double(stm, 10);
69                   z2 = sqlite3_column_double(stm, 11);
70                   z3 = sqlite3_column_double(stm, 12);
71                   z4 = sqlite3_column_double(stm, 13);
72
73                   r = sqlite3_column_double(stm, 14);
74                   g = sqlite3_column_double(stm, 15);
75                   b = sqlite3_column_double(stm, 16);
76                   a = sqlite3_column_double(stm, 17);
77
78                   axis = reinterpret_cast<const char*>(sqlite3_column_text(stm, 18))
                           ;
79
80                   char ax;
```

```
81                    if (axis == "x") ax = 'x';
82                    else if (axis == "y") ax = 'y';
83                    else if (axis == "z") ax = 'z';
84                    else ax = 0;
85
86                    double verts[12] =
87                    {
88                            x1, y1, z1,
89                            x2, y2, z2,
90                            x3, y3, z3,
91                            x4, y4, z4
92                    };
93                    double colors[4] = { r, g, b, a };
94
95                    Plane rect(verts, colors, ax);
96
97                    walls.push_back(rect);
98            }
99
100            /*
101            Logger log;
102            vector<string> output = { "Loaded walls on", currLevel };
103            log.logLine(output);
104            */
105
106            // Deconstructs the statement
107            sqlite3_finalize(stm);
108    }
109
110    void Level::loadDoors(sqlite3 *db)
111    {
112            doors.clear();
113            // Prepared Statement
114            sqlite3_stmt *stm;
115            // SQL command
116            string cmd;
117            // Connection Error Test
118            int err;
119            cmd = "SELECT * FROM doors WHERE LEVEL = \"" + currLevel + "\"";
120
121            err = sqlite3_prepare(db, cmd.c_str(), -1, &stm, 0);
122
123            if (err != SQLITE_OK)
124            {
125                    Logger log;
126                    vector<string> output = { "FATAL ERROR: Can't load doors while
                            loading", currLevel };
127                    log.logLine(output);
128
129                    exit(STATEMENT_ERROR);
130            }
131
132            // While we still get rows of output
133            while (sqlite3_step(stm) == SQLITE_ROW)
134            {
135                    double x1, x2, x3, x4,
```

```
136                             y1, y2, y3, y4,
137                             z1, z2, z3, z4,
138                             r, g, b, a;
139                 string id;
140                 string axis;
141
142                 id = reinterpret_cast<const char*>(sqlite3_column_text(stm, 0));
143                 x1 = sqlite3_column_double(stm, 2);
144                 x2 = sqlite3_column_double(stm, 3);
145                 x3 = sqlite3_column_double(stm, 4);
146                 x4 = sqlite3_column_double(stm, 5);
147
148                 y1 = sqlite3_column_double(stm, 6);
149                 y2 = sqlite3_column_double(stm, 7);
150                 y3 = sqlite3_column_double(stm, 8);
151                 y4 = sqlite3_column_double(stm, 9);
152
153                 z1 = sqlite3_column_double(stm, 10);
154                 z2 = sqlite3_column_double(stm, 11);
155                 z3 = sqlite3_column_double(stm, 12);
156                 z4 = sqlite3_column_double(stm, 13);
157
158                 r = sqlite3_column_double(stm, 14);
159                 g = sqlite3_column_double(stm, 15);
160                 b = sqlite3_column_double(stm, 16);
161                 a = sqlite3_column_double(stm, 17);
162
163                 a = sqlite3_column_double(stm, 17);
164
165                 axis = reinterpret_cast<const char*>(sqlite3_column_text(stm, 18))
                            ;
166
167                 char ax;
168                 if (axis == "x") ax = 'x';
169                 else if (axis == "y") ax = 'y';
170                 else if (axis == "z") ax = 'z';
171                 else ax = 0;
172
173                 double verts[12] =
174                 {
175                         x1, y1, z1,
176                         x2, y2, z2,
177                         x3, y3, z3,
178                         x4, y4, z4
179                 };
180                 double colors[4] = { r, g, b, a };
181
182                 Plane rect(verts, colors, ax);
183
184                 doors.push_back(Door(rect, id));
185         }
186
187         Logger log;
188         vector<string> output = { "Loaded doors on", currLevel };
189         log.logLine(output);
190
```

```cpp
191              // Deconstructs the statement
192              sqlite3_finalize ( stm );
193  }
194
195  void Level :: loadCylinders ( sqlite3 *db )
196  {
197              cylinders . clear ();
198              // Prepared Statement
199              sqlite3_stmt *stm ;
200              // SQL command
201              string cmd ;
202              // Connection Error Test
203              int err ;
204              cmd = "SELECT * FROM cylinders WHERE LEVEL = \"" + currLevel + "\"";
205
206              err = sqlite3_prepare ( db , cmd . c_str () , -1 , & stm , 0 );
207
208              if ( err != SQLITE_OK )
209              {
210                      Logger log ;
211                      vector < string > output = { "FATAL ERROR: Can't load cylinders while
                              loading", currLevel };
212                      log . logLine ( output );
213
214                      exit ( STATEMENT_ERROR );
215              }
216
217              // While we still get rows of output
218              while ( sqlite3_step ( stm ) == SQLITE_ROW )
219              {
220                      double xt , yt , zt ,
221                              xr , yr , zr ,
222                              r , g , b , a ,
223                              baseRadius , topRadius , height ;
224                      int stacks , slices ;
225
226
227                      xt = sqlite3_column_double ( stm , 1 );
228                      yt = sqlite3_column_double ( stm , 2 );
229                      zt = sqlite3_column_double ( stm , 3 );
230
231                      xr = sqlite3_column_double ( stm , 4 );
232                      yr = sqlite3_column_double ( stm , 5 );
233                      zr = sqlite3_column_double ( stm , 6 );
234
235                      baseRadius = sqlite3_column_double ( stm , 7 );
236                      topRadius = sqlite3_column_double ( stm , 8 );
237                      height = sqlite3_column_double ( stm , 9 );
238
239                      stacks = sqlite3_column_int ( stm , 10 );
240                      slices = sqlite3_column_int ( stm , 11 );
241
242                      r = sqlite3_column_double ( stm , 12 );
243                      g = sqlite3_column_double ( stm , 13 );
244                      b = sqlite3_column_double ( stm , 14 );
245                      a = sqlite3_column_double ( stm , 15 );
```

```
246
247
248                     double translate[3] = { xt, yt, zt };
249                     double rotate[3] = { xr, yr, zr };
250                     double colors[4] = { r, g, b, a };
251
252                     cylinders.push_back(Cylinder(baseRadius, topRadius, height, stacks
                            , slices, translate, rotate, colors));
253          }
254
255          Logger log;
256          vector<string> output = { "Loaded cylinders on", currLevel };
257          log.logLine(output);
258
259          // Deconstructs the statement
260          sqlite3_finalize(stm);
261 }
262
263
264 void Level::loadSwitches(sqlite3 *db)
265 {
266          switches.clear();
267          // Prepared Statement
268          sqlite3_stmt *stm;
269          // SQL command
270          string cmd;
271          // Connection Error Test
272          int err;
273          cmd = "SELECT * FROM switches WHERE LEVEL = \"" + currLevel + "\"";
274
275          err = sqlite3_prepare(db, cmd.c_str(), -1, &stm, 0);
276
277          if (err != SQLITE_OK)
278          {
279                  Logger log;
280                  vector<string> output = { "FATAL ERROR: Can't load switches while
                            loading", currLevel };
281                  log.logLine(output);
282
283                  exit(STATEMENT_ERROR);
284          }
285
286          // While we still get rows of output
287          while (sqlite3_step(stm) == SQLITE_ROW)
288          {
289                  double xt, yt, zt,
290                          xr, yr, zr;
291                  string target, s_type, id;
292                  int i_type;
293                  bool isOn;
294
295                  id = reinterpret_cast<const char*>(sqlite3_column_text(stm, 0));
296                  target = reinterpret_cast<const char*>(sqlite3_column_text(stm, 2)
                            );
297                  xt = sqlite3_column_double(stm, 3);
298                  yt = sqlite3_column_double(stm, 4);
```

```
299                    zt = sqlite3_column_double(stm, 5);
300
301                    xr = sqlite3_column_double(stm, 6);
302                    yr = sqlite3_column_double(stm, 7);
303                    zr = sqlite3_column_double(stm, 8);
304
305                    s_type = reinterpret_cast<const char*>(sqlite3_column_text(stm, 9)
                          );
306
307                    isOn = (bool)sqlite3_column_int(stm, 10);
308
309                    double translate[3] = { xt, yt, zt };
310                    double rotate[3] = { xr, yr, zr };
311
312                    if (s_type == "DOOR")
313                            i_type = T_DOOR;
314                    else if (s_type == "TERMINAL")
315                            i_type = T_TERMINAL;
316                    else if (s_type == "LEVEL_END")
317                            i_type = T_LEVEL_END;
318                    else
319                    {
320                            Logger log;
321                            vector<string> output = { "Failed to evaluate string type
                                  entry: ", s_type, "for switch ", id };
322                            log.logLine(output);
323
324                            exit(DATA_ENTRY_ERROR);
325                    }
326
327                    switches.push_back(Switch(translate, rotate, i_type, id, isOn));
328
329                    bool assigned = false;
330
331                    if (s_type == "LEVEL_END")
332                    {
333                            assigned = true;
334
335                            Logger log;
336                            vector<string> output = { "Switch ", id, " bound to end
                                  level" };
337                            log.logLine(output);
338                    }
339
340                    else if (s_type == "DOOR")
341                    {
342                            for (unsigned int i = 0; i < doors.size(); i++)
343                            {
344                                    if (doors[i].getID() == target)
345                                    {
346                                            Logger log;
347                                            vector<string> output = { "Binding switch
                                                  ", id, " to door", target };
348                                            log.logLine(output);
349
350                                            switches[switches.size() - 1].assign(&(
```

```
                                               doors[i]));
351
352                                            assigned = true;
353                                   }
354                            }
355                    }
356
357             else if (s_type == "TERMINAL")
358             {
359                    for (unsigned int i = 0; i < terminals.size(); i++)
360                    {
361                            if (terminals[i].getID() == target)
362                            {
363                                    Logger log;
364                                    vector<string> output = { "Binding switch
                                            ", id, " to terminal", target };
365                                    log.logLine(output);
366
367                                    switches[switches.size() - 1].assign(&(
                                            terminals[i]));
368
369                                    assigned = true;
370                            }
371                    }
372             }
373
374             if (!assigned)
375             {
376                    Logger log;
377                    vector<string> output = { "Failed to bind switch ", id, "
                                 to a ", s_type };
378                    log.logLine(output);
379
380                    exit(BINDING_ERROR);
381             }
382     }
383
384     Logger log;
385     vector<string> output = { "Loaded switches on", currLevel };
386     log.logLine(output);
387
388     // Deconstructs the statement
389     sqlite3_finalize(stm);
390 }
391
392 void Level::loadTerminals(sqlite3 *db)
393 {
394     terminals.clear();
395     // Prepared Statement
396     sqlite3_stmt *stm;
397     // SQL command
398     string cmd;
399     // Connection Error Test
400     int err;
401     cmd = "SELECT * FROM terminals WHERE LEVEL = \"" + currLevel + "\"";
402
```

```
403            err = sqlite3_prepare(db, cmd.c_str(), -1, &stm, 0);
404
405            if (err != SQLITE_OK)
406            {
407                    Logger log;
408                    vector<string> output = { "FATAL ERROR: Can't load terminals while
                            loading", currLevel };
409                    log.logLine(output);
410
411                    exit(STATEMENT_ERROR);
412            }
413
414            // While we still get rows of output
415            while (sqlite3_step(stm) == SQLITE_ROW)
416            {
417                    double xt, yt, zt,
418                           xr, yr, zr;
419                    string file, id;
420                    id = reinterpret_cast<const char*>(sqlite3_column_text(stm, 0));
421                    file = reinterpret_cast<const char*>(sqlite3_column_text(stm, 2));
422                    xt = sqlite3_column_double(stm, 3);
423                    yt = sqlite3_column_double(stm, 4);
424                    zt = sqlite3_column_double(stm, 5);
425
426                    xr = sqlite3_column_double(stm, 6);
427                    yr = sqlite3_column_double(stm, 7);
428                    zr = sqlite3_column_double(stm, 8);
429
430                    double translate[3] = { xt, yt, zt };
431                    double rotate[3] = { xr, yr, zr };
432
433                    Logger log;
434                    log.logLine(id);
435
436                    terminals.push_back(Terminal(translate, rotate, file, id));
437            }
438
439
440            Logger log;
441            vector<string> output = { "Loaded terminals on", currLevel };
442            log.logLine(output);
443
444            // Deconstructs the statement
445            sqlite3_finalize(stm);
446 }
447
448 void Level::loadTriggers(sqlite3 *db)
449 {
450            triggers.clear();
451            // Prepared Statement
452            sqlite3_stmt *stm;
453            // SQL command
454            string cmd;
455            // Connection Error Test
456            int err;
457            cmd = "SELECT * FROM triggers WHERE LEVEL = \"" + currLevel + "\"";
```

```
458
459            err = sqlite3_prepare(db, cmd.c_str(), -1, &stm, 0);
460
461            if (err != SQLITE_OK)
462            {
463                    Logger log;
464                    vector<string> output = { "FATAL ERROR: Can't load triggers while
                            loading", currLevel };
465                    log.logLine(output);
466
467                    exit(STATEMENT_ERROR);
468            }
469
470            // While we still get rows of output
471            while (sqlite3_step(stm) == SQLITE_ROW)
472            {
473                    string target, trigger, targetType, triggerType, id;
474                    int i_targetType, i_triggerType;
475
476                    id = reinterpret_cast<const char*>(sqlite3_column_text(stm, 0));
477                    trigger = reinterpret_cast<const char*>(sqlite3_column_text(stm,
                            2));
478                    target = reinterpret_cast<const char*>(sqlite3_column_text(stm, 3)
                            );
479                    triggerType = reinterpret_cast<const char*>(sqlite3_column_text(
                            stm, 4));
480                    targetType = reinterpret_cast<const char*>(sqlite3_column_text(stm
                            , 5));
481
482                    if (triggerType == "SWITCH")
483                            i_triggerType = T_SWITCH;
484                    else if (triggerType == "TERMINAL")
485                            i_triggerType = T_TERMINAL;
486                    else
487                    {
488                            Logger log;
489                            vector<string> output = { "Failed to evaluate string
                                    trigger type entry: ", triggerType, "for trigger ", id
                                    };
490                            log.logLine(output);
491
492                            exit(DATA_ENTRY_ERROR);
493                    }
494
495                    if (targetType == "SWITCH")
496                            i_targetType = T_SWITCH;
497                    else if (targetType == "TERMINAL")
498                            i_targetType = T_TERMINAL;
499                    else
500                    {
501                            Logger log;
502                            vector<string> output = { "Failed to evaluate string
                                    trigger type entry: ", targetType, "for trigger ", id
                                    };
503                            log.logLine(output);
504
```

```
505                             exit(DATA_ENTRY_ERROR);
506                     }
507
508                     triggers.push_back(Trigger(i_triggerType, i_targetType));
509
510                     bool assigned = bindTrigger(id, trigger, triggerType) &&
                                bindTarget(id, target, targetType);
511
512                     if (!assigned)
513                     {
514                             Logger log;
515                             vector<string> output = { "Failed to bind trigger ", id };
516                             log.logLine(output);
517
518                             exit(BINDING_ERROR);
519                     }
520         }
521
522         Logger log;
523         vector<string> output = { "Loaded trigger on", currLevel };
524         log.logLine(output);
525
526         // Deconstructs the statement
527         sqlite3_finalize(stm);
528 }
529
530 bool Level::bindTrigger(string id, string trigger, string triggerType)
531 {
532         if (triggerType == "SWITCH")
533         {
534                 for (unsigned int i = 0; i < switches.size(); i++)
535                 {
536                         if (switches[i].getID() == trigger)
537                         {
538                                 Logger log;
539                                 vector<string> output = { "Binding trigger ", id,
                                        " to trigger-switch", trigger };
540                                 log.logLine(output);
541
542                                 triggers[triggers.size() - 1].bindTrigger(&(
                                        switches[i]));
543
544                                 return true;
545                         }
546                 }
547         }
548
549         else if (triggerType == "TERMINAL")
550         {
551                 for (unsigned int i = 0; i < terminals.size(); i++)
552                 {
553                         if (terminals[i].getID() == trigger)
554                         {
555                                 Logger log;
556                                 vector<string> output = { "Binding trigger ", id,
                                        " to trigger-terminal", trigger };
```

```
557                                                 log.logLine(output);
558
559                                                 triggers[triggers.size() - 1].bindTrigger(&(
                                                        terminals[i]));
560
561                                                 return true;
562                                         }
563                                 }
564                 }
565
566                 return false;
567 }
568
569 bool Level::bindTarget(string id, string target, string targetType)
570 {
571
572                 if (targetType == "SWITCH")
573                 {
574                                 for (unsigned int i = 0; i < switches.size(); i++)
575                                 {
576                                                 if (switches[i].getID() == target)
577                                                 {
578                                                                 Logger log;
579                                                                 vector<string> output = { "Binding trigger ", id,
                                                                        " to target-switch", target };
580                                                                 log.logLine(output);
581
582                                                                 triggers[triggers.size() - 1].bindTarget(&(
                                                                        switches[i]));
583
584                                                                 return true;
585                                                 }
586                                 }
587                 }
588
589                 else if (targetType == "TERMINAL")
590                 {
591                                 for (unsigned int i = 0; i < terminals.size(); i++)
592                                 {
593                                                 if (terminals[i].getID() == target)
594                                                 {
595                                                                 Logger log;
596                                                                 vector<string> output = { "Binding trigger ", id,
                                                                        " to target-terminal", target };
597                                                                 log.logLine(output);
598
599                                                                 triggers[triggers.size() - 1].bindTarget(&(
                                                                        terminals[i]));
600
601                                                                 return true;
602                                                 }
603                                 }
604                 }
605
606                 return false;
607 }
```

```
608
609   void Level::loadLevel(std::string levelName)
610   {
611           Logger log;
612           vector<string> output = { "Starting to load", levelName };
613           log.logLine(output);
614
615           if (quadratic == NULL)
616           {
617                   quadratic = gluNewQuadric();
618           }
619
620           currLevel = levelName;
621
622           // Connection to SQL database
623           sqlite3 *db;
624           // 1 if error with DB
625           int connectErr = sqlite3_open("Data.db", &db);
626
627           if (connectErr != SQLITE_OK)
628           {
629                   Logger log;
630                   log.logLine("FATAL ERROR: Can't access database");
631
632                   exit(DATABASE_ERROR);
633           }
634
635           loadWalls(db);
636           loadDoors(db);
637           loadCylinders(db);
638           loadTerminals(db);
639
640           // Loading switches must be after doors/terminals to properly bind
641           loadSwitches(db);
642
643           // Loading triggers must be done last to properly bind
644           loadTriggers(db);
645
646           // Closes the database
647           sqlite3_close(db);
648
649           output[0] = "Finished loading";
650           log.logLine(output);
651
652           Cam.resetCam();
653
654           interactivity = false;
655
656           // Get out of wall
657           for (unsigned int i = 0; i < 10; i++)
658           {
659                   Cam.moveForward(1);
660           }
661
662           // Go dim for 5 seconds
663           HUD.goDim(5);
```

```
664  }
665
666  void Level::displayLevel()
667  {
668          for (auto i : doors)
669          {
670                  i.Display();
671          }
672
673          for (auto i : cylinders)
674          {
675                  i.Display();
676          }
677
678          for (auto i : switches)
679          {
680                  i.Display();
681          }
682
683          for (auto i : terminals)
684          {
685                  i.Display();
686          }
687
688          for (auto i : walls)
689          {
690                  i.Display();
691          }
692  }
```

3.1.23 Logger.h

```
 1  /************************************************************\
 2   * Logger.h                                                *
 3   * This file was created by Jeremy Greenburg               *
 4   * As part of The God Core game for the University of      *
 5   * Tennessee at Martin's University Scholars Organization  *
 6   *                                                         *
 7   * This file contains the declaration of the Logger class  *
 8   * Which writes messages to output.log because it's more   *
 9   * Reliable than stdout                                    *
10  \************************************************************/
11
12  #ifndef LOGGER_H
13  #define LOGGER_H
14
15  // To help find the user's document folder
16  #include <shlobj.h>
17
18  // std::vector
19  #include <vector>
20  // std::string
21  #include <string>
22
23  class Logger
24  {
25  private:
```

```
26          // Path to the log file
27          char CHAR_PATH[MAX_PATH];
28          std::string LOG_PATH;
29
30  public:
31          Logger();
32          // Erases the log file, called at the beggining of the program
33          void nuke();
34          // Writes to the log, either multiple lines or one line
35          void logLine(std::vector<std::string> input);
36          void logLine(std::string input);
37          // Writes the Camera Coordinates to the log file
38          void logCamCoords();
39
40  };
41
42  #endif
```

3.1.24 Logger.cpp

```
 1  /***********************************************************\
 2   * Logger.cpp                                              *
 3   * This file was created by Jeremy Greenburg               *
 4   * As part of The God Core game for the University of      *
 5   * Tennessee at Martin's University Scholars Organization  *
 6   *                                                         *
 7   * This file contains the defintion of the Logger class.   *
 8   * for more information, see Logger.h                      *
 9  \***********************************************************/
10
11  // Class declaration
12  #include "Logger.h"
13  // For Cam coords
14  #include "Globals.h"
15  // File I/O
16  #include <fstream>
17
18  #include <iostream>
19
20  using namespace std;
21
22  Logger::Logger()
23  {
24          HRESULT ret = SHGetFolderPath(NULL, CSIDL_PERSONAL, NULL,
                  SHGFP_TYPE_CURRENT, CHAR_PATH);
25          LOG_PATH = CHAR_PATH;
26          LOG_PATH += "\\The God Core\\output.log";
27  }
28
29  void Logger::nuke()
30  {
31          ofstream outfile(LOG_PATH); // Nukes everything within
32  }
33
34  void Logger::logLine(vector<string> input)
35  {
36          ofstream outfile(LOG_PATH, ios::app);
```

```
37
38          string output;
39
40          for (auto i : input)
41          {
42                  output += i;
43                  output += " ";
44          }
45          outfile << output << std::endl;
46  }
47
48  void Logger::logLine(string input)
49  {
50          ofstream outfile(LOG_PATH, ios::app);
51
52          outfile << input << std::endl;
53  }
54
55  void Logger::logCamCoords()
56  {
57          ofstream outfile(LOG_PATH, ios::app);
58
59          outfile << "Player Coordinates:\n";
60          outfile << "X: " << -Cam.x << endl;
61          outfile << "Y: " << -Cam.y << endl;
62          outfile << "Z: " << -Cam.z << endl;
63  }
```

3.1.25 MainMenu.h

```
 1  /************************************************************\
 2   * MainMenu.h                                               *
 3   * This file was created by Jeremy Greenburg               *
 4   * As part of The God Core game for the University of      *
 5   * Tennessee at Martin's University Scholars Organization  *
 6   *                                                          *
 7   * This file contains the decleration of the MainMenu class *
 8   * Which uses the Simple OpenGL Interface Library to load a *
 9   * png picture of the main menu, as well as provide button  *
10   * Interactivity                                            *
11  \************************************************************/
12
13  #ifndef MAIN_MENU_H
14  #define MAIN_MENU_H
15
16  // For loading pictures
17  #include <SOIL.h>
18  // Inherit 2D functionality
19  #include "TwoD.h"
20
21  // Make OpenGL happy
22  #include <cstdlib>
23  // openGL API
24  #include <GL\glut.h>
25
26  class MainMenu : public TwoD
27  {
```

```
28  public:
29          // Loads the picture up in memory
30          MainMenu();
31          // Handles drawing to the screen
32          void display();
33          // Handles and processes mouse clicks
34          void getClick(double x, double y);
35
36  private:
37          // Draws the main picture
38          void drawMainPic();
39          // DEBUG: draws boxes around all buttons
40          void drawClickBoxes();
41          // What the picture is bound to
42          GLint texture;
43  };
44
45  #endif
```

3.1.26   MainMenu.cpp

```
 1  /***********************************************************\
 2   * MainMenu.cpp                                            *
 3   * This file was created by Jeremy Greenburg               *
 4   * As part of The God Core game for the University of      *
 5   * Tennessee at Martin's University Scholars Organization  *
 6   *                                                         *
 7   * This file contains the defintion of the MainMenu class. *
 8   * for more information, see MainMenu.h                    *
 9  \***********************************************************/
10
11  // Class declaration
12  #include "MainMenu.h"
13  // isInMain
14  #include "Globals.h"
15  // Return codes
16  #include "Return.h"
17  // System log
18  #include "Logger.h"
19
20  #include "SaveManager.h"
21
22  using namespace std;
23
24  MainMenu::MainMenu()
25  {
26          texture = SOIL_load_OGL_texture
27                          (
28                                  "Resources\\Images\\Main.png",   // Image to load
29                                  SOIL_LOAD_AUTO,                                     //
                                       ???
30                                  SOIL_CREATE_NEW_ID,
31                                  SOIL_FLAG_MIPMAPS | SOIL_FLAG_NTSC_SAFE_RGB |
                                       SOIL_FLAG_COMPRESS_TO_DXT // !?!?!?!
32                          );
33
34          if (texture == 0)
```

```
35                {
36                        Logger log;
37                        vector<string> output = {"FATAL ERROR: SOIL cannot load image",
                                SOIL_last_result()};
38                        log.logLine(output);
39                        exit(SOIL_ERROR);
40                }
41  }
42
43  void MainMenu::drawMainPic()
44  {
45          glEnable(GL_TEXTURE_2D);
46
47          glBindTexture(GL_TEXTURE_2D, texture); // Prepares the texture for usage
48
49          glColor3d(1, 1, 1);
50          glBegin(GL_QUADS);
51          glTexCoord2d(0, 0);     glVertex2d(SCREENLEFT, SCREENTOP);
52          glTexCoord2d(0, 1); glVertex2d(SCREENLEFT, SCREENBOTTOM);
53          glTexCoord2d(1, 1); glVertex2d(SCREENRIGHT, SCREENBOTTOM);
54          glTexCoord2d(1, 0);     glVertex2d(SCREENRIGHT, SCREENTOP);
55
56          glEnd();
57
58          glDisable(GL_TEXTURE_2D);
59
60  }
61
62  void MainMenu::drawClickBoxes()
63  {
64          glColor3d(1, 0, 0);
65
66          // Start a new game
67          glBegin(GL_LINE_LOOP);
68          glVertex2d(SCREENRIGHT / 20.0, SCREENBOTTOM / 2.2);
69          glVertex2d(SCREENRIGHT / 20.0, SCREENBOTTOM / 1.9);
70          glVertex2d(SCREENRIGHT / 3.0, SCREENBOTTOM / 1.9);
71          glVertex2d(SCREENRIGHT / 3.0, SCREENBOTTOM / 2.2);
72          glEnd();
73
74          // Load game
75          glBegin(GL_LINE_LOOP);
76          glVertex2d(SCREENRIGHT / 10.0, SCREENBOTTOM / 1.57);
77          glVertex2d(SCREENRIGHT / 10.0, SCREENBOTTOM / 1.75);
78          glVertex2d(SCREENRIGHT / 3.5, SCREENBOTTOM / 1.75);
79          glVertex2d(SCREENRIGHT / 3.5, SCREENBOTTOM / 1.57);
80          glEnd();
81
82          // Options
83          glBegin(GL_LINE_LOOP);
84          glVertex2d(SCREENRIGHT / 8.5, SCREENBOTTOM / 1.35);
85          glVertex2d(SCREENRIGHT / 8.5, SCREENBOTTOM / 1.45);
86          glVertex2d(SCREENRIGHT / 3.9, SCREENBOTTOM / 1.45);
87          glVertex2d(SCREENRIGHT / 3.9, SCREENBOTTOM / 1.35);
88          glEnd();
89
```

```cpp
90              // Exit
91
92              glBegin(GL_LINE_LOOP);
93              glVertex2d(SCREENRIGHT / 6.5, SCREENBOTTOM / 1.16);
94              glVertex2d(SCREENRIGHT / 6.5, SCREENBOTTOM / 1.25);
95              glVertex2d(SCREENRIGHT / 4.5, SCREENBOTTOM / 1.25);
96              glVertex2d(SCREENRIGHT / 4.5, SCREENBOTTOM / 1.16);
97              glEnd();
98      }
99
100     void MainMenu::getClick(double x, double y)
101     {
102              // Start new game
103              if (x >= SCREENRIGHT / 20.0 && x <= SCREENRIGHT / 3.0)
104              {
105                      if (y >= SCREENBOTTOM / 2.2 && y <= SCREENBOTTOM / 1.9)
106                      {
107                              isInMain = false;
108                              songNum++;
109                              changeSong = true;
110                      }
111              }
112
113              // Load Game
114              if (x >= SCREENRIGHT / 10.0 && x <= SCREENRIGHT / 3.5)
115              {
116                      if (y >= SCREENBOTTOM / 1.75 && y <= SCREENBOTTOM / 1.57)
117                      {
118                              SaveManager Jesus; // Jesus Saves
119                              if (!Jesus.loadGame()); // null
120                              else isInMain = false;
121
122                      }
123              }
124
125              // Options
126              if (x >= SCREENRIGHT / 8.5 && x <= SCREENRIGHT / 3.9)
127              {
128                      if (y >= SCREENBOTTOM / 1.45 && y <= SCREENBOTTOM / 1.35)
129                      {
130                              // Jokes on me I never did get any options up
131                      }
132              }
133
134              // Exit
135              if (x >= SCREENRIGHT / 6.5 && x <= SCREENRIGHT / 4.5)
136              {
137                      if (y >= SCREENBOTTOM / 1.25 && y <= SCREENBOTTOM / 1.16)
138                      {
139                              exit(EXIT_OK);
140                      }
141              }
142     }
143
144     void MainMenu::display()
145     {
```

```
146         prepare2D ();
147
148         drawMainPic ();
149
150         // Disable once finished
151         // drawClickBoxes ();
152
153         glEnd ();
154
155         prepare3D ();
156 }
```

3.1.27   MusicManager.h

```
 1 /**************************************************************\
 2  * MusicManager.h                                            *
 3  * This file was created by Jeremy Greenburg                 *
 4  * As part of The God Core game for the University of        *
 5  * Tennessee at Martin's University Scholars Organization    *
 6  *                                                           *
 7  * This file contains the declaration of the MusicManager    *
 8  * Class, which uses the FMOD API to load .mp3 files into    *
 9  * Memory , play them when called , and release the memory    *
10  * When the song is no longer needed.                        *
11 \**************************************************************/
12
13 #ifndef MUSICMANAGER_H
14 #define MUSICMANAGER_H
15
16 // FMOD API
17 #include <fmod.hpp>
18
19 // Creates new type for ease of use
20 typedef FMOD::Sound* SoundClass;
21
22 class MusicManager
23 {
24 private:
25         // Pointer to dynamic system memory to load music
26         FMOD::System *m_pSystem;
27
28         // The path to the music folder
29         static const char* MUSIC_PATH;
30
31 public:
32         // Loads the song in memory
33         void makeSound(SoundClass *psound, const char *song);
34         // Plays the song (Always loops)
35         void playSound(SoundClass pSound, bool bLoop = true);
36         // Releases the song
37         void releaseSound(SoundClass psound);
38         // Initializes FMOD
39         MusicManager ();
40 };
41
42 #endif
```

3.1.28 MusicManager.cpp

```
1  /**************************************************************\
2   * MusicManager.cpp                                          *
3   * This file was created by Jeremy Greenburg                 *
4   * As part of The God Core game for the University of        *
5   * Tennessee at Martin's University Scholars Organization    *
6   *                                                           *
7   * This file contains the definition of the MusicManager     *
8   * Class. For more information, see MusicManager.h            *
9  \**************************************************************/
10
11 // Class definition
12 #include "MusicManager.h"
13
14 // Because concatenating char*'s are really hard
15 #include <string>
16
17 // Return codes
18 #include "Return.h"
19
20 // System log
21 #include "Logger.h"
22
23 using namespace std;
24
25 // Initialize the constant member of the class
26 const char* MusicManager::MUSIC_PATH = "Resources\\Music\\";
27
28 MusicManager::MusicManager()
29 {
30         Logger log;
31         if (FMOD::System_Create(&m_pSystem) != FMOD_OK)
32         {
33                 log.logLine("FATAL ERROR: FMOD unable to create system");
34                 exit(FMOD_ERROR);
35         }
36
37         int driverCount = 0;
38         m_pSystem->getNumDrivers(&driverCount);
39
40         // If you have no driver, you have bigger problems to worry about
41         if (driverCount == 0)
42         {
43                 // Report Error
44                 log.logLine("ERROR: FMOD unable to detect drivers");
45                 exit(FMOD_ERROR);
46         }
47
48         log.logLine("FMOD succesfully initialized");
49         // Initialize our Instance with 36 Channels
50         m_pSystem->init(36, FMOD_INIT_NORMAL, NULL);
51 }
52
53 void MusicManager::makeSound(SoundClass *psound, const char *song)
54 {
```

```
55          // MUSIC_PATH is placed in a nice string. Good string. Strings are friends
56          string fullPath = MUSIC_PATH;
57          // Now there is a full path to the song
58          fullPath += song;
59
60          m_pSystem->createSound(fullPath.c_str(), FMOD_DEFAULT, 0, psound);
61  }
62
63  void MusicManager::playSound(SoundClass pSound, bool bLoop)
64  {
65          if (!bLoop)
66                  pSound->setMode(FMOD_LOOP_OFF);
67          else
68          {
69                  pSound->setMode(FMOD_LOOP_NORMAL);
70                  pSound->setLoopCount(-1);
71          }
72
73          m_pSystem->playSound(pSound, NULL , false, 0);
74  }
75
76  void MusicManager::releaseSound(SoundClass pSound)
77  {
78          pSound->release();
79  }
```

3.1.29 Plane.h

```
 1  /**************************************************************\
 2   * Plane.h                                                    *
 3   * This file was created by Jeremy Greenburg                  *
 4   * As part of The God Core game for the University of         *
 5   * Tennessee at Martin's University Scholars Organization     *
 6   *                                                            *
 7   * This file contains the declaration of the Plane class      *
 8   * Which is used to hold the details of a 2D Plane and        *
 9   * draw  it to the screen                                     *
10  \**************************************************************/
11
12  #ifndef Plane_H
13  #define Plane_H
14
15  class Plane
16  {
17  private:
18          // Arrays containing the color and vertices of the Plane
19          double color[4];
20          // What axis is it aligned on (x y z)
21          char axis;
22          // The vertices of the corners
23          double vertices[12];
24  public:
25
26          // Paramaterized constructor, as there cannot be a Plane without vertices
27          // Can take an axis or can ignore exis
28          Plane(const double(&new_vertices)[12], const double(&new_color)[4], char
                  _axis);
```

```
29          Plane(const double(&new_vertices)[12], const double(&new_color)[4]);
30
31          // Part of the plane equation, calculated in constructor
32          double a, b, c, d;
33
34          // Determines if the player is in the bounds of the Plane (based on axis)
35          bool isInBounds();
36
37          // Returns the plane norm (Perpindicular line)
38          double getNorm();
39
40          // Mutate's the rectangles coordinates for the end of the game
41          void mutate();
42
43          // Print a Plane in 3D
44          void Display();
45          // Print a Plane in 2D
46          void Display2D();
47  };
48
49  #endif
```

3.1.30   Plane.cpp

```
1  /***************************************************************\
2   * Plane.cpp                                                   *
3   * This file was created by Jeremy Greenburg                   *
4   * As part of The God Core game for the University of          *
5   * Tennessee at Martin's University Scholars Organization      *
6   *                                                             *
7   * This file contains the definition of the Plane class        *
8   * For more information, see Plane.h                           *
9  \***************************************************************/
10
11  #include "Plane.h"
12
13  // For std::copy
14  #include <iterator>
15  #include <utility>
16
17  // max and min
18  #include <algorithm>
19
20  // OpenGL API
21  #include <GL\glut.h>
22
23  // For Cam coords
24  #include "Globals.h"
25
26  using namespace std;
27
28  Plane::Plane(const double (&new_vertices)[12], const double (&new_color)[4], char
        _axis)
29  {
30          // Copies the color
31          copy(begin(new_color), end(new_color), color);
32
```

```
33          // Copies the vertices
34          copy(begin(new_vertices), end(new_vertices), vertices);
35
36
37          // Somedays I wonder what I'm even doing \\
38          // When I forget what all this means: http://keisan.casio.com/exec/system
                /1223596129 \\
39
40          // Calculate vector equation ax + by + cz + d = 0
41          // Get two vectors from three of the corners
42          double AB[] = { vertices[3] - vertices[0], vertices[4] - vertices[1],
                vertices[5] - vertices[2] };
43          double AC[] = { vertices[6] - vertices[0], vertices[7] - vertices[1],
                vertices[8] - vertices[2] };
44          // Cross Product of AB and AC
45          a = (AB[1] * AC[2]) - (AB[2] * AC[1]);
46          b = (AB[2] * AC[0]) - (AB[0] * AC[2]);
47          c = (AB[0] * AC[1]) - (AB[1] * AC[0]);
48          d = (a * vertices[0] + b * vertices[1] + c * vertices[2]);
49
50          axis = _axis;
51  }
52
53  Plane::Plane(const double(&new_vertices)[12], const double(&new_color)[4])
54  {
55          // Copies the color
56          copy(begin(new_color), end(new_color), color);
57
58          // Copies the vertices
59          copy(begin(new_vertices), end(new_vertices), vertices);
60
61
62          // Somedays I wonder what I'm even doing \\
63                  // When I forget what all this means: http://keisan.casio.com/exec
                        /system/1223596129 \\
64
65  // Calculate vector equation ax + by + cz + d = 0
66  // Get two vectors from three of the corners
67          double AB[] = { vertices[3] - vertices[0], vertices[4] - vertices[1],
                vertices[5] - vertices[2] };
68          double AC[] = { vertices[6] - vertices[0], vertices[7] - vertices[1],
                vertices[8] - vertices[2] };
69          // Cross Product of AB and AC
70          a = (AB[1] * AC[2]) - (AB[2] * AC[1]);
71          b = (AB[2] * AC[0]) - (AB[0] * AC[2]);
72          c = (AB[0] * AC[1]) - (AB[1] * AC[0]);
73          d = (a * vertices[0] + b * vertices[1] + c * vertices[2]);
74
75          axis = 0;
76  }
77
78  void Plane::Display()
79  {
80          // Set's OpenGL's color to the color of the Plane
81          glColor4f(color[0], color[1], color[2], color[3]);
82
```

```cpp
83              glBegin(GL_QUADS);
84              glVertex3d(vertices[0], vertices[1], vertices[2]);
85              glVertex3d(vertices[3], vertices[4], vertices[5]);
86              glVertex3d(vertices[6], vertices[7], vertices[8]);
87              glVertex3d(vertices[9], vertices[10], vertices[11]);
88              glEnd();
89  }
90
91  void Plane::Display2D()
92  {
93              glColor4f(color[0], color[1], color[2], color[3]);
94
95              glBegin(GL_QUADS);
96              glVertex2d(vertices[0], vertices[1]);
97              glVertex2d(vertices[3], vertices[4]);
98              glVertex2d(vertices[6], vertices[7]);
99              glVertex2d(vertices[9], vertices[10]);
100             glEnd();
101 }
102
103 bool Plane::isInBounds()
104 {
105             if (axis == 'x')
106             {
107                     vector<double> X = { vertices[0], vertices[3], vertices[6],
                            vertices[9] };
108                     double maxX = *max_element(X.begin(), X.end());
109                     double minX = *min_element(X.begin(), X.end());
110
111                     return (-Cam.x <= maxX && -Cam.x >= minX);
112
113             }
114
115             else if (axis == 'y')
116             {
117                     vector<double> Y = { vertices[1], vertices[4], vertices[7],
                            vertices[10] };
118                     double maxY = *max_element(Y.begin(), Y.end());
119                     double minY = *min_element(Y.begin(), Y.end());
120
121                     return (-Cam.y <= maxY && -Cam.x >= minY);
122             }
123
124             else if (axis == 'z')
125             {
126                     vector<double> Z = { vertices[2], vertices[5], vertices[8],
                            vertices[11] };
127                     double maxZ = *max_element(Z.begin(), Z.end());
128                     double minZ = *min_element(Z.begin(), Z.end());
129
130                     return (-Cam.z <= maxZ && -Cam.z >= minZ);
131             }
132             else return false;
133 }
134
135 double Plane::getNorm()
```

```
136  {
137          return sqrt ( a * a + b * b + c * c );
138  }
139
140  void Plane :: mutate ()
141  {
142          // We're gonna mess stuff up , disable the axis so nothing funky happens
                  with collision
143          axis = ' ';
144
145          for ( unsigned int i = 0; i < 12; i ++)
146          {
147                  // 0 <= mutator <= 200
148                  double mutator = rand () % 201;
149                  // -100 <= mutator <= 100
150                  mutator -= 100;
151                  // -.01 <= mutator <= .01
152                  mutator /= 10000;
153
154                  vertices [ i ] += mutator ;
155          }
156  }
```

3.1.31 Return.h

```
 1  /**********************************************************\
 2   * Return . h                                              *
 3   * This file was created by Jeremy Greenburg              *
 4   * As part of The God Core game for the University of     *
 5   * Tennessee at Martin's University Scholars Organization *
 6   *                                                        *
 7   * This file contains varius return codes for when things *
 8   * Go horribly wrong ( and they do )                      *
 9   * ( just hopefully not during my senior defense )        *
10  \**********************************************************/
11
12  #ifndef RETURN_H
13  #define RETURN_H
14
15  #define EXIT_OK 0 // Indicates an intended exit
16  #define EXIT_EARLY 1 // If we exit OpenGL main loop early
17  #define FMOD_ERROR 2 // Fmod can't load sound
18  #define DATABASE_ERROR 3 // sqlite can't load database
19  #define STATEMENT_ERROR 4 // sqlite statement fails to execute
20  #define SOIL_ERROR 5 // SOIl fails to load image
21  #define DATA_ENTRY_ERROR 6 // Indicates an internal error in a database entry
22  #define BINDING_ERROR 7 // Error binding a trigger
23  #define FILE_NOT_FOUND 8 // A file is not found
24
25  #endif
```

3.1.32 Resource.h

```
 1  /**********************************************************\
 2   * Return . h                                              *
 3   * This file was created by Jeremy Greenburg              *
 4   * As part of The God Core game for the University of     *
```

```
 5   * Tennessee at Martin's University Scholars Organization   *
 6   *                                                          *
 7   * This file contains varius return codes for when things   *
 8   * Go horribly wrong (and they do)                          *
 9   * (just hopefully not during my senior defense)            *
10  \**************************************************************/
11
12  #ifndef RETURN_H
13  #define RETURN_H
14
15  #define EXIT_OK 0 // Indicates an intended exit
16  #define EXIT_EARLY 1 // If we exit OpenGL main loop early
17  #define FMOD_ERROR 2 // Fmod can't load sound
18  #define DATABASE_ERROR 3 // sqlite can't load database
19  #define STATEMENT_ERROR 4 // sqlite statement fails to execute
20  #define SOIL_ERROR 5 // SOIl fails to load image
21  #define DATA_ENTRY_ERROR 6 // Indicates an internal error in a database entry
22  #define BINDING_ERROR 7 // Error binding a trigger
23  #define FILE_NOT_FOUND 8 // A file is not found
24
25  #endif
```

3.1.33 SaveManager.h

```
 1  /**************************************************************\
 2   * SaveManager.h                                             *
 3   * This file was created by Jeremy Greenburg                 *
 4   * As part of The God Core game for the University of        *
 5   * Tennessee at Martin's University Scholars Organization    *
 6   *                                                           *
 7   * This file contains the declaration of the SaveManager     *
 8   * Class, which saves data by encrypting an array of strings *
 9   * And writing them to core.sav, or by reading in an array of*
10   * Strings from core.sav and decrypting them                 *
11  \**************************************************************/
12
13  #ifndef SAVEMANAGER_H
14  #define SAVEMANAGER_H
15
16  // Windows API
17  #include <shlobj.h>
18
19  // Because concatenating char*'s is really hard
20  #include <string>
21
22  class SaveManager
23  {
24  private:
25          // The path to core.sav
26          char CHAR_PATH[MAX_PATH];
27          std::string SAVE_PATH;
28
29          // Takes an unencrypted string and returns an encrypted string
30          std::string encrytData(std::string data);
31          // Takes an encrypted string and returns a decrypted string
32          std::string decryptData(std::string data);
33  public:
```

```
34          SaveManager ();
35          // Writes the array of encrypted strings to core.sav
36          void saveLevel ();
37          // Sets global variables to load game
38          bool loadGame ();
39          // Returns the decrypted string in core.sav
40          std :: string readSave ();
41          // Returns true if core.save exists
42          bool checkSave ();
43  };
44
45  #endif
```

3.1.34 SaveManager.cpp

```
1   /***************************************************************\
2    * SaveManager.cpp                                             *
3    * This file was created by Jeremy Greenburg                   *
4    * As part of The God Core game for the University of          *
5    * Tennessee at Martin's University Scholars Organization      *
6    *                                                             *
7    * This file contains the definition of the SaveManager class*
8    * For more information , see SaveManager.h                    *
9   \***************************************************************/
10
11  // Class definition
12  #include "SaveManager.h"
13
14  // File I/O
15  #include <fstream >
16
17  #include "Globals.h"
18
19  #include "Logger.h"
20
21  using namespace std;
22
23  SaveManager :: SaveManager ()
24  {
25          HRESULT ret = SHGetFolderPath (NULL , CSIDL_PERSONAL , NULL ,
                  SHGFP_TYPE_CURRENT , CHAR_PATH );
26          SAVE_PATH = CHAR_PATH;
27          SAVE_PATH += "\\The God Core\\core.sav";
28  }
29
30  string SaveManager :: encrytData (string data)
31  {
32          string ret_str;
33          for (unsigned int i = 0; i < data.length ()*3; i+=3)
34          {
35                  ret_str += data[i/3] + 48;
36                  ret_str += data[i/3] - 48;
37                  ret_str += data[i/3] + 53;
38          }
39          return ret_str;
40  }
41
```

```
42  string SaveManager::decryptData(string data)
43  {
44          string ret_str;
45          for (unsigned int i = 0; i < data.length(); i+=3)
46          {
47                  ret_str += data[i] - 48;
48          }
49
50          return ret_str;
51  }
52
53  string SaveManager::readSave()
54  {
55          Logger log;
56
57          ifstream save(SAVE_PATH);
58          log.logLine("Checking Save integrity.");
59
60          string enc_data; // Encrypted Data
61          string dcr_data; // Decrypted Data
62          save >> enc_data;// Read encrypted data from file
63          dcr_data = decryptData(enc_data); // Decrypt data
64
65          vector<string> output{ "Decrypted Data: ", dcr_data };
66          log.logLine(output);
67
68          save.close();
69
70          return dcr_data;
71  }
72
73  void SaveManager::saveLevel()
74  {
75          ofstream save(SAVE_PATH);
76
77          string input = curr_level + " " + to_string(songNum);
78
79          string encr_str = encrytData(input);
80
81          save << encr_str;
82
83          save.close();
84  }
85
86  bool SaveManager::loadGame()
87  {
88          // might change to vector<string> later
89          string data = readSave();
90          size_t pos = data.find(' ');
91
92          if (pos == string::npos) return false;
93          string savedLevel = data.substr(0, pos);
94          int savedSong = stoi(data.substr(pos + 1));
95
96          int temp_levelNum = getLevelNum(savedLevel);
97
```

```
98              if (temp_levelNum == -1) return false;
99
100             levelNum = temp_levelNum;
101             curr_level = getLevelString(levelNum);
102             songNum = savedSong;
103
104             loading = true;
105             changeSong = true;
106
107             return true;
108     }
109
110     bool SaveManager::checkSave()
111     {
112             ifstream save(SAVE_PATH);
113
114             if (save)
115             {
116                     return true;
117             }
118
119             else
120             {
121                     return false;
122             }
123     }
```

3.1.35 Switch.h

```
1   /***********************************************************\
2    * Switch.h                                                *
3    * This file was created by Jeremy Greenburg               *
4    * As part of The God Core game for the University of      *
5    * Tennessee at Martin's University Scholars Organization  *
6    *                                                         *
7    * This file contains the declaration of the Switch class  *
8    * Which is bound to a Door via pointer and can open and   *
9    * Close the door at will                                  *
10   \***********************************************************/
11
12  #ifndef SWITCH_H
13  #define SWITCH_H
14
15  // Door class
16  #include "Door.h"
17  #include "PoweredObject.h"
18  // Terminal Class
19  #include "Terminal.h"
20
21  // Types
22  #include "GCTypes.h"
23
24  class Switch  : public PoweredObject
25  {
26  private:
27          void* target; // The door that this switch activates
28          // Translation and rotation coordinates
```

```
29          double translate[3], rotate[3];
30
31          // One of the predefined types
32          GCtype targetType;
33
34          // Unique ID
35          std::string id;
36
37  public:
38          // Initializes the translation and rotation matrices
39          Switch(const double(&_translate)[3], const double(&_rotate)[3], GCtype
                  _type, std::string _id, bool _isOn);
40          // Binds the target pointer to an object
41          void assign(void* _target);
42          // Opens/Closes the door
43          void toggleTarget();
44          // Actually draws the switch
45          void Display();
46
47          // Get's the switch's ID
48          std::string getID();
49
50          // Gets the translation coordinates
51          double getX();
52          double getY();
53          double getZ();
54  };
55
56  #endif
```

3.1.36 Switch.cpp

```
1  /************************************************************\
2   * Switch.cpp                                               *
3   * This file was created by Jeremy Greenburg               *
4   * As part of The God Core game for the University of       *
5   * Tennessee at Martin's University Scholars Organization   *
6   *                                                          *
7   * This file contains the definition of the Switch class    *
8   * For more information, see Switch.h                       *
9  \************************************************************/
10
11  // Class decleration
12  #include "Switch.h"
13
14  // Allows copying arrays
15  #include <iterator>
16  #include <utility>
17  #include <algorithm>
18
19  #include "Globals.h"
20
21  // OpenGL API
22  #include <GL\glut.h>
23
24  using namespace std;
25
```

```
26  Switch::Switch(const double(&_translate)[3], const double(&_rotate)[3], GCtype
        _type, string _id, bool _isOn)
27  {
28          // Copies the color
29          copy(begin(_translate), end(_translate), translate);
30
31          // Copies the vertices
32          copy(begin(_rotate), end(_rotate), rotate);
33
34          targetType = _type;
35
36          target = NULL;
37
38          id = _id;
39
40          if (_isOn) activate();
41          else deactivate();
42
43  }
44
45  void Switch::assign(void* _target)
46  {
47          target = _target;
48  }
49
50  void Switch::toggleTarget()
51  {
52          switch (targetType)
53          {
54                  case T_DOOR:
55                  {
56                          Door* t = (Door*)target;
57                          t->isOpen = !t->isOpen;
58                          break;
59                  }
60                  case T_TERMINAL:
61                  {
62                          Terminal* t = (Terminal*)target;
63                          t->toggle();
64                          break;
65                  }
66                  case T_LEVEL_END:
67                  {
68                          levelNum++;
69                          curr_level = getLevelString(levelNum);
70                          loading = true;
71
72                          // TEMP
73                          songNum++;
74                          changeSong = true;
75                  }
76          }
77  }
78
79  void Switch::Display()
80  {
```

```
81          glPushMatrix();
82          glTranslated(translate[0], translate[1], translate[2]);
83          glRotated(rotate[0], 1, 0, 0);
84          glRotated(rotate[1], 0, 1, 0);
85          glRotated(rotate[2], 0, 0, 1);
86
87          glColor3d(0.9, 0.9, 0.9);
88          glutSolidCube(.1);
89
90          switch (targetType)
91          {
92          case T_DOOR:
93                  glColor3d(0, 1, 0);
94                  break;
95          case T_TERMINAL:
96                  glColor3d(1, 0, 0);
97                  break;
98          default:
99                  glColor3d(0, 0, 1);
100         }
101
102         // If powered off, recolor to black
103         if (!checkIfOn()) glColor3d(0, 0, 0);
104
105         glScaled(.5, .5, 1.5);
106         glutSolidCube(.1);
107
108         glPopMatrix();
109 }
110
111 string Switch::getID()
112 {
113         return id;
114 }
115
116 double Switch::getX()
117 {
118         return translate[0];
119 }
120
121 double Switch::getY()
122 {
123         return translate[1];
124 }
125
126 double Switch::getZ()
127 {
128         return translate[2];
129 }
```

3.1.37 Terminal.h

```
1 /************************************************************\
2  * Terminal.h                                               *
3  * This file was created by Jeremy Greenburg                *
4  * As part of The God Core game for the University of        *
5  * Tennessee at Martin's University Scholars Organization    *
```

```cpp
6    *                                                              *
7    * This file contains the declaration of the Terminal class    *
8    * Which draws and manages ingame computer terminals           *
9    * And has nothing to do with terminal illness I swear         *
10   \*************************************************************/
11
12   #ifndef TERMINAL_H
13   #define TERMINAL_H
14
15   #include "TwoD.h" // To inherit 2D class
16   #include "PoweredObject.h"
17
18   #include <cstdlib>
19
20   // For loading pictures
21   #include <SOIL.h>
22
23   #include "TextEngine.h" // To display text to screen
24
25   #include <string>
26
27   #include <GL\glut.h>
28
29   class Terminal : public TwoD, public PoweredObject // Inherit 2D functionality and
          power functionality
30   {
31   private:
32           // text = what the user is typing, input = completed input
33           std::string currentInput, currentText, error, file;
34           std::vector<std::string> history, prompts, content;
35           std::string id;
36
37           // Where to print each item
38           const double INPUT_LINE = SCREENBOTTOM / 7.0;
39           const double ERROR_LINE = INPUT_LINE - 30;
40           const double PROMPT_START = INPUT_LINE + 30;
41           const double CONTENT_START = PROMPT_START + 150;
42
43           // The banner texture
44           GLint bTexture;
45
46           // The user inputed number
47           int num;
48
49           // Print our text
50           TextEngine text;
51
52           // Translation and rotation matrices
53           double translate[3], rotate[3];
54
55           // Draws the actual terminal
56           void draw();
57
58           // Draws a standing terminal
59           void drawStanding();
60
```

```
61        // Draws a wall mounter terminal
62        void drawWallMounted();
63
64        // Processes the user input
65        void processInput();
66
67        // Parse the terminal file
68        void parseFile();
69
70        // The path to the Terminal Files
71        static const char* TERM_PATH;
72
73 public:
74        // Draws the 3D object in the world
75        void Display();
76        // Draws the 2D Terminal screen
77        void DisplayScreen();
78        // Shows the currently typed string
79        void getText(std::string text);
80        // Signifies a completed string to process
81        void getInput(std::string text);
82        // Returns an item in the terminal's log
83        std::string getHist(int count);
84        // Returns the number of items in the terminal's log
85        int getHistNum();
86
87        // Gets the translation coordinates
88        double getX();
89        double getY();
90        double getZ();
91
92        // Get the terminal's ID
93        std::string getID();
94
95        // To construct and initialize the terminal
96        Terminal(const double(&_translate)[3], const double(&_rotate)[3], std::::
              string _file, std::string _id);
97
98 };
99
100 #endif
```

3.1.38 Terminal.cpp

```
 1 /************************************************************\
 2  * Terminal.cpp                                            *
 3  * This file was created by Jeremy Greenburg               *
 4  * As part of The God Core game for the University of      *
 5  * Tennessee at Martin's University Scholars Organization  *
 6  *                                                         *
 7  * This file contains the definition of the Terminal class *
 8  * For more information , see Terminal.h                   *
 9 \************************************************************/
10
11 //
12 // Class declaration
13 #include "Terminal.h"
```

```cpp
14
15  // Planes
16  #include "Plane.h"
17
18  // For system logging
19  #include "Logger.h"
20
21  // Return codes
22  #include "Return.h"
23
24  // Global variables
25  #include "Globals.h"
26
27  // Logger
28  #include "Logger.h"
29
30  // File I/O
31  #include <fstream>
32
33  using namespace std;
34
35  const char* Terminal::TERM_PATH = "Resources\\Text\\";
36
37  void Terminal::getText(std::string text)
38  {
39          currentText = text;
40  }
41
42  void Terminal::getInput(std::string text)
43  {
44          currentInput = text;
45  }
46
47  string Terminal::getHist(int count)
48  {
49          int size = history.size();
50          if (history.empty())
51          {
52                  return "";
53          }
54
55          // If, somehow, a fool manages to get a variable that is out of bounds
56
57          else if (count >= size)
58          {
59                  return history.back();
60          }
61
62          else if (count < 0)
63          {
64                  return history.front();
65          }
66
67          else
68          {
69                  return history[size - count - 1];
```

```
70              }
71      }
72
73      int Terminal :: getHistNum ()
74      {
75              return history . size ();
76      }
77
78      void Terminal :: draw ()
79      {
80              // Completely black background
81              double colors [4] = { 0, 0, 0, 1 };
82              double vertices [12] =
83              {
84                      SCREENLEFT , SCREENTOP , -1,
85                      SCREENLEFT , SCREENBOTTOM , -1,
86                      SCREENRIGHT , SCREENBOTTOM , -1,
87                      SCREENRIGHT , SCREENTOP , -1
88              };
89
90              Plane background{ vertices , colors};
91              background . Display2D ();
92
93
94              // Gotta do the banner manually
95              glEnable ( GL_TEXTURE_2D );
96
97              glBindTexture ( GL_TEXTURE_2D , bTexture ); // Prepares the texture for usage
98
99              glColor3d (1, 1, 1);
100             glBegin ( GL_QUADS );
101             glTexCoord2d (0, 0);     glVertex2d ( SCREENLEFT , SCREENTOP );
102             glTexCoord2d (0, 1); glVertex2d ( SCREENLEFT , SCREENBOTTOM / 9.0);
103             glTexCoord2d (1, 1); glVertex2d ( SCREENRIGHT , SCREENBOTTOM / 9.0);
104             glTexCoord2d (1, 0);     glVertex2d ( SCREENRIGHT , SCREENTOP );
105
106             glEnd ();
107
108             glDisable ( GL_TEXTURE_2D );
109     }
110
111     void Terminal :: DisplayScreen ()
112     {
113             prepare2D ();
114
115             draw ();
116
117             // If we need to proces a command
118             if ( currentInput != "")
119             {
120                     processInput ();
121
122                     history . push_back ( currentInput );
123
124                     currentInput . clear ();
125             }
```

```cpp
126
127                 else
128                 {
129                         // Print all prompts
130                         for (unsigned int i = 0; i < prompts.size(); i++)
131                         {
132                                 text.printString(SCREENLEFT, PROMPT_START + 20 * i, 0, 1,
                                        0, prompts[i]);
133                         }
134
135                         // Print an error
136                         text.printString(SCREENLEFT, ERROR_LINE, 1, 0, 0, error);
137                         // Echo user text
138                         text.printString(SCREENLEFT, INPUT_LINE, 0, 1, 0, ":> " +
                                currentText);
139
140                         // If needed, print content
141                         if (num != -1 && num < (signed int)content.size())
142                         {
143                                 text.openFile(SCREENLEFT, CONTENT_START, 0, 1, 0, file,
                                        content[num]);
144                         }
145                 }
146
147         prepare3D();
148 }
149
150 void Terminal::processInput()
151 {
152         error = "";
153         if (currentInput == "exit" || currentInput == "Exit")
154         {
155                 isInTerminal = false;
156                 history.clear();
157         }
158
159         else if (currentInput == "clear" || currentInput == "Clear")
160         {
161                 num = -1;
162         }
163
164         else if (currentInput == "help" || currentInput == "Help")
165         {
166                 num = 0;
167         }
168
169         else
170         {
171                 string first, last;
172                 size_t pos = currentInput.find(" ");
173
174                 first = currentInput.substr(0, pos); // First half of string
175                 last = currentInput.substr(pos + 1); // Second half of string
176
177                 if (first == "read" || first == "Read")
178                 {
```

```
179                                num = atoi(last.c_str());
180                                if (num <= 0 || num >= (signed int)prompts.size())
181                                {
182                                        error = "ERROR: Invalid file number";
183                                        num = -1;
184                                }
185                        }
186
187                        else
188                        {
189                                error = "ERROR: Invalid Command: " + currentInput;
190                                num = -1;
191                        }
192                }
193 }
194
195 void Terminal::Display()
196 {
197        // Add two styles - Standing and wall mounted
198        glPushMatrix();
199
200        // Initial Positioning and rotation
201        glTranslated(translate[0], translate[1], translate[2]);
202        glRotated(rotate[0], 1, 0, 0);
203        glRotated(rotate[1], 0, 1, 0);
204        glRotated(rotate[2], 0, 0, 1);
205
206        //drawWallMounted();
207        drawStanding();
208
209        glPopMatrix();
210 }
211
212 void Terminal::drawStanding()
213 {
214        // Steel grey
215        glColor3d(.1, .1, .1);
216
217        // Draw Floor mount
218        glPushMatrix();
219        glTranslated(0, -1, 0);
220        glScaled(.5, .1, 1);
221        glutSolidCube(.5);
222        glPopMatrix();
223
224        // Draw leg
225        glPushMatrix();
226        glTranslated(0, -.6, 0);
227        glScaled(.1, .75, .1);
228        glutSolidCube(1);
229        glPopMatrix();
230
231        // Draw Monitor
232        glPushMatrix();
233        glScaled(.1, .5, .7);
234        glutSolidCube(1);
```

```
235
236            // Draw Screen
237            glPushMatrix();
238            // Change Screen based on power
239            if (checkIfOn())
240                    glColor3d(0, 1, 1);
241            else
242                    glColor3d(0, 0, 0);
243
244            glTranslated(-.3, 0, 0);
245            glutSolidCube(.7);
246
247            glPopMatrix();
248
249            glPopMatrix();
250 }
251
252 void Terminal::drawWallMounted()
253 {
254            glColor3d(0, 1, 1);
255            glutSolidSphere(1, 50, 50);
256 }
257
258 double Terminal::getX()
259 {
260            return translate[0];
261 }
262
263 double Terminal::getY()
264 {
265            return translate[1];
266 }
267
268 double Terminal::getZ()
269 {
270            return translate[2];
271 }
272
273 void Terminal::parseFile()
274 {
275            ifstream infile{ TERM_PATH + file};
276            string buff;
277
278            if (!infile)
279            {
280                    Logger log;
281                    vector<string> output = { "FATAL ERROR: File ", file, " NOT FOUND"
                            };
282                    log.logLine(output);
283                    exit(FILE_NOT_FOUND);
284            }
285
286            content.push_back("HELP"); // Help text is always the 0th tag in the
                    terminals
287
288            getline(infile, buff);
```

```cpp
289             prompts.push_back(buff); // Push back the file tag
290             getline(infile, buff);
291
292             while (buff != "<TAGS>")
293             {
294                     size_t pos = buff.find("--");
295                     if (pos != string::npos)
296                     {
297                             prompts.push_back(buff.substr(0, pos));
298                             content.push_back(buff.substr(pos + 3));
299                     }
300                     getline(infile, buff);
301             }
302
303 }
304
305 string Terminal::getID()
306 {
307         return id;
308 }
309
310 Terminal::Terminal(const double(&_translate)[3], const double(&_rotate)[3], string
        _file, string _id)
311 {
312         // Copies the color
313         copy(begin(_translate), end(_translate), translate);
314
315         // Copies the vertices
316         copy(begin(_rotate), end(_rotate), rotate);
317
318         bTexture = SOIL_load_OGL_texture
319                     (
320                             "Resources\\Images\\banner.png",   // Image to load
321                             SOIL_LOAD_AUTO,                                     // ???
322                             SOIL_CREATE_NEW_ID,
323                             SOIL_FLAG_MIPMAPS | SOIL_FLAG_COMPRESS_TO_DXT // !?!?!?!
324                             );
325
326         if (bTexture == 0)
327         {
328                 Logger log;
329                 vector<string> output = { "FATAL ERROR: SOIL cannot load terminal
                    banner", SOIL_last_result() };
330                 log.logLine(output);
331                 exit(SOIL_ERROR);
332         }
333
334         file = _file;
335
336         id = _id;
337
338         num = 0;
339
340         parseFile();
341 }
```

3.1.39 TextEngine.h

```
1   /************************************************************\
2    * TextEngine.h                                             *
3    * This file was created by Jeremy Greenburg               *
4    * As part of The God Core game for the University of       *
5    * Tennessee at Martin's University Scholars Organization   *
6    *                                                          *
7    * This file contains the declaration of the TextEngine class*
8    * Which uses glutBitmapCharacter to print strings into the *
9    * OpenGL window.                                           *
10  \************************************************************/
11
12  #ifndef TEXTENGINE_H
13  #define TEXTENGINE_H
14
15  // For string lengths in displaying text
16  #include <string>
17
18  // For multiple lines of text
19  #include <vector>
20
21  class TextEngine
22  {
23  private:
24          // The path to the game's text files (.log's)
25          static const char* TEXT_PATH;
26          // The offset between lines of characters
27          static const double LINE_OFFSET;
28
29          void displayText(
30                  // 2d start location of the text
31                  double x, double y,
32                  // rgb color of text
33                  double r, double g, double b,
34                  // glut font and text to be displayed
35                  void* font,
36                  std::vector<std::string> text);
37
38          // Searches a text file for text related to the tag, and returns all text
                    within the tag
39          std::vector<std::string> findText(std::string fileName, std::string tag);
40
41  public:
42          // Takes the location to display the text, color of the text,
43          // The file to read from, and a tag to search for
44          void openFile(double x, double y, double r, double g, double b,
45                  std::string fileName, std::string tag);
46
47          // Takes in a string to display
48          void printString(double x, double y, double r, double g, double b,
49                  std::string text);
50
51          // Returns text from fileName specified by tag
52          std::vector<std::string> getText(std::string fileName, std::string tag);
53  };
```

```
54
55  #endif
```

3.1.40 TextEngine.cpp

```
 1  /**************************************************************\
 2   * TextEngine.cpp                                           *
 3   * This file was created by Jeremy Greenburg               *
 4   * As part of The God Core game for the University of      *
 5   * Tennessee at Martin's University Scholars Organization  *
 6   *                                                          *
 7   * This file contains the definition of the TextEngine class *
 8   * For more information, see TextEngine.h                   *
 9  \**************************************************************/
10
11  // TextEngine declaration and std::string
12  #include "TextEngine.h"
13
14  // std::ifstream
15  #include <fstream>
16
17  // Standard I/O for debugging
18  #include <iostream>
19
20  // OpenGL API
21  #include <gl\glut.h>
22
23  using namespace std;
24
25  // Initializing the constants
26  const char* TextEngine::TEXT_PATH = "Resources\\Text\\";
27  const double TextEngine::LINE_OFFSET = 20;
28
29  void TextEngine::displayText(double x, double y,
30          double r, double g, double b,
31          void* font, vector<string> text)
32  {
33          vector<string>::iterator it;
34
35          // Iterates throguh the text vector and prints it to the screen
36          for (it = text.begin(); it != text.end(); it++)
37          {
38                  glColor3d(r, g, b);
39                  glRasterPos2d(x, y);
40
41                  for (unsigned int i = 0; i < it->length(); i++)
42                  {
43                          glutBitmapCharacter(font, (*it)[i]);
44                  }
45
46                  // Because glut does not print newlines
47                  y += LINE_OFFSET;
48          }
49  }
50
51  vector<string> TextEngine::findText(string fileName, string tag)
52  {
```

```
53            // The tags are listed between dollar signs
54            string fullTag = '$' + tag + '$';
55
56            string fullPath = TEXT_PATH + fileName ;
57
58            ifstream infile ( fullPath );
59
60            // Buffer to read in data
61            string buff ;
62            // Array to store strings
63            vector < string > data ;
64
65            // Find the string(s) to read in
66            getline ( infile , buff );
67            while ( infile && buff != fullTag )
68            {
69                    getline ( infile , buff );
70            }
71
72            // Store the string(s)
73            getline ( infile , buff );
74            while ( infile && buff != "$END$")
75            {
76                    data . push_back ( buff );
77                    getline ( infile , buff );
78            }
79
80            infile . close ();
81
82            return data ;
83  }
84
85  void TextEngine :: openFile ( double x , double y ,
86          double r , double g , double b ,
87          string fileName , string tag )
88  {
89          vector < string > input = findText ( fileName , tag );
90
91          displayText (x , y , r , g , b ,
92                  GLUT_BITMAP_HELVETICA_18 ,
93                  input );
94  }
95
96  vector < string > TextEngine :: getText ( string fileName , string tag )
97  {
98          vector < string > input = findText ( fileName , tag );
99
100         return input ;
101 }
102
103 void TextEngine :: printString ( double x , double y , double r , double g , double b ,
104         string text )
105 {
106         glColor3d (r , g , b );
107         glRasterPos2d (x , y );
108
```

```
109            for (unsigned int i = 0; i < text.length(); i++)
110            {
111                    glutBitmapCharacter(GLUT_BITMAP_HELVETICA_18, text[i]);
112            }
113
114            // Vertical spacing
115            y += LINE_OFFSET;
116 }
```

3.1.41   Triangle.h

```
 1  /**************************************************************\
 2   * Triangle.h                                                *
 3   * This file was created by Jeremy Greenburg                 *
 4   * As part of The God Core game for the University of        *
 5   * Tennessee at Martin's University Scholars Organization    *
 6   *                                                           *
 7   * This file contains the declaration of the Triangle class  *
 8   * Which is used to hold the details of a 2D Triangle and    *
 9   * draw  it to the screen                                    *
10  \**************************************************************/
11
12  #ifndef TRIANGLE_H
13  #define TRIANGLE_H
14
15  class Triangle
16  {
17  private:
18          // Arrays containing the colors and the xyz vertices of the triangles
19          double color[4], vertices[9];
20  public:
21          // Takes in the vertices and color of the triangle
22          Triangle(const double(&new_vertices)[9], const double(&new_color)[4]);
23          // Print the triangle in 3D
24          void Display();
25          // Print the triangle in 2D
26          void Display2D();
27  };
28
29  #endif
```

3.1.42   Triangle.cpp

```
 1  /**************************************************************\
 2   * Triangle.h                                                *
 3   * This file was created by Jeremy Greenburg                 *
 4   * As part of The God Core game for the University of        *
 5   * Tennessee at Martin's University Scholars Organization    *
 6   *                                                           *
 7   * This file contains the definition of the triangle class   *
 8   * For more information, see Triangle.h                      *
 9  \**************************************************************/
10
11  // Class declaration
12  #include "Triangle.h"
13
14  // For std::copy
```

```
15  #include <iterator>
16  #include <utility>
17
18  // OpenGL API
19  #include <GL\glut.h>
20
21  using namespace std;
22
23
24  Triangle::Triangle(const double(&new_vertices)[9], const double(&new_color)[4])
25  {
26          // Copies the color entry
27          copy(begin(new_color), end(new_color), color);
28
29          // Copies the vertices
30          copy(begin(new_vertices), end(new_vertices), vertices);
31  }
32
33  void Triangle::Display()
34  {
35          // Sets OpenGL's color to the triangle's color
36          glColor4f(color[0], color[1], color[2], color[3]);
37
38          // Draws the triangle
39          glBegin(GL_TRIANGLES);
40          glVertex3d(vertices[0], vertices[1], vertices[2]);
41          glVertex3d(vertices[3], vertices[4], vertices[5]);
42          glVertex3d(vertices[6], vertices[7], vertices[8]);
43          glEnd();
44  }
45
46  void Triangle::Display2D()
47  {
48          // Set's OpenGL's color to the triangle's color
49          glColor4f(color[0], color[1], color[2], color[3]);
50
51          // Draw's the triangle without the Z vertices
52          glBegin(GL_TRIANGLES);
53          glVertex2d(vertices[0], vertices[1]);
54          glVertex2d(vertices[3], vertices[4]);
55          glVertex2d(vertices[6], vertices[7]);
56          glEnd();
57  }
```

3.1.43 Trigger.h

```
1  /***************************************************************\
2   * Trigger.h                                                   *
3   * This file was created by Jeremy Greenburg                   *
4   * As part of The God Core game for the University of          *
5   * Tennessee at Martin's University Scholars Organization      *
6   *                                                             *
7   * This file contains the declaration of the Trigger class     *
8   * Which can be bound to a trigger-object that, upon use,       *
9   * Will activate a designated target-object.                   *
10  \***************************************************************/
11
```

```
12  #ifndef TRIGGER_H
13  #define TRIGGER_H
14
15  #include "Terminal.h"
16  #include "Switch.h"
17
18  #include "GCTypes.h"
19
20  class Trigger
21  {
22  private :
23          void* trigger ; // The object that activates the target
24          void* target ;  // The object that is activated by the target
25
26          GCtype triggerType ; // The type (defined from GCtypes.h) of the trigger
27          GCtype targetType ;  // The type(defined from GCtypes.h) of the target
28
29          void activateTarget ();
30
31  public :
32          // Get the object type of the trigger
33          int getTriggerType ();
34          // Attempts to trigger the target
35          bool tryToTrigger (void* input , GCtype type );
36          // Binds the triggering object
37          void bindTrigger (void* _trigger );
38          // Binds the target object
39          void bindTarget (void* _target );
40          // Constructor takes in trigger type and target type
41          Trigger (GCtype _triggerType , GCtype _targetType );
42
43  };
44
45  #endif
```

3.1.44   Trigger.cpp

```
 1  /************************************************************\
 2   * Trigger.cpp                                              *
 3   * This file was created by Jeremy Greenburg               *
 4   * As part of The God Core game for the University of      *
 5   * Tennessee at Martin's University Scholars Organization  *
 6   *                                                          *
 7   * This file contains the definition of the Trigger class  *
 8   * For more information , see Trigger.h                     *
 9  \************************************************************/
10
11  #include <cstdlib >
12  #include "Trigger.h"
13
14  int Trigger :: getTriggerType ()
15  {
16          return triggerType ;
17  }
18
19  void Trigger :: activateTarget ()
20  {
```

```
21            switch (targetType)
22            {
23                    case T_TERMINAL:
24                    {
25                            Terminal* t = (Terminal*)target;
26                            t->activate();
27                            break;
28                    }
29                    case T_SWITCH:
30                    {
31                            Switch* s = (Switch*)target;
32                            s->activate();
33                            break;
34                    }
35                    default:
36                    {
37                            break;
38                    }
39            }
40 }
41
42 bool Trigger::tryToTrigger(void* input, GCtype type)
43 {
44            // If this trigger is the correct type
45            if (triggerType != type) return false;
46
47            // If this trigger is the correct object
48            if (trigger != input) return false;
49
50            activateTarget();
51
52            return true;
53 }
54
55 void Trigger::bindTrigger(void* _trigger)
56 {
57            trigger = _trigger;
58 }
59
60 void Trigger::bindTarget(void* _target)
61 {
62            target = _target;
63 }
64
65 Trigger::Trigger(GCtype _triggerType, GCtype _targetType)
66 {
67            trigger = NULL;
68            target = NULL;
69            triggerType = _triggerType;
70            targetType = _targetType;
71 }
```

3.1.45 Triple.h

```
 1 /************************************************************\
 2  * Triple.h                                                *
 3  * This file was created by Jeremy Greenburg              *
```

```
 4   * As part of The God Core game for the University of        *
 5   * Tennessee at Martin's University Scholars Organization    *
 6   *                                                            *
 7   * This file contains the declaration of the Triple class    *
 8   * Which is just a simple 3-tuple really                     *
 9   \***********************************************************/
10
11  #ifndef TRIPLE_H
12  #define TRIPLE_H
13
14  class Triple
15  {
16  public:
17          double a, b, c;
18  };
19
20  // For converting to a triple
21  Triple makeTrip(double _a, double _b, double _c);
22
23  #endif
```

3.1.46   Triple.cpp

```
 1   /***********************************************************\
 2   * Triple.cpp                                                 *
 3   * This file was created by Jeremy Greenburg                  *
 4   * As part of The God Core game for the University of         *
 5   * Tennessee at Martin's University Scholars Organization     *
 6   *                                                            *
 7   * This file contains the definition of the TwoD class        *
 8   * For more information, see Triple.h                         *
 9   \***********************************************************/
10
11  #include "Triple.h"
12
13  Triple makeTrip(double _a, double _b, double _c)
14  {
15          Triple ret;
16          ret.a = _a;
17          ret.b = _b;
18          ret.c = _c;
19
20          return ret;
21  }
```

3.1.47   TwoD.h

```
 1   /***********************************************************\
 2   * TwoD.h                                                     *
 3   * This file was created by Jeremy Greenburg                  *
 4   * As part of The God Core game for the University of         *
 5   * Tennessee at Martin's University Scholars Organization     *
 6   *                                                            *
 7   * This file contains the declaration of the TwoD class       *
 8   * Which is used to hold the data and functionality for       *
 9   * Drawing in 2D with OpenGL                                  *
10   \***********************************************************/
```

```
11
12  #ifndef TWOD
13  #define TWOD
14
15  class TwoD
16  {
17  protected:
18          // The pixel boundaries of the screen
19          const double SCREENTOP = 0, SCREENBOTTOM = 1080,
20                  SCREENLEFT = 0, SCREENRIGHT = 1920;
21
22          // Prepares OpenGL draw in 2D
23          void prepare2D();
24
25          // "Resets" OpenGL to draw in 3D
26          void prepare3D();
27
28  };
29
30  #endif
```

3.1.48 TwoD.cpp

```
1  /****************************************************************\
2   * TwoD.cpp                                                     *
3   * This file was created by Jeremy Greenburg                    *
4   * As part of The God Core game for the University of           *
5   * Tennessee at Martin's University Scholars Organization       *
6   *                                                              *
7   * This file contains the definition of the TwoD class          *
8   * For more information, see TwoD.h                             *
9  \****************************************************************/
10
11  #include "TwoD.h"
12
13  // OpenGL API
14  #include <gl\glut.h>
15
16  void TwoD::prepare2D()
17  {
18          // Disable depth testing
19          glDisable(GL_DEPTH_TEST);
20          // Disable writing to the z buffer
21          glDepthMask(GL_FALSE);
22          // Disables lighting
23          glDisable(GL_LIGHTING);
24
25          // Create an orthogonal matrix to write on
26          glMatrixMode(GL_PROJECTION);
27          glPushMatrix();
28          glLoadIdentity();
29          glOrtho(SCREENLEFT, SCREENRIGHT, SCREENBOTTOM, SCREENTOP, -1, 1);
30          glMatrixMode(GL_MODELVIEW);
31          glPushMatrix();
32          glLoadIdentity();
33  }
34
```

```
35  void TwoD::prepare3D()
36  {
37          // Discards the orthogonal matrices
38          glMatrixMode(GL_PROJECTION);
39          glPopMatrix();
40          glMatrixMode(GL_MODELVIEW);
41          glPopMatrix();
42
43          // Enable depth testing
44          glEnable(GL_DEPTH_TEST);
45          // Enables writing to the z buffer
46          glDepthMask(GL_TRUE);
47          // Renable lighting
48          glEnable(GL_LIGHTING);
49  }
```

3.2   Database

3.2.1   Walls

ID	LEVEL	X1	X2	X3	X4	Y1	Y2	Y3	Y4	Z1	Z2	Z3	Z4	R	G	B	A	Axis
lvlceiling	LEVELZERO	-5	-5	8	8	1	1	1	1	-4	1	1	-4	0.70	0.70	0.70	1	0
lvlfloor	LEVELZERO	-5	-5	8	8	-1	-1	-1	-1	-4	1	1	-4	0.70	0.70	0.70	1	0
room0frntlftwall	LEVELZERO	5	5	5	5	-1	1	1	-1	-4	-4	-2.5	-2.5	0.29	0.29	0.29	1	z
room0frntrghtwall	LEVELZERO	5	5	5	5	-1	1	1	-1	-0.5	-0.5	1	1	0.29	0.29	0.29	1	z
room0backwall	LEVELZERO	-5	-5	-5	-5	-1	1	1	-1	-4	-4	1	1	0.29	0.29	0.29	1	z
room0rghtwall	LEVELZERO	-5	-5	5	5	-1	1	1	-1	1	1	1	1	0.29	0.29	0.29	1	x
room0frnttopwall	LEVELZERO	5	5	5	5	0.5	1	1	0.5	-2.5	-2.5	-0.5	-0.5	0.29	0.29	0.29	1	z
room1lftwall	LEVELZERO	5	5	8	8	-1	1	1	-1	-4	-4	-4	-4	0.29	0.29	0.29	1	x
room1rghtwall	LEVELZERO	5	5	8	8	-1	1	1	-1	1	1	1	1	0.29	0.29	0.29	1	x
room1frntbotwall	LEVELZERO	8	8	8	8	-1	-0.5	-0.5	-1	-4	-4	1	1	0.29	0.29	0.29	1	z
room1frnttopwall	LEVELZERO	8	8	8	8	0.5	1	1	0.5	-4	-4	1	1	0.29	0.29	0.29	1	z
room0lftlftwall	LEVELZERO	-5	-5	-1.5	-1.5	-1	1	1	-1	-4	-4	-4	-4	0.29	0.29	0.29	1	x
room0ftrghtwall	LEVELZERO	1.5	1.5	5	5	-1	1	1	-1	-4	-4	-4	-4	0.29	0.29	0.29	1	x
room0fttopwall	LEVELZERO	-1.5	-1.5	1.5	1.5	0.70	1	1	0.70	-4	-4	-4	-4	0.29	0.29	0.29	1	x
room0frntmidwall	LEVELZERO	8	8	8	8	0.5	-0.5	-0.5	0.5	-4	-4	1	1	0.12	0.56	1	0.598	z
lv1Floor	LEVELONE	30	30	-30	-30	-1	-1	-1	-1	40	-5	-5	40	0.70	0.70	0.70	1	0
lv1HangarCeiling	LEVELONE	15	15	-15	-15	5	5	5	5	5	-5	-5	5	0.70	0.70	0.70	1	0
lv1HangerFrntLeftWall	LEVELONE	15	15	1	1	-1	5	5	-1	5	5	5	5	0.80	0	0	1	x
lv1HangerFrntRghtWall	LEVELONE	-1	-1	-15	-15	-1	5	5	-1	5	5	5	5	0.80	0	0	1	x
lvl1HangerFrntTopWall	LEVELONE	1	1	-1	-1	1	5	5	1	5	5	5	5	0.80	0	0	1	x
lvl1HangerRghtWall	LEVELONE	-15	-15	-15	-15	-1	5	5	-1	5	5	-5	-5	0.80	0	0	1	z
lvl1HangerLeftWall	LEVELONE	15	15	15	15	-1	5	5	-1	5	5	-5	-5	0.80	0	0	1	z
lvl1HangerBckWall	LEVELONE	15	15	-15	-15	-1	5	5	-1	-5	-5	-5	-5	0.12	0.56	0	0.402	x
lvl1HallLftWall1	LEVELONE	3	3	3	3	-1	1	1	-1	5	5	11	11	0.80	0	0	1	z
lvl1HallRghtWall1	LEVELONE	-3	-3	-3	-3	-1	1	1	-1	5	5	8	8	0.80	0	0	1	z
lvl1HallLftWall2	LEVELONE	3	3	3	3	-1	1	1	-1	13	13	19	19	0.80	0	0	1	z
lvl1HallRghtWall2	LEVELONE	-3	-3	-3	-3	-1	1	1	-1	10	10	15	15	0.80	0	0	1	z
lvl1HallRghtWall3	LEVELONE	-3	-3	-3	-3	-1	1	1	-1	17	17	19	19	0.80	0	0	1	z
lvl1HallCeiling	LEVELONE	15	15	-15	-15	1	1	1	1	5	29	29	5	0.70	0.70	0.70	1	0
lvl1HallLftEnd	LEVELONE	7.5	7.5	1	1	-1	1	1	-1	19	19	19	19	0.80	0	0	1	x
lvl1HallRghtEnd	LEVELONE	-3	-3	-1	-1	-1	1	1	-1	19	19	19	19	0.80	0	0	1	x
lvl1LftRoom1LeftWall	LEVELONE	-3	-3	-15	-15	-1	1	1	-1	19	19	19	19	0.80	0	0	1	x
lvl1LftRoom1RghtWall	LEVELONE	-3	-3	-15	-15	-1	1	1	-1	12.5	12.5	12.5	12.5	0.80	0	0	1	x
lvl1LftRoomsBckWall	LEVELONE	-15	-15	-15	-15	-1	1	1	-1	19	19	5	5	0.80	0	0	1	z
lvl1Room4LeftWall	LEVELONE	7.5	7.5	7.5	7.5	-1	1	1	-1	19	19	29	29	0.80	0	0	1	z
lvl1Room4RghtWall	LEVELONE	-7.5	-7.5	-7.5	-7.5	-1	1	1	-1	19	19	29	29	0.80	0	0	1	z
lvl1Room3LeftWall	LEVELONE	20	20	20	20	-1	1	1	-1	10	10	-3	-3	0.80	0	0	1	z
lvl1Room3BckWall	LEVELONE	20	20	15	15	-1	1	1	-1	-3	-3	-3	-3	0.80	0	0	1	x
lvl1Room3FrntWall1	LEVELONE	20	20	6	6	-1	1	1	-1	10	10	10	10	0.80	0	0	1	x
lvl1Room3FrntWall2	LEVELONE	4	4	3	3	-1	1	1	-1	10	10	10	10	0.80	0	0	1	x
lvl1Room2LftWall	LEVELONE	7.5	7.5	7.5	7.5	-1	1	1	-1	19	19	10	10	0.80	0	0	1	z
lvl1Room3Ceiling	LEVELONE	20	20	15	15	1	1	1	1	-3	10	10	-3	0.70	0.70	0.70	1	0
lvl1Room4BckWallLft	LEVELONE	7.5	7.5	1	1	-1	1	1	-1	29	29	29	29	0.80	0	0	1	x
lvl1Room4BckWallRght	LEVELONE	-1	-1	-7.5	-7.5	-1	1	1	-1	29	29	29	29	0.80	0	0	1	x
lvl2Room0BckWallLft	LEVELTWO	5	5	1	1	-1	1	1	-1	2	2	2	2	0	0.80	0	1	x
lvl2Room0BckWallRght	LEVELTWO	-5	-5	-1	-1	-1	1	1	-1	2	2	2	2	0	0.80	0	1	x
lvl2Room0LftWallLft	LEVELTWO	-5	-5	-5	-5	-1	1	1	-1	2	2	1	1	0	0.80	0	1	z
lvl2Room0LftWallRght	LEVELTWO	-5	-5	-5	-5	-1	1	1	-1	-1	-1	-2	-2	0	0.80	0	1	z
Lvl2Room0FrtnWall	LEVELTWO	-5	-5	5	5	-1	1	1	-1	-2	-2	-2	-2	0	0.80	0	1	x
Lvl2Room0RghtwallLft	LEVELTWO	5	5	5	5	-1	1	1	-1	-2	-2	-1	-1	0	0.80	0	1	z
Lvl2Room0RghtWallRght	LEVELTWO	5	5	5	5	-1	1	1	-1	1	1	2	2	0	0.80	0	1	z
Lvl2Room0FakeDoor	LEVELTWO	-1	-1	1	1	-1	1	1	-1	2	2	2	2	0.29	0.29	0.29	1	x
Lvl2LftHallBckWall	LEVELTWO	-5	-5	-18	-18	-1	1	1	-1	1	1	1	1	0	0.80	0	1	x
lvl2RghtHallBckWall	LEVELTWO	5	5	18	18	1	-1	-1	1	1	1	1	1	0	0.80	0	1	x
Lvl2LftHallLftWall	LEVELTWO	-18	-18	-18	-18	1	-1	-1	1	1	1	-15	-15	0	0.80	0	1	z
lvl2RghtHallRghtWall	LEVELTWO	18	18	18	18	1	-1	-1	1	1	1	-15	-15	0	0.80	0	1	z
lvl2LftHallFrntWallRght	LEVELTWO	-5	-5	-9	-9	-0.5	0.5	0.5	-0.5	-1	-1	-1	-1	0	0.80	0	1	x
lvl2LftHallLftWall	LEVELTWO	-11	-11	-15	-15	-0.5	0.5	0.5	-0.5	-1	-1	-1	-1	0	0.80	0	1	x
lvl2LftHallTopStrip1	LEVELTWO	-5	-5	-15	-15	1	0.5	0.5	1	-1	-1	-1	-1	0	0.80	0	1	x
lvl2LftHallBotStrip1	LEVELTWO	-5	-5	-15	-15	-0.5	-1	-1	-0.5	-1	-1	-1	-1	0	0.80	0	1	x
lvl2RghtHallFrntWallLft	LEVELTWO	5	5	9	9	-0.5	0.5	0.5	-0.5	-1	-1	-1	-1	0	0.80	0	1	x
lvl2RghtHallFrntWallRght	LEVELTWO	11	11	15	15	-0.5	0.5	0.5	-0.5	-1	-1	-1	-1	0	0.80	0	1	x
lvl2RghtHallTopStrip1	LEVELTWO	5	5	15	15	0.5	1	1	0.5	-1	-1	-1	-1	0	0.80	0	1	x
lvl2RghtHallBotStrip1	LEVELTWO	5	5	15	15	-0.5	-1	-1	-0.5	-1	-1	-1	-1	0	0.80	0	1	x
lvl2LftRoomsRghtWall	LEVELTWO	-5	-5	-5	-5	1	-1	-1	1	-2	-2	-12	-12	0	0.80	0	1	z
lvl2RghtRoomsLftWall	LEVELTWO	5	5	5	5	1	-1	-1	1	-2	-2	-12	-12	0	0.80	0	1	z

ID	LEVEL	X1	X2	X3	X4	Y1	Y2	Y3	Y4	Z1	Z2	Z3	Z4	R	G	B	A	Axis
lvl2LftHallBckWall2	LEVELTWO	-18	-18	-1	-1	1	-1	-1	1	-15	-15	-15	-15	0	0.80	0	1	x
lvl2RghtHallBckWall2	LEVELTWO	18	18	1	1	1	-1	-1	1	-15	-15	-15	-15	0	0.80	0	1	x
lvl2BckWall	LEVELTWO	-9	-9	9	9	1	-1	-1	1	-12	-12	-12	-12	0	0.80	0	1	x
lvl2LftDividerLft	LEVELTWO	-15	-15	-11	-11	1	-1	-1	1	-5.5	-5.5	-5.5	-5.5	0	0.80	0	1	x
lvl2LftDividerRght	LEVELTWO	-9	-9	-5	-5	1	-1	-1	1	-5.5	-5.5	-5.5	-5.5	0	0.80	0	1	x
lvl2LftDividerTop	LEVELTWO	-11	-11	-9	-9	0.5	1	1	0.5	-5.5	-5.5	-5.5	-5.5	0	0.80	0	1	x
lvl2LftDividerTBot	LEVELTWO	-11	-11	-9	-9	-0.5	-1	-1	-0.5	-5.5	-5.5	-5.5	-5.5	0	0.80	0	1	x
lvl2RghtDividerLft	LEVELTWO	15	15	11	11	1	-1	-1	1	-5.5	-5.5	-5.5	-5.5	0	0.80	0	1	x
lvl2RghtDividerRght	LEVELTWO	9	9	5	5	1	-1	-1	1	-5.5	-5.5	-5.5	-5.5	0	0.80	0	1	x
lvl2RghtDividerTop	LEVELTWO	11	11	9	9	0.5	1	1	0.5	-5.5	-5.5	-5.5	-5.5	0	0.80	0	1	x
lvl2RghtDividerTBot	LEVELTWO	11	11	9	9	-0.5	-1	-1	-0.5	-5.5	-5.5	-5.5	-5.5	0	0.80	0	1	x
lvl2Ceiling	LEVELTWO	-18	-18	18	18	1	1	1	1	2	-20	-20	2	0.70	0.70	0.70	1	0
lvl2Floor	LEVELTWO	-18	-18	18	18	-1	-1	-1	-1	2	-20	-20	2	0.70	0.70	0.70	1	0
lvl2LftInnerWallLft	LEVELTWO	-15	-15	-15	-15	-1	1	1	-1	-1	-1	-2	-2	0	0.80	0	1	z
lvl2LftInnerWallCtr	LEVELTWO	-15	-15	-15	-15	-1	1	1	-1	-4	-4	-8	-8	0	0.80	0	1	z
lvl2LftInnerWalLRght	LEVELTWO	-15	-15	-15	-15	-1	1	1	-1	-10	-10	-12	-12	0	0.80	0	1	z
lvl2RghtInnerWallLft	LEVELTWO	15	15	15	15	-1	1	1	-1	-1	-1	-2	-2	0	0.80	0	1	z
lvl2RghtInnerWallCtr	LEVELTWO	15	15	15	15	-1	1	1	-1	-4	-4	-8	-8	0	0.80	0	1	z
lvl2RghtInnerWalLRght	LEVELTWO	15	15	15	15	-1	1	1	-1	-10	-10	-12	-12	0	0.80	0	1	z
lvl2EndHallLft	LEVELTWO	-1	-1	-1	-1	-1	1	1	-1	-15	-15	-20	-20	0	0.80	0	1	z
lvl2EndHallRght	LEVELTWO	1	1	1	1	-1	1	1	-1	-15	-15	-20	-20	0	0.80	0	1	z
lvl3StrtWallLft	LEVELTHREE	-1	-1	-13	-13	-1	1	1	-1	1	1	1	1	0	0	0.80	1	x
lvl3StrtWallRght	LEVELTHREE	1	1	13	13	-1	1	1	-1	1	1	1	1	0	0	0.80	1	x
lvl3Room0RghtWall	LEVELTHREE	-8	-8	-8	-8	-1	1	1	-1	1	1	-1	-1	0	0	0.80	1	z
lvl3Room0/1RghtWall	LEVELTHREE	-8	-8	-8	-8	-1	1	1	-1	-3	-3	-7	-7	0	0	0.80	1	z
lvl3Room1/2RghtWall	LEVELTHREE	-8	-8	-8	-8	-1	1	1	-1	-9	-9	-13	-13	0	0	0.80	1	z
lvl3Room2/3RghtWall	LEVELTHREE	-8	-8	-8	-8	-1	1	1	-1	-15	-15	-19	-19	0	0	0.80	1	z
lvl3Room3RghtWall	LEVELTHREE	-8	-8	-8	-8	-1	1	1	-1	-21	-21	-23	-23	0	0	0.80	1	z
lvl3Room7LftWall	LEVELTHREE	8	8	8	8	-1	1	1	-1	1	1	-1	-1	0	0	0.80	1	z
lvl3Room7/6LftWall	LEVELTHREE	8	8	8	8	-1	1	1	-1	-3	-3	-7	-7	0	0	0.80	1	z
lvl3Room6/5LftWall	LEVELTHREE	8	8	8	8	-1	1	1	-1	-9	-9	-13	-13	0	0	0.80	1	z
lvl3Room5/4LftWall	LEVELTHREE	8	8	8	8	-1	1	1	-1	-15	-15	-19	-19	0	0	0.80	1	z
lvl3Room4LftWall	LEVELTHREE	8	8	8	8	-1	1	1	-1	-21	-21	-23	-23	0	0	0.80	1	z
lvl3Ceiling	LEVELTHREE	-13	-13	13	13	1	1	1	1	1	-23	-23	1	0.70	0.70	0.70	1	0
lvl3Floor	LEVELTHREE	-13	-13	13	13	-1	-1	-1	-1	1	-23	-23	1	0.70	0.70	0.70	1	0
lvl3Room0/1Divid	LEVELTHREE	-13	-13	-8	-8	-1	1	1	-1	-5	-5	-5	-5	0	0	0.80	1	x
lvl3Room1/2Divid	LEVELTHREE	-13	-13	-8	-8	-1	1	1	-1	-11	-11	-11	-11	0	0	0.80	1	x
lvl3Room2/3Divid	LEVELTHREE	-13	-13	-8	-8	-1	1	1	-1	-17	-17	-17	-17	0	0	0.80	1	x
lvl3Room7/6Divid	LEVELTHREE	13	13	8	8	-1	1	1	-1	-5	-5	-5	-5	0	0	0.80	1	x
lvl3Room6/5Divid	LEVELTHREE	13	13	8	8	-1	1	1	-1	-11	-11	-11	-11	0	0	0.80	1	x
lvl3Room5/4Divid	LEVELTHREE	13	13	8	8	-1	1	1	-1	-17	-17	-17	-17	0	0	0.80	1	x
lvl3FrntWall	LEVELTHREE	-13	-13	13	13	-1	1	1	-1	-23	-23	-23	-23	0	0	0.80	1	x
lvl3InnerLftWall	LEVELTHREE	-3	-3	-3	-3	-1	1	1	-1	-2	-2	-20	-20	0	0	0.80	1	z
lvl3InnerRghtWall	LEVELTHREE	3	3	3	3	-1	1	1	-1	-2	-2	-20	-20	0	0	0.80	1	z
lvl3Room0LftWall	LEVELTHREE	-13	-13	-13	-13	-1	1	1	-1	1	1	-1	-1	0	0	0.80	1	z
lvl3Room0/1LftWall	LEVELTHREE	-13	-13	-13	-13	-1	1	1	-1	-3	-3	-7	-7	0	0	0.80	1	z
lvl3Room1/2LftWall	LEVELTHREE	-13	-13	-13	-13	-1	1	1	-1	-9	-9	-13	-13	0	0	0.80	1	z
lvl3Room2/3LftWall	LEVELTHREE	-13	-13	-13	-13	-1	1	1	-1	-15	-15	-19	-19	0	0	0.80	1	z
lvl3Room3LftWall	LEVELTHREE	-13	-13	-13	-13	-1	1	1	-1	-21	-21	-23	-23	0	0	0.80	1	z
lvl3Room7RghtWall	LEVELTHREE	13	13	13	13	-1	1	1	-1	1	1	-1	-1	0	0	0.80	1	z
lvl3Room7/6RghtWall	LEVELTHREE	13	13	13	13	-1	1	1	-1	-3	-3	-7	-7	0	0	0.80	1	z
lvl3Room6/5RghtWall	LEVELTHREE	13	13	13	13	-1	1	1	-1	-9	-9	-13	-13	0	0	0.80	1	z
lvl3Room5/4RghtWall	LEVELTHREE	13	13	13	13	-1	1	1	-1	-15	-15	-19	-19	0	0	0.80	1	z
lvl3Room4RghtWall	LEVELTHREE	13	13	13	13	-1	1	1	-1	-21	-21	-23	-23	0	0	0.80	1	z
lvl3Entrance	LEVELTHREE	-1	-1	1	1	-1	1	1	-1	1	1	1	1	0.29	0.29	0.29	1	x
lvl3Room0FakeDoor	LEVELTHREE	-13	-13	-13	-13	-1	-1	-1	-1	-1	-1	-3	-3	0.5	0.5	0.5	1	z
lvl3Room1FakeDoor	LEVELTHREE	-13	-13	-13	-13	-1	1	1	-1	-7	-7	-9	-9	0.5	0.5	0.5	1	z
lvl3Room2FakeDoor	LEVELTHREE	-13	-13	-13	-13	-1	1	1	-1	-13	-13	-15	-15	0.5	0.5	0.5	1	z
lvl3Room3FakeDoor	LEVELTHREE	-13	-13	-13	-13	-1	1	1	-1	-19	-19	-21	-21	0.5	0.5	0.5	1	z
lvl3Room7FakeDoor	LEVELTHREE	13	13	13	13	-1	1	1	-1	-1	-1	-3	-3	0.5	0.5	0.5	1	z
lvl3Room6FakeDoor	LEVELTHREE	13	13	13	13	-1	1	1	-1	-7	-7	-9	-9	0.5	0.5	0.5	1	z
lvl3Room5FakeDoor	LEVELTHREE	13	13	13	13	-1	1	1	-1	-13	-13	-15	-15	0.5	0.5	0.5	1	z
lvl3Room4FakeDoor	LEVELTHREE	13	13	13	13	-1	1	1	-1	-19	-19	-21	-21	0.5	0.5	0.5	1	z
lvl3Room8BckWall	LEVELTHREE	-3	-3	3	3	-1	1	1	-1	-12	-12	-12	-12	0	0	0.80	1	x
lvl3Room9BckWall	LEVELTHREE	-3	-3	3	3	-1	1	1	-1	-15	-15	-15	-15	0	0	0.80	1	x
lvl3Room8FrntWallLft	LEVELTHREE	-3	-3	-1	-1	-1	1	1	-1	-2	-2	-2	-2	0	0	0.80	1	x
lvl3Room8FrntWallRght	LEVELTHREE	3	3	1	1	-1	1	1	-1	-2	-2	-2	-2	0	0	0.80	1	x
lvl3Room9FrntWallLft	LEVELTHREE	-3	-3	-1	-1	-1	1	1	-1	-20	-20	-20	-20	0	0	0.80	1	x
lvl3Room9FrntWallRght	LEVELTHREE	3	3	1	1	-1	1	1	-1	-20	-20	-20	-20	0	0	0.80	1	x
lvl4Room0LftWall	LEVELFOUR	-5	-5	-5	-5	-1	1	1	-1	-2	-2	2	2	0.29	0.29	0.29	1	z
lvl4Room0RghtWall	LEVELFOUR	5	5	5	5	-1	1	1	-1	2	2	-2	-2	0.29	0.29	0.29	1	z
lvl4Room0BckLft	LEVELFOUR	-5	-5	-1	-1	-1	1	1	-1	2	2	2	2	0.29	0.29	0.29	1	x
lvl4Room0BckRght	LEVELFOUR	1	1	5	5	-1	1	1	-1	2	2	2	2	0.29	0.29	0.29	1	x
lvl4Room0Ceiling	LEVELFOUR	5	5	-5	-5	1	1	1	1	2	-2	-2	2	0.70	0.70	0.70	1	0
lvl4Room0Floor	LEVELFOUR	5	5	-5	-5	-1	-1	-1	-1	2	-2	-2	2	0.70	0.70	0.70	1	0
lvl4Entrance	LEVELFOUR	-1	-1	1	1	-1	1	1	-1	2	2	2	2	0.5	0.5	0.5	1	x
lvl4Room1LftWall	LEVELFOUR	10	10	10	10	-5	5	5	-5	-2	-2	-12	-12	0.598	0.598	0.598	1	z
lvl4Room1RghtWall	LEVELFOUR	-10	-10	-10	-10	-5	5	5	-5	-2	-2	-12	-12	0.598	0.598	0.598	1	z
lvl4Room1FrntWall	LEVELFOUR	10	10	-10	-10	-5	5	5	-5	-12	-12	-12	-12	0.598	0.598	0.598	1	x
lvl4Room1BckLftWall	LEVELFOUR	-10	-10	-5	-5	-5	5	5	-5	-2	-2	-2	-2	0.598	0.598	0.598	1	x
lvl4Room1BckRghtWall	LEVELFOUR	10	10	5	5	-5	5	5	-5	-2	-2	-2	-2	0.598	0.598	0.598	1	x
lvl4Room1Ceiling	LEVELFOUR	10	10	-10	-10	5	5	5	5	-2	-12	-12	-2	0.70	0.70	0.70	1	0
lvl4Room1Floor	LEVELFOUR	10	10	-10	-10	-5	-5	-5	-5	-2	-12	-12	-2	0.70	0.70	0.70	1	0
lvl4Room0FrntWall	LEVELFOUR	-5	-5	5	5	-1	1	1	-1	-2	-2	-2	-2	0.101	0.101	0.101	0.902	x
lvl2LftBckWindowLft	LEVELTWO	-15	-15	-11	-11	1	-1	-1	1	-12	-12	-12	-12	0	0.80	0	1	x
lvl2BckWindowTop	LEVELTWO	-11	-11	-9	-9	0.5	1	1	0.5	-12	-12	-12	-12	0	0.80	0	1	x
lvl2BckWindowBot	LEVELTWO	-11	-11	-9	-9	-0.5	-1	-1	-0.5	-12	-12	-12	-12	0	0.80	0	1	x
lvl2RghtBckWindowLft	LEVELTWO	15	15	11	11	1	-1	-1	1	-12	-12	-12	-12	0	0.80	0	1	x
lvl2RghtWindowTop	LEVELTWO	11	11	9	9	0.5	1	1	0.5	-12	-12	-12	-12	0	0.80	0	1	x
lvl2RghtWindowBot	LEVELTWO	11	11	9	9	-0.5	-1	-1	-0.5	-12	-12	-12	-12	0	0.80	0	1	x
lvl2lftWindow1	LEVELTWO	-11	-11	-9	-9	-0.5	0.5	0.5	-0.5	-1	-1	-1	-1	0	1	1	0.29	x
lvl2lftWindow2	LEVELTWO	-11	-11	-9	-9	-0.5	0.5	0.5	-0.5	-5.5	-5.5	-5.5	-5.5	0	1	1	0.29	x
lvl2lftWindow3	LEVELTWO	-11	-11	-9	-9	-0.5	0.5	0.5	-0.5	-12	-12	-12	-12	0	1	1	0.29	x
lvl2RghtWindow1	LEVELTWO	11	11	9	9	-0.5	0.5	0.5	-0.5	-1	-1	-1	-1	0	1	1	0.29	x
lvl2RghtWindow2	LEVELTWO	11	11	9	9	-0.5	0.5	0.5	-0.5	-5.5	-5.5	-5.5	-5.5	0	1	1	0.29	x
lvl2RghtWindow3	LEVELTWO	11	11	9	9	-0.5	0.5	0.5	-0.5	-12	-12	-12	-12	0	1	1	0.29	x

3.2.2 Doors

ID	LEVEL	X1	X2	X3	X4	Y1	Y2	Y3	Y4	Z1	Z2	Z3	Z4	R	G	B	A	axis
room0room1Door	LEVELZERO	5	5	5	5	-1	0.5	0.5	-1	-2.5	-2.5	-0.5	-0.5	0.90	0.90	0.90	1	z
lvl0ExitDoor	LEVELZERO	-1.5	-1.5	1.5	1.5	-1	0.70	0.70	-1	-4	-4	-4	-4	0.5	0.5	0.5	1	x
lvl1HangerHallDoor	LEVELONE	1	1	-1	-1	-1	1	1	-1	5	5	5	5	0.90	0.90	0.90	1	x
lvl1Room2Door	LEVELONE	3	3	3	3	-1	1	1	-1	11	11	13	13	0.90	0.90	0.90	1	z
lvl1Room0Door	LEVELONE	-3	-3	-3	-3	-1	1	1	-1	17	17	15	15	0.90	0.90	0.90	1	z
lvl1Room1Door	LEVELONE	-3	-3	-3	-3	-1	1	1	-1	10	10	8	8	0.90	0.90	0.90	1	z
lvl1Room3Door	LEVELONE	6	6	4	4	-1	1	1	-1	10	10	10	10	0.90	0.90	0.90	1	x
lvl1Room4Door	LEVELONE	1	1	-1	-1	-1	1	1	-1	19	19	19	19	0.90	0.90	0.90	1	x
lvl1ExitDoor	LEVELONE	1	1	-1	-1	1	-1	-1	1	29	29	29	29	0.5	0.5	0.5	1	x
lvl2Room0Door	LEVELTWO	-15	-15	-15	-15	-1	1	1	-1	-2	-2	-4	-4	0.90	0.90	0.90	1	z
lvl2Room1Door	LEVELTWO	-15	-15	-15	-15	-1	1	1	-1	-8	-8	-10	-10	0.90	0.90	0.90	1	z
lvl2Room2Door	LEVELTWO	15	15	15	15	-1	1	1	-1	-8	-8	-10	-10	0.90	0.90	0.90	1	z
lvl2Room3Door	LEVELTWO	15	15	15	15	-1	1	1	-1	-2	-2	-4	-4	0.90	0.90	0.90	1	z
lvl2ExitDoor	LEVELTWO	-1	-1	1	1	-1	1	1	-1	-20	-20	-20	-20	0.5	0.5	0.5	1	x
lvl3Room0Door	LEVELTHREE	-8	-8	-8	-8	-1	1	1	-1	-1	-1	-3	-3	0.13	0.5504	0.13	1	z
lvl3Room1Door	LEVELTHREE	-8	-8	-8	-8	-1	1	1	-1	-7	-7	-9	-9	0.13	0.5504	0.13	1	z
lvl3Room2Door	LEVELTHREE	-8	-8	-8	-8	-1	1	1	-1	-13	-13	-15	-15	0.70	0.13	0.13	1	z
lvl3Room3Door	LEVELTHREE	-8	-8	-8	-8	-1	1	1	-1	-19	-19	-21	-21	1	0.5504	0	1	z
lvl3Room7Door	LEVELTHREE	8	8	8	8	-1	1	1	-1	-1	-1	-3	-3	0.13	0.5504	0.13	1	z
lvl3Room6Door	LEVELTHREE	8	8	8	8	-1	1	1	-1	-7	-7	-9	-9	1	0.5504	0	1	z
lvl3Room5Door	LEVELTHREE	8	8	8	8	-1	1	1	-1	-13	-13	-15	-15	0.70	0.13	0.13	1	z
lvl3Room4Door	LEVELTHREE	8	8	8	8	-1	1	1	-1	-19	-19	-21	-21	1	0.5504	0	1	z
lvl3Room8Door	LEVELTHREE	-1	-1	1	1	-1	1	1	-1	-2	-2	-2	-2	0.90	0.90	0.90	1	x
lvl3Room9Door	LEVELTHREE	-1	-1	1	1	-1	1	1	-1	-20	-20	-20	-20	0.90	0.90	0.90	1	x

3.2.3 Switches

ID	LEVEL	target	xt	yt	zt	xr	yr	zr	type	startOn
lvl0Door1	LEVELZERO	room0room1Door	5	0	-3	0	90	0	DOOR	1
lvl0Door2	LEVELZERO	t˙lvlzero˙room1	7	0	-4	0	0	0	TERMINAL	1
lvl0END	LEVELZERO	NULL	-2	0	-4	0	0	0	LEVEL˙END	0
lvl1END	LEVELONE	NULL	1.5	0	29	0	0	0	LEVEL˙END	0
lv1Door1	LEVELONE	lvl1HangerHallDoor	1.5	0	5	0	0	0	DOOR	1
lvl1Door2	LEVELONE	lvl1Room2Door	3	0	10	0	90	0	DOOR	1
lvl1Door3	LEVELONE	lvl1Room0Door	-3	0	18	0	90	0	DOOR	1
lvl1Door4	LEVELONE	lvl1Room1Door	-3	0	11	0	90	0	DOOR	1
lvl1Door5	LEVELONE	lvl1Room4Door	1.5	0	19	0	0	0	DOOR	1
lvl1Door6	LEVELONE	lvl1Room3Door	6.5	0	10	0	0	0	DOOR	0
lvl2Door0	LEVELTWO	lvl2Room0Door	-15	0	-5	0	90	0	DOOR	1
lvl2Door1	LEVELTWO	lvl2Room1Door	-15	0	-7	0	90	0	DOOR	0
lvl2Door2	LEVELTWO	lvl2Room2Door	15	0	-7	0	90	0	DOOR	1
lvl2Door3	LEVELTWO	lvl2Room3Door	15	0	-5	0	90	0	DOOR	0
lvl2END	LEVELTWO	NULL	-1	0	-18	0	90	0	LEVEL˙END	0
lvl3Room0	LEVELTHREE	lvl3Room0Door	-8	0	-0.5	0	90	0	DOOR	1
lvl3Room1	LEVELTHREE	lvl3Room1Door	-8	0	-6.5	0	90	0	DOOR	1
lvl3Room2	LEVELTHREE	lvl3Room2Door	-8	0	-12.5	0	90	0	DOOR	0
lvl3Room3	LEVELTHREE	lvl3Room3Door	-8	0	-18.5	0	90	0	DOOR	0
lvl3Room7	LEVELTHREE	lvl3Room7Door	8	0	-0.5	0	90	0	DOOR	1
lvl3Room6	LEVELTHREE	lvl3Room6Door	8	0	-6.5	0	90	0	DOOR	0
lvl3Room5	LEVELTHREE	lvl3Room5Door	8	0	-12.5	0	90	0	DOOR	0
lvl3Room4	LEVELTHREE	lvl3Room4Door	8	0	-18.5	0	90	0	DOOR	0
lvl3Room0END	LEVELTHREE	NULL	-13	0	-0.5	0	90	0	LEVEL˙END	0
lvl3Room1END	LEVELTHREE	NULL	-13	0	-6.5	0	90	0	LEVEL˙END	0
lvl3Room2END	LEVELTHREE	NULL	-13	0	-12.5	0	90	0	LEVEL˙END	0
lvl3Room3END	LEVELTHREE	NULL	-13	0	-18.5	0	90	0	LEVEL˙END	0
lvl3Room7END	LEVELTHREE	NULL	13	0	-0.5	0	90	0	LEVEL˙END	0
lvl3Room6END	LEVELTHREE	NULL	13	0	-6.5	0	90	0	LEVEL˙END	0
lvl3Room5END	LEVELTHREE	NULL	13	0	-12.5	0	90	0	LEVEL˙END	0
lvl3Room4END	LEVELTHREE	NULL	13	0	-18.5	0	90	0	LEVEL˙END	0
lvl3Room8	LEVELTHREE	lvl3Room8Door	-1.5	0	-2	0	0	0	DOOR	0
lvl3Room9	LEVELTHREE	lvl3Room9Door	1.5	0	-20	0	0	0	DOOR	1

3.2.4 Terminals

ID	LEVEL	tag	xt	yt	zt	xr	yr	zr
t'lvlzero'room1	LEVELZERO	lv0TM1.tm	7	0	-2	0	0	0
t'lvl1Room0	LEVELONE	lvl1TM0.tm	-13	0	15	0	180	0
t'lvl1Room1	LEVELONE	lvl1TM1.tm	-13	0	8	0	180	0
t'lvl1Room3	LEVELONE	lvl1TM2.tm	17	0	-2	0	90	0
t'lvl2Room0	LEVELTWO	lvl2TM0.tm	-6	0	-3	0	0	0
t'lvl2Room1	LEVELTWO	lvl2TM1.tm	-6	0	-9	0	0	0
t'lvl2Room2	LEVELTWO	lvl2TM2.tm	6	0	-9	0	180	0
t'lvl2Room3	LEVELTWO	lvl2TM3.tm	6	0	-3	0	180	0
t'lvl3Room0	LEVELTHREE	lvl3TM0.tm	-10	0	-4	180	-90	0
t'lvl3Room1	LEVELTHREE	lvl3TM1.tm	-10	0	-10	180	-90	0
t'lvl3Room2	LEVELTHREE	lvl3TM2.tm	-10	0	-16	180	-90	0
t'lvl3Room3	LEVELTHREE	lvl3TM3.tm	-10	0	-22	180	-90	0
t'lvl3Room4	LEVELTHREE	lvl3TM4.tm	10	0	-22	180	-90	0
t'lvl3Room5	LEVELTHREE	lvl3TM5.tm	10	0	-16	180	-90	0
t'lvl3Room6	LEVELTHREE	lvl3TM6.tm	10	0	-10	180	-90	0
t'lvl3Room7	LEVELTHREE	lvl3TM7.tm	10	0	-4	180	-90	0
t'lvl3Room8	LEVELTHREE	lvl3TM8.tm	0	0	-11	0	90	0
t'lvl3Room9	LEVELTHREE	lvl3TM9.tm	0	0	-16	0	-90	0

3.2.5 Triggers

ID	LEVEL	Trigger	Target	TriggerType	TargetType
tr'lvl0End	LEVELZERO	t'lvlzero'room1	lvl0END	TERMINAL	SWITCH
tr'lvl1Switch	LEVELONE	t'lvl1Room0	lvl1Door6	TERMINAL	SWITCH
tr'lvl1End	LEVELONE	t'lvl1Room3	lvl1END	TERMINAL	SWITCH
tr'lvl2Door1	LEVELTWO	t'lvl2Room0	lvl2Door3	TERMINAL	SWITCH
tr'lvl2Door2	LEVELTWO	t'lvl2Room3	lvl2Door1	TERMINAL	SWITCH
tr'lvl2End	LEVELTWO	t'lvl2Room1	lvl2END	TERMINAL	SWITCH
tr'lvl3Sec1	LEVELTHREE	t'lvl3Room9	lvl3Room3	TERMINAL	SWITCH
tr'lvl3Sec2	LEVELTHREE	t'lvl3Room9	lvl3Room4	TERMINAL	SWITCH
tr'lvl3Sec3	LEVELTHREE	t'lvl3Room9	lvl3Room6	TERMINAL	SWITCH
tr'lvl3Sec4	LEVELTHREE	t'lvl3Room9	lvl3Room8	TERMINAL	SWITCH
tr'lvl3Sec5	LEVELTHREE	t'lvl3Room8	lvl3Room5	TERMINAL	SWITCH
tr'lvl3Sec6	LEVELTHREE	t'lvl3Room8	lvl3Room2	TERMINAL	SWITCH
tr'lvl3End	LEVELTHREE	t'lvl3Room2	lvl3Room2END	TERMINAL	SWITCH

3.2.6 Cylinders

ID	XT	YT	ZT	XR	YR	ZR	base Radius	top Radius	height	stacks	slices	R	G	B	A	LEVEL
lifePodCenter	-4	1	-1.5	90	0	0	0.5	0.5	2	50	50	0	0.80	0.8195	1	LEVELZERO
lifePodBot	-4	0	-1.5	90	0	0	0.0503	1	1	50	50	0.5	0.5	0.5	1	LEVELZERO
lifePodTop	-4	1	-1.5	90	0	0	1	0.5	0.5	50	50	0.5	0.5	0.5	1	LEVELZERO
kiosk1	5	1	25	90	0	0	0.5	0.5	2	50	50	0	0.75	1	0.598	LEVELONE
kiosk2	-5	1	25	90	0	0	0.5	0.5	2	50	50	0	0.75	1	0.598	LEVELONE
power1	-2	1	-5	90	0	0	0.5	0.5	2	50	50	1	0.8397	0	1	LEVELTHREE
power2	2	1	-5	90	0	0	0.5	0.5	2	50	50	1	0.8397	0	1	LEVELTHREE
power3	-2	1	-18	90	0	0	0.5	0.5	2	50	50	1	0.8397	0	1	LEVELTHREE
power4	2	1	-18	90	0	0	0.5	0.5	2	50	50	1	0.8397	0	1	LEVELTHREE
power1BaseBot	-2	0	-5	90	0	0	0.70	0.70	1	50	50	0.201	0.201	0.201	1	LEVELTHREE
power1BaseTop	-2	1	-5	90	0	0	0.70	0.70	0.5	50	50	0.201	0.201	0.201	1	LEVELTHREE
power2BaseBot	2	0	-5	90	0	0	0.70	0.70	1	50	50	0.201	0.201	0.201	1	LEVELTHREE
power2BaseTop	2	1	-5	90	0	0	0.70	0.70	0.5	50	50	0.201	0.201	0.201	1	LEVELTHREE
power3BaseBot	-2	0	-18	90	0	0	0.70	0.70	1	50	50	0.201	0.201	0.201	1	LEVELTHREE
power3BaseTop	-2	1	-18	90	0	0	0.70	0.70	0.5	50	50	0.201	0.201	0.201	1	LEVELTHREE
power4BaseBot	2	0	-18	90	0	0	0.70	0.70	1	50	50	0.201	0.201	0.201	1	LEVELTHREE
power4BaseTop	2	1	-18	90	0	0	0.70	0.70	0.5	50	50	0.201	0.201	0.201	1	LEVELTHREE

3.3   Images

3.3.1   Main Menu

3.3.2   Terminal Banner

3.3.3   Game Icon

3.4   Music