Josiah Craw
35046080

OpenGL Assignment

COSC363

Assignment 1

The scene contains, two alien robots that crawl around, one inside the castle with the spaceship and the other guarding the doorway, the alien spacecraft which hovers up and down while appearing to spin, a simple castle, and a cannon.

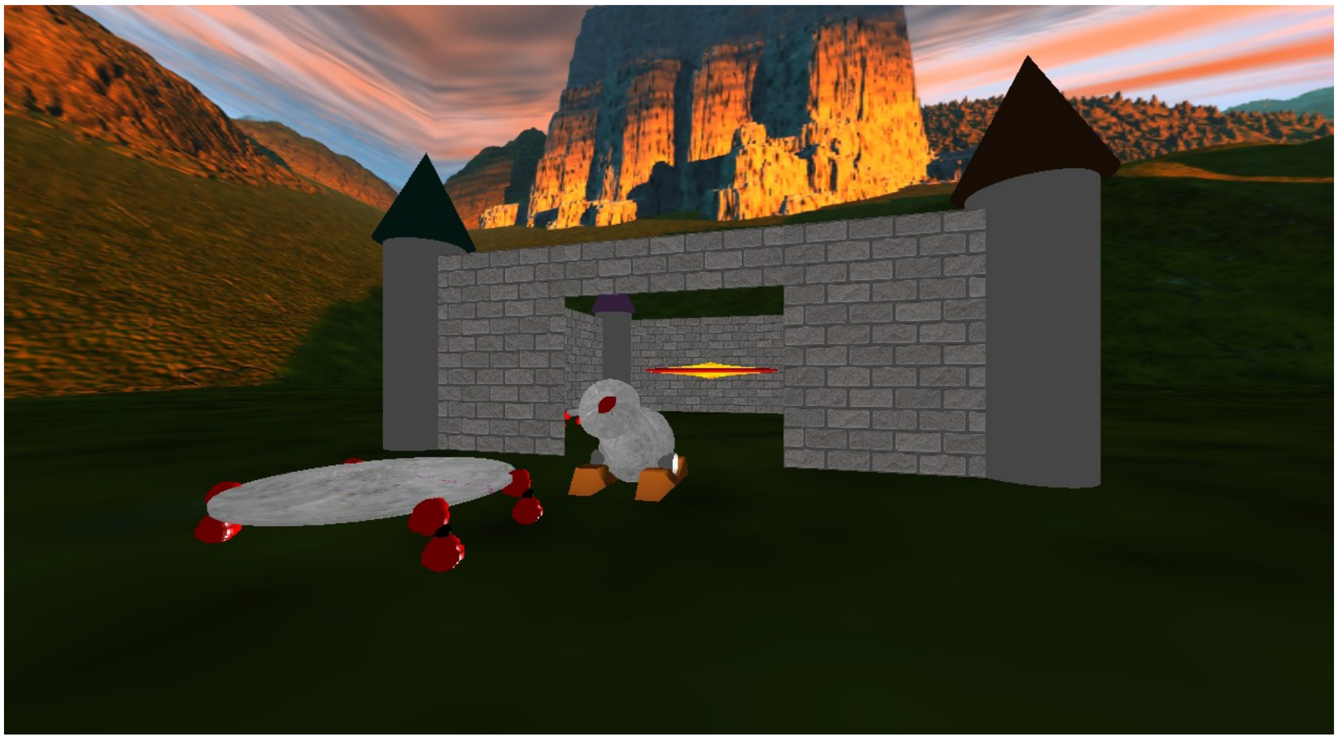

Figure 1: Main view detailing all objects

The Alien spacecraft is hovering in the centre of the castle, with the spheres on the outside rotate about the centre, causing it to appear to be rotating. The spacecraft is constructed of various vertices, using the mathematical formula,

$$y=x^2$$

This formula is then inverted to create the points, with the y values being incremented by one for each point for the flat plane of the spacecraft, which is then revolved to create the shape (**cfg/verts/shipBody.verts**). To finish the edge a second set of vertices are defined to taper off the harsh edge (**cfg/verts/edge.verts**) are also revolved

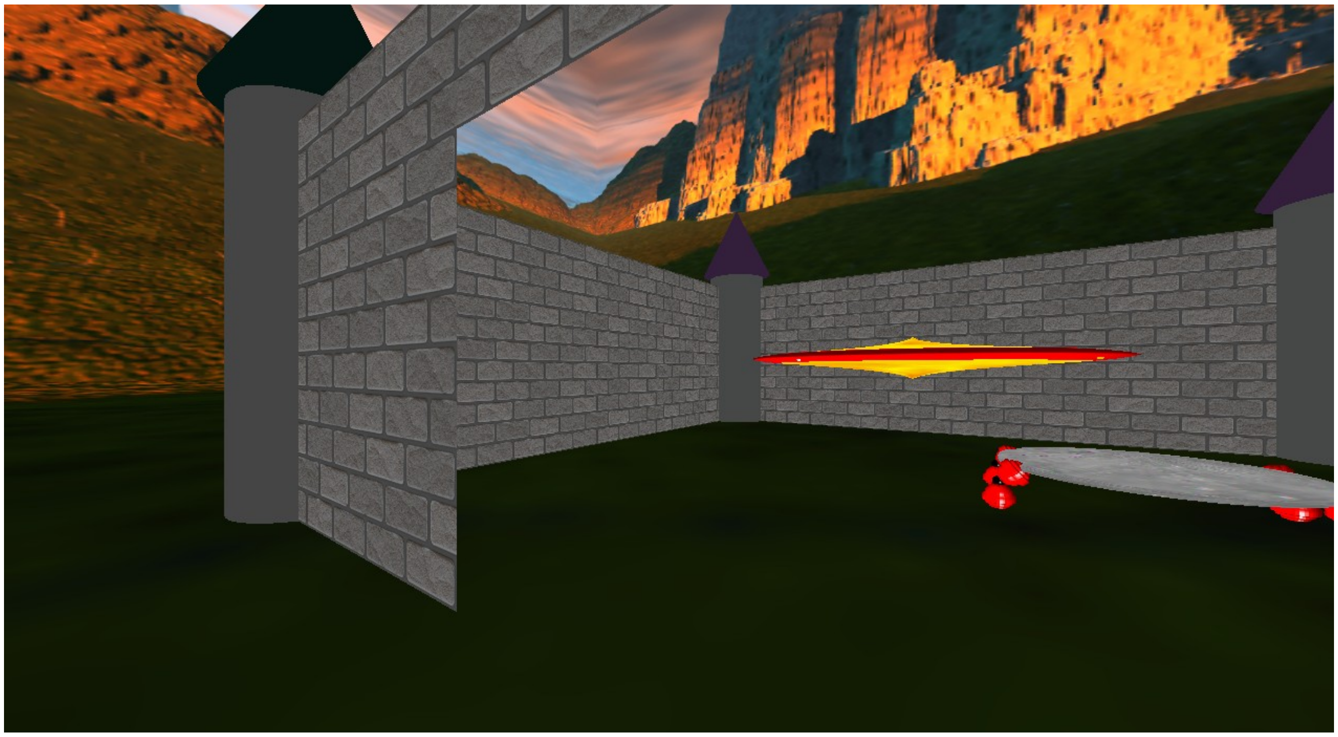

Figure 2: Spacecraft and entry to castle

The hovering of the ship is defined using a set of points (**cfg/paths/up.path**), these points describe the eventual time based transformation of the ship, with the y direction being changed.

A similar method is used to create the robots, with the body being the same as the body for the spaceship and the legs were constructed buy revolving vertices and then the resulting shapes were used to construct the leg.

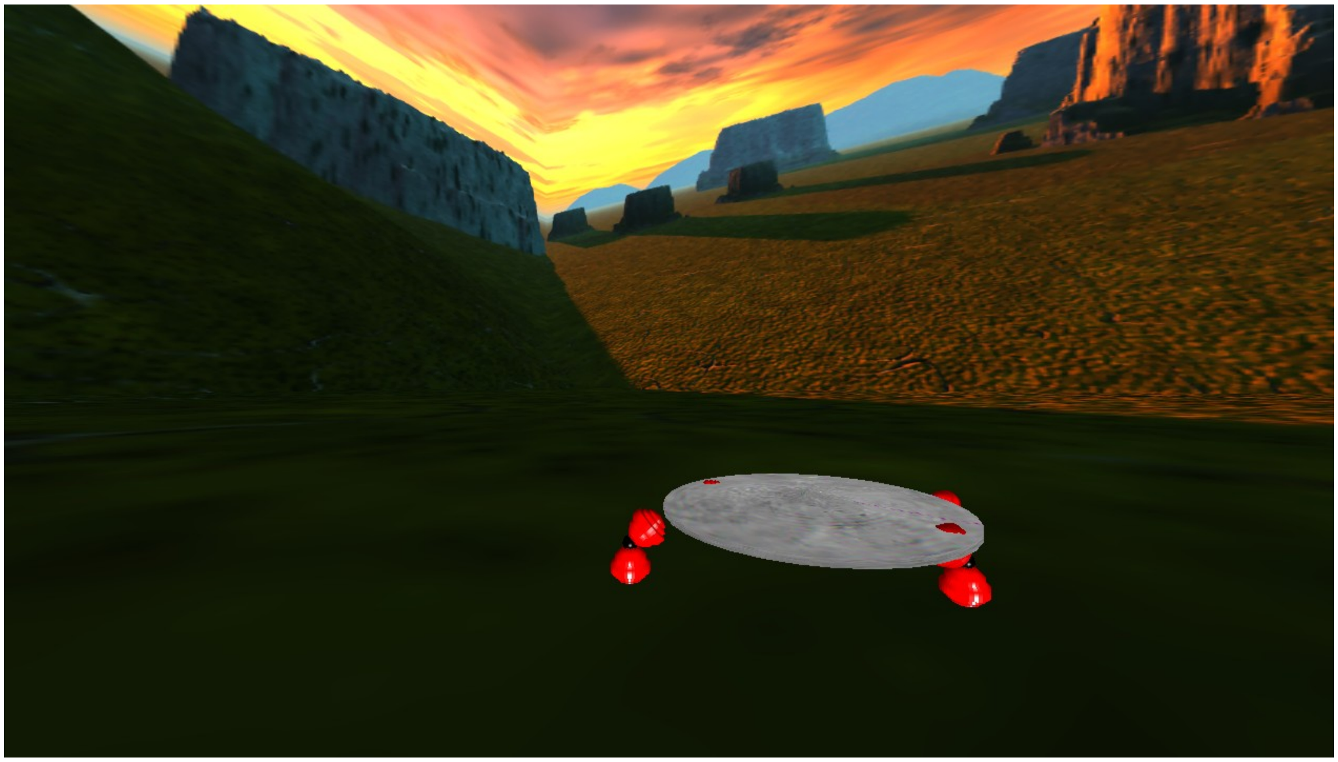
Figure 3: Robot

The movement of the robots, is defined as the motion of the legs (**cfg/paths/walk.path**), and the motion of the entire robot (**cfg/paths/robot**(*x*)**.path**). These coordinates describe the movements, and the time and position that they should rotate.

The cannon is constructed similarly, using a revolved 2D surface defined in (**cfg/verts/cannon.verts**). With the base extruded from the points contained in (**cfg/verts/cannon_leg.verts**).

The firing of the cannonball and the flight path of the spaceship for takeoff are defined in (**cfg/paths/cannonball.path**) and (**cfg/paths/flyAway.path**) respectively. These variable define the relative motion of the opbjects as they are transformed over time.

The relative path of the cannonball is defined by

$$y=-(x^2-140)/100 + 200$$

This is used to define a wide parabola to model the expected flight path of the cannon ball, with 200 added to define a maximum height of 200, the x intercepts of this function are then -141 and 141. The subtraction is to ensure the parabola starts from the origin and moves in the positive x direction. The points used have the x incremented and the response of the y mapped.

The relative path of the spaceship loosely follows the Fibonacci sequence to make the ship appear as if speeding away.

The controls have the left mapped to the left arrow, right to the right arrow, forward to the up arrow, reverse to the down arrow, cannon fire to the ‘c’ key and spaceship takeoff to the ‘s’ key.

The main feature of the program is that it is modular and can be updated while the view is running and update realtime or can be changed after the program is compiled. This is possible though various configuration files and definition files as mentioned above.

Configuration:

1 Texturing

The textures are loaded through the file **cfg/textures.cfg** in the form “*nameOfTexture.ext,fileType(0 or 1 .tga and .bmp respectively),(1 if the texture is to be clamped, any other number if not)*”
The number of textures is recorded in **cfg/numTex.cfg** as *numOfTextures***,**

2 Objects and Transform

Objects are loaded into the program using the command *loadObjFromFile(filename, texId[], bool pressed);* With the bool pressed being used to control button activated sequences, and the textures being passed to texture objects. The files are stored in **cfg/obj/**

2.1 Objects examples

Quads:

QUADS,
x,y,z,s,t,
x,y,z,s,t,
x,y,z,s,t,
x,y,z,s,t,
END,

Cylinder:

CYLINDER,
base_rad,top_rad,height,slices,stacks,

Cone:

CONE,
base_rad,height,slices,stacks,

Sphere:

SPHERE,
radius,slices,stacks,

Rectangle:

RECTANGLE,
width,height,depth,

Revolve:

The file inputted is a .vert file in the form,

x,x,x,x,x,x,x,x,
y,y,y,y,y,y,y,y,y,
z,z,z,z,z,z,z,z,z,

REVOLVE,filename,numVertices,slices,

Extrude:

The file inputted is a .vert file in the form,

x,x,x,x,x,x,x,x,
y,y,y,y,y,y,y,y,y,
z,z,z,z,z,z,z,z,z,

EXTRUDE,filename,numVertices,width,height,depth,

2.2 Transformations examples

Texture:

TEXTURE,textureIndex,param

param = 1,2,3

where 1 is GL_REPLACE, 2 is GL_REPLACE & GL_MODULATE and 3 is GL_MODULATE.

ENDTEX, (Used after the object to be textured has been called)

Scale:

SCALE,scaleX,scaleY,scaleZ,

Translate:

TRANSLATE,x,y,z,

Rotate:

ROTATE,xDir,yDir,xDir,angle,

Colour:

COLOUR,r,g,b,

Motion:
Rotate,

MOVE,ROTATE,anglePerUpdate,angleLimit,rotationalMotionIndex,reverse(0 or 1),x,y,z,
The angle per update defines the amount the object rotates by every update, angleLimit defines the max angle, r rotationalMotionIndex defines the index of the array of current angles this object is referring to and the angle is updated as a global variable, everse defines if the motion should reverse after the angle reaches the angleLimit, x, y and z define the axis which is being rotated.

Linear,

MOVE,LINEAR,filename,numPoints,linearMotionIndex,pressRequired,repeat,
The filename is the name of the path file (**cfg/paths/*.path**), numPoints is the number of points in the path file, pressRequired (1 if true 0 if false) indicates if this motion is only to be activated on keypress, and repeat (1 if true 0 if false) indicates if the motion is to repeat when completed.

The path files for linear movement are structured as follows,

x,x,x,x,x,x,x,x,x,x,x,x,x,x,x,x,x,x,x,x,x,
y,y,y,y,y,y,y,y,y,y,y,y,y,y,y,y,y,y,y,y,y,
z,z,z,z,z,z,z,z,z,z,z,z,z,z,z,z,z,z,z,z,z,
rotateIndex,angle,rotateIndex,angle,END,

Where rotateIndex is the ith value of (x,y,z) where the rotation occurs and angle is the final relative angle of the object being transformed.