COSC 363 Assignment 1 Report

James Brazier, jbr185, 66360439

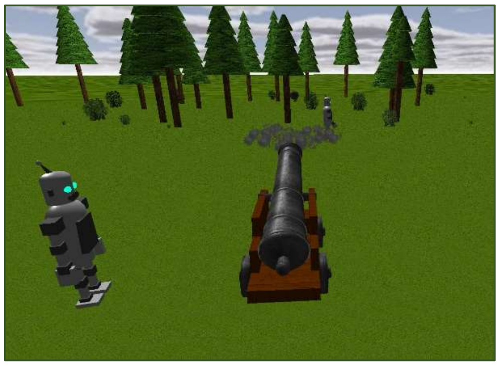
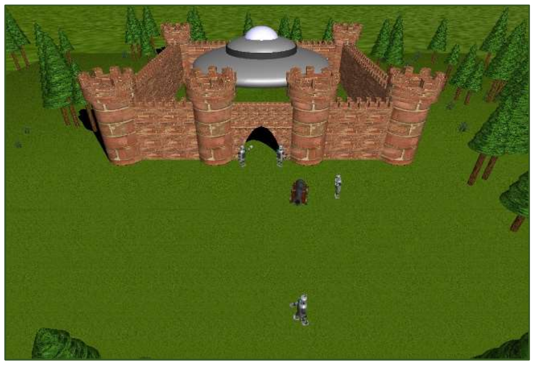
Alien Invasion!

Aliens are planning an invasion of our planet! But before they can begin sending large forces to the surface to conquer us, they have sent a small vessel to analyse and assess our planetary defences. Fortunately for us we spotted the unidentified object early in its mission and employed an EMP device to attempt to disable it, forcing it to land in an old abandoned fort in a small wood with unknown damage. The aliens now in a dangerous and unknown land have started to assess their situation from the safely of their ship by deploying robotic servants. But, with encroaching forces, they have had to employ what weapons they can find, to defend themselves.

The scene is fairly simple; we have the old fortress surrounded by trees and other foliage in the rear of the scene with the alien UFO largely composing the centre. The Aliens have four deployed robots; two guarding the entrance loitering, one scouting outside, and one analysing the nearby cannon just outside the front of the fort.

The objects of the scene

Most of the objects in the scene are composed of GLU shapes and a self-made cube, with a few exceptions. This is so that they can be textured using the GLU quadric functions. The cube also has a texture scale function to allow large cubes to have more than one repeated image to reduce the blurry textures in the scene.

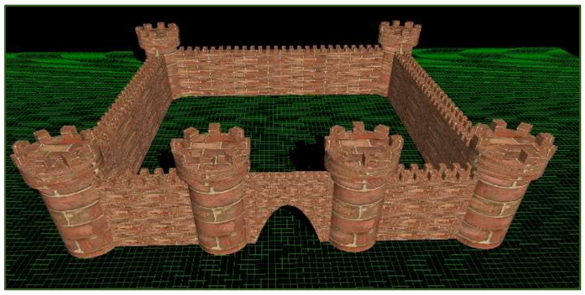
The Castle

The Castle is made of a scaling set of sub-objects that change length and relative position based on the given size of the castle. There are 6 tower objects at the four corners and two for the gate, each made of two cylinders and a set of crenulation cubes. Connecting these are 5 wall objects composing of a single large cube, and again, a set of crenulations.

The castle has a simple brick texture for all objects and is the only object that casts a shadow for is simplicity in colouring and lighting. Unfortunately, the shadow doesn't look good from long distances.

The castle also has a parabolic archway for the entrance, which is generated from the equation:

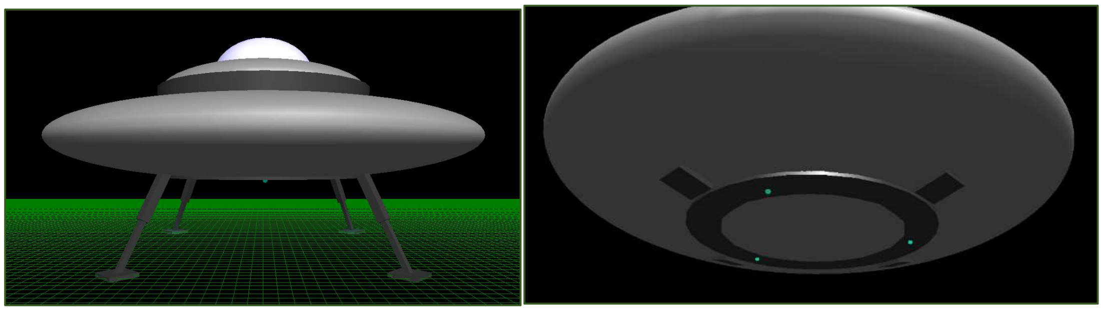
$$y = -0.32x^2 + 8$$



The Space Ship

The Space Ship is made of a set of spheres scaled to make large disks, a set of extendable landing legs, and a rotating cylinder with spotlights on the underside. The ship has specular lighting to show it's advanced and perfect structure.

The ship starts in a landed position with the ability to lift off and fly (with the 'L' key). When flying the legs retract and the spinning cylinder gains speed causing the spotlights to cast an oppressive light over the fort.



The Robots

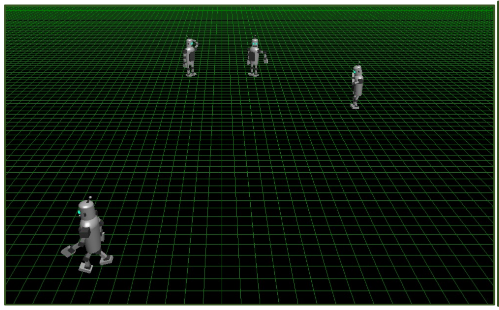
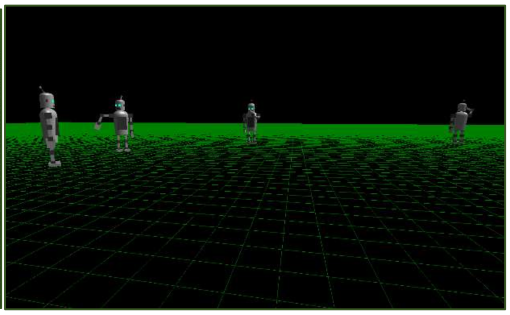
There are four robots in the scene, each are children of the robot superclass meaning that they share common functionality and are created the same way with the same objects. They all have different animations which are in the Units.cpp file. All have specular lighting in the same vain as their vessel.

Each are composed off 6 limbs:

> The head; which is made of a large cylinder with a sphere rounding off the top, two cylindrical eyes (with no lighting effects), a mouth attachment made of two cylinders and a cube, two cube ears, and an antenna to which they are supposedly controlled from.
>
> The torso; which is a swashed cylinder and sphere for the main body and a simple cube on the front to retract from the bland nature of the cylinder.
>
> Two arms; which are created from a set of cubes scaled to form the hand, fore and upper arm, wrist, shoulder, and elbow. The arm is built sequentially from the hand to the shoulder allowing for several rotations approximating a human's arm movement.
>
> Two Legs; which are created very similarly to the arms and allow for approximate human leg movement.

Each of the robots animate as follows:

> The left guard scratches their head with their right arm
>
> The right guard waves their forearm as per the classic robot dance movement and alternates from their left to their right arm and back.
>
> The first scout examines the cannon, leaning down to view it, scratching its chin.
>
> The final robot is pacing left to right looking to the distance, watching for threats.

The Cannon

The cannon similarly is broken into sub-objects; the base and the barrel. The barrel is a set of cylinders and rounding spheres stacked to approximate an 18th century cannon barrel. The base is a set of cubes forming a plank on wheels with a brace for the barrel.

The cannon barrel is built around the cylinder in the base so that the rotation of the barrel for aiming is accurate to its ‘connection’ to the base.

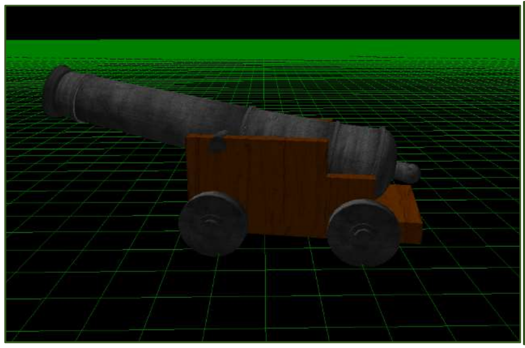
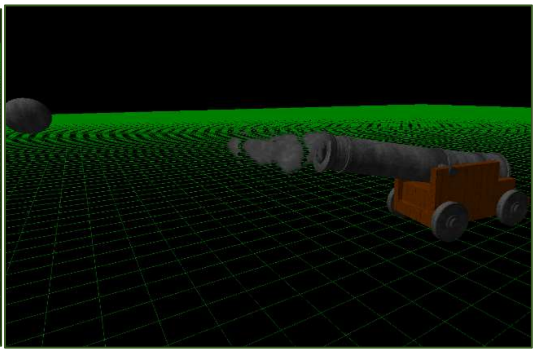
The cannon can be fired with the ‘F’ key, which in turn will start an animation of the cannon recoil, spawn some smoke billboarded quads (particles), and launch a cannon ball. This ball then undergoes a physics approximating animation that uses the velocity, direction of the ball and ‘gravity’ to find the vertical and horizontal movement of the ball, modified by the angle the cannon is facing, over time.

$$At\ each\ Timestep\colon\ Vel_{new} = Vel_{current} - (Gravity + Friction)$$

$$Pos_{new} = Pos_{Current} + Vel_{new}$$

The cannon firing sequence was quite a tricky task to get to work. The cannon ball and smoke particles need to be detached from the cannon transformations so that rotating the cannon doesn’t rotate the flying ball/smoke. So, the X, Y, Z position of the end of the barrel is calculated and the ball/smoke source is placed there. Then the velocity is modified by the same sine and cosine functions to get a correct directional velocity.

See: Particle.cpp, Cannon.cpp & CannonBall.cpp

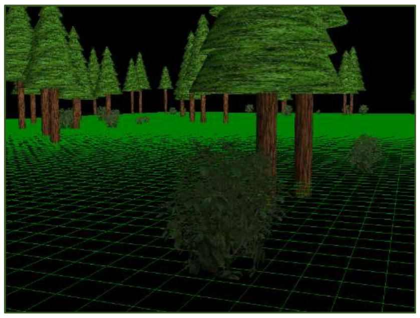



The Foliage

There are two main foliage items; trees and bushes. the trees are made of a trunk and a set of cylinders that form the leaves by coning out. They are controlled by a Tree class that takes the height of the trunk and the height of the leaves, which controls how many cylinders there are. The bushes are simply a hash shape of quads textured with an alpha layered image of a bush.

The bushes and trees are placed around the map by a simple random generator for the X and Z coordinates. There is also a no-go zone for the placement of the objects which will cause a new coordinate set to generate, to ensure there is no foliage in the important parts of the scene.

See: genFoliage() **in Program.cpp**



The Sky Box

The skybox is simple, it is a modified cube with a texture that is mapped to the sides to create the somewhat lacklustre illusion of more world to the scene.

There is however an additional part to the scene; by pressing 'P' a day/night cycle starts, though not the prettiest, it simulates the sun moving around the scene and in turn changes the shadow of the castle. This eventually culminates to the light source for the sun disappearing and plunging the scene into darkness (a.k.a. ambient lighting only). Pressing and holding 'O' will speed this process up.

The Floor

The floor is an edited excerpt from the labs code for the cannon OFF model, but smaller and creates a quad every OpenGL unit. These quads are also individually textured with a piece of the ground of the skybox.

Other Features

The Camera

The gluLookAt() function and the camera modification has been encapsulated in an object to keep the gritty details of how the different camera modes are handled away from view. The object as several functions for changing the look point and moving the camera (as per the control scheme below) which instead of calculating the position and look point every display update, only is recalculated when the parameters change. Is case of the object bind modes, the camera is recalculated along with the object in the refresh loop.

Object Hierarchy

As object orientated programming is the norm now days, the objects in the scene have been encapsulated into simple GLobjects classes. These contain position variables and an orientation angle for the object, as well as an abstract/pure virtual display function to override. These objects also have interfaces for the different types of GLobjects in the scene, such as: GLanimated, for animated objects that must be updated in the refresh loop. GLtextured, for objects that need to load textures. GLlight, for objects that are, or have, light sources so that the GL_LIGHT can be enabled in initialization.

Texture management

As there are objects that individual load textures for themselves as part of the object hierarchy, there is a central texture loading location in the form of the TextureManager object. This object has a single function; loadTexture(**string&** fileName, **void**(*loadFunc)(**string&**)) to load textures from a file. This function will first check the included map structure for the file name, to see if it has been loaded before, and returns that texture name. Otherwise the texture is loaded and stored under the file name in the map.

Controls Summery

	Camera Mode			
Key	**Third Person**	**First Person Roam**	**UFO View**	**Cannon View**
Movement / View				
W	Move view point forward	Move forward	-	Move barrel up
A	Move view point left	Move left	-	Turn cannon left
S	Move view point back	Move back	-	Move barrel down
D	Move view point right	Move right	-	Turn cannon right
Up	Rotate camera up	Look up	-	
Down	Rotate camera down	Look down	-	
Left	Rotate camera left	Look left	-	
Right	Rotate camera right	Look right	-	
Left Shift	Toggle ‘Run’			
Right Alt	Zoom in		-	
Right Ctrl	Zoom out		-	
Other				
R	Change camera mode			
P	Toggle Day/Night cycle			
O	Hold to speed up Day/Night cycle			
F	Fire cannon			
L	Toggle UFO state (land/lift-off)			