Assignment 1: A 3D Model Gallery | Student name: Ahmad Al | Student ID: 14749959

This report shows a scene created using OpenGL, and discusses the code that was implemented, from features to models and more.

Scene Description

The initial scene opens to a camera pointed at a concrete building with surrounding trees; and a door that can be opened and closed. The background (Skybox) used was a snowy terrain background with trees and gloomy skies above.

The following models are in the building:

- A simple character of a human that walks forwards and backwards in the building
- A toy car that moves in a circle on the floor
- A cannon model, that contains a projectile that can be fired with the press of a button
- A snowman that jumps up and down
- A desk that contains a robot paraboloid (surface of revolution model), which rotates.

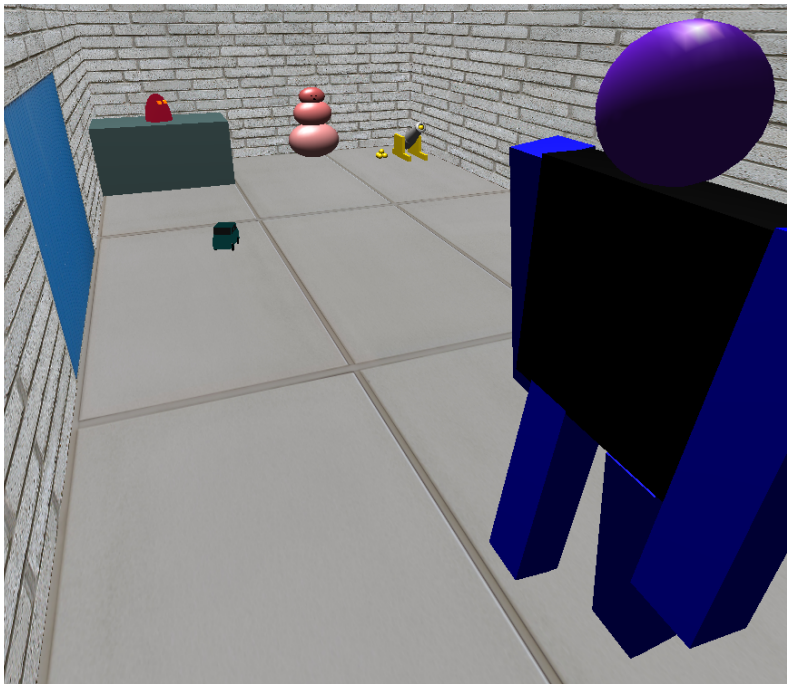

Figure 1: Inside the building and the models present

List of Control Functions

Keyboard:

- **Key 'f':** When this key is pressed, the cannon fires a projectile (can only be used once)
- **Key 'o':** This key is used to open and close the building door

Special Keys:

- **Key 'F1':** Used to switch the camera view (from general view to Humanoid POV)

Figure 2: Scene outside the building

Model Descriptions and Extra Features Added

The Building and the Door:
The building was made by drawing polygons for all the faces, that included the top, bottom, right, left, back and front faces. Texture coordinates are then assigned to allow for textures to be loaded and displayed on these polygons. The front contained 4 polygon structures in it, of which one of those faces was a door. The texture coordinates are calculated for the 4 polygons on the front of the building as one polygon, to combine the textures of the front of the building while allowing for a door.
The door was simply a polygon with a texture loaded on which opened when the key (‘o’) was pressed and closed when the key (‘o’) was pressed again. This was done by changing the coordinates of the door when this key is triggered.

Humanoid:
This model was made by only using GLUT objects. The hands, body, and legs were made by using cubes that were scaled and transformed. The head being the only exception that was made using a sphere.
The humanoid was made to move along a straight line forwards and backwards using the timer by translating the humanoid along the z-axis in increments. Once certain z-axis coordinates are reached, the humanoid was made to walk in the opposite direction (to avoid the humanoid walking through the building).
The humanoid also posses a second camera mode (POV) from his eyes which will be discussed later on in this report.

Toy Car:
The toy car was made solely by vector drawing using quads, except for the wheels which were made using the GLUT object Torus. The car is initially drawn with careful detail, then scaled up. The car is then rotated about the y-axis to give it a circular motion around the floor of the building.

Shooting Cannon Gun (Physics Model):
The most difficult object to implement. Initially a ‘.off’ mesh file is loaded to draw the central cannon part. The base is then drawn around the cannon and a cannon bullet is added to the cannon. More cannon bullets are added on the floor around the base to make the scene more visually pleasing.
When the fire key (‘f’) is pressed, the cannon launches a projectile (the projectile is initially aimed upwards, starting by travelling upwards then slowly descending) with a trajectory based on gravitational and velocity equations which then stops when it hits the floor. The button press was done by using a boolean that makes the firing motion start.

The trajectory was calculated using the following equations and variables:

- Gravity = -9.8m/s
- Delta T (time) = ((glutGet(GLUT_ELAPSED_TIME))/1000)-time_when_f_pressed;
  Where GLUT_ELAPSED_TIME is the time since the program was initialised and ‘time_when_f_pressed’ represents the time at which the fire key ‘f’ was pressed. The times are divided by 1000 to convert the units from milliseconds to seconds.
- Velocity = Gravity * Delta T
- Position_y = Velocity * Delta T, it is important to note that the position values were modified by multiplying and adding numbers to compensate for the scaling and the transformations done to the cannon and projectile.
- Position_x = Previous Position X – Velocity, this is done to calculate the position along the x axis.

Through the above equations with an implementation of global variables and a timer, the Velocity, Delta T, Position_y and Position_x are constantly updated as the projectile is travelling.

Jumping Snowman
The jumping snowman was made by combining 3 spheres, scaling them and transforming them to align to appear as a snowman, with eyes and a nose. The jumping was done by incrementally translating the snowman in the y-axis until it reached a certain height, then back down until it reached the ground, giving it a bouncing motion.

The Spinning Robot Paraboloid (Surface of Revolution), made using Mathematical Equations
This represents a robot with eyes that rotates around himself. The robot was drawn as a paraboloid using the following mathematical equations:

```
float e = sin(M_PI * N_LAT / ((LAT - 1)) - M_PI / 2.0f) / 5;
float r = cos(M_PI * N_LAT / ((LAT - 1)) - M_PI / 2.0f);
float q = (cos(M_PI * N_LONG / (LONG - 1)) * r) / 5;
float w = (sin(M_PI * N_LONG / (LONG - 1)) * r * 2) / 5;

vertices[N_LAT][N_LONG][0] = q;          //  x
vertices[N_LAT][N_LONG][1] = w;   // y
vertices[N_LAT][N_LONG][2] = e;          // z
```

Where LAT and LONG represent the latitude and longitude respectively, and N_LAT and N_LONG represent the count of latitude and longitude respectively. This shape was made based on mathematical equations for Paraboloids of Revolution and through extensive trial and error runs.
After the shape is drawn, eyes are added, they are both scaled and a rotation is introduced using the timer around the y-axis to make the robot spin.

Trees and Desk
These objects were created using only GLUT and GLU objects. The trees were creating by drawing cylinders for the base, then placing two cones on-top of each other, then they were replicated. The desk is simply a solid cube.

Two Camera Modes
The switching camera was done by using a boolean control to switch between the modes when the key ‘F1’ is pressed. The normal general view is the initial camera view, when ‘F1’ is pressed, the camera view changes to what the Humanoid see from his eyes. The camera lookAT parameters are changed to the Humanoid point of view when ‘F1’ is pressed. To preserve the previous camera view, the coordinates are saved before ‘F1’ is pressed and assigned to variables, once ‘F1’ is pressed again, the camera coordinates are changed to saved variables. This ensures that the previous camera view is preserved so that the camera perspective does not change between camera modes.

Difficulties and Challenges
It took some time to adjust to the programming methods of openGL. However, the project started to get harder rapidly. The most challenging and difficult parts of the program was the physics model and the paraboloid robot. The physics models most challenging part was getting a realistic trajectory for the projection. Since the model was scaled up adjustments had to be made to get the required trajectory through trial and error and modifying the equations. The challenge faced with the paraboloid robot was getting the correct equations. Furthermore, when the equations were correct, there were many errors with some vertices bing drawn in random directions, despite the overall shape being almost complete. Through more trial and error, the correct shape was acquired with the correct number of vertices in the array to be drawn. Other difficulties arose in drawing the objects using polygons or quads, in specific, the measurements and in some cases the texture coordinates where combined texturing was required.

Lighting
Two light sources are present in the scene, one pointing inside the building and one outside. Most of the models inside the building have specular reflections added.

Resources and References

1. Skybox graphics, CMM Custom Map Makers, from http://www.custommapmakers.org/skyboxes.php
2. Building floor and building door, Texture Library, from http://texturelib.com/
3. Building outside brick texture, Textures.com from https://www.textures.com/
4. Cannon model file (.off), University of Canterbury, from http://learn.canterbury.ac.nz/