THE UNIVERSITY OF NOTTINGHAM

G53GRA COURSEWORK

Frozen

You Li
psyyl6@nottingham.ac.uk
4256681

May 12, 2017

Contents

List of Figures

1 Overview

This project has aiming at building a 3D graphics application. As required in the specification, Hierarchical Modeling, animation viewing and projection, lighting, texturing, and mouse/keyboard control were implemented in this project.

This project is built with textured sky box, hierarchical modeled forests, a movable lake, the yellow sunlight and aurora, an animated mountain and an animated object loaded from obj file. Most of the code were updated from demos and lab files. The distinction requirements were reached, and improvement on the 3D presentation has been made and anesthetics effects were considered.

The 3D scene and the code were explained in this report, and the improvement from lab code and demo codes were also pointed out during the explanation. The scene was built under the inspiration of the Disney movie Frozen. This is an integrity of the graphics knowledge learn from lab and lectures, also a try of implementation of existing art works.

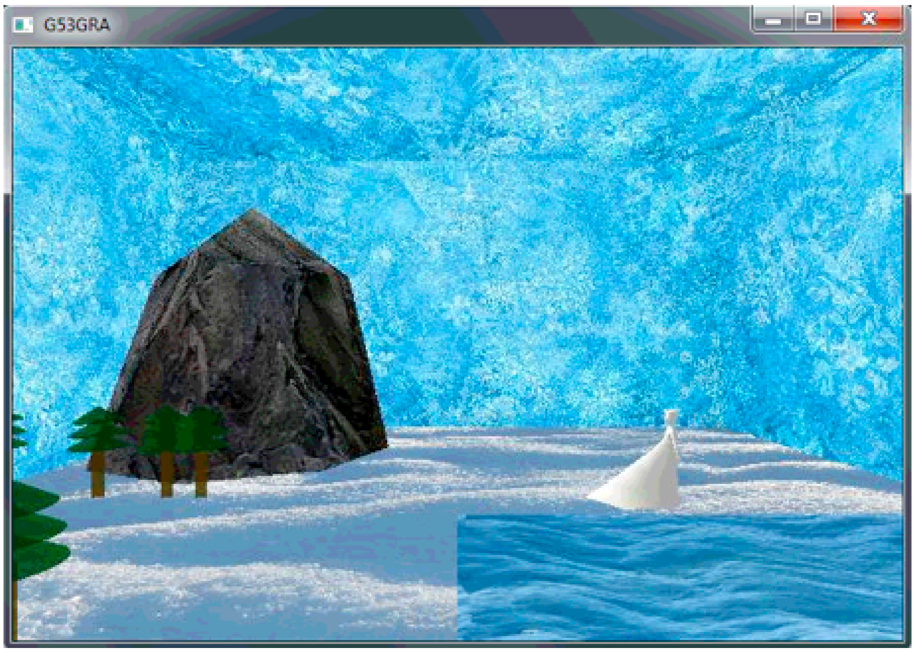

Figure 1: overview screen shot

2 Hierarchical Modeling

As seen in the scene, there are a forest in the lest hand side. Drag the mouse to clear view. The code for drawing a tree is shown as below. The tree is build with

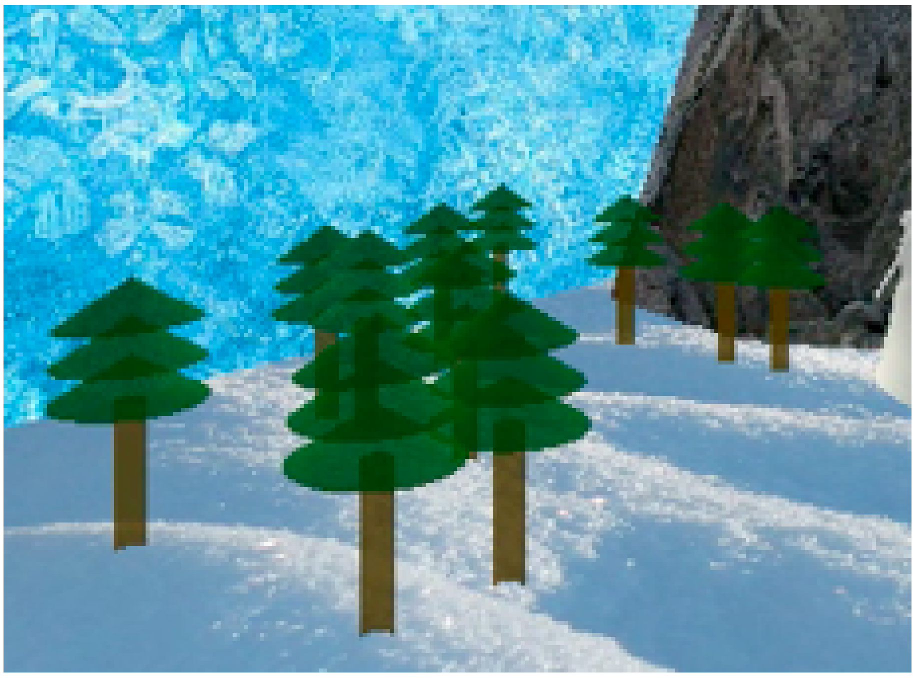

Figure 2: forest screen shot

a cylinder and three cones. The cylinder was referenced from lab code, and for the cone, glutSolidCone() was used. the base was set at 50 pixels,height at 30. faces was set at 300 to make it has a smooth view.

For the tree, cylinder was draw as the base of the tree, then push matrix at the same coordinate of translation to draw the tops. The distance for each tree is 25 pixels. Then popMatrix to go back after drawing.

```
void Tree::Display()
{

        glPushMatrix();
        glPushAttrib(GL_ALL_ATTRIB_BITS);

        glTranslatef(pos[0], pos[1], pos[2]);
        glScalef(scale[0], scale[1], scale[2]);
        glPushMatrix();
        //cylinder
                glTranslatef(pos[0], pos[1], pos[2]);
                glScalef(scale[0], scale[1], scale[2]);
                cylinder(100, 10);
                //cone
                glPushMatrix();
                        glTranslatef(0.0f, 100.f+pos[1], 0.0f);
                        glRotatef(-90, 1.0f, 0.0f, 0.0f);
                        glColor4f(0.f, 0.25f, 0.f,0.85f);
                        glutSolidCone(50, 30, 300, 50);
                        glPushMatrix();
                                glTranslatef(0.0f, 0.f, 25.0f);
                                glScalef(0.9f, 1.0f, 1.0f);
                                glutSolidCone(50, 30, 300, 50);
                                glPushMatrix();
                                        glTranslatef(0.0f, 0.f, 25.0f);
                                        glScalef(0.8f, 1.0f, 1.0f);
                                        glutSolidCone(50, 30, 300, 50);
                                glPopMatrix();
                        glPopMatrix();
                glPopMatrix();
        glPopMatrix();
        //glBindTexture(GL_TEXTURE_2D, 0);
        //glDisable(GL_TEXTURE_2D);

        glPopAttrib();
        glPopMatrix();
}
```

For the Forest, this class inherit from Displayable objects, and set a Tree object. Then draw multiple trees using Display function. The coordinate of the tree was set as the related position of the forest, and further improvement such as generate random coordinate is possible to be implemented.

In this way, we do not need add too many tree objects in the Scene, and only the forest added to the Objects to be displayed.

```
void Forest::Display(){
        glPushMatrix();
        glPushAttrib(GL_ALL_ATTRIB_BITS);

        glTranslatef(pos[0], pos[1], pos[2]);
        glScalef(scale[0], scale[1], scale[2]);

        tree->position(-233.f, 0.f, 0.0f);
        tree->Display();
        ...
        glPopAttrib();
        glPopMatrix();
}
```

Apart from the tree, other objects also implemented with hierarchical modeling, but just take the tree object as an example here.

3 Animation

An animated object Elsa which loaded from obj file was created at the center of the scene. The point, facet and normal value were read from the obj file, which provided a good view for the object. The code was updated from demo code lamp and demo code planet with the help of Will. The improvement of the code were read from file and integrate the planet code with other shapes. Also, the moving path has been updated. During loading the obj file, the coordinate were read from file for each face and normals. Then draw the object for each normal of each faces.

Since the object was loaded as a whole object, so the animation of each part is disabled. So I added the animation around the position of this object. Elsa rotate around the point(0,0.2, 0) at orbitSpeed of 100 and axisSpeed of 100. The update function is listed as follows

```
void Elsa::Update(const double& deltaTime){
        if (move){
                axisRotation +=
        (axisRotationSpeed*static_cast<float>(deltaTime)+pos[0]);
                if (axisRotation > 360.0f)
                {
                        axisRotation =
            axisRotation - 360.0f;
                }
                orbitRotation +=
        orbitRotationSpeed*static_cast<float>(deltaTime);
                if (orbitRotation > 360.0f)
                {
                        orbitRotation =
            orbitRotation - 360.0f;
                }
        }

}
```

The mountain is inherited from Textured Sphere from assignment2, and the improvement have made on texture blending function. Also, some animations were updated. The mountain was oval shaped, with changeable height and base. It can also change its normal numbers by pressing "+" and "-", and the height could be changed via pressing u and i.

4 Viewing and Projection

In this project, the camera is set up at position

```
        eyePosition[0] = 0.0f;
        eyePosition[1] = 300.0f;
        eyePosition[2] = 1.3f * static_cast<float>
    (Scene::GetWindowHeight())
    / static_cast<float>(tan(M_PI / 6.0)); //0.0f;
```

where x is at the center of the box, the y is height at 300 pixel. Since the sky box is set up at size of

```
        static_cast<float>(Scene::GetWindowHeight())
    / static_cast<float>(tan(M_PI / 6.0)); //0.0f;
```

Therefore the viewing position is at the near front of the sky box. The view point is able to change by pressing 'w', 'a', 's', 'd'. Then press ' ' (space) to get back to original view. The view could also changed by mouse dragging and pressing. The size of the sky box will be changed once the screen size is reset.

Following images are the screen shots from different view positions.

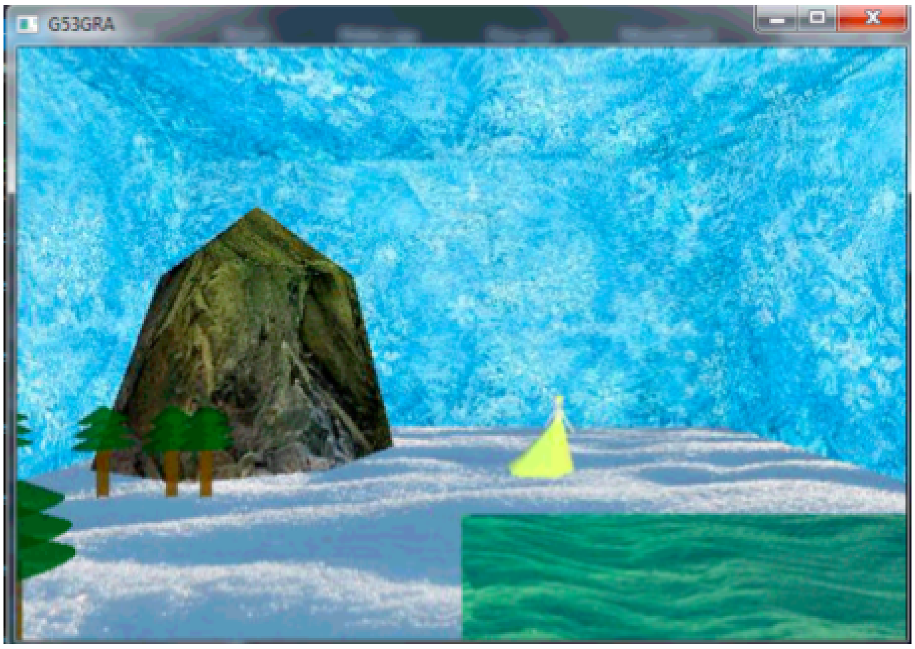

Figure 3: front screen shot

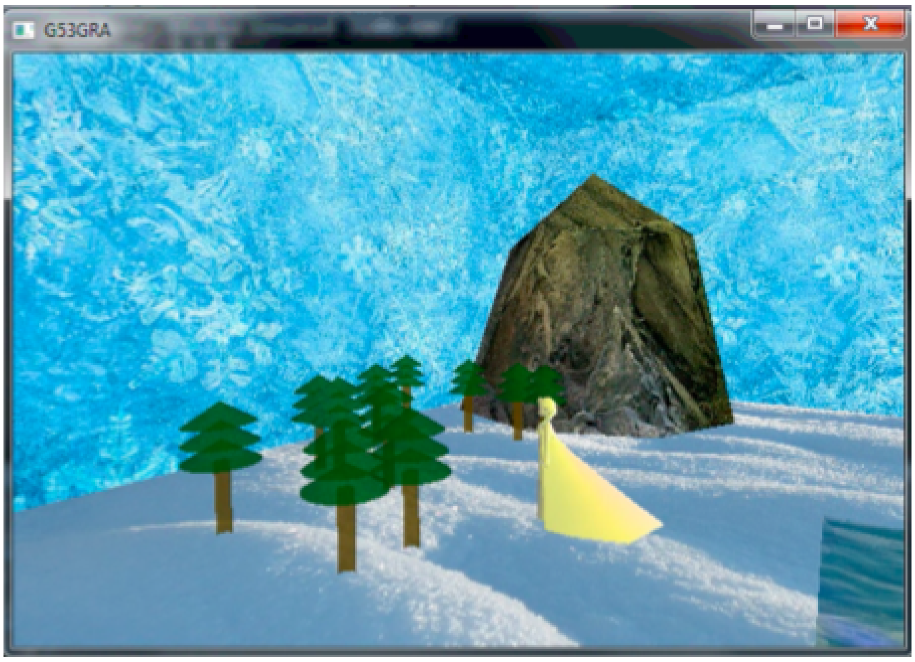

Figure 4: left screen shot

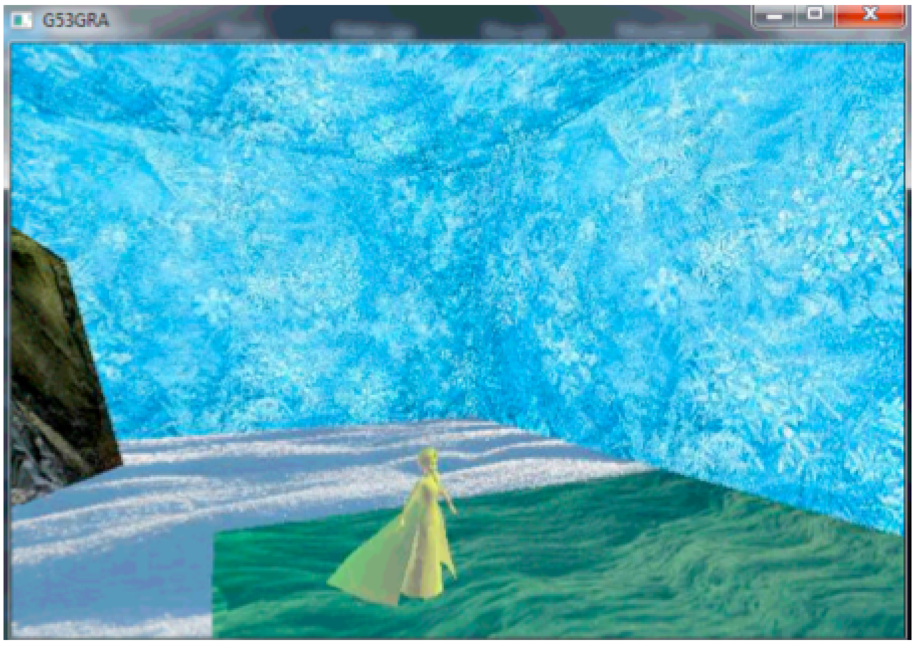

Figure 5: right screen shot

5 Mouse and Keyboard control

The animated objects are able to be controlled by mouse and keyboard input. Overall, this was approached by implementing HandelKey function in each object. and for each input type, use switch case to set separate values for the object. The details are stated in Aimation and Viewing and projection parts.

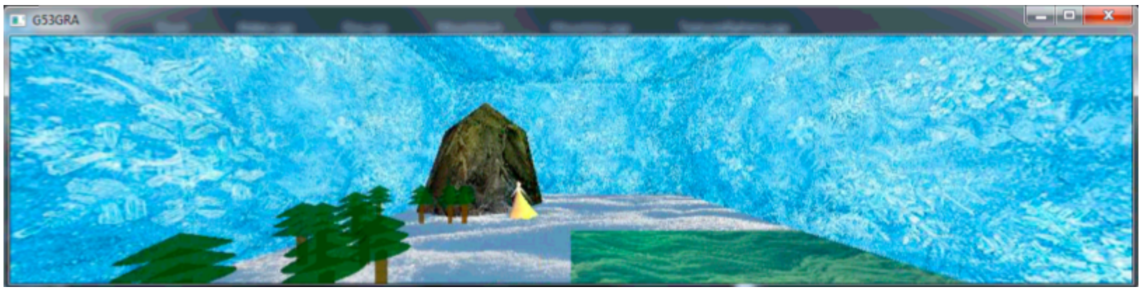

Figure 6: long screen shot

6 lighting

In this project, A single sun light and three moving lights of yellow, red and green were implemented. The light are set as ambient and is able to be turned on and off by pressing 0, 1, 2, 3. The light source is at the top of the box, but the lighting effects are able to be seen inside the box.

7 Texturing

In this project, the sky box was textured with two images, which has different view of the walls and the floor. The Mountain was textured with blended texture of stone and grass. Other projects are simple colored using openGL functions.