COMP3069 Computer Graphics Coursework 2 Report

Gaole Dai

20124917

scygd1@nottingham.edu.cn

School of Computer Science University of Nottingham Ningbo China

Introduction

The Christmas holiday is approaching. Building a 3D scene and attaching it to an electronic gift card will be a brilliant idea. Therefore, the main theme of this 3D scene is Christmas. After collecting some relative information and carefully reviewing all the lecture knowledge, I decided to build an outdoor snowy street view of a *Christmas Village* considering the feasibility. After one month of development, all the design details mentioned in the proposal have been implemented. The main proposed scene and actually developed scene are shown in Figure 1.

This report will outline the related development details and what I have done to meet the requirements according to the order of the criteria mentioned in the coursework sheet.

(a) Proposed Scene [1]

(b) Christmas Village

Figure 1: Plan and Actual 3D Scene

Development Tools and Environment

1. Hardware (CPU): Intel(R) Core(TM) i7-10510U CPU @ 1.80GHz
2. Hardware (GPU): NVIDIA GeForce MX250

3. Software for code programming: Microsoft Visual Studio 2019 Community Version
4. Software for 3D model building: Blender 2.93
5. Programming Language: `C++`
6. OpenGL Library: GLFW (Because of the library conflict, I chose to use GLFW instead of the GLUT taught by lab, it is a great opportunity to extend what I have learned in the classroom to conduct self exploration.)

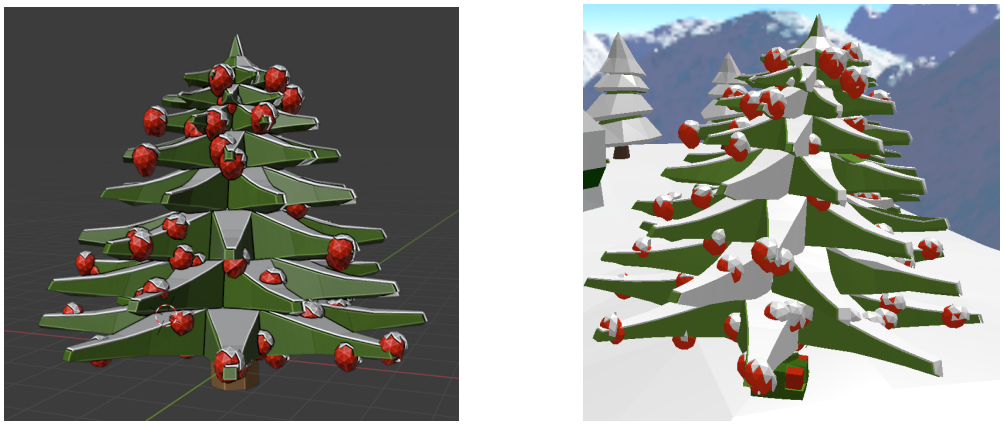
Design Details

3D Models

There are more than eighteen self-built models in the 3D scene, more than I had planned in the proposal. The main objects I have included in the scene are three kinds of Christmas trees, three modes of Santa Claus, three kinds of houses, five kinds of gifts, snowman, cloud, sleigh, fence, wooden bridge, iced lake, basement floor, two types of candy cane, Xmas village sign, rock, smoke, snow, well and castle. Most of the objects are built using Blender. The implementation method and screenshots of each model are in the following paragraphs.

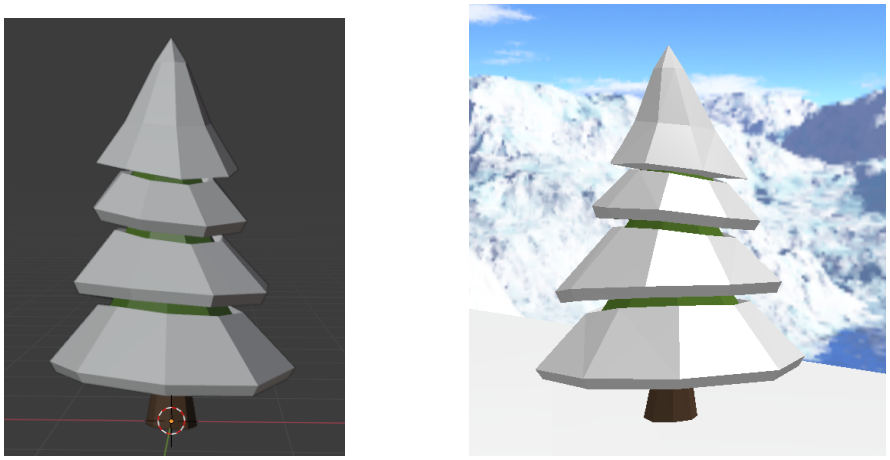
Christmas Tree There are three types of Christmas trees in the scene, they are the decorative Christmas tree (Figure 2), Christmas tree with round canopy (Figure 3), and the Christmas tree without ornaments (Figure 4).

The green part of the Christmas tree is built up layer by layer, first build a base, then replicate, transfer, rotate and scale. The red ornaments randomly distributed on the Christmas tree are obtained by a particle system, and the white snow is produced by selecting the similarity faces and then by an operation called extrude faces along normals. The truck is an eight-sided cylinder.



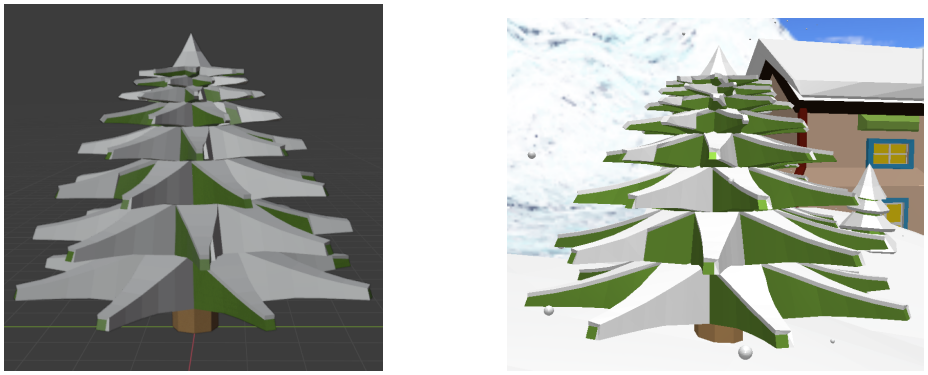
(a) Decorated Tree in Blender (b) Decorated Tree in the Scene

Figure 2: Decorated Christmas Tree



(a) Round Tree in Blender (b) Round Tree in the Scene

Figure 3: Round Christmas Tree



(a) Bare Tree in Blender (b) Bare Tree in the Scene

Figure 4: Bare Christmas Tree

Santa Claus In the animation, the status of the Santa Claus changed, respectively, standing Santa (Figure 5), bending Santa (Figure 6) and sitting Santa (Figure 7) . Santa Claus is built from one side of the rectangular leg upwards, using the `Subdivide` function to segment the plane area and then `Extrude` the different shapes of objects, the symmetry of the two sides using the `Mirror` function.

(a) Santa Claus Standing in Blender

(b) Santa Claus Standing in the Scene

Figure 5: Santa Claus Standing

(a) Santa Claus Bend in Blender

(b) Santa Claus Bend in the Scene

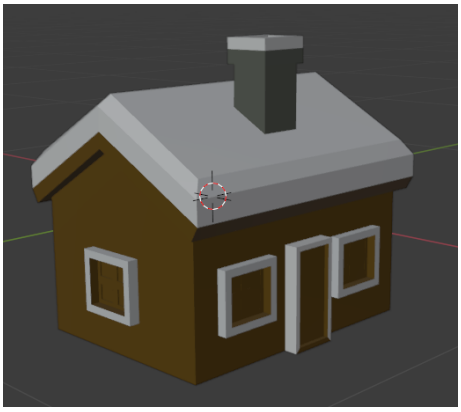
Figure 6: Santa Claus Bend

(a) Santa Claus Sit in Blender

(b) Santa Claus Sit in the Scene

Figure 7: Santa Claus Bend

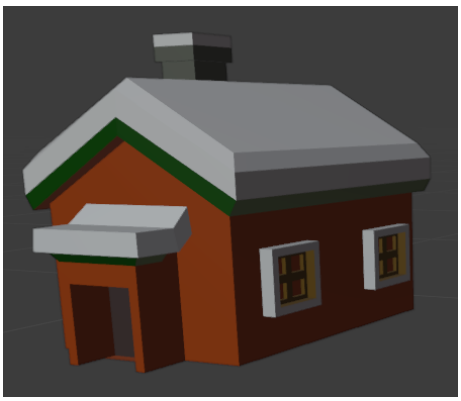
Houses There are a total of three types of houses in the scene: the brown basic house with five windows on the left side of the scene (Figure 8), then the red house on the right side of the scene (Figure 9), which comes with a garage, and finally the double-storey house on the backside of the scene (Figure 10). Each house is topped with a white thickness to simulate snow.



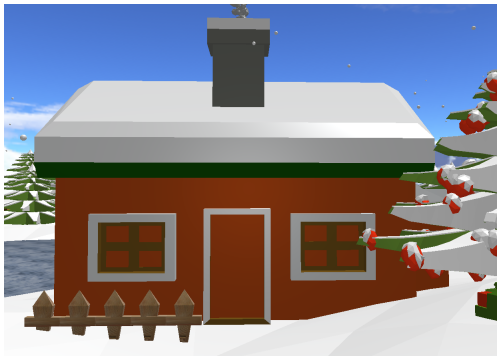
(a) Brown House in Blender

(b) Brown House in the Scene

Figure 8: Brown House

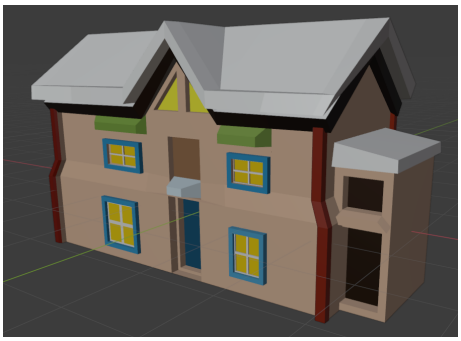


(a) Red House in Blender



(b) Red House Bend in the Scene

Figure 9: Red House

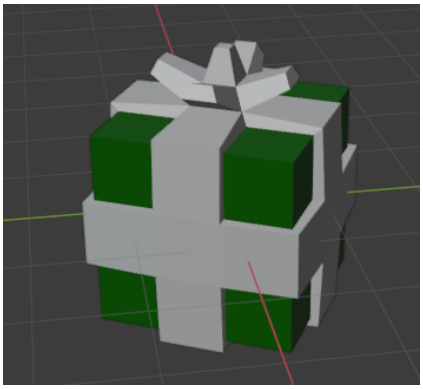


(a) Double-Storey House in Blender

(b) Double-Storey House in the Scene

Figure 10: Double-Storey House

Gift There are five colors of gifts in total as shown in Figure 11 and they are obtained by changing the color through the same gift model. The shape of the gift is a cube, the top strap is built by dividing the surface of the square and then `Extrude` along each face.

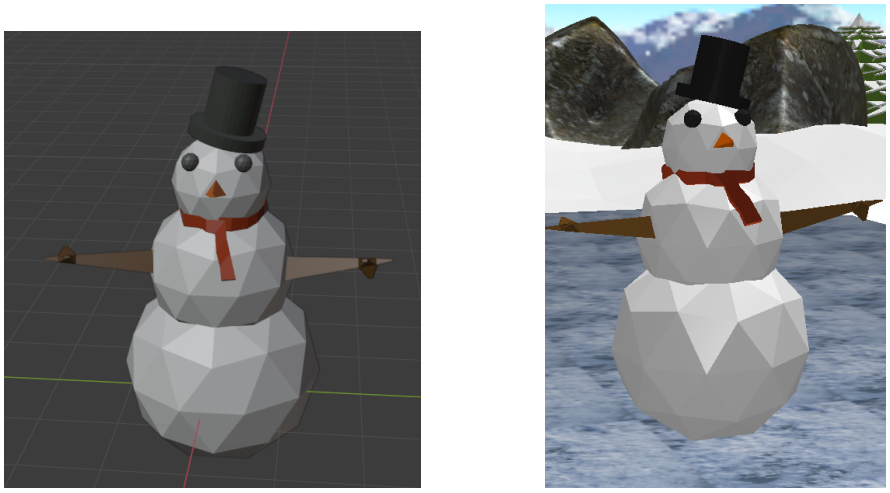


(a) Gifts in Blender

(b) Gifts in the Scene

Figure 11: Gifts

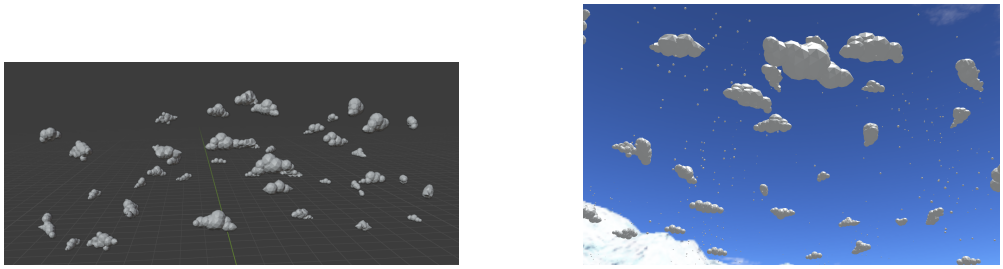
Snowman The snowman in the scene follows the same hierarchy modeling method as in coursework 1. The body part is composed of three spheres, the arm is `Extrude` and `Scale` a certain triangular face in the sphere, the black hat is a cylinder, and the scarf is a rectangular plane.



(a) Snowman in Blender (b) Snowman in the Scene

Figure 12: Snowman

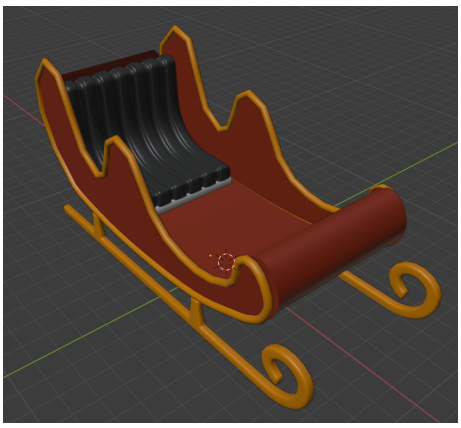
Cloud The clouds above the scene are made of several spheres accumulated together and turned into different shapes by `Scale` and `Rotate`.



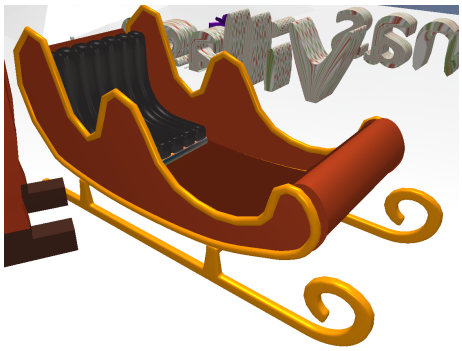
(a) Cloud in Blender (b) Cloud in the Scene

Figure 13: Cloud

Santa's Sleigh Santa's sleigh shown in Figure 14 is formed by three parts, skis, carriage and black cushion. The bottom part of the bent skis is obtained by spinning a face of the rectangle, the red carriage is built by transforming a circular curve and then using `Triangulate Faces`Triangulate Faces, the two sides of the symmetry is also used `Mirror` function.



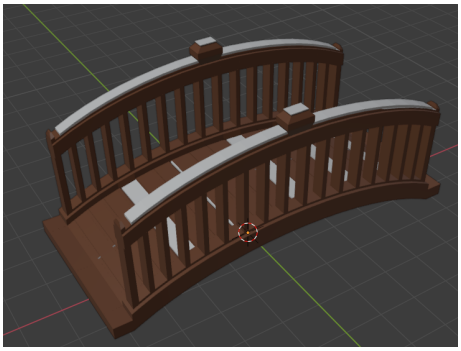
(a) Sleigh in Blender



(b) Sleigh in the Scene

Figure 14: Sleigh

Wooden Bridge As shown in Figure 15, the bottom of the bridge is rectangular scaled into a roof shape, and then using the Bevel function to get the arch bottom of the bridge. Use the `Loop Cut and Slide` function to split the bottom surface, then apply the `Extrude Region and Move` to get the handrail of the bridge.



(a) Bridge in Blender

(b) Bridge in the Scene

Figure 15: Bridge

Castle The castle (Figure 16) consists of four columns, four walls and the house inside, the columns are rectangular, the uppermost face of the rectangle is divided and pulled up to get the effect of a piece of brick. The house within the walls is similar to the previously mentioned house.

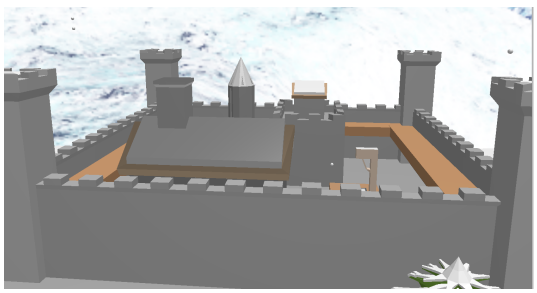
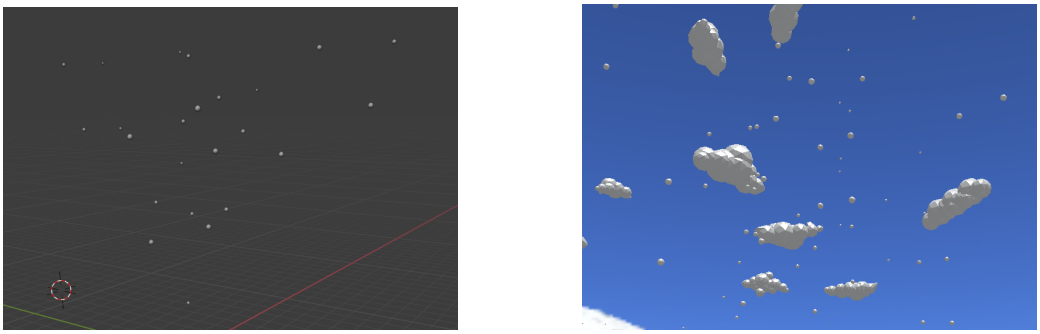


Figure 16: Castle

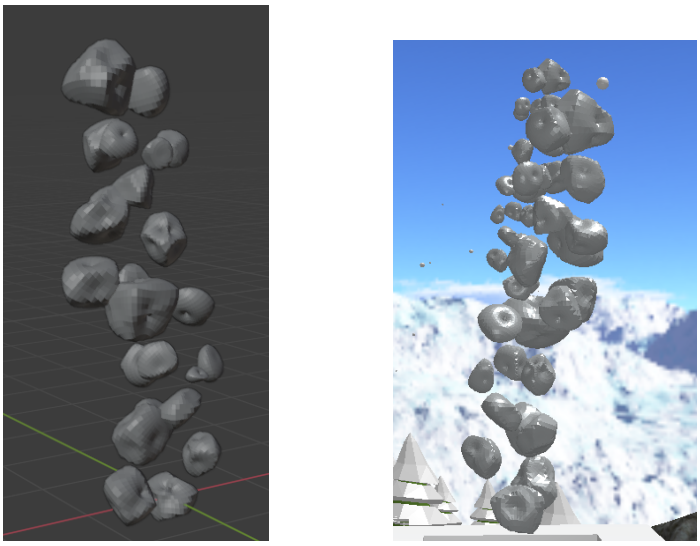
Snow Due to time and technical constraints, the snow in the scene is made up of several spheres, and the snow is spread throughout the scene through the transformation operation and animation of the `C++` code.



(a) Snow in Blender (b) Snow in the Scene

Figure 17: Snow

Smoke Also due to time and technical constraints, the smoke in the scene is composed of several smooth rectangles.



(a) Smoke in Blender (b) Smoke in the Scene

Figure 18: Smoke

Transformation

All objects are initially at the origin and are moved to the corresponding position by the transformation operation (scaling, translating, rotating). Sample programming code for transformation is shown below:

```
    // move the gift object to the target position
    model = glm::translate(model, glm::vec3(2.0, -0.2, 40.0));
    // rotate the gift object
    model = glm::rotate(model, glm::radians(-15.0f), glm::vec3(0.0, 1.0, 0.0));
    // change the size of the gift object
    model = glm::scale(model, glm::vec3(0.5, 0.5, 0.5));

```

Viewpoint

Viewpoint switching can be performed in several ways:

1. Custom viewpoint changing with mouse and keyboard operations; use the mouse to change the camera direction and use keyboard WASD or arrow keys to change the position of the camera. The function `processInput()` is intended to handle the user's keyboard operations and the mouse operation is processed in function `mouse_callback()`. Sample programming code for viewpoint changing is shown below:

```
    // if press key W or UP, the camera will move forward
    if ((glfwGetKey(window, GLFW_KEY_W) == GLFW_PRESS) || (glfwGetKey(window, GLFW_KEY_UP) == GLFW_PRESS)) {
        camera.ProcessKeyboard(FORWARD, deltaTime * cameraSpeed);
    }
    // if press key S or DOWN, the camera will move backward
    if ((glfwGetKey(window, GLFW_KEY_S) == GLFW_PRESS) || (glfwGetKey(window, GLFW_KEY_DOWN) == GLFW_PRESS)) {
        camera.ProcessKeyboard(BACKWARD, deltaTime * cameraSpeed);
    }
    // if press key A or LEFT, the camera will move left
    if ((glfwGetKey(window, GLFW_KEY_A) == GLFW_PRESS) || (glfwGetKey(window, GLFW_KEY_LEFT) == GLFW_PRESS)) {
        camera.ProcessKeyboard(LEFT, deltaTime * cameraSpeed);
    }
    // if press key D or RIGHT, the camera will move right
    if ((glfwGetKey(window, GLFW_KEY_D) == GLFW_PRESS) || (glfwGetKey(window, GLFW_KEY_RIGHT) == GLFW_PRESS)) {
        camera.ProcessKeyboard(RIGHT, deltaTime * cameraSpeed);
    }

```

2. The viewpoint changing speed could be adjusted by pressing key Q. And the speed could be restored to the initial speed by pressing key R. Implementation code is shown below:

```
    // if press key Q, the camera move speed will accelerate
    if ((glfwGetKey(window, GLFW_KEY_Q) == GLFW_PRESS)) {
        cameraSpeed = 4.0;
```

```
    }
    // if press key R, the camera move speed will return to original speed
    if ((glfwGetKey(window, GLFW_KEY_R) == GLFW_PRESS)) {
        cameraSpeed = 1.0;
    }

```

3. The viewpoint could be changed back to the initial point by pressing key 0. Implementation code is shown below:

```
    // set camera position
    glm::vec3 initCameraPos = glm::vec3(0.0f, 8.0f, 65.0f);
    Camera camera(initCameraPos);
    // if press key number 0, the camera will be set to the original position
    if (glfwGetKey(window, GLFW_KEY_0) == GLFW_PRESS) {
        camera = initCameraPos;
    }

```

4. The viewpoint can be moved automatically by pressing key 1 (flying camera function). The whole movement will last sixteen seconds, the first eight seconds the camera will move forward, then in the next four seconds the camera will move in a circular motion to the right, and then the last four seconds the camera will move in a circular motion to the left. The functionality is achieved with the following code:

```
        autoModeTimer += deltaTime;
        if (autoModeTimer >= autoModeDuration) {
            camera = initCameraPos;
            autoMode = false;
            autoModeTimer = 0.0;
        }
        else {
            const float radius = 25.0f;
            float timeInverval = fmod(autoModeTimer, autoModeDuration);
            float camX, camY, camZ;
            if (timeInverval <= 8) {
                camX = -6.0 * timeInverval / 8.0;
                camZ = -40.0 * timeInverval / 8.0;
                camY = -3.0 * timeInverval / 8.0;
                camera = initCameraPos + glm::vec3(camX, camY, camZ);
            }
            else if (timeInverval <= 12) {
                camX = sin(PI * 2 * (timeInverval- 8.0) / 15.0) * 20;
                camZ = cos(PI * 2 * (timeInverval - 8.0) / 15.0) * 20;
                camera = glm::vec3(camX, 5.0, camZ);
            }
            else{
                camX = sin(-PI * 2 * (timeInverval - 12.0) / 40.0) * 50;
                camZ = cos(-PI * 2 * (timeInverval - 12.0) / 40.0) * 50;
```

```
                camera = glm::vec3(camX, 7.0, camZ);
            }
        }

```

Animation

There are five animations in the scene:

1. Animation of *snowing* using translation operation: the snow keeps falling from the sky naturally. To make the snow moving more realistic, I applied two snow animation systems, one with a duration of fifteen seconds, another with a duration of thirty seconds. Therefore, the snow could fill the scene.

2. Animation of the *snowman* using translation and rotation operations: the snowman moves continuously in a cycle on the lake with a random radius. The duration of the snowman move in one circle is fifteen seconds.

3. Animation of the *Santa Claus and his sleigh* using translation and rotation operations: the animation includes five keyframes, the first is from standing Santa Claus to bending Santa Claus which denotes the process of the Santa Claus jump into his sleigh, the second would be the bending Santa Claus changes to sitting Santa Claus in the sleigh, and then he would move with the sleigh up in a circle, in the fourth keyframe, he would throw a gift which would then fall under the Christmas tree, and finally, the Santa Claus would go back to the origin with his sleigh.

4. Animation of the *smoke* from the chimney using translation and scale operations: the smoke would float from two houses' chimney. For realism, I also applied two smoke animation systems, one with a duration of twelve seconds, another with a duration of eight seconds. The smoke would gradually become smaller while floating upward.

5. Animation of the *cloud* using rotating operations: the cloud would constantly move around the origin point and rotate in a circle for sixty seconds.

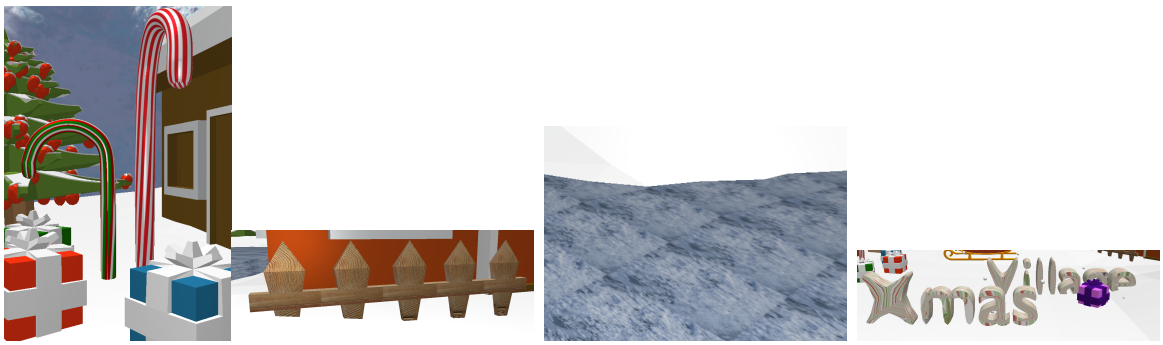
   Sample Code for animation is shown below:

```
// timer counts the total time of the scene displaying
float timeInterval=fmod(timer,smokeAnimationDuration);
smokeAnimationTranslate=5.0*timeInterval/smokeAnimationDuration;
smokeScale=0.6-0.5*timeInterval/smokeAnimationDuration;

```

Texturing

As shown in Figure 19, many objects have applied the texture for realism. The two candy canes with different striped textures, the fence with wooden texture, the lake with ice texture, the skybox with mountain and sky textures, the Xmas Village Sign with Christmas pattern texture.



(a) Candy Canes (b) Fence (c) Iced Lake (d) Xmas Village Sign

Figure 19: Texturing

Some of the objects in the scene are applied to the textures via texturing algorithms, namely, the fragment shader and vertex shader. The code from [2] of the fragment shader from skybox is shown below. The `TexCoords` is the UV coordinates of the texture which is obtained from the vertex shader, and the texture is accessed by calling function `texture()`.

```
    #version 330 core
    out vec4 FragColor;
    in vec3 TexCoords;
    uniform samplerCube skybox;
    void main()
    {
        FragColor = texture(skybox, TexCoords);
    }

```

Other objects like the iced lake is textured with the following code:

```
    // load the texture for the lake
    unsigned int whiteSnowTexture = loadTexture("Resources/Textures/lake
    .png");
    glActiveTexture(GL_TEXTURE0);
    glBindTexture(GL_TEXTURE_2D, whiteSnowTexture);
    renderBase();

```

Lighting

The *Pong lighting model* is applied to this 3D scene, which combined three lightings (ambient, diffuse and specular) with different parameters. To enrich the environment, three types of light including directional light, point light and spotlight in nature are stimulated and applied to the scene [3]. The parameters for each type of light are tuned to best render the environment. Also, other sets of parameters are applied with a different point light color green. By pressing key number 2, the lighting mode will be changed to dark/night mode as shown in Figure 20, I placed the point light at the Xmas village sign, so there would be a luminous light effect.

Figure 20: Dark Mode

Different parameters for the lighting is set with programming code below:

```
// set night light effect to the object shader with point light position and color accordingly
// ------------------------------------------------------------------
void LightEffectNight(Shader shader, glm::vec3 pointLightPositions[], glm::vec3 pointLightColors[]) {
    shader.use();
    // directional light
    shader.setVec3("dirLight.direction", -0.2f, -1.0f, -0.3f);
    shader.setVec3("dirLight.ambient", 0.3f, 0.3f, 0.3f);
    shader.setVec3("dirLight.diffuse", 0.4f, 0.4f, 0.4f);
    shader.setVec3("dirLight.specular", 0.7f, 0.7f, 0.7f);

    // point light
    shader.setVec3("pointLights[0].position", pointLightPositions[1]);
    shader.setVec3("pointLights[0].ambient", pointLightColors[0]*0.05f);
    shader.setVec3("pointLights[0].diffuse", pointLightColors[0]*1.0f);
    shader.setVec3("pointLights[0].specular", pointLightColors[0]*1.0f);
    shader.setFloat("pointLights[0].constant", 1.0f);
    shader.setFloat("pointLights[0].linear", 0.09);
    shader.setFloat("pointLights[0].quadratic", 0.032);

    // spotLight
    shader.setVec3("spotLight.position", camera.Position);
    shader.setVec3("spotLight.direction", camera.Front);
    shader.setVec3("spotLight.ambient", 0.0f, 0.0f, 0.0f);
    shader.setVec3("spotLight.diffuse", 1.0f, 1.0f, 1.0f);
    shader.setVec3("spotLight.specular", 1.0f, 1.0f, 1.0f);
    shader.setFloat("spotLight.constant", 1.0f);
    shader.setFloat("spotLight.linear", 0.09);
    shader.setFloat("spotLight.quadratic", 0.032);
}

```

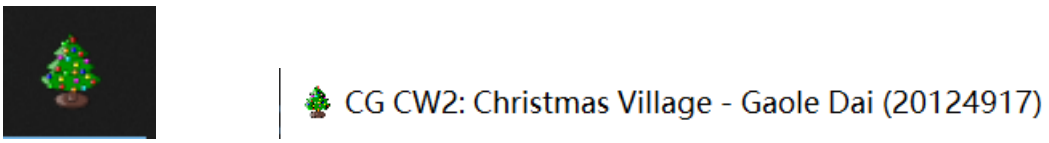
Creative Ideas

I want to highlight some creative ideas and additional features in this 3D scene.

1. Harmony environment with the background Christmas music and footsteps sound effect.
2. Impressive animation: Santa Claus and his sleigh animation.
3. Different keyboard operations as shown in table below for better interaction and more user friendly.

Keyboard shortcut	Function
W/Arrow Up	Camera move forward
A/Arrow Left	Camera move leftward
S/Arrow Down	Camera move backward
D/Arrow Right	Camera move rightward
0	Camera back to initial position
1	Camera auto move mode
2	Lighting: dark/night mode
3	Lighting: back to day mode
Q	Camera moving speed up
P	Replay the Santa Claus animation
R	Restore camera moving speed

4. Details: I have changed the windows icon for better aesthetics appearance.



(a) Icon (b) Windows Icon

Figure 21: Windows Icon

Conclusion

Reflection

Overall, I have achieved all the designs mentioned in the project proposal, and added more features during the development. Through the development, I have grasped many computer graphics knowledge, and put the theoretical knowledge into real-life implementation. I have downloaded some CC licence images from the internet, which I want to contribute to here.

1. Rock texture from `https://ambientcg.com/view?id=Rock020`;

2. Candy cane's texture from https://cutewallpaper.org/download.php?file=/22/cute-candy-canes-wallpapers/1235814362.jpg and https://www.vecteezy.com/vector-art/3423537-seamles-striped-pattern-christmas-candy-cane-texture;
3. Xmas village sign texture from https://www.wallpaperup.com/1019692/Textures_Christmas_texture.html;
4. Iced lake texture from https://pxhere.com/en/photo/1070208;
5. Christmas music from https://pixabay.com/music/search/genre/christmas/;
6. Footsteps sound effect from https://freesound.org/search/?q=footsteps+snow&f=&s=num_downloads+desc&advanced=0&g=1;
7. Skybox from https://opengameart.org/sites/default/files/skybox-pack_0.zip

Lastly, I want to thank all the lecture and lab materials and a website learnopengl.com where I obtained some source code and learned many applications.

Future Work

Some improvements of the scene could be developed in the future, including the shadow effect, more decorations on the outdoor houses, and adding indoor.

References

[1] Ivanix88. Christmas ball - download free 3d model by ivanix88 (@ivanix88) [3686898], 2019. [Online; accessed November 14, 2021].

[2] JoeyDeVries. Learnopengl/6.2.skybox.fs at master · joeydevries/learnopengl. [Online; accessed November 28, 2021].

[3] Light casters, https://learnopengl.com/lighting/light-casters. [Online; accessed December 3, 2021].