CPT205 3D Modelling Project

Information and Computing Science
Rui Sang 2251576

1 Introduction

1.1 Scene Overview

This project recreates a serene park scene beside Dushu Lake using OpenGL and FreeGLUT. The scene features a church, lakeside paths, the lake, surrounding forest, and a small platform, offering users a virtual experience similar to visiting the real location. Various graphics techniques were used to achieve dynamic and static effects, enhancing the visual appeal.

Figure 1: Overview

Figure 1 shows the reference and 3D scene setup. This report covers the code framework, project design, graphics techniques like hierarchical modeling, lighting, and texture mapping, and includes an operation guide for user interaction.

1.2 Code Framework and Structure

The diagram below illustrates the basic framework structure of the code for my entire project.

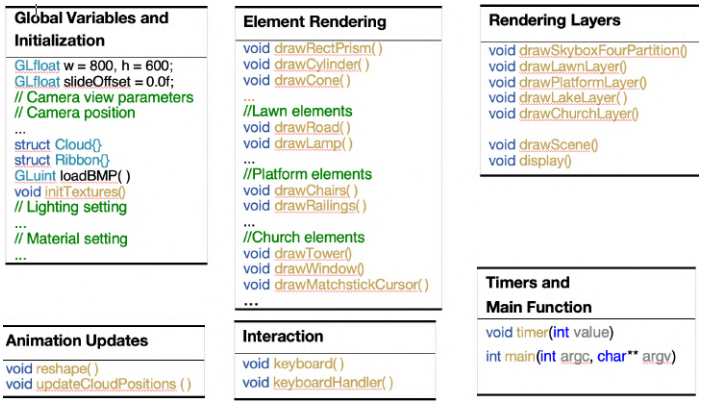


Figure 2: Realization of water ripple

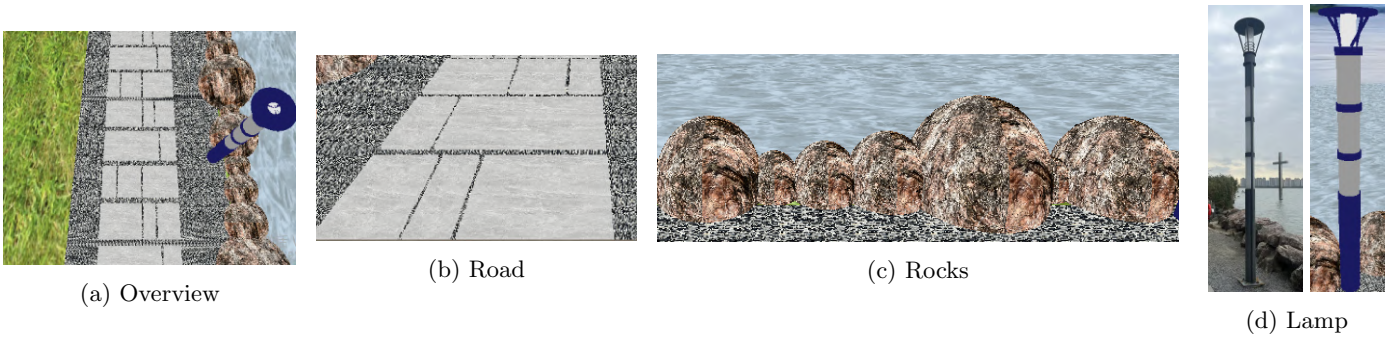
2 Design and Implementation

2.1 Scene Elements

First, I constructed the scene in sections: lawn elements, lake elements, church elements, and platform elements. Finally, I used functions like `void drawSkyboxFourPartition()`, `void drawLawnLayer()`, `void drawPlatformLayer()`, `void drawLakeLayer()`, **and** `void drawChurchLayer()` to bring together all parts for easier overall transformation and adjustments.

2.1.1 Static Elements

(1) Lawn Elements First, the lawn area is defined by drawing a rectangle and applying a suitable grass texture.



(a) Overview (b) Road (c) Rocks (d) Lamp

Figure 3: Elements on the lawn

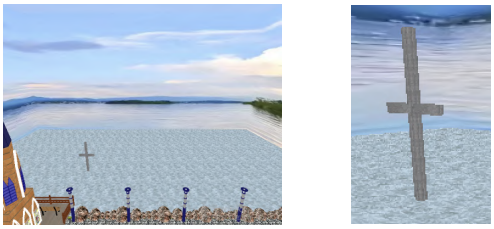
Stone Path: The stone path is created by first drawing a basic rectangular outline and then using the function `drawRoad()` to lay down a pathway made up of cuboids of varying sizes. This is achieved by defining a basic element and repeatedly calling it within a loop. Two realistic textures are mapped onto the path to enhance its detail.

Lakeside Rocks: The rocks are created by calling the built-in function `glutWireIcosahedron()` to simulate the irregular and wrinkled surfaces. Six rocks of different sizes form a group, which is then replicated within the lawn area to form the lakeside stones. Textures are mapped to simulate the details of the rocks.

Street Lamp: The street lamp is modeled as a replica of a real lamp. Multiple calls to the function `drawCylinder()` are used to create the cylindrical components at different heights, and the angles are adjusted to form the lamp's frame.

Tree: The tree is constructed with three different shades of green to form the leaves using `drawCuboid()` to create depth. The trunk is made with `drawCone()`. The full tree is created by combining these parts, and multiple calls are made to form a group of trees.

(2) Lake Elements The lake surface is defined by drawing a basic rectangle, onto which a water ripple texture is applied to simulate realism. The surface color is also adjusted to enhance the effect.

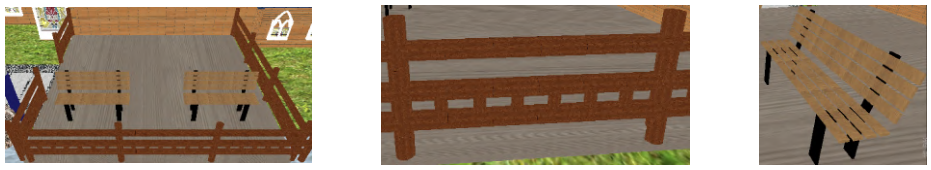


(a) Overview (b) cross

Figure 4: Elements on the lake

Cross: A cross is formed by calling the `drawRectPrism()` function twice, setting different dimensions for height, width, and depth, and intersecting the two prisms to create a 3D cross.

(3) Platform Elements The platform is created using the function `drawCuboid()` to draw a flattened 3D platform, and a texture is applied.



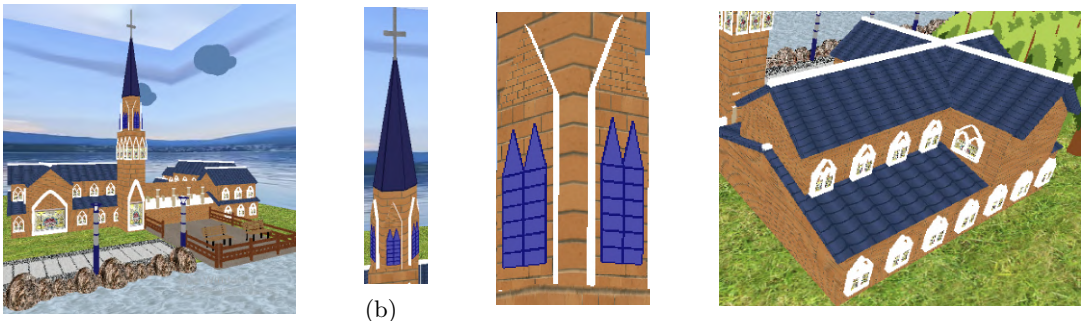
(a) Overview (b) Handrail (c) Chair

Figure 5: Elements on the platform

Railing: The railing is formed by repeatedly calling `drawCylinder()` to create a basic cylindrical element, which is then positioned around the platform's edges, leaving an open entrance for access.

Chair: The bench is constructed by assembling multiple cuboids, with the backrest tilted at an angle. A black frame is drawn to support the structure, and appropriate wood textures are applied to render the desired look.

(4) Church Elements ***Tower***: The tower consists of a combination of geometric shapes, starting with a quadrilateral prism, followed by a hexagonal prism and hexagonal pyramid to form the base. On each side of the hexagonal prism, a protruding structure with a window is added by adjusting the angles.



(a) Overview (b) Tower (c) Structure (d) Buildings

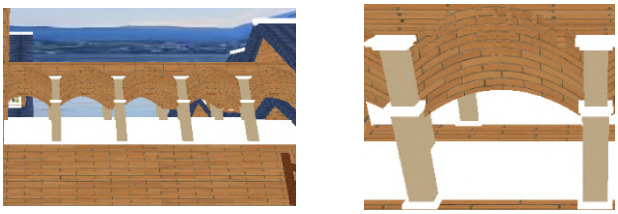
Figure 6: Elements of the church

Building: The building features a classic gabled roof. The base is a rectangular prism, on top of which a triangular prism is placed. Two flat rectangles are added to each side of the triangular prism to form the roof, and a long white rectangular prism acts as a ridge beam where the two roof surfaces meet.

(a) (b) (c)

Figure 7: Three kinds of windows

Windows: A window frame is designed and encapsulated in the function `drawWindowFrame()`. Three types of windows are designed, with variations in how the glass panes are divided, either horizontally or both horizontally and vertically.



(a) Overview (b) Columns and arches

Figure 8: Bridge between the buildings

Bridge: The bridge begins with a Bezier curve to form the arched structure, which is then filled in to form a solid framework. The framework is combined with rectangular prisms for the top and base of the bridge. Additional columns are added to each side by combining multiple prisms and trapezoidal shapes, replicating the real-world structure.

2.1.2 Dynamic Elements

Clouds in the Sky: The clouds are simulated by repeatedly calling the `glutSolidSphere()` function to form clusters that resemble piled-up clouds. The cloud movement is animated using `glutTimerFunc()` to simulate their translation across the sky.

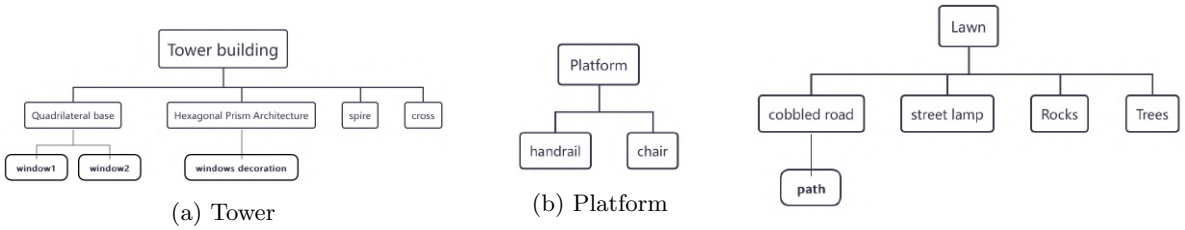
Lake Surface Dynamics: The lake's dynamic effect is achieved by applying a ripple texture and setting it to translate, thus simulating the gentle movement of the lake surface.

3 Graphics Techniques

3.1 Geometry Creation and Hierarchical Modeling

Geometry creation is primarily achieved by drawing basic geometric shapes, including rectangles, cuboids, cylinders, spheres, icosahedrons, and prisms. These basic shapes are used to build complex scene elements by calling custom drawing functions such as `drawCuboid()`, `drawCylinder()`,etc.

Hierarchical modeling is mainly reflected in the combination of each object in the scene, allowing different parts to transform together. Figure 8 shows examples of hierarchical modeling in my project.



(a) Tower (b) Platform (c) Lawn

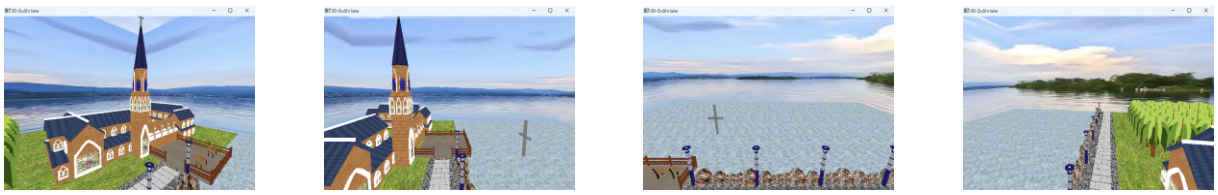
Figure 9: Three examples of hierarchical modeling

3.2 Transformation Techniques

In this project, my transformation techniques are reflected in two aspects: 1. Adjusting the positions of different elements to achieve the desired layout and size proportions. 2. Rendering dynamic elements. Both of these aspects are achieved using functions like `glTranslatef()`, `glRotatef()`, and `glScalef()`.

3.3 Viewing and Projection

Viewing: In this project, a dynamic camera view is used to allow users to freely observe the virtual scene. The camera position is initially defined by `X0`, `Y0`, and `Z0`, and the function `updateCameraPosition()` is called to make the camera rotate around the fixed reference point `XRef`, `YRef`, and `ZRef`. At the same time, the `gluLookAt()` function is used to set the camera's viewing direction, with the up direction set along the Y-axis, ensuring the camera maintains a stable perspective while rotating around the reference point, providing a dynamic and immersive viewing experience. The figure below shows the scene observed as the horizontal view is continuously adjusted.



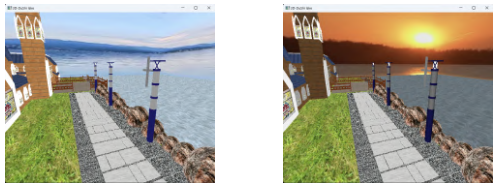
(a) (b) (c) (d)

Figure 10: Different scenes after switching angles

Projection: In the `display()` function, a perspective projection is used. The projection matrix is set using `glMatrixMode(GL_PROJECTION)`, and the `gluPerspective()` function ensures that the size of the objects in the scene changes with the viewing distance, enhancing the sense of depth and realism.

3.4 Lighting and Materials

I design the scene's lighting with `Global ambient light` for foundational illumination and `Parallel light` to simulate natural lighting. `Point lights` are added as streetlights, illuminating the scene at night. Skybox texture switching creates a smooth transition from daytime to dusk, enhancing the evening atmosphere and immersion.



(a) Morning (b) Evening

Figure 11: Lighting effect

3.5 Texture Mapping

To create a realistic scene, most of the textures used in this project are sourced from photographs taken on-site. These textures are then allocated using the `glBindTexture()` and `glTexImage2D()` functions. In my project, I employed two main methods for texture mapping:

a. Vertex-Specified Texture Coordinate Mapping: This method involves using `glTexCoord2f()` to explicitly set texture coordinates for each vertex. It is suitable for simple planes or regular shapes, such as the church walls, roofs, lawn, lake surface, and other large, flat elements.

b. GLU Quadric Texture Mapping: Using GLU functions like `gluQuadricTexture()` along with `gluSphere()`, this method automatically maps textures onto quadric surfaces. For example, the railings are cylinders that benefit from this approach. Similarly, for rocks represented as icosahedrons, due to the difficulty of accurate texture mapping, I created corresponding quadric surfaces to properly apply the textures.

4 Animation and Interaction

Animation: The dynamic elements in the scene include the translation of the clouds in the sky and the rippling effect of the lake surface.

Interaction: The interactions in this project are primarily controlled through keyboard inputs:

(a) Press 'm': The clouds move across the sky by updating their position coordinates, reappearing on the left after exiting on the right, creating a continuous loop. The lake's ripple effect is simulated by modifying texture mapping coordinates.

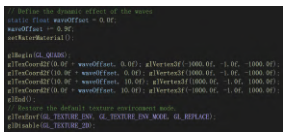


Figure 12: Realization of water ripple

(b) Press 'Up', 'Down', 'Left', 'Right': These keys let the user adjust the camera position, offering a comprehensive view of the scene and enhancing the user experience.

(a) original scene (b) press 'up' (c) press 'down' (d) press 'right' (e) press 'left'

Figure 13: Different scenes after press keyboard

(c) Press '1', '2': These keys control the time of day and the corresponding lighting effects in the scene: '1' for morning, '2' for evening.

5 Conclusion

This project uses OpenGL and FreeGLUT to create a 3D scene rich in natural beauty and dynamic effects. By combining geometry, hierarchical modeling, transformations, viewing and projection, lighting and materials, and texture mapping, the scene presents a diverse virtual park environment of Dushu lake.