1. Introduction

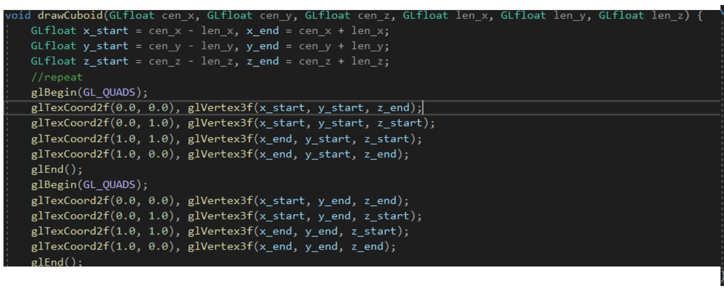
In the Assessment 2 of the CPT205, we are required to create a well-thought 3-dimensional scene. Initially, I wanted to create an outdoor scene. However, it is difficult for me to decide which outdoor scenes should be built. I went to the campus of XJTLU, Bailu Park and so on to take pictures and looked for inspiration. Finally, I decide to create a virtual 3D scene of Canglang pavilion, a famous place of Suzhou, China. There are a pavilion, chairs, guards, lamp, trees, a beach, a sea, roads etc. in this scene, you could visit in a large park and go to the pavilion through road. More details would be demonstrated in the below.

全国重点文物保护单位
沧浪亭
中华人民共和国国务院二〇〇六年六月公布
苏州市人民政府二〇〇六年六月立

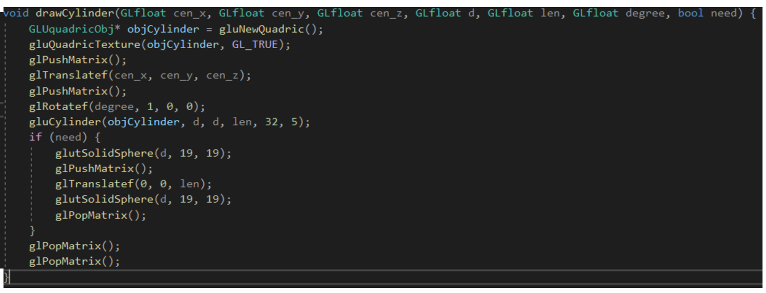
2. Brief instruction section

2.1 Creation of geometry

Here, I list some libs that I use to create the geometry. I use glBegin() and glEnd() to draw both the 2D and 3D scenes. I use GL_POLYGON to create the objects and use glVertex3f() to decide the objects' position. As for draw objects, to avoid trouble. Firstly, I draw basic models like cuboid and cylinder. They will be used for many times. Hence, I packaged them. When need to use them, I call them directly.



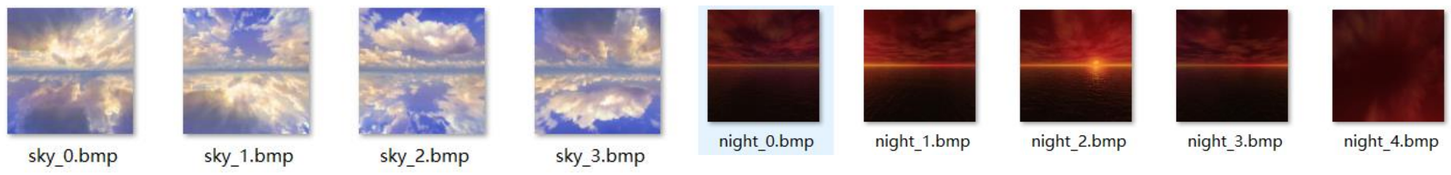
```
void drawCuboid(GLfloat cen_x, GLfloat cen_y, GLfloat cen_z, GLfloat len_x, GLfloat len_y, GLfloat len_z) {
    GLfloat x_start = cen_x - len_x, x_end = cen_x + len_x;
    GLfloat y_start = cen_y - len_y, y_end = cen_y + len_y;
    GLfloat z_start = cen_z - len_z, z_end = cen_z + len_z;
    //repeat
    glBegin(GL_QUADS);
    glTexCoord2f(0.0, 0.0), glVertex3f(x_start, y_start, z_end);
    glTexCoord2f(0.0, 1.0), glVertex3f(x_start, y_start, z_start);
    glTexCoord2f(1.0, 1.0), glVertex3f(x_end, y_start, z_start);
    glTexCoord2f(1.0, 0.0), glVertex3f(x_end, y_start, z_end);
    glEnd();
    glBegin(GL_QUADS);
    glTexCoord2f(0.0, 0.0), glVertex3f(x_start, y_end, z_end);
    glTexCoord2f(0.0, 1.0), glVertex3f(x_start, y_end, z_start);
    glTexCoord2f(1.0, 1.0), glVertex3f(x_end, y_end, z_start);
    glTexCoord2f(1.0, 0.0), glVertex3f(x_end, y_end, z_end);
    glEnd();
```



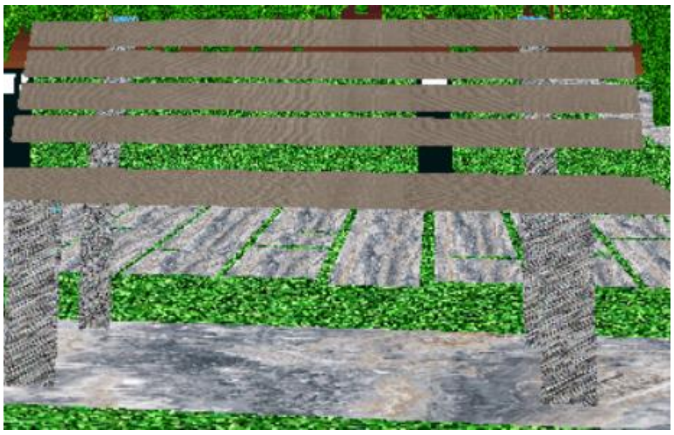
```
void drawCylinder(GLfloat cen_x, GLfloat cen_y, GLfloat cen_z, GLfloat d, GLfloat len, GLfloat degree, bool need) {
    GLUquadricObj* objCylinder = gluNewQuadric();
    gluQuadricTexture(objCylinder, GL_TRUE);
    glPushMatrix();
    glTranslatef(cen_x, cen_y, cen_z);
    glPushMatrix();
    glRotatef(degree, 1, 0, 0);
    gluCylinder(objCylinder, d, d, len, 32, 5);
    if (need) {
        glutSolidSphere(d, 19, 19);
        glPushMatrix();
        glTranslatef(0, 0, len);
        glutSolidSphere(d, 19, 19);
        glPopMatrix();
    }
    glPopMatrix();
    glPopMatrix();
}
```

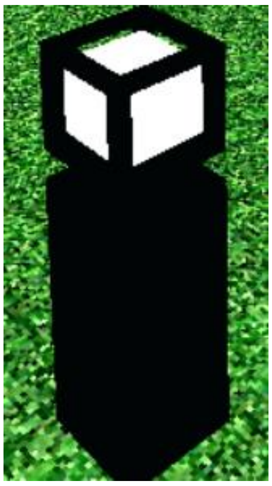
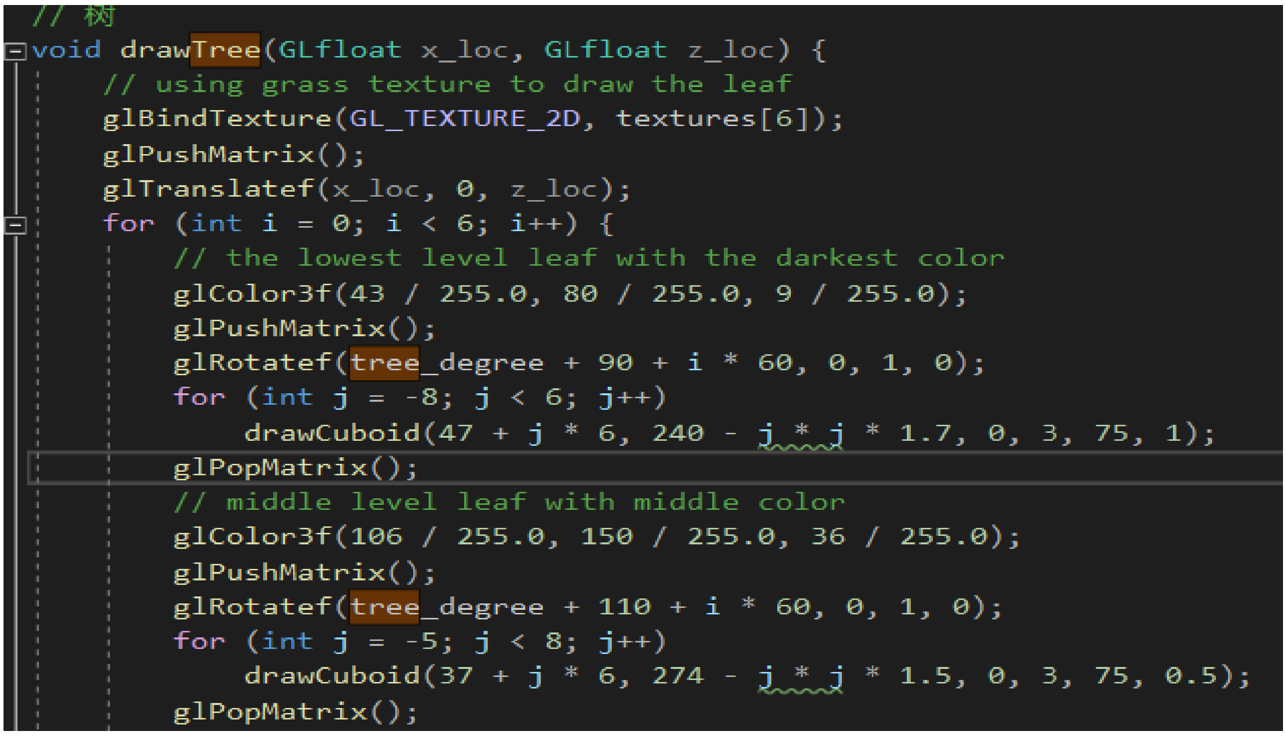
2.2 Hierarchical modelling

As we all know, it is not easy to create a large 3D scene. It requires us to hierarchical modelling.
Firstly, I learn knowledge about skybox. I find relevant materials about day and night skybox. It is a super large cube. Every surface has a texture. It surrounds our 3D scene and it is too far away to arrive in its boundary. In addition, the cloud is consists of different size of balls. They generate randomly.



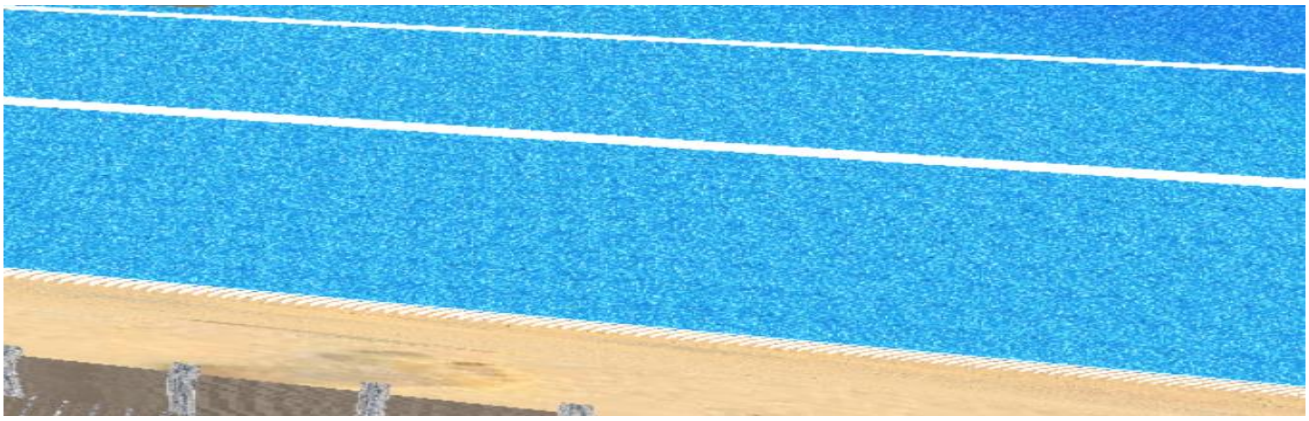
The road has grass (A layer of grass texture on the cuboid foundation), trees(Composed of a cone and many cuboids), street lights(Composed of a luminous white head and a black cuboid body) , chairs(consists of a cuboid foundation, a seat, a back and 4 legs)



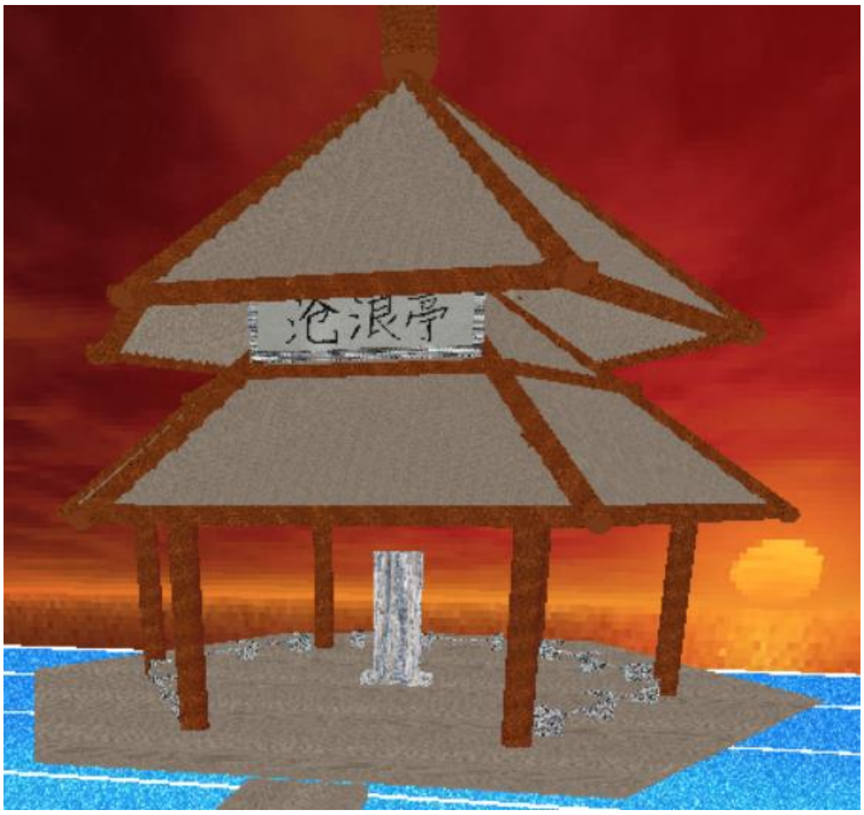



```
// 树
void drawTree(GLfloat x_loc, GLfloat z_loc) {
    // using grass texture to draw the leaf
    glBindTexture(GL_TEXTURE_2D, textures[6]);
    glPushMatrix();
    glTranslatef(x_loc, 0, z_loc);
    for (int i = 0; i < 6; i++) {
        // the lowest level leaf with the darkest color
        glColor3f(43 / 255.0, 80 / 255.0, 9 / 255.0);
        glPushMatrix();
        glRotatef(tree_degree + 90 + i * 60, 0, 1, 0);
        for (int j = -8; j < 6; j++)
            drawCuboid(47 + j * 6, 240 - j * j * 1.7, 0, 3, 75, 1);
        glPopMatrix();
        // middle level leaf with middle color
        glColor3f(106 / 255.0, 150 / 255.0, 36 / 255.0);
        glPushMatrix();
        glRotatef(tree_degree + 110 + i * 60, 0, 1, 0);
        for (int j = -5; j < 8; j++)
            drawCuboid(37 + j * 6, 274 - j * j * 1.5, 0, 3, 75, 0.5);
        glPopMatrix();
```

I also use a sea texture and sea_wave to create a sea and beach.



In the opposite bank, I create the Canglang pavilion.



2.3 Transformations

The movement can be divided into two types: transformation and rotation, which can be realized by function glTranslatef() and glRotatef().

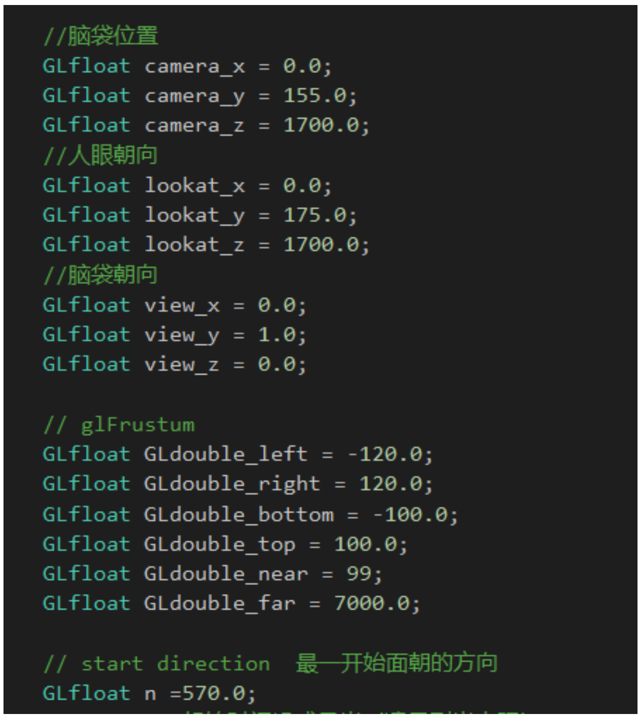
The main purpose of transformation is to move the object to where it needs to be, without having to think about this displacement when building every part of the object. This is ideal for use on objects that require multiple reuses. In this project, the transformation for each object is placed inside the function of the object. When the external functions need to call it, they only need to pass the coordinates of the centre of an object.

Rotation is used to position an object at an inclined angle without having to spend time on calculating the position of the point coordinates, like the recline of a chair back. Besides, rotating the object around the distance to create multiple times can also give the object a more substantial visual effect. For example, the branches of trees that have mentioned earlier are obtained by rotation.

In addition, transformation and rotation are also used in animation. By constantly changing the distance of offset and the angle of rotation, the effect of animation can be created. Like the waving of spindrifts and the swaying of trees.

2.4 viewing and projection

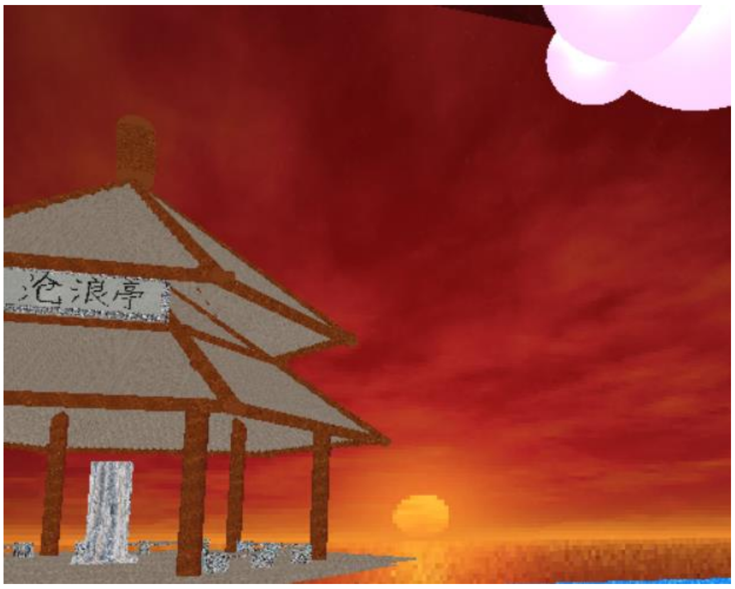
In this project, function glMatrixMode(GL_PROJECTION) is used to realize projection, and glFrustum() is used to create a vision crystal of perspective shape. Function gluLookAt() defined the position of the camera and the look at point.

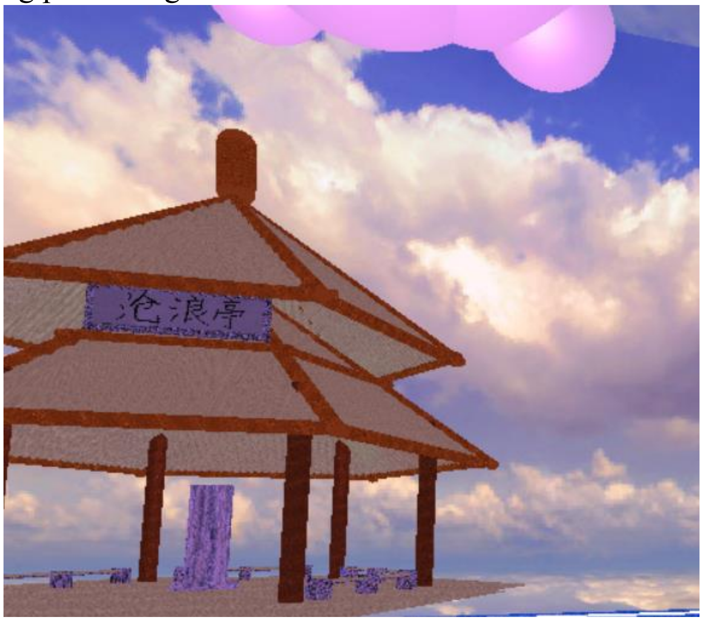


```
//脑袋位置
GLfloat camera_x = 0.0;
GLfloat camera_y = 155.0;
GLfloat camera_z = 1700.0;
//人眼朝向
GLfloat lookat_x = 0.0;
GLfloat lookat_y = 175.0;
GLfloat lookat_z = 1700.0;
//脑袋朝向
GLfloat view_x = 0.0;
GLfloat view_y = 1.0;
GLfloat view_z = 0.0;

// glFrustum
GLfloat GLdouble_left = -120.0;
GLfloat GLdouble_right = 120.0;
GLfloat GLdouble_bottom = -100.0;
GLfloat GLdouble_top = 100.0;
GLfloat GLdouble_near = 99;
GLfloat GLdouble_far = 7000.0;

// start direction  最一开始面朝的方向
GLfloat n =570.0;
```

2.5 lighting and materials

There are two lighting types here.
Point light (The program creates a point light at the position of the sun to represent the sun, whose light decaying at a certain coefficient in all directions. The point light has different colours at different time states, which is used to create the overall colour change and make the scene colour more consistent with the sky map).
Parallel light (the sun is present only in the day, there is a difference in the overall intensity of the light between day and night. This distinction is made by switching parallel lights on and off.





2.6 Texture are used in many places.

2.7 interactions
Click WASD, you can rotate the view sight.

Click ↑ ↓ ← →, you can move step.

Click mouse wheel, you can fly or down.