CG project report

Program explanation

According to the Interim report, there is a piece of land displayed in the project, objects described in previous report has now been constructed already, model parts in the following will be discussed in detail. However, there are some changes in the second work stage comparing to the work plan in interim report, due to the time limitations, I delete some works expected to be done. The scene embodies the peace and fine of the countryside in Autumn, and also express my love and yearning to the autumn night of the countryside.

Requirements obtained in Program

In the project requirements sheet, there are some requirements for the project, which are Models, Translation and Scaling, Texturing, Animations and Viewpoint, lighting effect, this part will discuss each requirement respectively, to show how the project meet them.

Models

Several objects are loaded in to the scene, waterwheel [1], lamps [2],(showed in **Figure.1-1**), architectures (temples [3], desks[4], **Figure.1-2**), plants(grass, mushrooms, trees, falls,[5] **Figure.1-3**), in these objects, architectures and plants are combined with the mainland, which includes a river and some rocks, bridges and stairs, reservoir, in the sky, there are some stars [6] and a blue moon (showed in **Figure.1-4**). Almost objects in the project are constructed by blender, some of them are modified based on internet sources.

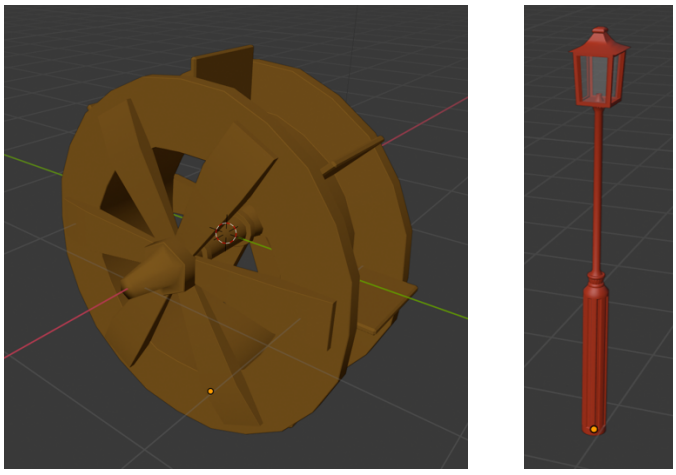


Figure.1-1

Figure.1-2

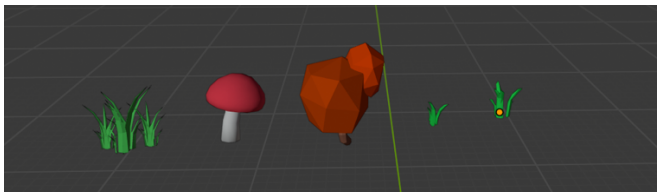


Figure.1-3

Figure.1-4

Having learnt in the lab, I wrote load_obj function to load object into my project based on ***tinyobjloader.h*,** **Figure.1-5** shows the function details.

```
void load_obj(tinyobj::attrib_t attrib, vector<tinyobj::shape_t> shapes,vector<tinyobj::material_t> materials, bool tex,bool emit)
{
    for (int j = 0; j < shapes.size(); j++) {
        int a = 0;
        for (int p = 0; p < shapes[j].mesh.indices.size(); p += 3) {
            int nind1 = shapes[j].mesh.indices[p].normal_index;
            int nind2 = shapes[j].mesh.indices[p + 1].normal_index;
            int nind3 = shapes[j].mesh.indices[p + 2].normal_index;
            if (emit) {
                float s[] = {1, 1, 1, 1.0};
                glMaterialfv(GL_FRONT_AND_BACK, GL_EMISSION, s);
            } else {
                glMaterialfv(GL_FRONT_AND_BACK, GL_DIFFUSE, materials[shapes[j].mesh.material_ids[a]].diffuse);
            }
            a++;
            int ind1 = shapes[j].mesh.indices[p].vertex_index;
            int ind2 = shapes[j].mesh.indices[p + 1].vertex_index;
            int ind3 = shapes[j].mesh.indices[p + 2].vertex_index;
            glBegin(GL_TRIANGLES);
            if (tex)
                glTexCoord2d(attrib.texcoords[shapes[j].mesh.indices[p].texcoord_index],
                             t: attrib.texcoords[shapes[j].mesh.indices[p].texcoord_index + 1]);
            glNormal3f( nx: attrib.normals[3 * nind1 + 0],  ny: attrib.normals[3 * nind1 + 1],  nz: attrib.normals[3 * nind1 + 2]);
            glVertex3f( x: attrib.vertices[3 * ind1 + 0],  y: attrib.vertices[3 * ind1 + 1],  z: attrib.vertices[3 * ind1 + 2]);
            if (tex)
                glTexCoord2d( s: attrib.texcoords[shapes[j].mesh.indices[p + 1].texcoord_index],
                             t: attrib.texcoords[shapes[j].mesh.indices[p + 1].texcoord_index + 1]);
            glNormal3f( nx: attrib.normals[3 * nind2 + 0],  ny: attrib.normals[3 * nind2 + 1],  nz: attrib.normals[3 * nind2 + 2]);
            glVertex3f( x: attrib.vertices[3 * ind2 + 0],  y: attrib.vertices[3 * ind2 + 1],  z: attrib.vertices[3 * ind2 + 2]);
            if (tex)
                glTexCoord2d( s: attrib.texcoords[shapes[j].mesh.indices[p + 2].texcoord_index],
                             t: attrib.texcoords[shapes[j].mesh.indices[p + 2].texcoord_index + 1]);
            glNormal3f( nx: attrib.normals[3 * nind3 + 0],  ny: attrib.normals[3 * nind3 + 1],  nz: attrib.normals[3 * nind3 + 2]);
            glVertex3f( x: attrib.vertices[3 * ind3 + 0],  y: attrib.vertices[3 * ind3 + 1],  z: attrib.vertices[3 * ind3 + 2]);
            glEnd();
        }
    }
}
```

Figure.1-5

The following **Figure.1-6** shows the overall scene in this project, the whole scene has red as main color style, and the time in the scene world is night, so the surroundings is black, with four lamps lighting the scene.

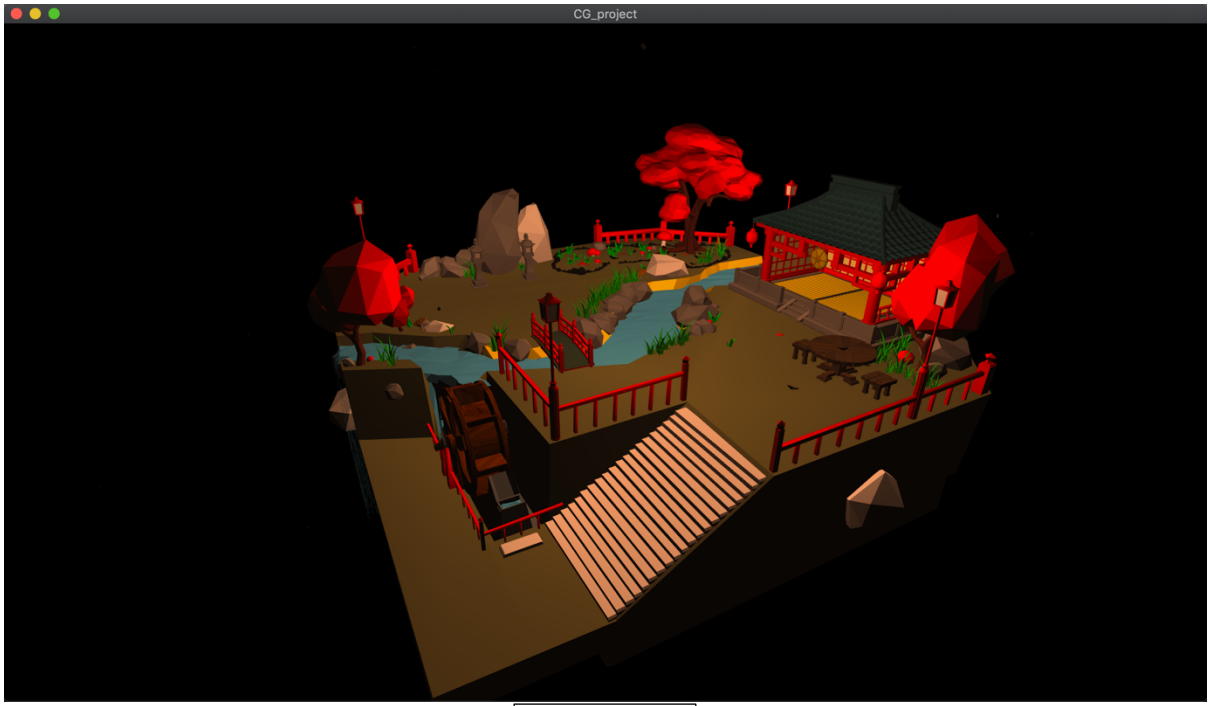


Figure.1-6

Transformation

When placing those objects, I did translate these objects to proper position. In the code files, I borrowed the framework from lab6 of Computer Graphics, there is a black scene with a colorful cube and grids, I firstly delete them from the scene, and load waterwheel first, then, I translate it to the origin point, let it rotate by Y-axis, in a constant speed, then load the mainland object, adjust its position by ***glTranslatef()***, so that the waterwheel will display at the right position in the mainland.

Texture

This part is used to set some unique textures on some objects or some parts of the objects, such as green leaves of trees, brown wood-like texture of the tree bark, waterwheel as well. Stone texture on roads, bridges, etc. in the final scene, I set textures to the waterwheel and the moons, for other objects, they have only materials without texture. Since this scene aims to display the autumn night scene, materials colors with light effectiveness can better display the environment of the night. Finally, in the project, the following objects are textured, and showed in **Figure.2-1**.

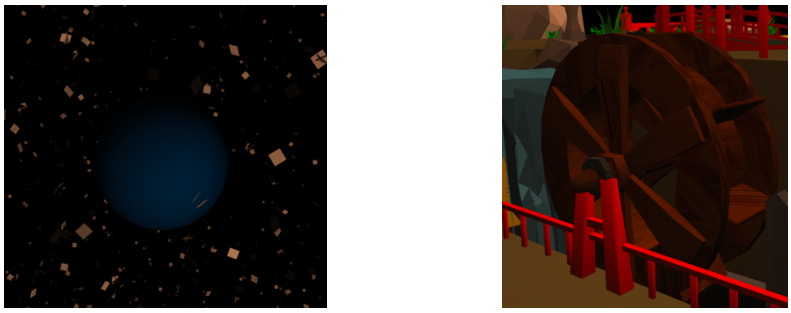


Figure.2-1

According to interim report and teaching materials, I wrote some functions to load image and create relative textures for some objects, I mainly use ***std_image.h*** file to load texture sources, codes in **Figure.2-2** states the function loading image as source of specific textures and those in **Figure.2-3** states function setting textures to faces of objects.

```
void loadTex(char *imgName){
// set up properties of the texture
    glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_WRAP_S, GL_REPEAT);
    glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_WRAP_T, GL_REPEAT);
    glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MIN_FILTER, GL_LINEAR);
    glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MAG_FILTER, GL_LINEAR);
// generate texture;
    int width, height, channel;
    unsigned char *data = stbi_load(imgName, &width, &height, &channel, 0);
    if(data != NULL){
        glTexImage2D(GL_TEXTURE_2D, 0, GL_RGB, width, height, 0, GL_RGB, GL_UNSIGNED_BYTE, data);
    }
    else{
        printf(" failed to load texture\n");
    }
    stbi_image_free(data);
}
```

Figure.2-2

```
glEnable(GL_TEXTURE_2D);
glEnable(GL_COLOR_MATERIAL);
unsigned int texture[2];
//unsigned int texture;
glGenTextures(2, texture);
glBindTexture(GL_TEXTURE_2D, texture[0]);
loadTex("qiu.jpg");
//record the initial position of the scene
glBegin(GL_QUADS);
glTexCoord2f(0,0);
glVertex3f(0.46,0,0.46);
glTexCoord2f(0,1);
glVertex3f(0.46,0,-0.46);
glTexCoord2f(1,1);
glVertex3f(-0.46,0,-0.46);
glTexCoord2f(1,0);
glVertex3f(-0.46,0,0.46);
glEnd();
```

Figure.2-3

Animation

Animation makes some objects to be dynamic, and there are some objects are expected to keep moving in specific styles. Including rotating waterwheel, running water, falling leaves, waterfalling, etc. Usually this part of works is combined closely with particle system. Particle system is mainly used to create water splashing effects in the project. In the project, water runs in the river, when goes through the waterwheel, water will splash, in order to implement those vivid scenes, particle system is considered to be used. I've found a thesis [4] explains comprehensively about how to implement waterfall scene based on particle system in OpenGL, in second work stage, I managed to implement waterfall effect in the project, while due to time limits, I give up implementing the leaves falling effect. **Figure.3-1** displays the waterfall results that I realized in the project, about water movement. Other animation examples such as the waterwheel rotation that I have implemented in my project displayed in **Figure.3-2**.

Figure.3-2

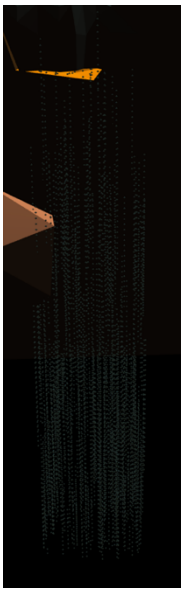


Figure.3-1

Viewpoint

Viewpoints in the project is changeable so that it is able to see everything in the project by change the viewpoint. Press WASD to move the viewpoint forward, left, backwards and right, E for upwards and Q for downwards, besides, the perspective will follow the movement of the mouse, then it is quite easy to see the whole scene by this operation. However, in the final version of the project, I did not implement the operation that dragging mouse to rotate the perspective but move it directly by moving the mouse instead.

Lighting

Lighting system adds light effects in the scene, in this project, as stated in the interim report, there will be an automatic moving sun(an illuminant sphere) emit light to the mainland , to simulate daytime of the land, the sphere will start at the same horizontal level as the right side of the land, and stop at the left side of the land at the same horizontal level as well, and it will cost a few minutes for it to move from right to the left. Moreover, it means night is coming when the sphere stops at the left side of the land, then streetlamp in the land will give off, so that it is still clear to see the whole scene. If there still exists some time, light effects from the fire worm will be take into consideration.

However, in the final version of the project, there is a moon(lighting effectiveness shows in the following, **Figure.4-3**) displayed in my project rather than sun, and the whole scene is night, with some stars in the sky, the scene is unable to change to daytime either. Since time limitations, lighting effectiveness stated above did not implement already yet, but more lighting effectiveness is in lamp luminous in the scene. There are four lamps in the scene, press button 1,2,3,4 respectively in the keyboard can turn on and off specific lamps, which the effectiveness shows in the following. (**Figure.4-1** shows the scene when all the lamps are on, **Figure.4-2** shows the scene when some lamps are off)

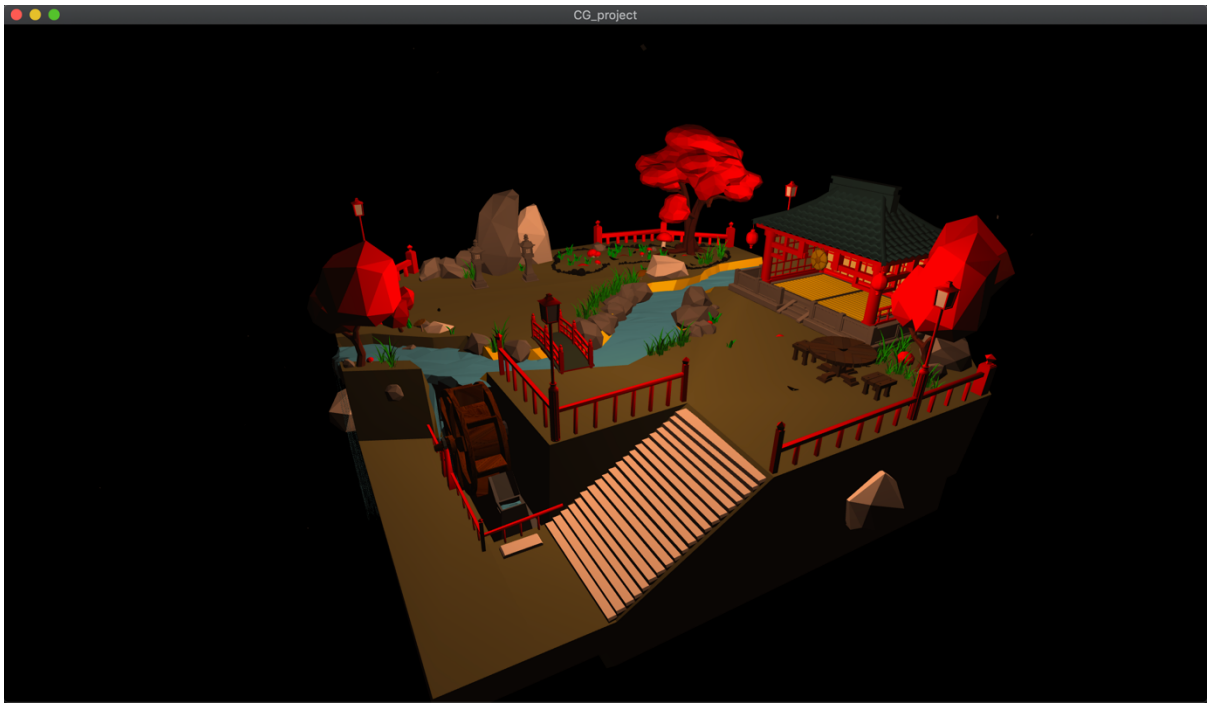
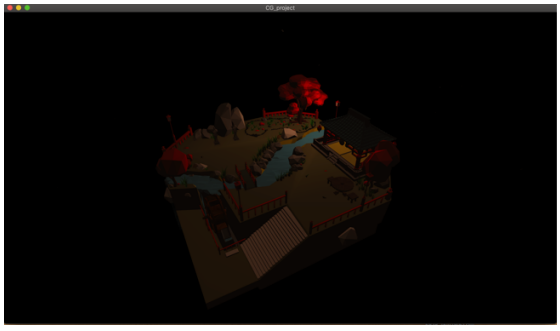
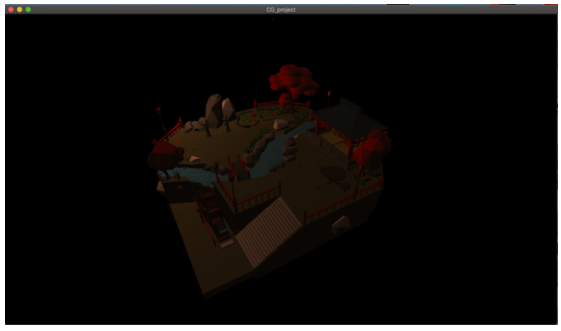
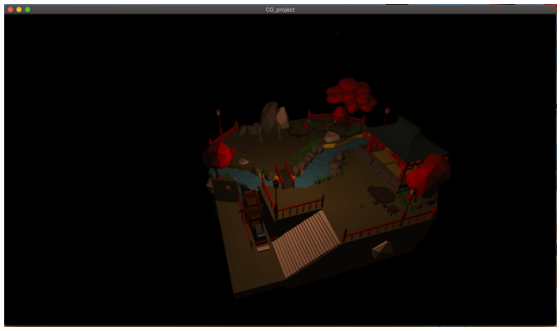


Figure.4-1







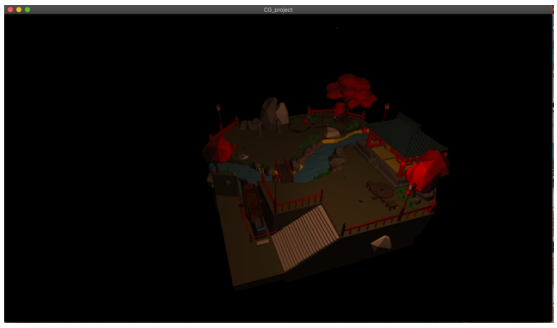


Figure.4-2

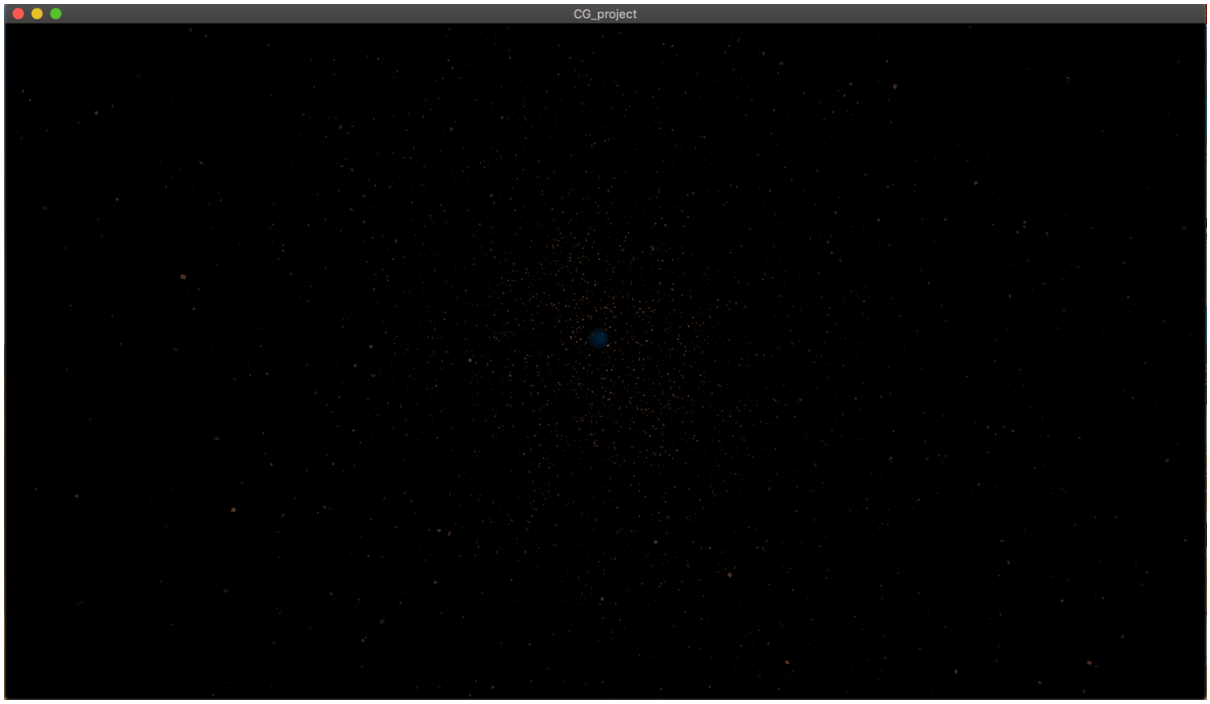


Figure.4-3

Creative Ideas

The project is aimed to express the beauty of autumn night, so I figure that the color style and objects performances are properly fit the theme. The main colors I used in the scene are red and orange, because leaves in falls are getting red and falling from the tree to the grounds, with lighting effectiveness, the peace and beauty in the autumn night are fully displayed then. And I did not implement skybox to surround my scene, but use some tiny objects distributed in the space over the scene to simulate the star night, with a blue moon. As a result, the theme of autumn night is able to be fully embodied with these objects altogether in the scene. Moreover, I implemented the waterfall effectiveness in my scene, it then looks more dynamic in the scene, despite the water in the river is static, waterfall effectiveness is able to make the whole scene more vivid and other animations such as the rotating of waterwheel, it can be regarded as the effect of the waterfall. Some performances of the content in this part can be found in the previous figures.

Commend lines and file descriptions

Project run by the following commend lines:

g++ -std=c++17 -lglfw -framework OpenGL -framework GLUT -Wno-deprecated-declarations cg_project.cpp -o cg_project && ./cg_project

There are totally 13 files in the project, three objects loaded in the project, each object has one **.obj** and **.mtl** file; two texture source images and two built-in header files, **tinyobjloader.h** used to load object file, **stb_image.h** used to load texture source images, two implemented header files, **lab06_framework.h** and **loadFunclib.h**, which first is from framework of lab6

in computer graphics, the other one contains some functions for image loading and object loading, which are used in **cg_project.cpp** file.

Summary

With the end of the semester, the project also met its ending. In this process, there are many requirements, as displayed in previous parts, I'd met all of the requirements, at least all of the requirements can be found in the scene, despite some of them are not satisfying. Firstly, there are not enough animations in my scene, in previous plan, there ought to have leaf falling effectiveness in the scene, and water running animations, which are not implemented in the final version project. Secondly, the whole scene is a little bit dark than expected, so that some object seems not very clear, the lighting effectiveness are expected to be more familiar then. Finally, the star night and moon are not very obvious in the scene, as showed in **Figure4-3**, stars and moon are difficult to see. As a result, there are still some deficiencies in my project, in there are some time in the future, these problems are main focus.

Reference List

[1]: https://sketchfab.com/3d-models/gd51-darren-t4a3-2f1e5edfdd354df7aa2579443d7cd088

[2]: https://sketchfab.com/3d-models/basic-streetlantern-73f702e0f8a64c30b487a914d80ac32c

[3]: https://sketchfab.com/3d-models/day-1-temple-019a4bedbb5243dbbe4d3d425b836a09

[4]:https://sketchfab.com/search?features=downloadable&q=desk&sort_by=-likeCount&type=models

[5]: https://sketchfab.com/3d-models/the-river-2a8453f6f5834671ab82a3afc1d6bd26

[6]: https://sketchfab.com/3d-models/star-cluster-15k-stars-model-51148b78a37a4a72b22d8e06f4293e07