

Report

Computer Graphics (COMP3069 UNNC)

Xinran TANG 20126518

1. How to use the program

- i. Use Visual Studio 2019.
- ii. Create a new Project -> Visual C++ -> Empty Project

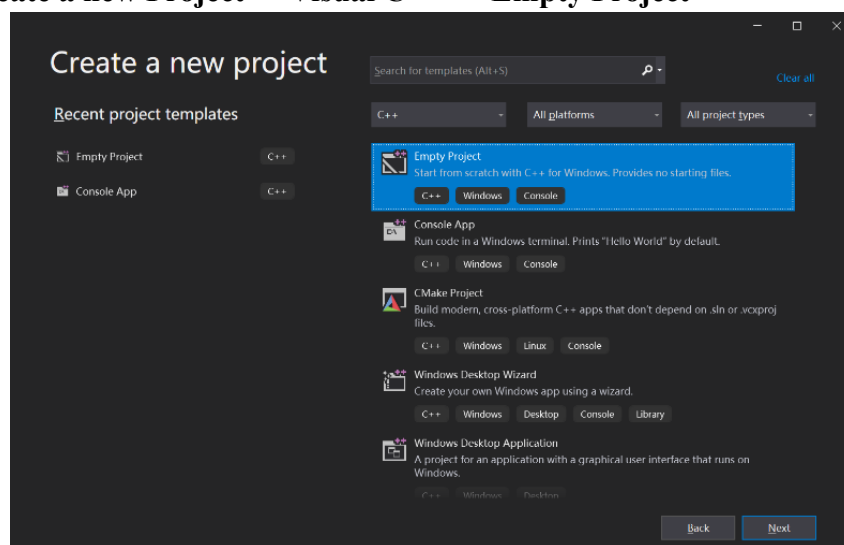


Figure 1: create a new project

- iii. Copy sources
 - Copy the files in 'CW2_lib' into the created project

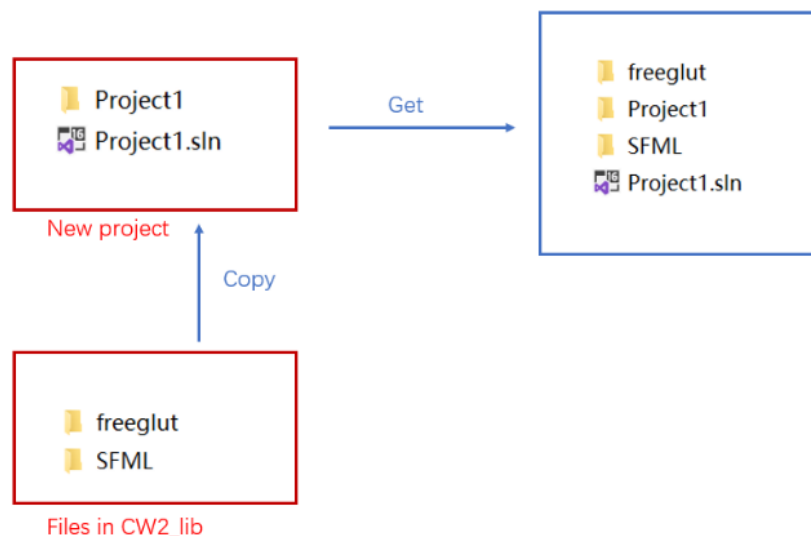


Figure 2: Copy the files in CW2_lib into the created project

- Copy the files in 'CW2_code' into the 'Project'

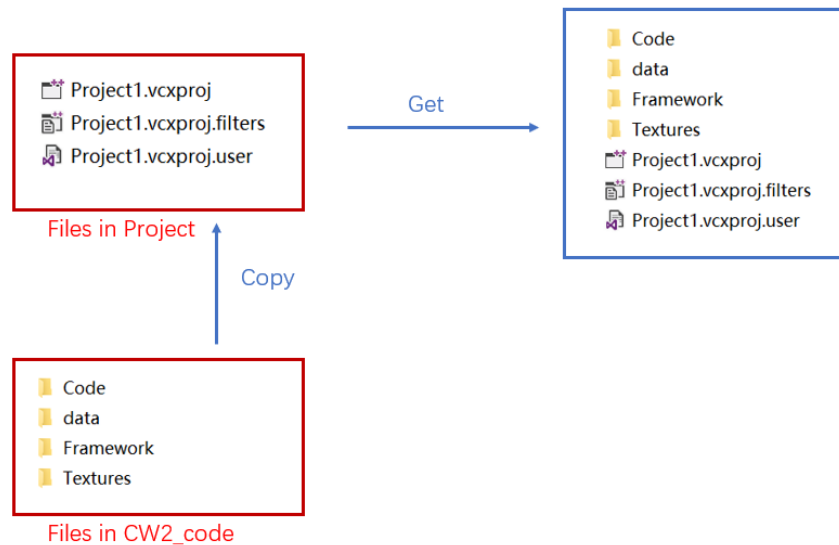


Figure 3: Copy the files in CW2_code into the Project

- Open the project in Visual Studio 2019



Figure 4: Open the project in Visual Studio 2019

- Then right click **Code**, **data**, **Framework**, and **Textures** folders, select **include in the project**

iv. Configuration: Right click Project and open Properties

Please configure the SFML on your computer if an error about SFML occurs.

- C/C++ -> General -> Additional Include Directories

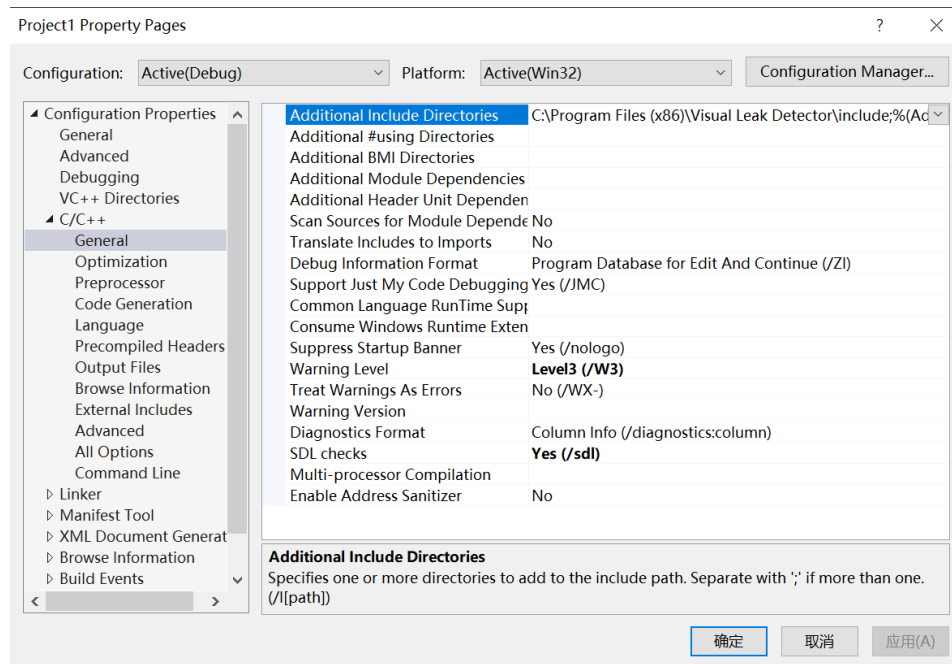
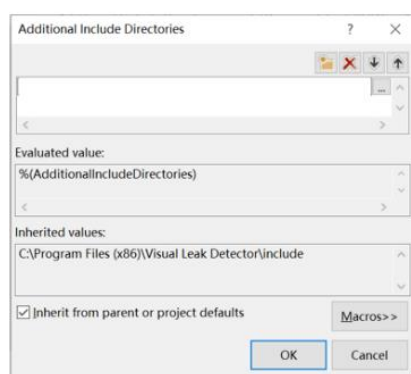


Figure 5: add Additional Include Directories

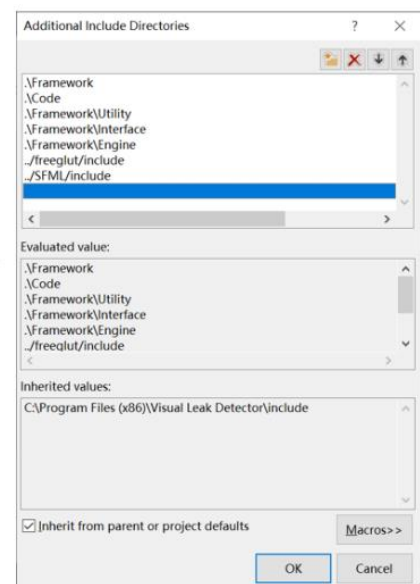
Add the following lines into **Additional Include Directories**:

.\Framework
.\Code
.\Framework\Utility
.\Framework\Interface
.\Framework\Engine
../freelut/include
../SFML/include



Before

Get



After

Figure 6: After adding Additional Include Directories

➤ Linker -> General -> Additional Library Directories

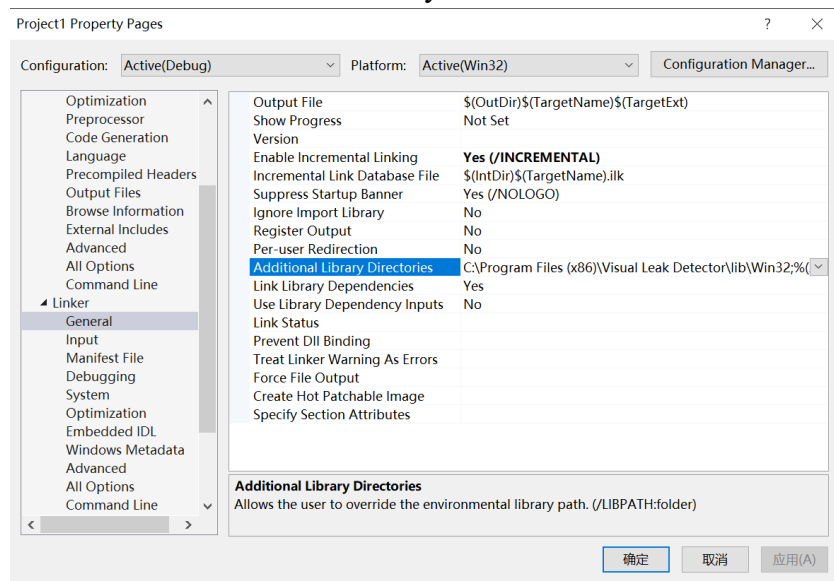


Figure 7: Add Additional Library Directories

Add the following lines into **Additional Library Directories**

../freelut/lib

../SFML/lib

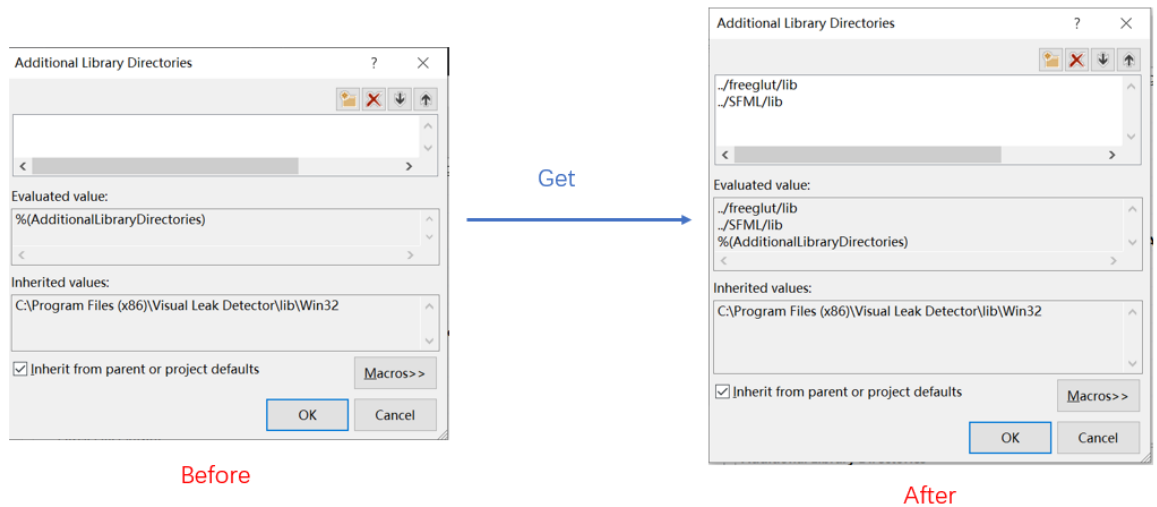


Figure 8:After adding Additional Library Directories

➤ Linker -> Input -> Additional Dependencies:

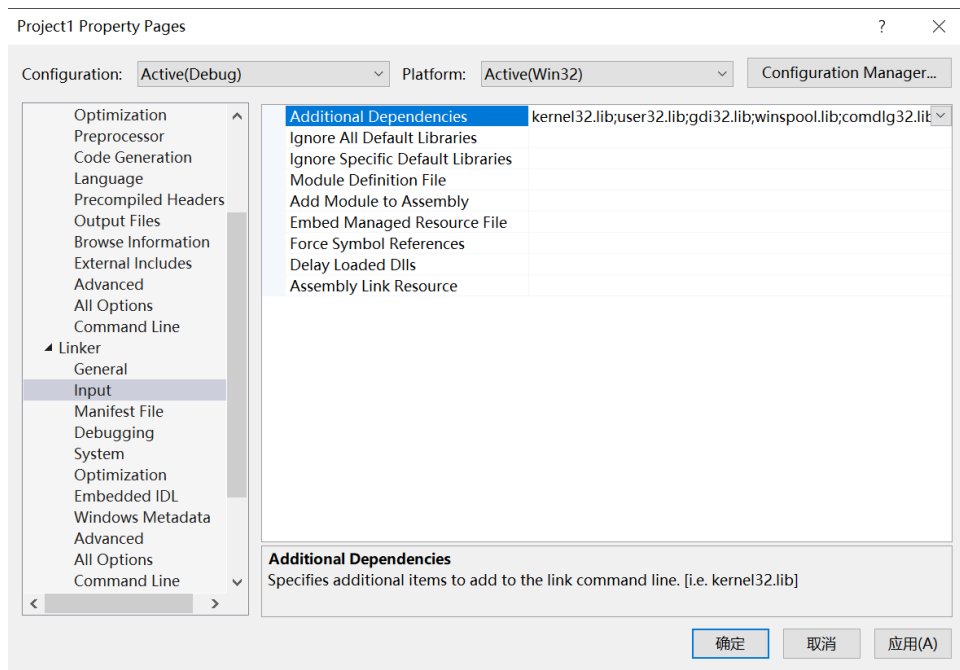


Figure 9: Add Additional Dependencies

Add the following lines into **Additional Dependencies**:

sfml-graphics-d.lib

sfml-window-d.lib

sfml-system-d.lib

sfml-audio-d.lib

sfml-network-d.lib

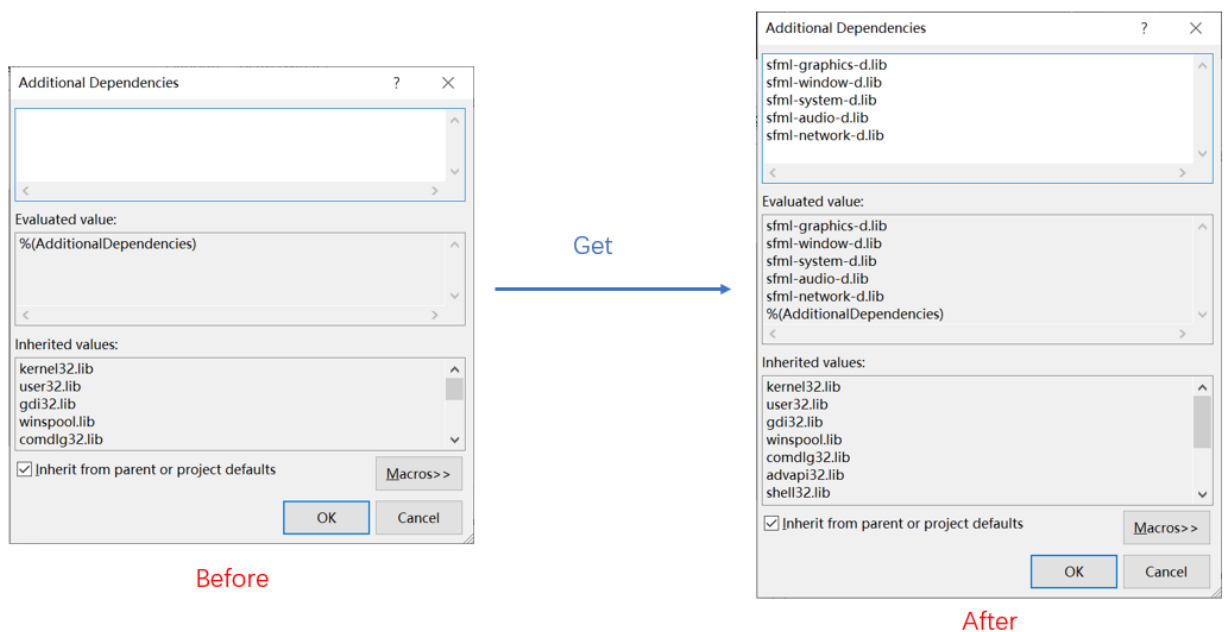


Figure 10: After adding Additional Dependencies

v. Run the program

Click the **Local Windows Debugger** to run the program



Figure 11: Run the program

vi. Use keyboard or mouse to move the camera

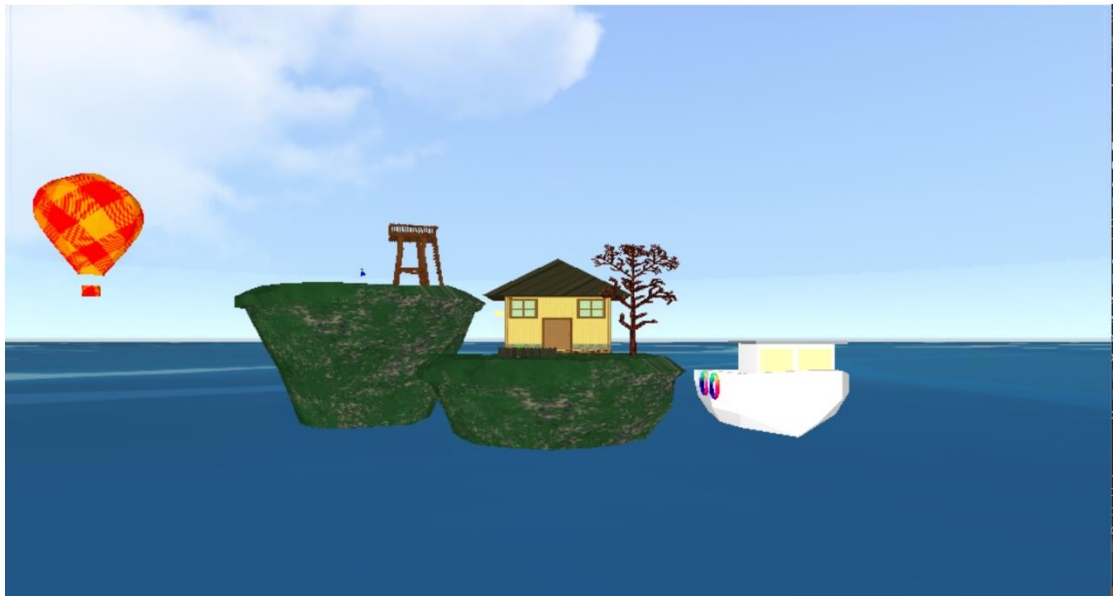
‘W’: move forward; ‘A’: move left; ‘S’: move backward; ‘D’: move right; ‘Q’: move downward; ‘E’: move upward.

vii. Object interactions

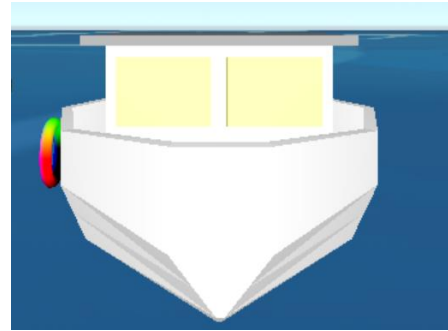
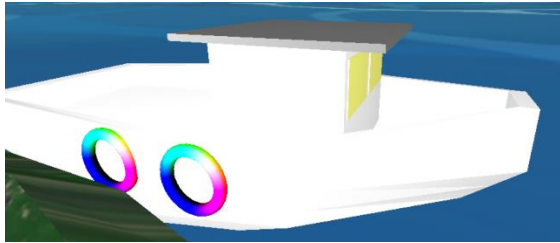
‘B’: change the viewpoint to the boat; ‘H’: change the viewpoint to the inside of the house; ‘N’: change the viewpoint to the left platform; ‘C’: pull the carrot up or put it down; ‘M’: frozen or unfrozen the water; ‘K’: turn off the light; ‘L’: turn on the light.

2. Scene Screenshots

➤ **outside view**



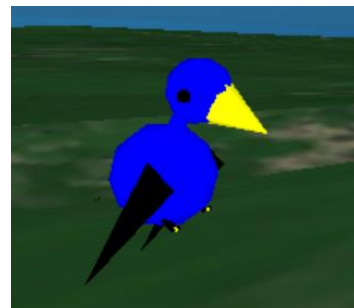
➤ **outside object – boat**



- outside object – yard (house, tree, fence, carrot, butterfly)



- outside object – left yard (bird, platform)



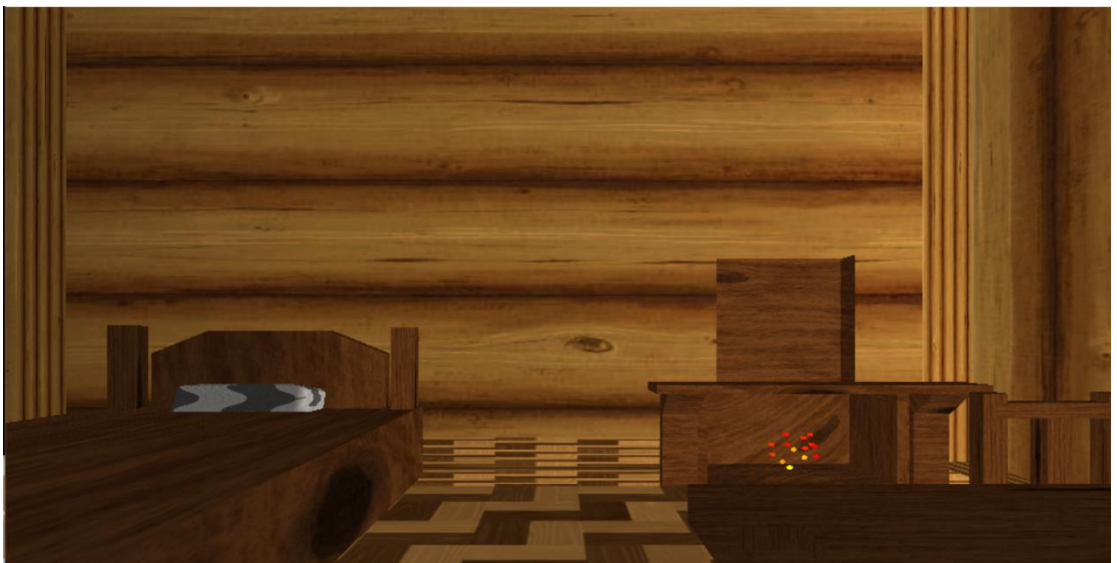
- outside object – balloon



- inside view – light on



- inside view – light off



3. Project Requirement

3.1 3D models

For 3D models, I used a combination of own created OpenGL models, own created blender models, and imported models.

➤ For own created **OpenGL models**:

- house
The walls of the house are created by drawing cubes and scratching them to the appropriate sizes. Then the texture images are added to each cube. The roof of the house is created by drawing tetrahedrons, and texture is added to the surface of the tetrahedron. The front wall of the house has a different texture than the other walls, which have a door and two windows.
- fence
The fence is created by drawing cubes and scratching them to the appropriate sizes. Five pieces of cubes in a group to form a well-proportioned fence shape: From low to high and high to low. Then the texture images are added to each cube.
- part of the ship
The top part of the ship is created by drawing cube and setting colors to them. Two lifebuoys are created by drawing torus and setting them in a colorful appearance.
- table
The table is created by drawing cubes and scratching them to the appropriate sizes. The texture images are added to each cube.
- fire in the fireplace
The fire in the fireplace is created by drawing glutSolidSphere and scratching them to the appropriate sizes. The color of the flames becomes red from inside to outside to make it looks more realistic.
- bird
The body, head, neck, wings, tail, beak, legs, and talons of the bird are created separately and then combined. The body and head are created by drawing glutSolidSphere, the neck, legs, and talons are created by drawing glutSolidCube, the wings and tail are created by GL_TRIANGLES of three points, the beak is created by drawing glutSolidCone.

➤ For own created **blender models**:

The following objects are created by using **blender** software and exported as a .obj file.

- wings of the butterfly
- carrot
- chair
- bed
- pillow
- fireplace
- balloon
- base stone

➤ For **imported models**:

- skybox (cite from Skybox.cpp, Skybox.h, stb_image.h provided by module convenor)
- water (cite from Water.cpp, Water.h provided by module convenor)
- tree (cite from the website: https://blog.csdn.net/qq_39019698/article/details/80376416)
- left platform (cite from the website: <https://free3d.com/>)
- part of the ship (cite from the website: <https://free3d.com/>)

3.2 Transformation of models

Each object is transformed to fit the size and position of the overall landscape, using `glTranslatef()`, `glScalef()` and `glRotatef()`.

3.3 Different viewpoint to the scene environment

In this program, there is a camera object, and you can change the camera view by pressing 'W', 'A', 'S', 'D', 'Q' or 'E'. Apart from that, you can also press 'B' to change the viewpoint of the boat and see the scene from a further distance. Press 'H' can change the viewpoint to the inside of the house and see the layout of the room. You can change the viewpoint to the left yard by pressing 'N'.

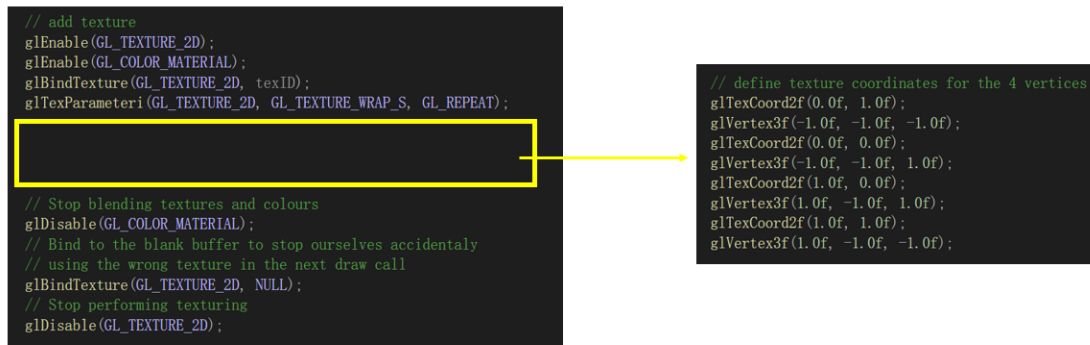
3.4 Animations for some objects in the scene

Object butterfly, boat, balloon, bird, carrot, and the fire in the fireplace have different animations as shown in the video.

3.5 Texturing

➤ OpenGL models

For OpenGL models, the functions from the GL library for mapping maps are used. The code shown at the left of Figure 12 is the frame used for each mapping, and the code at the right of Figure 12 shows mapping the points of the image to the points on the surface of the object.



```
// add texture
glEnable(GL_TEXTURE_2D);
glEnable(GL_COLOR_MATERIAL);
glBindTexture(GL_TEXTURE_2D, texID);
glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_WRAP_S, GL_REPEAT);

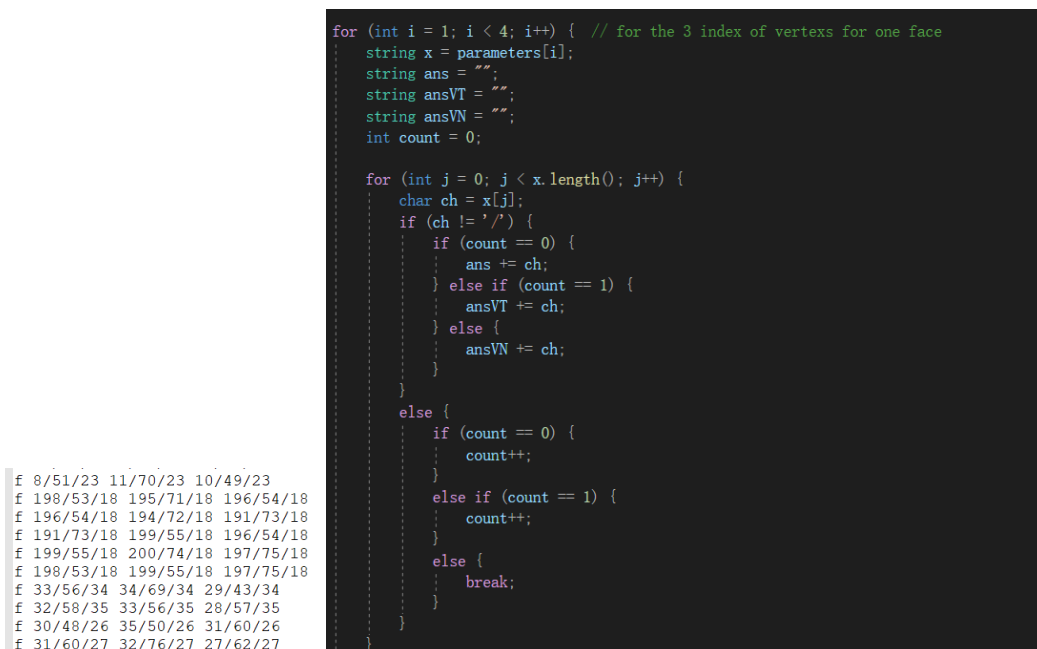
// Stop blending textures and colours
glDisable(GL_COLOR_MATERIAL);
// Bind to the blank buffer to stop ourselves accidentally
// using the wrong texture in the next draw call
glBindTexture(GL_TEXTURE_2D, NULL);
// Stop performing texturing
glDisable(GL_TEXTURE_2D);
```

```
// define texture coordinates for the 4 vertices
glTexCoord2f(0.0f, 1.0f);
glVertex3f(-1.0f, -1.0f, -1.0f);
glTexCoord2f(0.0f, 0.0f);
glVertex3f(-1.0f, -1.0f, 1.0f);
glTexCoord2f(1.0f, 0.0f);
glVertex3f(1.0f, -1.0f, 1.0f);
glTexCoord2f(1.0f, 1.0f);
glVertex3f(1.0f, -1.0f, -1.0f);
```

Figure 12: Texturing code

➤ blender models

For blender Models, I read the code in the .obj file and modified the code in ObjectLoader.cpp. After locating to f, 'vt' and 'vn' are separated. 'vt' stores texture coordinates and 'vn' is normal. Only 'vt' is used for mapping.



```
f 8/51/23 11/70/23 10/49/23
f 198/53/18 195/71/18 196/54/18
f 196/54/18 194/72/18 191/73/18
f 191/73/18 199/55/18 196/54/18
f 199/55/18 200/74/18 197/75/18
f 198/53/18 199/55/18 197/75/18
f 33/56/34 34/69/34 29/43/34
f 32/58/35 33/56/35 28/57/35
f 30/48/26 35/50/26 31/60/26
f 31/60/27 32/76/27 27/62/27
```

```
for (int i = 1; i < 4; i++) { // for the 3 index of vertices for one face
    string x = parameters[i];
    string ans = "";
    string ansVT = "";
    string ansVN = "";
    int count = 0;

    for (int j = 0; j < x.length(); j++) {
        char ch = x[j];
        if (ch != '/') {
            if (count == 0) {
                ans += ch;
            } else if (count == 1) {
                ansVT += ch;
            } else {
                ansVN += ch;
            }
        } else {
            if (count == 0) {
                count++;
            } else if (count == 1) {
                count++;
            } else {
                break;
            }
        }
    }
}
```

Figure 13: Part of the .obj file and Separate 'vt' and 'vn' from 'f'

3.6 Lighting

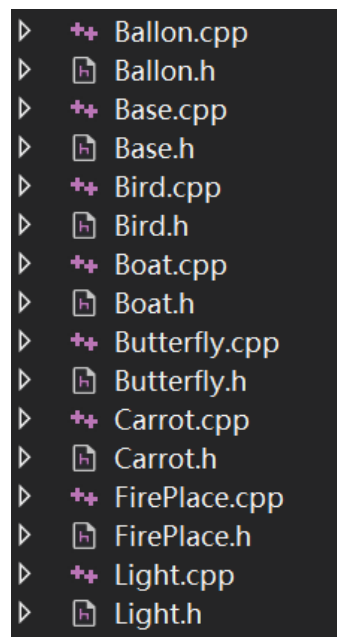
GL_LIGHT0 and GL_LIGHT1 are used in the project, GL_LIGHT0 is the environmental light, and GL_LIGHT1 is used for the flashlight when GL_LIGHT0 is off.

3.7 Good readability for program code

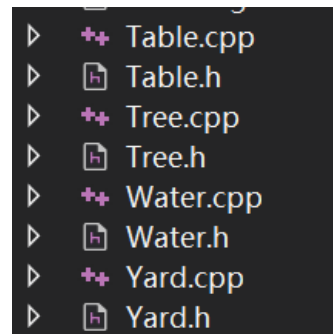
The project is developed by using object-oriented programming and based on the given framework (pure glut library). All objects are created in 'MyScene' class, in 'Initialise' function, and added to the scene by 'AddObjectToScene' function. All objects inherit 'Displayable' class, some also inherit 'Animation' or 'Input' classes.

➤ **folder structure:**

- **Code** folder: all objects that will be shown on the scene.
each object has its own .cpp and .h file named by its own appellation.



```
▶ ++ Ballon.cpp
▶ [h] Ballon.h
▶ ++ Base.cpp
▶ [h] Base.h
▶ ++ Bird.cpp
▶ [h] Bird.h
▶ ++ Boat.cpp
▶ [h] Boat.h
▶ ++ Butterfly.cpp
▶ [h] Butterfly.h
▶ ++ Carrot.cpp
▶ [h] Carrot.h
▶ ++ FirePlace.cpp
▶ [h] FirePlace.h
▶ ++ Light.cpp
▶ [h] Light.h
```



```
▶ ++ Table.cpp
▶ [h] Table.h
▶ ++ Tree.cpp
▶ [h] Tree.h
▶ ++ Water.cpp
▶ [h] Water.h
▶ ++ Yard.cpp
▶ [h] Yard.h
```

- **Framework** folder: provided by module convenor.
- **data** folder: all .obj files exported from the blender.
- **Texture** folder: all images used for texturing objects on the scene.
- **freeglut** folder, **SFML** folder: libraries.

4. Creative Idea

➤ **Carrot**

The carrot in the fence can be pulled up or put down by pressing the keyboard 'C'.



➤ **Butterfly**

The wings of the butterfly can rotate to imitate the real butterfly and the butterfly can fly back and forth.

➤ **Bird**

The bird in the left yard can jump up and down based on gravity.

➤ **Fire in the fireplace**

In order to make the fire looks more realistic, the upper flames are transformed repeatedly, and the color of the fire becomes red from inside to outside.

➤ **Flashlight**

When the light is off by pressing 'K', the flashlight will be open immediately. The flashlight can illuminate the area at a 50-degree cut Angle in front of the camera.

➤ **Music**

The background music is added by the SFML, which is a multimedia library providing a simple interface for each component of the PC to simplify the development of games and multimedia applications.

5. Citation

- [1] framework G53GRA: cite from the module convenor.
- [2] skybox, water, objLoader: cite from the module convenor.
- [3] images of the skybox: <http://www.humus.name/index.php?page=Textures&start=8>
- [4] images of the textures: <https://www.sharetextures.com/textures/>
- [5] imported 3D blender models: <https://free3d.com/>
- [6] imported tree model: https://blog.csdn.net/qq_39019698/article/details/80376416