Report for the final project in 3014 Computer Graphics 2018
a1695329 Yong Yang — Director, cameraman, and scriptor in both senses. &&
a1677895 Fengbei Liu — Artisan, scenic designer, and space cowboy's creator.
https://github.cs.adelaide.edu.au/a1677895/CGA3P2/releases/tag/1.0
User manual:

1. ESC key: Exit.
2. WASD keys for movement.
3. M key: Free the mouse so that it's visible and can move outside the OpenGL window. Press M again to control the camera again.
4. K key to look at the house instantly.
5. L key to look at the mysterious figure instantly.

Introduction:
We first talk about the two basic classes, Object and Camera that we are proud of, and then the ordeals we faced when implementing the relatively advanced techniques of environment mapping, shadow mapping, and particles. The marks that we think the program covered will be listed last.

Just to be pedantic, we use the term "we" even when we are describing the work of one person. The author of functionalities is identified by Authored by __ right after the functionality name. Also, we used our English names (Max for Yang and Thomas for Liu) in the code.

Details:
1. Object.h/Object.cpp. Authored by Liu. Originality: Quite original. We are following the red book's instruction for rotation and scaling operations, namely move to origin first, then rotate/scale, and translate back.
Object is the main class for processing all the properties of an Object, including extracting information from obj file, rendering and all sorts of operations support object modification.

- Each translation and scale operations have absolute and relative methods, help controller focusing on logic.
- Each object contains its own model matrix as model matrix only affect object itself. This improves the readability of the code and makes it easier for debugging.
- The material properties are passed to corresponding shader for light calculation.

2. Camera.h/Camera.cpp. Authored by Yang. Originality: Quite original. We followed the camera movement logic in main.cpp on learnOpenGL, but the functions, except a couple, are all implemented by us.
Camera is the class that yields the view matrix used for rendering objects.

- Cameras can move and look around, also with instant functions (teleport and setLookTarget) and gradual functions (move a bit, and turn a bit).
- The camera class has two instances only in our program, the main camera controlled by the mouse and WASD, and the secondary camera that is at the same position of main camera, but always looks at a target, and whose output is rendered at the bottom left corner.

3. Environment mapping. Authored by Liu. Originality: Fifty fifty. The original idea is followed from LearnOpengl Websites. Render the skybox on the surface of the object. Use I calculate the distance between camera and object , use R sample skybox from the environment. Skybox will project on the object perfectly.

4. Shadow mapping. Authored by Yang. Originality: Quite unoriginal. We followed the examples of both learnOpenGL and OpenGL-tutorial. We only had time to let objects project shadows on the ground, but not on other objects. We do have percentage-closer filtering, but the depth map has to cover such a large field, the shadows still look rather pixelated. The Peter Panning glitch of the house can be alleviated by using front face culling, but that will introduce problems for the rock's shadows, so we commented out depth map face culling.

5. Particles. Authored by Yang. Originality: Fifty fifty. The particle class was copied from learnOpenGL's class that creates 2D squares for that… Breakout protagonist. We wanted to make fire out of it first, but we didn't want to use code found online, because what good is a fireball if it doesn't have its own personality? We spent a dreadful amount of time, but failed horribly. So we got some snowflakes instead to cool down. The physics for snowflakes is quite easy. We used point sprites, by the way.

To sum up the functionalities we implemented:
Sunlight: directional light [1 mark]
Mysterious camera light: spot light [1 mark]
Trees, rocks, and tavern: loading obj files [3 marks]
Main camera and secondary camera (on the bottom left corner): multiple cameras [2 marks]
Texture on objects: texture mapping [1 mark]
Background: skybox [1 mark]
Mysterious effect on mysterious figure: depth cue [1 mark]
Shaders for all kinds of stuff for multipass: multiple vertex/fragment shaders [2 marks]
Mysterious figure: environment mapping [2 marks]
Pixel Shadows: multi-pass method: shadow mapping [3 marks]
Snowflake: procedural generation of particles [3 marks]
Background music: simple sounds [1 mark]

Yang was the project architect, and he was the tech guy that did most of the R&D (not rest and dine, I hardly have time for either) and debugging for the team. He's the first author of Shadows [3], Cameras [2], and Snowflakes[3] (sounds like a random third-class novel doesn't it) which are hopefully worth at least 8 marks.
Liu did most of the fundamental stuff. He's also the first author of Lights [2], Objects[3 + 1], and Boxes [1 + 2], Music[1], and Fog[1], that add up to 11 marks.
Neither of us can do anything without our own shader programs, so the 2 marks for multiple shaders cannot really be claimed by one author.
So we think our program covers about 21 marks. There are alpha blending features on snowflakes, but nothing else. There's actually also point light emitting from the camera, but we couldn't think of a reasonable …… reason for it.

Finally, we are happy with a 50/50 share of the loot.