CPT205 Assignment 2 Technical Report

Name	Zihan	Deng
ID Number	2144196	
Programme	Information and Computing Science	
Module Title	Computer Graphics	
Module Code	CPT205	
Assignment Title	Assessment 2 – 3D Modelling Project	

Introduction

This program, based on the Freeglut library, provides a simple implementation of modeling the SIP North Campus of the university, vividly showcasing the daily life on campus. The program incorporates lighting and texture features to create more realistic scene constructions. In terms of animation, the program includes pedestrians and cars with lively walking animations, featuring straight-line walking and driving, as well as flags fluttering in the wind. In terms of interaction, the program achieves realistic camera spatial movement and rotation through mouse and keyboard interactions.

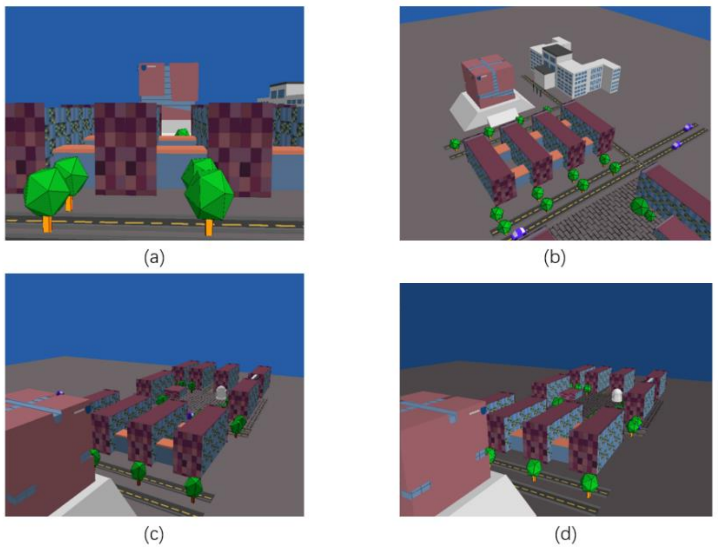

Figure 1. A simple demonstration of the program. Includes viewpoint movement and lighting changes

Program Demonstration:

The program's usage adheres to daily logic and is straightforward. Upon startup, as shown in **Figure** 1 (a), users can click and drag within the window using the mouse to observe the surroundings, similar to turning their heads. To move the camera position for a more comprehensive view of the campus, users can use the `WASD` keys for forward, backward, left, and right movements in the x-z plane, respectively. The spacebar and the `C` key control the camera's ascent and descent along the y-axis. **Figures** 1 (b) and (c) illustrate scenes that users may see after performing camera position movements. To simulate different scenes throughout the day on campus, holding down the `Q` or `E` key can control the movement of the light source, as shown in **Figures** 1 (c) and (d). Finally, pressing the digit "`1`" resets the camera to its initial position, and pressing the Esc key exits the program.

Implementation of Static Objects

The structures in this program are composed of static, simple objects, modeling buildings such as CB, FB, and the buildings of science and engineering complexes. Among them, there are objects with similar appearances, allowing for efficient code reuse. It is worth noting that in the three-dimensional scene, solid objects enclosed by two-dimensional planes usually lack lighting effects. Therefore, it is necessary to define the normal vectors for each face in the programming. Second, the source code for trees and shrubs, serving as decorations in the program, comes from materials used in school laboratory courses.

In the programming process, the concept of hierarchical modeling is applied. Since OpenGL's graphics rendering is based on changes in coordinate systems, maintenance of the stack in memory is necessary to read or save previous coordinate parameters.

In terms of lighting, although OpenGL provides multiple initial light sources for invocation, this program only uses light source number 0. This light source simulates sunlight with an infinite position, providing global illumination to the environment.

Regarding materials, the program reads and maps previously drawn bitmap files onto different object surfaces. The program utilizes four texture maps, as shown in **Figure** 2. S.bmp and SGLS.bmp represent the appearance of the exterior walls of the science and engineering buildings' different faces, BG.bmp depicts the ground of the North Campus square, and FBGLS.bmp portrays the style of windows for FB. Among these four texture maps, BG.bmp is from internet sources.

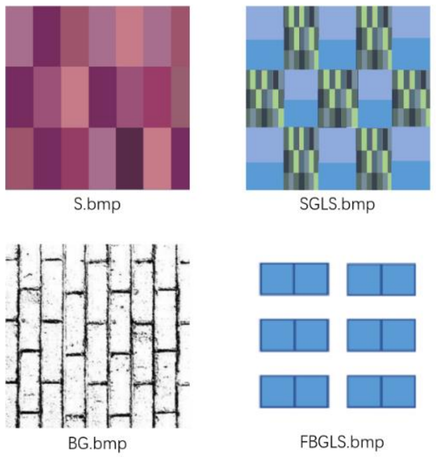

Figure 2. Material (BG.bmp source: https://blog.csdn.net/L_Stock/article/details/128113151)

Implementation of Animation

This program implements three objects with animations, as shown in Figure 3. For the movement of the car and pedestrians, simple callback functions are employed. Their positions are linearly increased, and the scene is redrawn. After reaching a certain position, their coordinates are reset to create a looping effect. The animation of the flag is inspired by the flag animation in CW1, using periodic functions to linearly increase the flag's parameters and segment it into stripes.

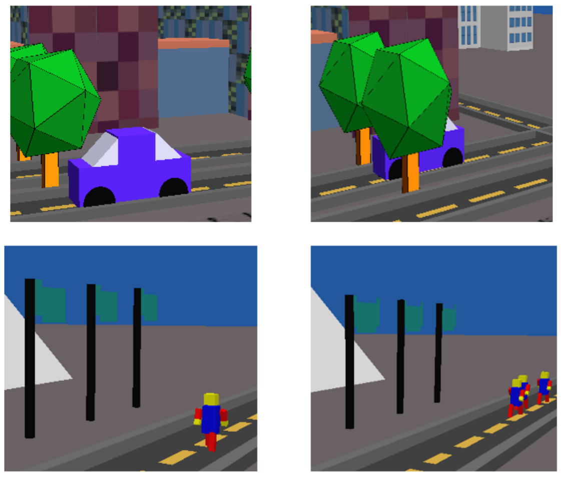

Figure 3. Animation

Implementation of Interaction

To allow users to observe the campus environment from any angle, the program implements user-friendly human-computer interaction. For camera movement, the program follows the logic used in various first-person 3D games, enabling movement through the WASD keys and rotating the view through mouse clicks and drag operations. Since rotating the view changes the camera coordinate system, the direction vector needs to be dynamically adjusted. This functionality is achieved by trigonometric transformations on the x-z plane vector in the program. Mouse movement operations are implemented through the use of `` `glutMotionFunc` ``. Regarding light source movement, Q and E control the light source's movement along the x and z axes. To prevent unlimited increases in parameters, coordinate limitations have been added.

Discussion

In summary, the program clearly constructs a 3D scene and simulates real-world scenarios through techniques such as lighting, textures, and hierarchical modeling. However, there is still room for optimization in the subsequent stages of the program. Firstly, the complexity of the code can be improved by using more efficient data structures or algorithms to streamline them. Secondly, one could consider implementing more sophisticated lighting effects, customizing reflection effects for different objects, and enhancing texture details to further refine the program.