CSCI 3260 Principles of Computer Graphics

Assignment Two: Texturing and Lighting (15%)

Due Time: 11:59pm, Nov 10 (Saturday), 2018

Late penalty: 10% per day.

Fail the course if you copy

I. Introduction

In this assignment, you are required to build an even more realistic and complex scene with OpenGL. To achieve this task, you will experience more features in OpenGL, including lighting, complex model building and loading, texture mapping and interactive events. You are about to use primitive drawing or load a 3D model from an .obj file directly and then view/model the transformation to create this 3D scene. Texture mapping and lighting will be employed to make the scene and objects more realistic. Mouse/keyboard inputs and window event handling will help realizing the interactive animation.

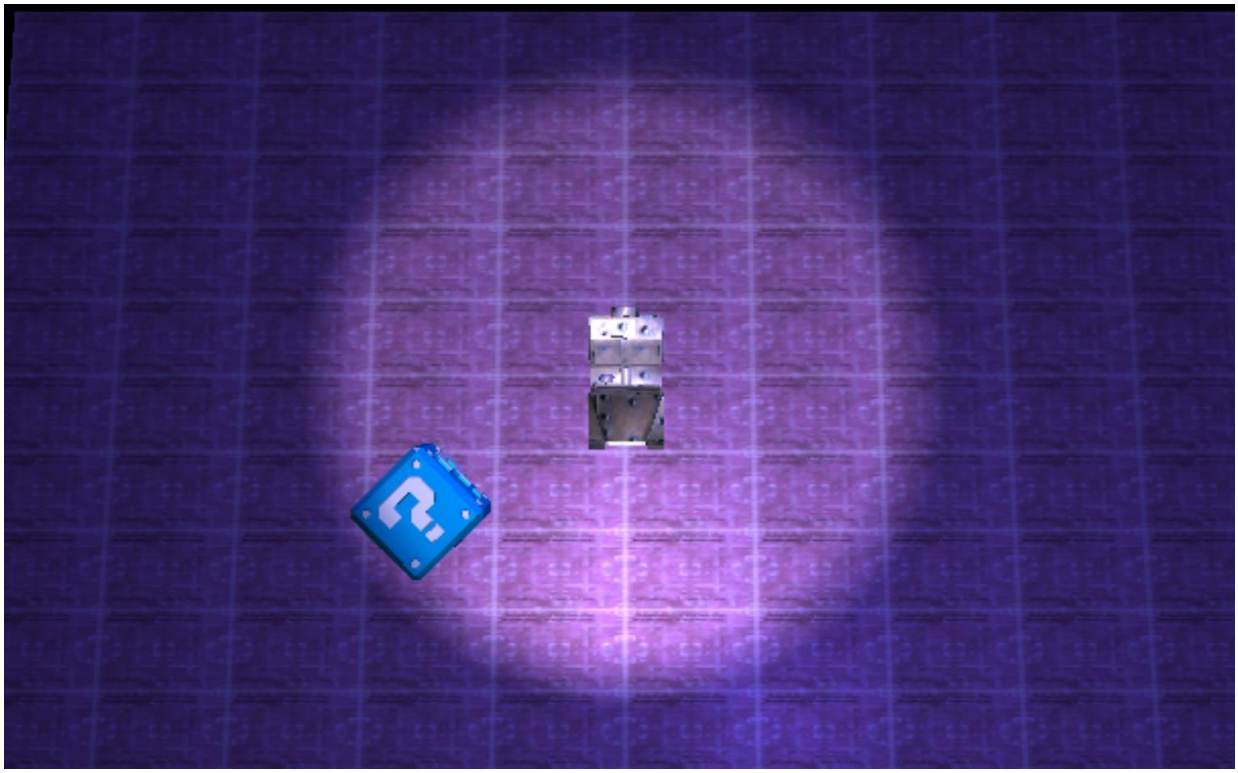

Fig. 1: The scene drawn by the demo program.

In this assignment, there are three models in the scene. One of them (the ground plane) is relatively simple, others (jeep and block) are complex. We can design the vertex attributes of the ground plane by ourselves. However, for the jeep and block, they are so complicated that we need to load the models by .obj files. In addition,

the three models are rendered with different textures and lighting effects. The scene displayed can be controlled by the user's interactive input. You can also enrich the scene you modeled in assignment 1.

II. Implementation Details

Task 1: Loading complex object

Use the Open Asset Import Library, or the function '**bool loadOBJ()**' which we have given to load at least two complex models, i.e. the jeep and block in the demo program. In this part, you can use **bool loadOBJ()** function by modifying the '**void sendDataToOpenGL()**' subroutines.

We have provided the models in the demo program, i.e. plane.obj, jeep.obj and black.obj. You are encouraged to download other .obj file from Internet or use Blender to design your own objects.

Task 2: Texture Mapping & Lighting

You need to map different textures to the three models, i.e. the ground plane, jeep and block in the demo program. You first need to create one OpenGL texture and set the texture parameter by modifying the '**GLuint loadBMP_custom(const char * imagepath)**' subroutines. Then, load and bind textures to different models in '**void sendDataToOpenGL()**' and '**void paintGL(void)**' subroutines, respectively.

Here, we have also provided the textures of models in the demo program, i.e. block_texture.bmp, jeep_texture.bmp. You are also encouraged to download other textures from Internet.

In addition, the 3D scene should be illuminated with at least two light sources. One should be an environment light. For the other light sources, such as diffuse and specular light, you can decide the position and color by yourself. The main purpose of adding such light sources is to produce the diffuse light and specular light effects on the models. You can do this by modifying the '**void paintGL(void)**' subroutines.

Task 3: Interactive Events and Animation

In this task, you are required to implement the following interactive events and animation:

(a) Lighting control

Press key "q" and key "w" to increase and reduce brightness of diffuse light, respectively.

Press key "z" and key "x" to increase and reduce brightness of specular light, respectively.

(b) Texture control

Press key '1'~'3' to switch the texture for the ground plane, we also provided three textures which can be applied on the ground plane. (i.e. theme1.bmp~ theme3.bmp)

(c) Model animation and control

One complex model can rotate about one axis and we can use key "s" to control either stop or continue the rotation. (See animation of the block in the demo program)

Press arrow keys "↑ ↓ ←→" to control the movements of the other complex model on the ground plane. Specifically, "↑ ↓" indicate forward and backward movement respectively. "←→" indicate left and right rotation respectively. (See animation of the jeep in the demo program)

(d) View control

Control the position of camera by mouse, which means:

When mouse moves up, the whole scene you see moves down.

When mouse moves left, the whole scene you see moves right.

Press key ' ' (key “space”) to toggle between states of controlling and not controlling.

In this task, you may modify the following subroutines:

void keyboard(unsigned char key, int x, int y) ;

void move(int key, int x, int y) ;

void PassiveMouse(int x, int y);

Bonus Task: Enhance the visual effect of your scene (maximum 20%)

OpenGL provides many functions for your program to create various visual effects. You can study them by yourself and introduce them into the assignment. Here are some suggested improvements:

- Loading more complex models and map other textures onto them to form a meaningful scene, for example, car park. (10%)
- Using Skybox to render the whole environment. (10%)
- Using different types of lighting source, such as Pointlight, Spotlight, Arealight. (10%)
- Moving the lighting in the environment. (10%)
- Draw points or lines to trace the movement of one of complex models (10%)
- Any other interesting effects

III. Grading Scheme

Your assignment will be graded by the following marking scheme:

Basic (80%)

Loading the complex model	10%
Texture mapping	20%
Lighting of environment light	5%
Lighting of light source you designed	15%
Lighting control	10%
Texture control	5%
Model animation	10%
View control	5%
Bonus	20%
Total:	**100%**

Note: no grade will be given if the program is incomplete or fails compilation.

IV. Guidelines to submit programming assignments

1) You are suggested to write your programs on Windows, since there will be enough technical support. If you developed the program in other platforms, make sure your program can be compiled and executed on Windows as the program will only be tested on this platform. The official IDE is Visual Studio C++ 2017.

2) Modify the provided *main.cpp* & *VertexShaderCode.glsl* & *FragmentShaderCode.glsl*. No other additional .cpp or .h files are allowed. You could create additional .glsl files if you would like to implement shadow mapping technique. Type your full name and student ID in *main.cpp*. ***Missing such essential information will lead to mark deduction (up to 10 points).***

3) We only accept OpenGL code written in programmable pipeline. No points will be given if your solution is written in fixed pipeline.

4) Zip the source code file (i.e. *main.cpp & VertexShaderCode.glsl & FragmentShaderCode.glsl*), the executable file (e.g., *openGL.exe*), the readme file (i.e., *readme.txt*), the .obj file you create or download and the .bmp file you download in a .zip. Name it with your own student id (e.g. *1155012345.zip*).

5) Submit your assignment via eLearn Blackboard. (https://blackboard.cuhk.edu.hk/)

6) Please submit your assignment before 11:59 p.m. of the due date. Late submission will be penalized by 10% deduction per day.

7) In case of multiple submissions, only the latest one will be considered.

8) ***Fail the course if you copy.***