CSCI 3260 Principles of Computer Graphics

Assignment Two: Texturing and Lighting (15%)

Due Time: 11:59pm, Nov 11 (Monday), 2019

Late penalty: 5 points per day.

Late submission after two weeks past deadline will not be accepted.

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 a primitive drawing or load a 3D model from a .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 in realizing the interactive animation.

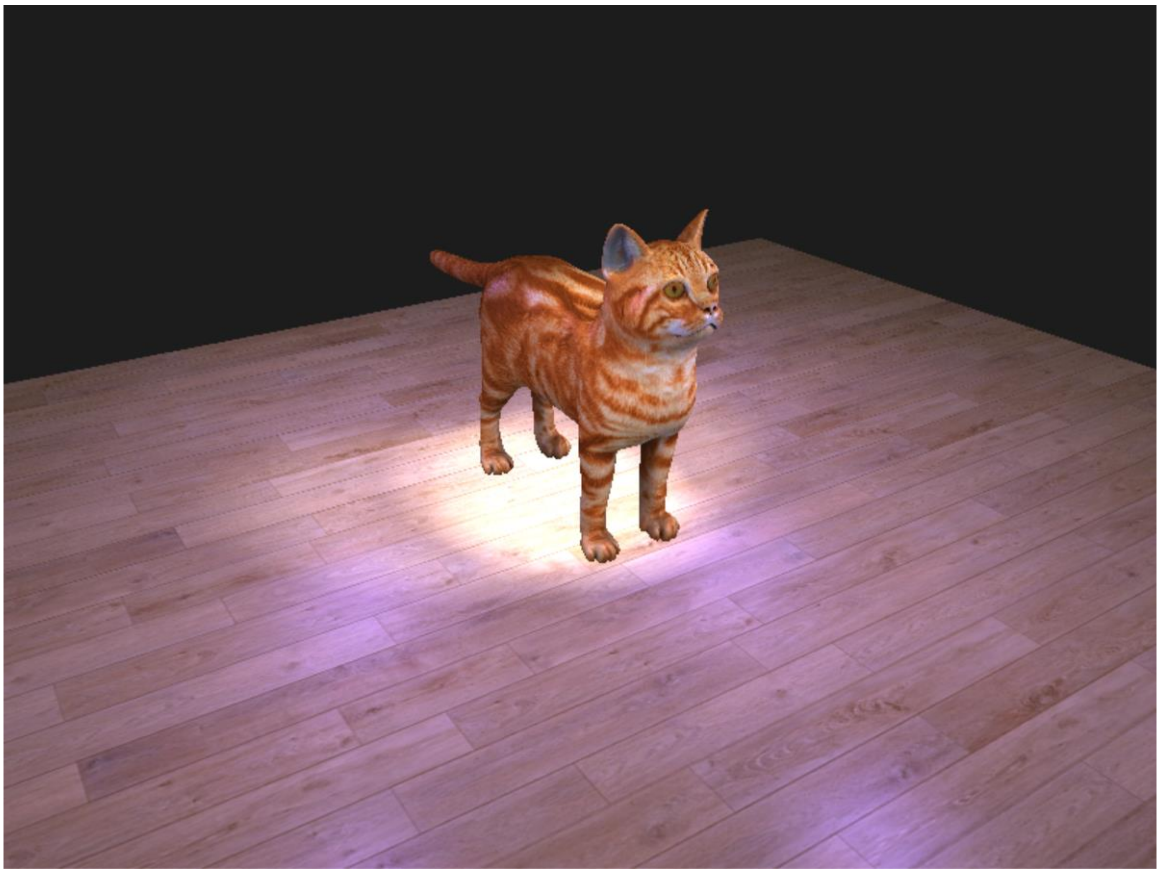

Fig. 1: The scene drawn by the demo program.

In this assignment, there are two models in the scene. One of them (the ground plane) is relatively simple, the other (cat) is complex. We can design the vertex attributes of the ground plane by ourselves. However, for the cat, it is so complicated that we need to load the models by .obj files. Besides, the ground plane and the cat 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 created in assignment 1.

II. Implementation Details

Task 1: Loading complex object

Use the Open Asset Import Library, or the function '`Model loadOBJ(const char* objPath)`' which we have given to load at least one complex model, i.e. the cat in the demo program. In this part, you can use '`Model loadOBJ(const char* objPath)`' function by modifying the '`void sendDataToOpenGL()`' subroutines.

We have provided the models in the demo program, i.e. floor.obj and cat.obj. You are encouraged to download other .obj files from the Internet or use Blender to design your objects.

(You need to inspect the cat.obj because if you directly draw the cat, it will be very huge. Specifically, you need to do some transformations.)

Task 2: Texture Mapping & Lighting

You need to map different textures to the two models, i.e. the ground plane and the cat in the demo program. We will use the stb image library (see "Dependencies/stb_image") to load the texture images. You are required to change the texture of the cat by using keyboard interaction. You first need to create one OpenGL texture and set the texture parameter by modifying the '`GLuint loadTexture(const char* texturePath)`' 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. floor/floor_diff.jpg, floor/floor_spec.jpg, cat/cat_01.jpg, cat/cat_02.jpg. You are also encouraged to download other textures from the Internet.

Also, the 3D scene should be illuminated with at least two light sources. One should be an environment (directional) light. For the other light sources, 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 "w" and key "s" to increase and reduce the brightness of directional light, respectively. At least one light can rotate about one axis and we can use key "p" to control either stop or continue the movement. (See the animation of the point light in the demo program)

(b) Texture control

Press key '1'-'2' to switch the texture for the cat, we also provided two textures which can be applied on the cat. (i.e. cat/cat_01.jpg – cat/cat_02.jpg)

(c) Animation and control

Press arrow keys " ↑ ↓ ←→ " to control the movements of the cat model on the ground plane. Specifically,

"↑ ↓" indicate forward and backward movement respectively. "←→" indicate left and right rotation respectively. (See the animation of the cat in the demo program)

(d) View control

Control the position of camera by mouse, which means:

When the left button clicked and the mouse moves up, the whole scene you see moves down.
When the left button clicked and the mouse moves left, the whole scene you see moves right.
(See the demo program. The right-click function does not require.)

In this task, you may modify the following subroutines:

void mouse_callback(int button, int state, int x, int y);
void motion_callback(int x, int y);
void keyboard_callback(unsigned char key, int x, int y);
void special_callback(int key, 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, a car park or a zoo. (10%)
- Using different types of lighting sources to make meaningful scenes, such as the combination of Pointlight, Spotlight, etc. (10%)
- Shadow mapping on the complex models. (10%)
- Draw points or lines to trace the movement of one of the 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 (directional) 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 on 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++ 2019.
2) Modify the provided *main.cpp* & *VertexShaderCode.glsl* & *FragmentShaderCode.glsl*. You could create additional .glsl files if you would like to implement the 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 the programmable pipeline. No points will be given if your solution is written in the 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 image file you download in a .zip. Name it with your 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. In the case of multiple submissions, only the latest one will be considered. Late submission will be penalized by 5 points deduction per day.**
7) ***Fail the course if you copy.***