CSCI 3260 Principles of Computer Graphics

Assignment 2: Texturing and Lighting

Due Time: 11:59pm, Nov 1 (Sunday), 2020

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 captured in the demo

In this assignment, there are two models in the scene. One of them (the background sea) is relatively simple, the other (the dolphin) is complex. We can design the vertex attributes of the background sea by ourselves. However, for the dolphin, it is so complicated that we need to load the models by .obj files. Besides, the sea and the dolphin 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 which 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 dolphin 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.sea.obj and dolphin.obj. You are encouraged to download other .obj files from the Internet or use Blender to design your objects.

(You need to inspect the dolphin.obj because if you directly draw the dolphin, 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 sea surface and the dolphin 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 dolphin by using keyboard interaction. You firstly need to generate one OpenGL texture and set the texture parameter by modifying the '`void Texture::setupTexture(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 the two models in the demo program, You are also encouraged to download other textures from the Internet or draw/filter texutures by yourself.

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.

(b) Texture control

Press key '1' and '2' to switch two different textures for the dolphin, and we have also provided two textures which can be applied on the dolphin. (i.e. dolphin/dolpihin_01.jpg, dolphin/dolphin_02.jpg)

Press key '3' and '4' to switch two different textures for the sea surface, and we have also provided two textures which can be applied on the dolphin. (i.e. sea/sea_01.jpg, sea/sea_02.jpg)

(c) Object control

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

(d) View control

Control the view of camera by mouse, which means:

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

In this task, you may modify the following subroutines to implement above requirements:

```
void mouse_button_callback(GLFWwindow* window, int button, int action, int mods)
{
    // Sets the mouse-button callback for the current window.
}
void cursor_position_callback(GLFWwindow* window, double x, double y)
{
    // Sets the cursor position callback for the current window
}
void scroll_callback(GLFWwindow* window, double xoffset, double yoffset)
{
    // Sets the scoll callback for the current window.
}
void key_callback(GLFWwindow* window, int key, int scancode, int action, int mods)
{
    // Sets the Keyboard callback for the current window.
}
```

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:

- Load more complex models and map other textures onto them to form a meaningful scene as you like. (10%)
- Use 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	15%
Texture mapping	15%
Lighting of environment (directional) light	10%
Lighting of light source you designed	10%
Lighting control	10%
Texture control	10%
Object control	10%
View control	10%
Bonus	10%
Total:	**100%**

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

IV. Guidelines to submit programming assignments

1) You can write your programs on Windows and macOS. Previously, the official grading platform is Windows with Visual Studio. If we encounter problems when execute/compile your program, you may have to show your demo to the tutor in person.

2) Modify the provided main.cpp & VertexShaderCode.glsl & FragmentShaderCode.glsl and provide all your code in this file. It is not recommended to create or use other additional .cpp or .h files. 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) We only accept OpenGL code implemented with GLFW and GLEW. No points will be given if you use other windowing and OpenGL extension libraries (unless you have strong enough reasons).

5) Zip the source code file (i.e. main.cpp & VertexShaderCode.glsl & FragmentShaderCode.glsl), the executable file (i.e., openGL.exe) (if you use Windows for your assignment), and the readme file (i.e., readme.txt) in a .zip (see Fig. 3). Name it with your own student id (e.g. 1155012345.zip).

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

7) Please submit your assignment before 11:59 p.m. of the due date. Late submission will be penalized by 5 points deduction per day.

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

9) Fail the course if you copy.