Programming Assignment 04: Shaded Rendering

1. Overview

In this assignment you will use OpenGL Shading Language(GLSL) to write your own vertex and fragment shaders to compute illumination and shading of meshes.

Basic Requirements

1. **Generate Triangle Meshes (TODO 1):**
    1. Use Element Buffer Object (EBO) to draw the cube. The cube provided in the start code is drawn with Vertex Buffer Object (VBO). In the DisplayableCube class draw method, you should switch from VBO draw to EBO draw. To achieve this, please first read through VBO and EBO classes in GLBuffer. Then you rewrite the self.vertices and self.indices in the DisplayableCube class. Once you have all these down, then switch the line vbo.draw() to ebo.draw().
    2. Generate Displayable classes for an ellipsoid, torus, and cylinder with end caps. These classes should be like the DisplayableCube class and they should all use EBO in the draw method.
    PS: You must use the ellipsoid formula to generate it, scaling the Displayable sphere doesn't count
2. **Set Normal Rendering (TODO 2):** As a visual debugging mode, you'll implement a rendering mode that visualizes the vertex normals with color information. In Fragment Shader, use the vertex normal as the vertex color (i.e. the rgb values come from the xyz components of the normal). The value for each dimension in vertex normal will be in the range -1 to 1. You will need to offset and rescale them to the range 0 to 1.
3. **Illuminate your meshes (TODO 3):** Use the illumination equations we learned in the lecture to implement components for
    1. Diffuse
    2. Specular
    3. Ambient

    You'll implement the missing part in the Fragment shader source code. This part will be implemented in OpenGL Shading Language. Your code should iterate through all lights in the Light array.
4. **Set up lights (TODO 4):** Set up lights:
    1. Use the Light struct which is defined above and the provided Light class to implement illumination equations for 3 different light sources.
        - Point light
        - Infinite light
        - Spotlight with radial and angular attenuation
    2. In the Sketch file Interrupt_keyboard method, bind keyboard interfaces that allow the user to toggle on/off specular, diffuse, and ambient with keys S, D, A.
5. **Create your scenes (TODO 5):**
    1. We provide a fixed scene for you with preset lights, material, and model parameters. This scene will be used to examine your illumination implementation, and you should not modify it.
    2. Create 3 new scenes (can be static scenes). Each of your scenes must have
        - at least 3 differently shaped solid objects

    - each object should have a different material
    - at least 2 lights
    - All types of lights should be used
3. Provide a keyboard interface that allows the user to toggle on/off each of the lights in your scene model: Hit 1, 2, 3, 4, etc. to identify which light to toggle.

6. **Texture Mapping (BONUS FOR 480, 10 extra credits) (TODO 6/BONUS 6)**

    1. Set up each object's vertex texture coordinates(2D) to the self.vertices 9:11 columns (i.e. the last two columns). Tell OpenGL how to interpret these two columns: you need to set up attribPointer in the Displayable object's initialize method.
    2. Generate texture coordinates for the torus and sphere. Use "./assets/marble.jpg" for the torus and "./assets/earth.jpg" for the sphere as the texture image. There should be no seams in the resulting texture-mapped model.

Extra Credit (10 points each, you may only receive up to 10 pts extra)

7. **Normal Mapping (BONUS 7)**

    1. Perform the same steps as Texture Mapping above, except that instead of using the image for vertex color, the image is used to modify the normals.
    2. Use the input normal map ("./assets/normalmap.jpg") on both the sphere and the torus.

8. **Artist Rendering (advanced) (BONUS 8)**

    Look at Section 10.3, "Artistic Shading" in Shirley/Marschner (4th ed.).

    1. Implement line drawing in shader code
    2. Implement cool-to-warm shader code

Programming Style

9. For any modified or newly added source file, you should include a brief explanation about what you did in this file at the file heading and add your name to the author list. Your code should be readable with sufficient comments. You should use consistent variable naming and keep reasonable indentation. In python, we prefer to use reStructuredText format docstring, for more details about this format, please check **here**.

2. Resources

2.1 Start code

A Python Program skeleton, which includes basic classes, methods, and main pipeline, is provided for you. You are expected to complete the parts marked with TODO. There are comments in the skeleton code that will help guide you in writing your own subroutines.

2.2 Environment Setup

Installing the appropriate programming environment should be covered in a lab session. For more step-by-step instructions, please check the environment set up on Blackboard.

2.3 User Interface

The user interface to the program is provided through mouse buttons and keyboard keys.

Left Mouse Dragging: Rotate the camera
Middle Mouse Dragging: Translate the camera
A: Toggle Ambient Light
D: Toggle Diffuse Light
S: Toggle Specular Light
Left Arrow: Go back to last scene
Right Arrow: Next Scene
1,2,3, ...: Toggle lights in current scene

2.4 Video Demo

We prepared a video demo for you, hope this can help you better understand your tasks and speed up your debugging process. Please check it on the Blackboard.

3. Submission (due by midnight, Tuesday, 12/7)

3.1 Source Code

Your program's source files are to be submitted electronically on Gradescope. The code you submit should conform to the program assignment guidelines.

3.2 Demo

Part of your grade for this programming assignment will be based on your giving a short demo (5 minutes) during the CS480/680 scheduled labs following the assignment due date. You will be expected to talk about how your program works.

4. Grading

Requirements	**CS480 Credits**
Generate Triangle Meshes: Ellipsoid, Torus, and Cylinder with end caps	20
Implement EBO for defining your meshes	10
Generate normals for your meshes, and implement normal visualization	5
Illuminate your meshes with diffuse, specular, and ambient components	25
Support 3 different light types (point, infinite, spotlight)	10
Create 3 different scenes	20
Texture mapping	10(extra)
Normal mapping	10(extra)
Artist Rendering	10(extra)
Programming Style	10

5. Code Distribution Policy

You acknowledge this code is only for the course learning purpose. You should never distribute any part of this assignment, especially the completed version, to any publicly accessible website or open repository without our permission. Keeping the code in your local computer or private repository is allowed. You should never share, sell, gift, or copy the code in any format with any other person without our permission.