Programming Assignment 3

Date of announcement: 21^{st} Feb 2019
Submission deadline: 14^{th} Mar 2019

Description

In this assignment you will learn how to implement the two-pass shadow mapping algorithm, and implement multiple lights in OpenGL using programmable shaders.

Implementation Specifications - Grading Criteria

Develop an OpenGL application with the following functionality and features:

- The minimum version of OpenGL should be 3.1 and up.
- OpenGL must be used in *retained mode*.
- Include comments throughout the source code to explain each command/block of commands.
- Load the OBJ files from assignment #1 and render them with triangles
- Prepare all necessary OpenGL data structures. You should use an element buffer object to implement the vertex list.
- Create a GLFW window of size 800×800 with double buffering support.
- Render the object on display.
- The application should use a perspective view to display the object and use the depth buffer for hidden surface removal.
- The application must handle the following input (same as assignment #1):
  - the user can move the camera using the mouse i.e. moving forward/backward while left button is pressed $\rightarrow$ move into/out of the scene
  - the user can move the camera using the keyboard i.e. W moves forward, S moves backwards, A moves left, D moves right, left arrow rotates the camera left about the up vector $\rightarrow R_{up_L}$, right arrow rotates the camera right about the up vector $\rightarrow R_{up_R}$, up arrow rotates the camera upwards about the right vector $\rightarrow R_{right_U}$, down arrow rotates the camera downwards about the right vector $\rightarrow R_{right_D}$
  - the user can rotate the object using keyboard input i.e. B rotates the object about the X axis, N rotates the object about the Y axis, E rotates the object about the Z axis
  - the user can move the object using keyboard input i.e. J moves the object along the +X axis, L moves the object along the -X, I moves the object along the +Y axis, K moves the object along the -Y, PgUp moves the object along the +Z axis, PgDown moves the object along the -Z axis
  - the user can uniformly scale the object using keyboard input i.e. O scales up the object by a factor of 10%, P scales up the object by a factor of -10%
- Part A: This functionality is enabled by pressing the function key 'F1'. The scene should use multiple lights. The positions and colors for each light are given below. All lights are point-light sources.
  1. color $(0.2, 0.05, 0.05)$ at location $(10.0, 15.0, 5.0)$
  2. color $(0.05, 0.2, 0.05)$ at location $(-10.0, 15.0, 5.0)$
  3. color $(0.05, 0.05, 0.2)$ at location $(0.0, 15.0, 5.0)$
  4. color $(0.05, 0.05, 0.05)$ at location $(0.0, 0.0, 25.0)$

- Part B: This functionality is enabled by pressing the function key 'F2'. The scene contains a single light source at position $(0.0, 20.0, 10.0)$ and color $(0.8, 0.2, 0.2)$. Implement the two-pass shadow mapping algorithm using render-to-texture. Assume that the light is a spot-light source which points towards the origin of the world
- the ambient, diffuse, and specular coefficients should be $k_a = 0.25, k_d = 0.75, k_s = 1.0$ respectively (for parts A and B).

Submission (electronic submission through Moodle only)

Please create a zip file containing your C/C++ code, vertex shader, fragment shader, a readme text file (.txt). In the readme file document the features and functionality of the application, and anything else you want the grader to know i.e. control keys, keyboard/mouse shortcuts, etc.

Additional Information

- You can use the skeleton code provided during the lab sessions to get started.
- A video demonstrating the functionality is posted on YouTube: `https://youtu.be/IKPxeOzYlx8`

Evaluation Procedure

You MUST demonstrate your solution program to the lab instructor during lab hours. You will be asked to download and run your submitted code, demonstrate its full functionality and answer questions about the OpenGL programming aspects of your solution. Major marking is done on the spot during demonstration. Your code will be further checked for structure, non-plagiarism, etc. However, ONLY demonstrated submissions will receive marks. Other submissions will not be marked.