CSc 305 - Spring 2020
Introduction to Computer Graphics
Assignment 3
University of Victoria

Due: March 15, 2020 at 23:55. **Late assignments will not be accepted.**

1 Overview

This assignment covers the design and implementation of a real-time system that renders a scene with OpenGL containing a cube, a point-light, and diffuse shading on the cube. The scene is rendered through a virtual pinhole camera using perspective projection. The colour of the cube, the lights, and the position of the camera are left up to you.

2 Specification

The marks for this assignment are divided between **basic features** and **advanced features**. Your submission must implement **all** of the basic features (which are collectively worth 60% of the grade). To earn the remaining part of the grade, your submission will also implement one or more advanced features. If you do not implement all of the basic features, you will not receive any marks for advanced features. You are therefore encouraged to make sure all of the basic features are fully working before attempting any advanced features.

Basic Features

Your assignment must implement the following basic features.

- Render a scene containing a cube.
- Implement a virtual pinhole camera and perspective projection.
- Implement a point light.
- Implement diffuse shading.

The dimensions of the window are left up to you, as is the colour of the background. The success of your final render is based on whether the cube is rendered and shaded correctly and not purely on your implementation.

Advanced Features

For full marks, your assignment must also implement some of the advanced features below. Note that some features are worth more marks than others (based on their difficulty). You may implement as many advanced features as you want, but your final mark will be capped at 100%.

- **Advanced Rendering Features:** For all features in this category, you must provide a way of switching between the different render outputs without having to modify the source code. You may accomplish this using specific keys or by using a UI built with ImGUI. Live-reloading of shaders is also acceptable.
  - [1 mark] Implement a directional light. The position and direction of the light is up to you.
  - [1 mark] Implement specular reflection.
  - [2 mark] Implement specular shading with multiple light sources. The type of light sources is left entirely up to you but there must be *at least* two light sources.
- **Advanced Camera Features:** For all features in this category, you may implement these as a replacement of the default fixed camera described in the basic features.
  - [2 marks] Implement a first-person view camera. The camera must implement the following functionality:
    1. The camera needs to be able to move left and right along the **v** vector as defined for the pinhole camera.
    2. The camera needs to be able to slide forwards and backwards along the **w** vector.
    3. The camera must be able to change its pitch and yaw angles. The yaw axis will be defined as the **u** vector and the pitch axis will be the **v** vector.

    The controls for the camera are left entirely up to you. An example would be to use the arrow keys to move the camera, and the mouse to determine the yaw and pitch angles. In order to simplify this, you will need to invoke `glfwSetInputMode(window, GLFW_CURSOR, GLFW_CURSOR_DISABLED)` in order to lock the cursor at the centre of the screen.
- **Advanced Geometry Features:** For all features in this category, you must provide a way of switching between the different geometry types without having to modify the source code. You may accomplish this using specific keys or by using a UI built with ImGUI. For all features in this category, you do not have to include the mesh you used with your submission.
  - [2 marks] Load and render a simple mesh. Examples of meshes can be found on the conneX site.
  - [3 marks] Load and render Sponza. The Sponza mesh is conformed by different sets of geometry, and each must be rendered with a different colour. Be aware that Sponza is a large mesh and therefore the position of the lights and camera will have to take this into account. It is also recommended that you test this compiling your code in Release to speed up loading times.

You will be provided with an assignment bundle that will contain the following:

1. A full CMake setup for the assignment similar to the one seen in the labs,
2. a blank `main.cpp` file.
3. a blank `assignment.hpp` header.

Your submission must include the following:

1. One `main.cpp` file containing your implementation.
2. One `assignment.hpp` file containing the definitions of any auxiliary data structures, functions, etc that you may require. Note that you may leave this file blank if necessary.
3. Any and all shader files required for your submission. Note the naming convention established in the labs.
4. A `README.txt` file containing the following information:
    - Which optional features you have implemented.
    - Any controls for these features, including keyboard and mouse controls, lines of code to be commented/uncommented in the shaders, etc.

Please note that the full assignment bundle does not need to be provided in your submission, only the individual files listed above. Please submit the files individually instead of submitting a single archive. It is your responsibility to ensure that all optional features and their respective controls are clearly stated in the `README` file.

3 Evaluation

This assignment is worth 7% of your final grade and will be marked out of 10 during a live demo with a member of the teaching team. During the demo you will be asked to compile and run your code, as well as explain details about your implementation. The demos will take place during the labs in the week of March 16th and will be marked according to the following rubric.

Basic Features (6 marks)	**Marks**
Render a cube	2 marks
Use a pinhole camera with perspective projection	1 marks
Uses a point light	1 marks
Uses diffuse shading	2 marks
Advanced Features (up to 4 marks)	**Marks**
Implements a directional light	1 mark
Implements specular reflection	1 mark
Implements specular shading with multiple light sources	2 marks
Implements a first-person view camera	2 marks
Loads and renders a simple mesh	2 marks
Loads and renders Sponza	3 marks

Your implementation is expected to follow best practices for object oriented design in C++ (using a similar level of object integration as the code you have seen during the labs). If your code does not use an object-oriented design, or exhibits bad style, the evaluator may deduct up to two marks from your grade.

Your implementation must use modern OpenGL functions. If your code utilizes functions from the fixed function pipeline (OpenGL 2.2) you will receive a mark of zero. If your code uses non-DSA functions, the evaluator may deduct up to two marks from your grade.

Your code must compile and run correctly in ECS 354. If your code cannot be compiled or if it crashes during the demo, you will receive a mark of zero. If your program displays a black screen with no output, or if it presents OpenGL errors, or if your shaders do not compile correctly, you will receive a mark of zero. If you do not attend your scheduled lab, you will receive a mark of zero.

In the event that demos are not possible due to the closure of the University, the marking will be performed by using the provided `README` file and the rubric show above. In this case, if you do not submit the `README` file, you will receive a mark of zero.