Assignment 2 – CS360A

3D rendering and shading

Due date: Sep 22, 2024, 11:59pm

Grade: 100 points (10% of the course grade)

In this assignment, you will implement three different shading algorithms: (1) Flat (per-face) shading, (2) Gouraud (per-vertex) shading, and (3) Phong (per-fragment) shading. We have discussed these shading methods in detail in our class. You will have one canvas divided into three viewports, as shown below. Each viewport will have its own objects and will allow exclusive interaction using a mouse, i.e., when you are interacting with objects in one viewport, the objects in the other two viewports should stay as it is (see video for demonstration). You can use your own color for the objects but the structure and orientation of the objects in the scene should be the same. To allow interaction, you will have to keep track of your mouse click location, detect which viewport has been selected, and apply all transformations only to the selected viewport. You can consult the code in HelloIITK to see how you can create multiple viewports.

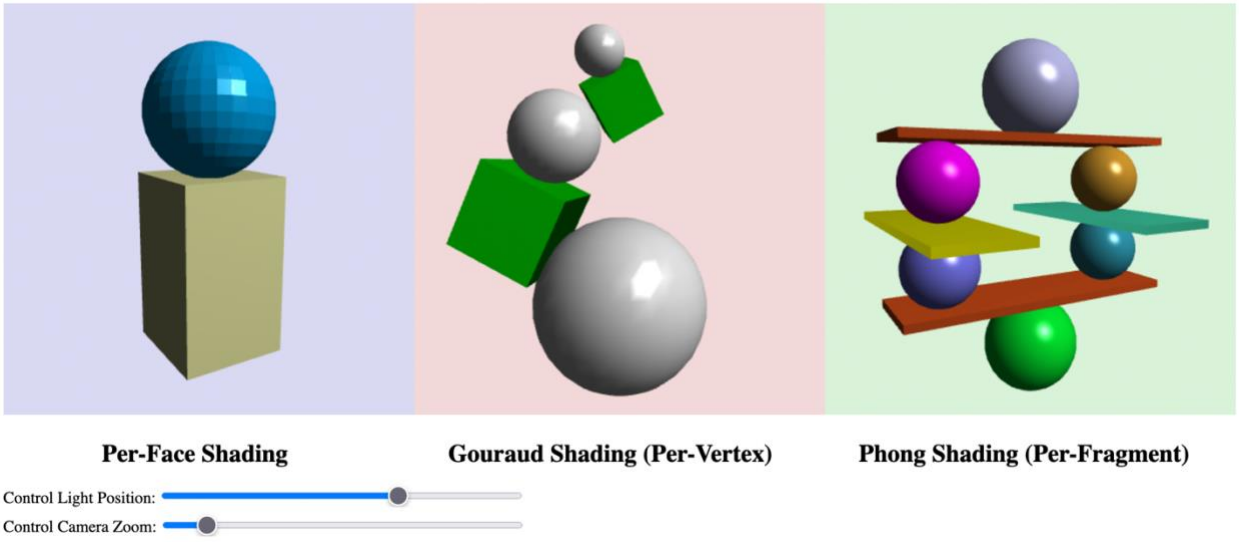
From left to right, viewport one will have flat shading, viewport two will have per-vertex shading, and viewport three will have per-fragment shading. To do these, you will have to write six shader routines- (1 vertex shader, 1 fragment shader for flat shading), (1 vertex shader, 1 fragment shader for per-vertex shading), and (1 vertex shader, 1 fragment shader for per-fragment shading). So, in your drawScene() function, you will have three viewports, and each viewport will use one of those shading methods. For this, you will have to switch between different shaders. Please check your slides where I have discussed the structure of your drawScene() function and showed you how you can switch to a different shading method.

You are going to add two sliders to your interface. Slider 1 will allow moving the light in the X-direction, and slider 2 will allow Z-axis zoom in/out operation. For zooming, you can just map the slider value to the z coordinate of your camera position to get this effect. Please check the video for the desired effect when the sliders are changed. The sample code for the slider has already been provided to you.

Functions for Sphere and Cube generation with normal vectors is provided. Copy the functions in your project. I have not given the drawCube() and drawSphere() methods. You should know how to write those and add normal to it. A version of drawCube() without the normal is given in the previous projection example code.

You will use the glMatrix-0.9.5.min.js JavaScript matrix-vector manipulation library to perform transformations and projections using the APIs provided by it. Example usage of this library is discussed in class, and sample codes are provided as references. You are not required to understand everything provided in this library. Other than the glMatrix-0.9.5.min.js library, no other additional library is allowed to be used to complete your assignment.

Here is the scene that you will create (for all interactions, see the attached video)



I suggest that you follow the divide-and-conquer strategy for this assignment. There are several pieces that are put together to form the scene:

1. Ensure you can tender both the sphere and cube and your projection is working correctly.
2. Check that you can apply transformations to them.
3. Create multiple viewports and see if you can render different objects on different viewports.
4. Implement the viewport interactions. Now, implement flat shading and apply it to make sure you are getting the correct effects.
5. Implement the other two shading models one by one and make them used in each viewport separately.
6. Add the slider 2 for camera zooming.

When all features are working fine, you can build the scenes for each viewport. It is easier to do it step by step and build one capability at a time, ensuring it is working correctly before moving on to the next effect.

Check the video to understand how the specular reflection changes based on the shading model, and you should see the same effect when your code is working correctly.

How to submit?

The **HelloIITK** portal will be set up for submission. There will be a time limit set. Please start early and finish it by the deadline. Your submission should contain three files, one main JavaScript, one HTML file, and the glMatrix.js file. Zip everything into a single compressed file and upload it. Your code should just run out-of-the-box on TA's computer without needing to do any modification. You can test it in both Chrome and Firefox before submitting it. Name your compressed file as "**Lastname_rollnumber_Assignment1.zip**", and replace lastname, rollnumber with your last name and roll number.

Grading:

We will grade your submitted version only. So DO NOT MISS THE DEADLINE, else you may get 0.