CSE 333/533 - Monsoon 2023
Assignment 3: Lighting and shading
Due date: 23:59, 13th Oct. 2023

The given code renders a cube in single color and no shading. Note that the camera can be rotated around the central object using mouse. The exercises below ask you to implement simple lighting and shading computations in OpenGL.

1. **Switch-on lighting in the given program using a point light source**
   (a) Replace the cube with any parametric surface of your choice (refer to Lab 2). Generate normals for the object. Lighting computations (e.g. in the Phong model) require normals along with vertex data. Compute per-vertex normals (remember to normalize the normals - i.e. make sure normals are unit vectors in all your computations). For a parametric surface $f(u, v) : \mathbb{R}^2 \mapsto \mathbb{R}^3$, the normal (normalized) at a point can be calculate as

$$\hat{n} = \frac{\partial f/\partial u \times \partial f/\partial v}{\|\partial f/\partial u \times \partial f/\partial v\|}.$$

   (b) Add a fixed point light source
   In programmable OpenGL, lighting computations are done entirely in the shaders. A point light source is identified by the location of the light and its color. Add these variables to your vertex shader as uniforms and pass-on certain values from your C++ program.
   (c) Use Phong lighting to render the object with specular highlights.
   Modify your vertex shader to include all three lighting components: ambient, diffuse, and specular. Use Phong exponent α = 32. Notice that the back side of the object will not be pitch dark since you have added an ambient component.
   (d) Modify your shaders to render the object using Phong shading.
   Use per-fragment Phong shading to generate a better shaded object. Again use the Phong lighting model with all three components. Any per-fragment computation has to be performed in a fragment shader, therefore you need to move shading computations from the vertex shader to the fragment shader.

[Functionality: (a) 5 marks (b) 5 marks (c) 10 marks, (d) 5 marks; Code quality and doc: 5 marks, Total: **30 marks**]

2. **Create a spot light source**
   (a) A spotlight is a light source that emits light in a cone of directions. As a result only object within the cone of illumination are lit. In OpenGL you can represent a spot light with a position, a direction, and a cutoff angle θ (half cone angle) that defines the cone of illumination. In your code, implement a spot light such that only part of the object falls inside the cone of illumination. Choose the cutoff angle appropriately (anything between 15°-30° is good). Perform all lighting calculations in the fragment shader.
   (b) You will observe that the above spot light has hard edges on the boundary of the illumination cone (where the light cone intersects with the object). Implement smooth soft edges for your spot light by creating an inner cone (with cutoff angle θ_{in}) and an outer cone (with cutoff angle θ_{out}). The position and directions of these two cones are same. Keep the θ_{in} same as cutoff angle in part (a), and θ_{out} to be

slightly more (perhaps 5°-10°) than θ_{in}. The intensity of a fragment falling in between these two cones should drop down from 1 to 0 in a smooth fashion. For an angle ϕ at a fragment, linearly interpolate the intensity by multiplying with the fraction $f = (\phi - \theta_{in})/(\theta_{out} - \theta_{in})$.

[Functionality: (a) 5 marks, (b) 10 marks; Code quality and doc: 5 marks, Total: **20 marks**]

Deliverables (as a single zipped file **Assignment03_<studentID>.zip**) containing:

- C/C++ code (make sure to upload full code and do not include any intermediate object files, delete any other temporary files).
- 2~3 page PDF Report written with with **Latex/MS Word**. Use the acmlarge option (single column) (see sample-acmlarge.tex if writing with Latex). Include screenshots within the report itself (and DO NOT attach separately).

Total marks for this assignment: 50 marks

Note: Your code should be written by you and be easy to read. You are NOT permitted to use any code that is not written by you. (Any code provided by the TA can be used with proper credits within your program).