Lab 6 - Lighting and Shading

You may work in pairs on this assignment. To receive credit, push your code to your git repository by 11:59PM on March 27th.

In this lab, you'll be implementing the Phong reflection model to shade the cube from previous labs. The Phong reflection model combines diffuse reflections which mimic rough surfaces, specular reflections which mimic shiny surfaces and ambient reflections which mimic the small amount of light that is uniformly scattered around the scene.

Part 1 - Normals

To calculate any of the lighting equations, we need to have the normal at the point we're shading. The normal is part of the underlying geometry. We need to specify the normal at each point, just like we specify position. Modify the initializeCube method to create a normal buffer, just like we're doing for position. You'll need a normal for every vertex, so I recommend copying the position array and modifying the vectors to represent the normal at each point.

Part 2 - Vertex Shader

We'll be calculating our lighting equations in our fragment shader, but we'll need access to the cube's position and normal data first. First, add a normal input variable on your vertex shader. Then, create position and normal outputs in the vertex shader and copy over the corresponding data. Be sure to transform the data into world space first. To transform normal vectors you'll need to multiply by the inverse transpose of your model matrix and normalize the resulting vector, whereas your position vectors can just be multiplied by your model matrix. If you just multiplied the normals by the model matrix, you'd see shading errors when your model matrix includes scaling.

Part 3 - Fragment Shader

a. Light

We'll need a light source if we want to shade our cube. In your fragment shader, create a uniform vec3 variable that will store our light position. Make sure to initialize it in initializeCube (a good starting value is 0,10,0).

b. Ambient

Implement ambient reflections by specifying a small constant multiplier that represents light that was scattered around the whole scene (a good starting point is .1). Multiply it by the color of your cube.

b. Diffuse

Implement diffuse reflections using the light position and the position and normals on your cube. You'll need to calculate the light direction (a unit vector that points in the direction of your light source), and then take the dot product of that vector with your normal. Clamp the dot product between 0 and 1, as we don't want any negative values. Add the resulting value to your ambient value before multiplying it by the color of your cube.

c. Specular (5 pts extra credit)

To calculate specular reflections we'll need to calculate the view direction (a unit vector that points toward the camera) and a reflection direction (a unit vector of our light direction reflected about the normal). We'll also need a shininess variable, which is a floating point or integer value. The higher the shininess value the shinier the surface (0 would be not shiny and all, 128 would be very shiny). Take the dot product of the view direction and reflection direction vectors, clamp the value to between 0 and 1 and raise it to the power of your shininess variable.

d. Camera space light (5 pts extra credit)

Try transforming your light position so the uniform variable is actually specified in camera space. Invert your view matrix to get a camera space to world space transform and use it transform your light position. Now as you move the camera around your light will always be shining on what you're looking at.

Things to notice

When the light is directly above a surface is when the diffuse shading will be brightest. Rotate the cube around and see how the cube looks from different angles. If you've implemented specular reflections, you should be able to see a specular highlight when the camera is in line with the light's reflection. When the material is very shiny, the highlight will be small, but defined, whereas when the material isn't very shiny the highlight will be larger and less present.

Recommended Reading

Phong reflection model