CS334/ECE30834: Assignment #2 - GPU it!
Basic Lighting and Normals Calculation

Out: February 7, 2022
Back/Due: February 21, 2022, 11:29 AM

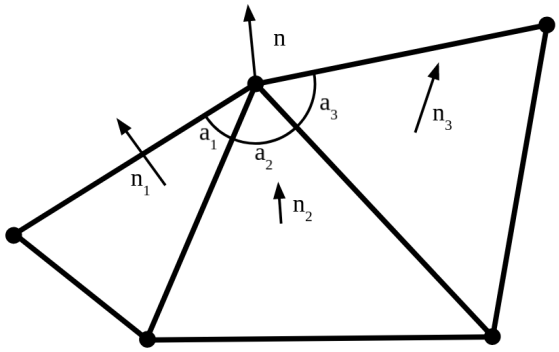
Objective:
This assignment is designed to give you some experience with GPU programming using shaders.

Summary:
For this assignment, you are given a scene with a model and several light sources. You will first have to compute surface normals, which will then be used as input to the shaders. In the shaders, you will implement the Phong lighting model using the lights in the scene and the surface normals. Finally, you will give some material properties to simulate the appearance of a few preset materials.

Specifics:

1. Start with the templates from the course website (choose FreeGLUT or Qt). The templates support Windows and Linux environments. Compile and run the templates. Both templates come with keyboard and mouse controls that allow you to rotate the view, move and rotate the light sources, change the viewing modes, and alter the light and material properties. Read the command-line output for an overview of the provided controls. Additionally, the Qt template comes with some GUI widgets that do the same thing. Upon starting the program, a config file `config.txt` is read that gives default values for the abovementioned properties. Feel free to modify this file when testing your code. You may also provide an alternative configuration file as a command-line argument.
2. **(35%) Normals Calculation.** Lighting models require knowledge of the surface normal direction in order to produce realistic effects. In previous assignments, surface normals were read directly from the mesh files, but in this assignment the mesh loader has been modified to ignore them. Instead, you will calculate surface normals from the geometry as it is loaded. Specifically, you are to calculate two types of surface normals: flat normals and smooth normals.
   Flat Normals. (Also known as face normals). Although the normal is constant for each triangle, in this implementation each vertex of a triangle stores the normalized vector that is exactly perpendicular to the triangle's plane. This can be computed using the cross product of the triangle's edges. Be sure to respect the winding order of the vertices, otherwise your normals will be backwards.
   Smooth Normals. To calculate smooth normals, each vertex will store the weighted

average of all the face normals of triangles that use that vertex. The face normals are weighed by the angle of the two edges incident on the vertex, as in the figure below.



$n = \text{normalize}(n_1a_1 + n_2a_2 + n_3a_3)$

Make your changes in `mesh.cpp` in the `load` function. Use the `Vertex` struct to store the face and smooth normals per vertex. Note that the number of vertices equals 3 * the number of triangles – in other words, vertices that are shared between several triangles are duplicated for each triangle. This is because each vertex stores a flat and a smooth normal. While the smooth normal will be the same for each duplicated vertex, the flat normal will be different. You can use the Normals shading mode when running the program to test your implementation.

Flat normals

Smooth normals

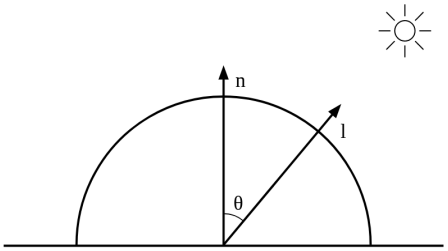
3. **(55%) Phong Lighting.** After completing normals calculation, you can begin implementing the Phong illumination model. Lighting calculations happen in the fragment shader, where you will have access to all needed lighting and material information. Each light source affects the scene with three components: ambient, diffuse, and specular, described below. Each light can be enabled or disabled. When disabled, the light should have no effect on the scene. Lights can also be either point-lights or directional lights. If a light is a directional light, then light direction does not depend on the fragment position. Lights also have an associated light color, which modulates each of the below components, and a light position, which determines the direction of the light. Additionally, the object in the scene has its own color, which is multiplied by the

contribution of each light, as well as material properties that determine the relative strength of each of the below components.

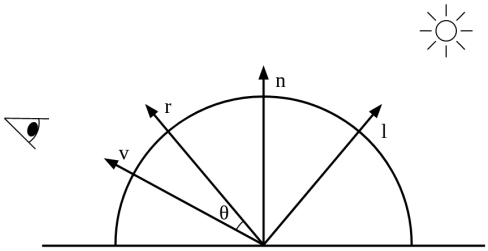
Ambient. Ambient light is an approximation of indirect lighting, where light rays bounce around the scene before striking the object. In practice, this is implemented as a simple additive term that does not depend on light or view directions. The strength of the ambient term is determined by the `ambStr` uniform, which has a value between 0 and 1.

Diffuse. Diffuse lighting approximates the amount of light that hits the surface based on the angle between the surface normal and the incident light vector. The more closely aligned the normal vector and the light direction are, the brighter the surface. This is modeled by the cosine of the angle between these two vectors. Additionally, the strength of the diffuse term is modified by the `diffStr` uniform.

Specular. Specular highlighting occurs when light is reflected off the surface directly toward the viewer. The incident light direction is reflected across the surface normal, and the cosine of the angle between the viewing direction and the reflected light direction determines the brightness of the highlight. The highlight is further modified by an exponent, `specExp`, that determines the glossiness of the object, and the strength of the specular term is given by `specStr`.



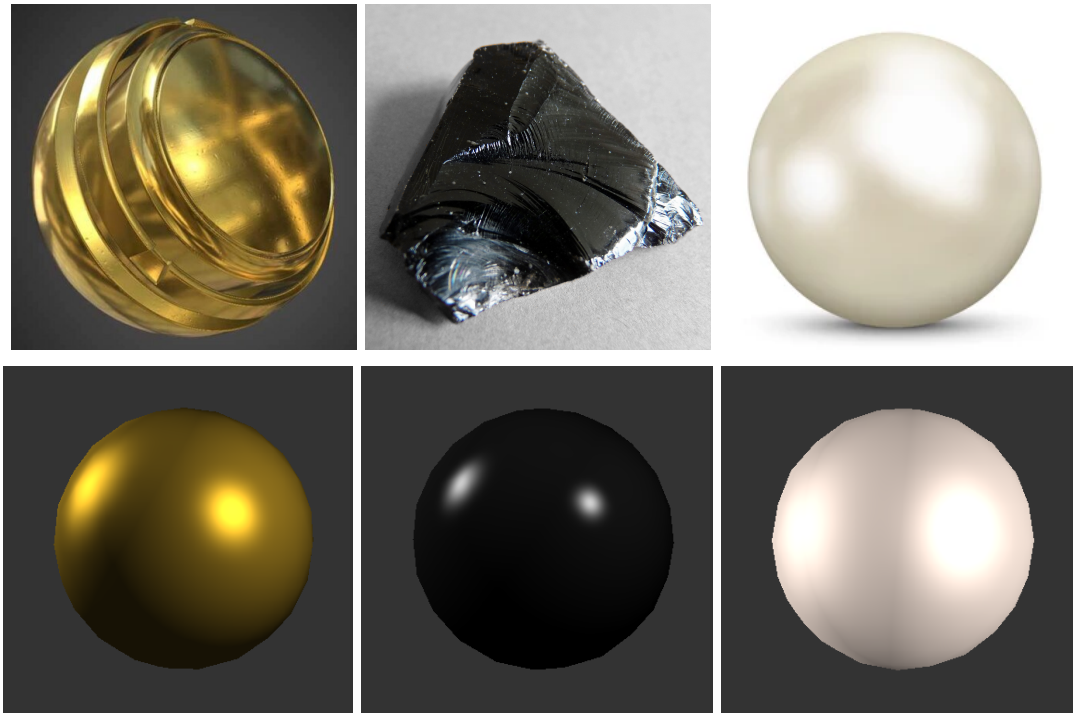
Diffuse



Specular

Write your implementation in the fragment shader, `shaders/f.glsl`. Here you have access to an array of light sources, with each light's information contained in the `LightData` struct. This data is automatically updated for you as the lights change. There are also uniforms for the camera position, object color, and other material properties for you to use.

4. **(10%) Material Presets.** After completing the Phong illumination model, you will use your implementation to determine material properties to approximate the appearance of three preset materials: gold, obsidian, and pearl. There are three config files for you to write the material property values in: `config_gold.txt`, `config_obsidian.txt`, and `config_pearl.txt`. Adjust the values to match the appearance of the below examples as closely as possible. You should set the light positions and colors as well.



Gold Obsidian Pearl

5. **(+10%) Extra credit: Gouraud Shading.** In the above Phong implementation, the vertex normals are interpolated and all lighting calculations are done for each pixel. This is referred to as Phong shading. Another type of shading, called Gouraud shading, performs all lighting computations for each vertex, and interpolates the resulting color. For extra credit, re-implement the Phong illumination model, but using Gouraud shading instead. You will have to implement this using the vertex shader, which does not have all the needed information pre-written for you. You will need to add uniforms and light data to the vertex shader, as well as an additional color output to be interpolated.
6. The functions/methods requiring your implementation will be marked "`TODO`"; however, depending on your particular implementation there might be other places for you to change code. You are expected to add/modify the code as necessary to make sure the application runs smoothly.

Turn-in:
To give in the assignment, please use Brightspace. Give in a zip file with your complete project (project files, source code, and precompiled executable). The assignment is due BEFORE class on the due date. It is your responsibility to make sure the assignment is delivered/dated before it is due. If you wish to receive confirmation of receipt, please ask by email in advance.

Don't wait until the last moment to hand in the assignment!

For grading, the program will be compiled on Linux and run from the terminal (with Visual Studio as a fallback – please try to avoid platform-specific code (e.g., don't #include <windows.h>)), run without command line arguments, and the code will be inspected. If the program does not compile, zero points will be given. If you have more questions, please ask on Piazza!

Good luck!