

## CSC 471 – Lighting /Shading – 10% final grade

**Program 3: Due Tuesday, Nov. 8th at 11:59pm.** The late policy on the syllabus applies. This programming assignment should be done **individually!** You may talk to one another about the program, but you may not look at someone's working code.

**Objectives: learn** about shading, including the difference between Gouraud, Phong and a silhouette shader. Create a program that can set up a scene with multiple meshes with different material properties on the two different meshes.

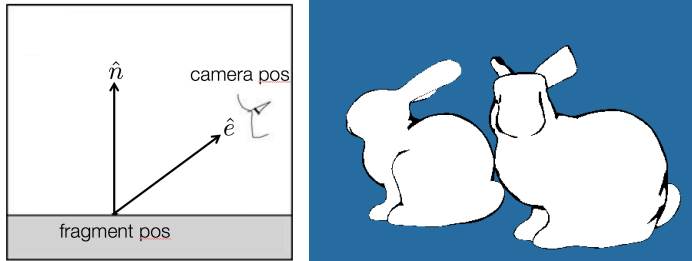
You may start with any of the prior code to read in an obj mesh (that is specified on the command line). Note that your program must compute the normals for the input mesh – even for a mesh with normals provided, you must compute the normals. **\*\*Your code will be tested on files that do not include normal, so your code must deal with this!\*\*** To complete this assignment you must:

1. Read in and display two copies of a mesh (one slightly rotated). Write code to compute normals for the mesh. Start with a normal per face and then compute the weighted average normal per vertex of all neighboring face normals. While debugging, use the shader we used for P2B, which displayed the normal of the mesh as a color. Make sure that your computed normal look reasonable! **When toggling through the shaders using the “p” key (see #4), the normal should be displayed with this original coloring.** (See below figure)
2. You will also need to add code to specify lighting in your world. Your program should have at least one light source at a reasonable position (Please have at least one light start at  $\{-2, 2, 2\}$  that is white  $\{1, 1, 1\}$ ). Allow the user to change the position of the light using the keyboard – specifically **if the user presses the ‘q’ key the light should move to the left and when the user presses the ‘e’ key the light should move to the right.** All lighting should be done via your GLSL shaders.
3. Next add support to allow the meshes to be drawn as smooth shaded (Gouraud) shading. Implement the shading by computing and setting a color per vertex in your vertex shader. Your shading should include diffuse, specular and ambient lighting. Gouraud shading should be implements in one shader pair.
4. Next implement the full Phong shading via a fragment shader, that is the normals are interpolated across the face and shading is computed per pixel. **The ‘p’ key should allow the user to toggle between the various shading models.** Be sure to toggle between the different shaders by toggling between programs on the CPU. Something along the lines of:

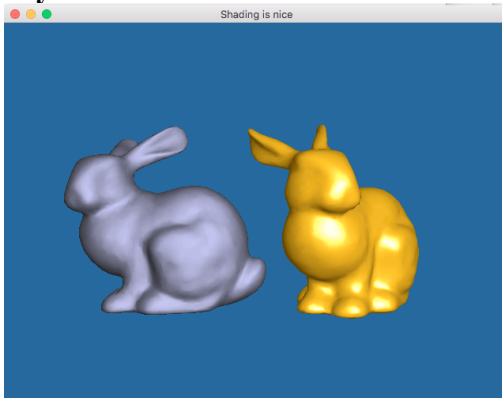
```
if(something) {
    glUseProgram(pid0);
} else {
    glUseProgram(pid1);
}
```

*For both shaders work slowly and be sure you keep all vectors in the same space!*  
No vectors used for lighting should be transformed in perspective space.

- Next, write a silhouette shader, that draws the ‘outline edges’ of the mesh. You will do this, by coloring all fragments white, except for those that have a normal that is almost perpendicular to the eye vector (see below image, which illustrates the eye vector and the output of a silhouette shader).



- You will need a way to specify different materials for the mesh (which will control its shading). Three default materials will be provided – you must include these in your implementation these as your lighting results will be compared against these. You will need to add another material that you find interesting. Provide support to toggle through all four of the different material **via the ‘m’ key**.



- Please leave the camera controls as they are
- Add basic mesh transforms such that the ‘a’ and ‘d’ keys **hooked** up to rotate the meshes around the Y axis.

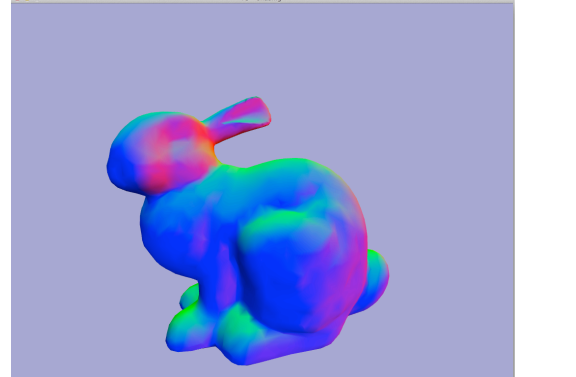
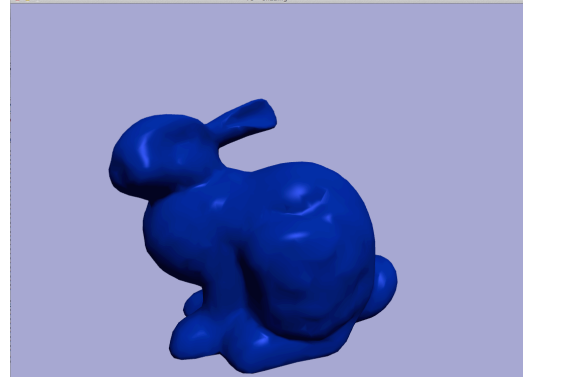
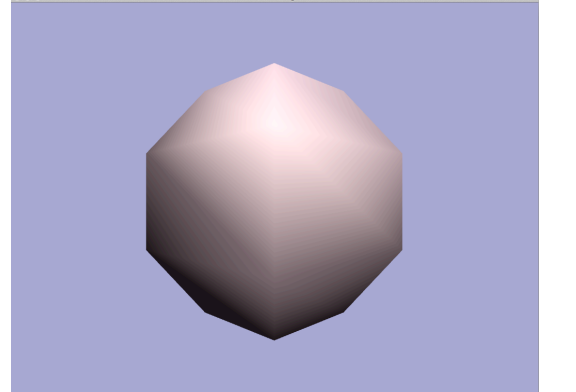
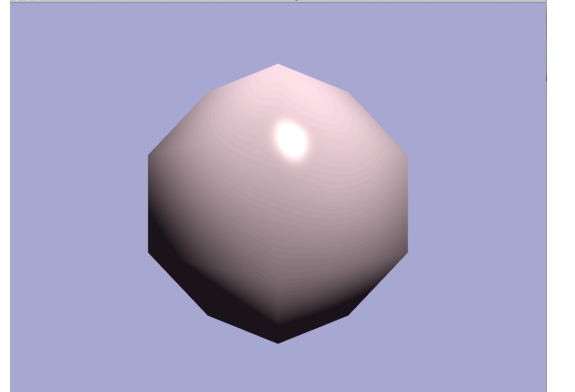
In summary, your code needs to support:

- Command line argument to load different obj files (and compute their normals)
- “a” and “d” key to rotate meshes
- “q” and “e” key to move the light position along the X axis
- “p” key to toggle between four shaders: colored normals, Gouraud, Phong, silhouette
- “m” key to toggle between materials

When not specified exactly, you must make reasonable design choices to support the necessary functionality for the program. Point breakdown (A completely functional program is worth 100 points total.):

- 10 pts for two meshes shaded with correct normal (mapped to rgb)
- 10 pts for correctly computed normal (must be computed in your program)
- 14 pts for smooth shading (Gouraud)

- 16 pts for Phong fragment shader
- 14 pts for silhouette shader
- 10 pts for translating the light
- 10 pts for support of materials
- 16 point for overall functionality (i.e. multiple shaders, rotation, key mappings etc.)

	
Example output of normal mapped to color	Example of a Phong shaded mesh
	
Example of a Gouraud shaded mesh with a different material	Example of the same mesh Phong shaded