CSC 471 - Lighting:

Program 3: Due Friday, May 10th 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 surface representations (especially indexed face sets). Learn about lighting. Learn flat and smooth shading of different object materials

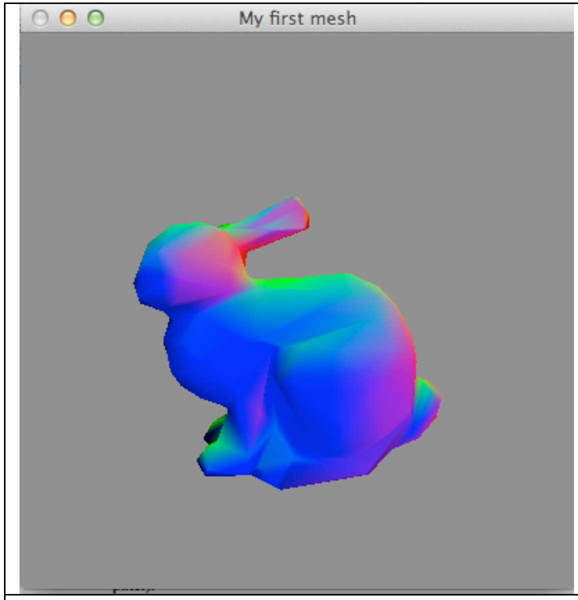
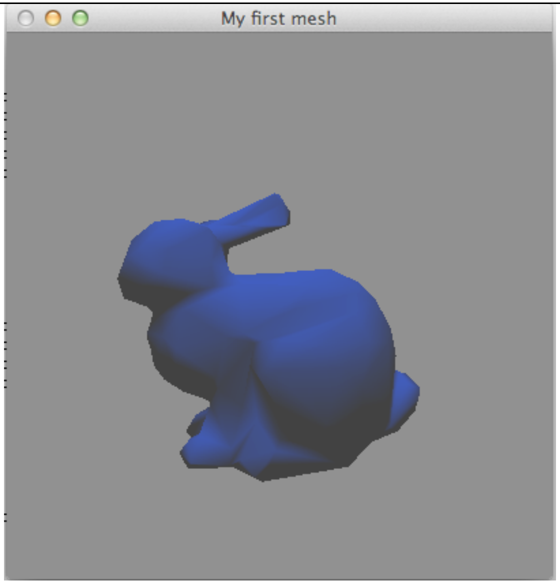
Your program will need to be able to read in a mesh file. A base program is provided that partially reads in an indexed face set (.m format) – you will need to add the functionality to store the data once it is read in and display it. This format will be described in class, but essentially the file contains a list of vertices and then a list of faces, which index the vertex list.

- The first thing you want to do is complete the parser code to not read the data but to store the data in a VBO and IBO and then display the data (don't worry if it looks 2D at this point). Be careful that your code always centers the mesh and resizes it to be a reasonable size
- Next integrate your transform code from program 2 into this new program in order for the mesh to be able to be manipulated. This means that the mesh should be able to be rotated (using the virtual trackball), translated and scaled just as in program 2. Carefully read the base code and make sure you understand the basic set up for transforms. Add a reset key "r" to bring the mesh back to the origin of the world with the original size and orientation.
- Next, you will need 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. Include a mode to visualize your normals as color data. When the user presses the 'n' key, normal should be displayed.
- 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 (for example above and to the right of the object). Please visualize the light as a small cube in your world. Allow the user to change lighting parameters (e.g. position or color). As a minimum, use a point light source and allow the user to change the position of the light (using the keyboard – specifically if the user presses the 'z' key the light should move to the left and when the user presses the 'x' key the light should move to the right). All lighting should be done via your GLSL shaders.
- Next add support to allow the mesh to be drawn as smooth shaded (Gouraud) shading. Implement the shading by computing and setting a color per vertex in your vertex shader.
- Next implement a fragment shader which is a full Phong shader implementation (That is the normals are interpolated across the face and shading is computed per pixel).
- You will need a way to specify different materials for the mesh (which will control its shading). At minimum allow the user to select between two different material properties for the mesh from a menu. By default one of your materials

must be a shiny blue material – specifically a material with the following parameters: diffuse {0, .08, .5}, specular {.4, .4, .4}, ambient {.2, .2, .2}, shininess = 200

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

15 pts for full mesh display
30 pts for smooth shading (gouraud)
20 pts for Phong fragment shader
15 pts for translating the light
20 point for overall functionality (i.e. lighting working under transformations, and viewing changes, etc.)

	
Example output of normal mapped to color	Example of a diffuse shaded mesh