

**CSC 471: Program 3: Due Tuesday May 12th, 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 in OpenGL. Learn flat and smooth shading of different object materials. Learn about projection.

- Your program will need to be able to read in a mesh file. I have provided you with a working parser for the mesh format we will be using. This program reads in a file from the command line and then displays it as a wireframe mesh. We will be using a simplified version of an indexed face set for our mesh format. 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. Please download MeshParser\_release3.cpp (from the class webpage) and compile it (note that the program expects to be run from the command line with one argument which is the filename of the mesh to load – leave this as it is, as I will want to test your program on different files). In addition, you can download all or one of the example meshes provided in the data directory (delete the “csc471.html” part of the url and replace it with the word “data”, which is a directory, I recommend starting with bunny500.m). You will need to augment this base code for your assignment. Note that the code also draws a glut wireframe sphere in the scene. Leave this in the scene, as it will allow you to see the difference between orthographic and perspective viewing.
- The first thing you want to do is integrate your transform code from program 2 into this new program in order for the mesh to be able to be manipulated. (Leave the sphere where it is, you do not need to allow it 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 respect the transforms in place to center the mesh at the origin and scale it to a reasonable size. 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 want to "flesh out" the wireframe mesh and sphere to draw solid colored polygonal objects.
- Next, you will want 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). 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 a keyboard event or mouse event). For light sources, you can restrict transformations to translations (if point source) and also rotation (if directional lighting). You may allow the user to select which object they want to transform (mesh or light source) from a menu. We will be discussing lighting in class, some relevant calls are: `glLight()` and be sure to turn them on with `glEnable(GL_LIGHTING)` and also turn on the specific light source as well. Code examples will be provided.

- Next, you will want to specify different colors for the mesh. Note that enabling lighting in OpenGL, disables any glColor calls, thus you will need to use “materials” to set colors for the mesh. At minimum allow the user to select between two different material properties for the mesh from a menu. In addition, allow the user to toggle from wireframe, to flat or constant (one color per polygon) shading, to smooth (Gouraud) shading. Have the drawing mode toggle (wireframe/flat/smooth) be a menu option. One relevant call is glMaterial().
- Next allow the projection to toggle between orthographic and perspective projection by pressing the “v” key on the keyboard. Relevant calls are: gluPerspective() and gluLookAt().

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.):

5 pts filled in mesh

5 pts for material colors for the mesh

10 pts for toggling between orthographic and perspective viewing

15 pts for flat shading

15 pts for smooth shading

10 pts for translating the light

30 pts for correct object transformations (16 for rotation, 7 for translation, 7 for scaling)

10 point for overall functionality (i.e. lighting working under transformations, and viewing changes, etc.)