

**Fall 2009**  
**22C:151 Introduction to Computer Graphics**  
**Assignment 6**

**Due: Monday October 19th at 11:59pm**

*Note: I'm giving you extra time to help deal with midterms!*

**Goal:** Load a complex triangle mesh into your OpenGL program, add a user interface for rotating the object, and illuminate the object using standard Phong illumination.

**Problem 1 (30 points):** Write a OpenGL/GLUT program which renders 3D geometry into a  $640 \times 480$  window. Use the whole window as the viewport (i.e., a  $640 \times 480$  viewing area). Post sample images from each part of the problem on your webpage.

- **Part A (4 points):** Load a triangle mesh into your program. The easiest way to do this is to use Nate Robins' `glm.c` and `glm.h` (which are posted on the web page) to read in OBJ files. Read the header file! Sample geometry is available on the web page. Render an object at the origin using a single color (i.e., no lighting). GLM should be able to do this for you. Set the viewpoint to something sensible, so you can see the object.
  - Please do *NOT* reload your model(s) every frame! Do *NOT* put `glmReadOBJ()` in your `display()` routine! If your program is slower than about 50 frames per second, you are reloading every frame. Doing so will lose all 4 points in Part A.
- **Part B (4 points):** Add a virtual trackball (attached to the left mouse button) that allows you to rotate your object using mouse input. You may use the trackball code posted on the web page. The idea of a virtual trackball is that different motions on the 2D screen get mapped to rotations about different axes in 3D.
- **Part C (4 points):** Add a GLUT menu, which allows you to choose between displaying (at least) the Al Capone, Stanford Dragon, and Utah Teapot models. See the GLUT specification on the web page for information.
- **Part D (8 points):** Add a white light to the scene and enable OpenGL's Phong lighting. Set the light's ambient color to (0.2, 0.2, 0.2, 1.0) and the diffuse and specular to (1.0, 1.0, 1.0, 1.0).
- **Part E (2 points):** Make sure to enable depth testing so you only see the front of the object.
- **Part F (3 points):** Render the Al Capone object using the materials defined in the 'al.mtl' file. GLM will automatically load this file (when you read the OBJ file), and should allow you easily render this way (use `GLM_SMOOTH | GLM_MATERIAL`). You should look at the `glm.c` code to understand what is happening.
- **Part G (2 points):** Add a keyboard callback or menu entries to allow the user to change the material type of the object. Material properties for approximating some common materials are available on the class web page.
- **Part H (3 points):** Add another virtual trackball (attached to the middle mouse button) that rotates the light around the object.

**NOTE:** A "README" file is required in order to get full credit! It is worth the 2 or 3 minutes it takes to write such a file, as it guarantees we know how to compile and run your program, and you can make note of any odd behavior or strange bugs (which may mask required functionality).

**NOTE:** A sample executable is available on the web page for you to see my expectations for this homework. The trackball code I used in the example is different (i.e., broken) than the version on the class page, so your motion should be more intuitive.