

## CS 770/870

### Assignment 3: Basic 3D Viewing and Modeling

September 29, 2008

**Due: Monday, 10/13 by 23:59:59**

**Last allowed submission: 10/17 23:59:59**

The purpose of this assignment is to give you practice using basic OpenGL 3D modeling and viewing features.

1. **3D Viewing.** Implement some *glui* widgets to allow the user to interactively modify the 3D OpenGL viewing parameters. The user should be able to switch between perspective and parallel projection and change all the parameters for *gluLookAt*, *glOrtho*, *glFrustum* and *glPerspective*. You can test your viewing without implementing the 3D modeling part of the assignment by using objects from the *GlutSolids* classes.
2. **3D shapes.** You should extend/modify our abstract *Shape* class to support 3D transformations. You should implement at least 3 subclasses that include 3D components. Chapter 6 in the text shows lots of examples of pre-defined 3D models that can be used in the OpenGL world. One possible goal of a *Shape* child is to wrap an OO class around one of the *glu* objects. However, you should have at least one class that has multiple subcomponents. At least one of your *Shape* classes should contain at least one component that is an instance of one of your other *Shape* classes.
3. **Predefined scene.** At start up, the program should invoke code that displays a predefined scene that contains at least 2 instances of each of your predefined shapes, and could contain *GlutSolid* objects..

#### Point allocation

- 10 Predefined scene with at least something interesting; nested solids and/or *GlutSolids*
- 25 *glui* widgets for parallel projections – and correct results.
- 15 additional *glui* widgets for perspective projection – and correct results
- 25 your own creations of 3D objects; 3 subclasses of solids (need not be hierarchical)
- 15 nested solid definitions

Features that might qualify for the last 10 points include copying or deletion of shape instances, defining new scenes, and/or other features.

#### Note:

Points are earned by correctly implementing features robustly. Points are deducted for bugs, incorrect implementation, poor style, poor design decisions, etc.