

# CS 770/870

## Assignment 2: Transformations, GUIs, & Commands

September 15, 2008

**Due: Monday, 9/29 by 23:59:59**

**Last allowed submission: 10/3 23:59:59**

The purpose of this assignment is to give you practice using OpenGL transformations in a simple object-oriented context, as well as give you a brief introduction to using GLUT and supporting commands in a simple graphics application. Using the framework suggested by the demo program, you need to implement a program with the following features:

1. **Transformable shapes.** Your program should support at least 3 classes representing simple shapes. These could be the same classes that you used for Assignment 1, except your implementation for this assignment should define all the coordinates in such a way that they are relative to the center of the shape. In other words, if the user adds an instance of the shape at location (0,0) in the window, the shape will have its center at (0,0) Each instance of each shape must have its own unique name, and must support the following commands:
  - **Rotate to** *deg* degrees from the orientation specified in the shape's original definition, with positive values being clockwise. Rotations should cause the object to spin about its center.
  - **Translate to** *tx* pixels horizontally and *ty* pixels vertically from the middle of the window, with positive directions to the right and up. Translations should specify the new location of the object's center.
  - **Scale to** a scale factor of *sx* horizontally and *sy* vertically from the original size. A scale should apply in the reference frame of the object—horizontal and vertical with respect to the original orientation of the object, as opposed to the orientation it has after rotation.

The results of the commands should be visible on the next refresh cycle.

2. **GLUI window.** It should include the following components:
  - **List of object names.** This list must contain the names of all objects that appear on the GLUT drawing window. Selecting one of these names should designate that object as the *current* object until a different name is selected from the list. Your program should draw a rectangle around the current shape; it would make sense for this to be a new method the Shape class to “highlight” the object.
  - **A group of scrollbars to control drawing.** Each scrollbar should be labeled, and should be logically attached to a parameter of each of the commands listed in part one: *deg*, *tx*, *ty*, *sx*, and *sy*. When one of the scrollbars is moved, it should modify the associated attribute of the *current* object (selected in the list of object names).
3. **File input.** You should be able to read a file containing a simple command language for defining a *scene* of objects. The file name should be specified on the command line. The following commands are required.

Command	Arguments	Notes
d	type name	Define an instance of class <i>type</i> with specified <i>name</i> . Both arguments are strings. You do not have to use the class name as the <i>type</i> , an abbreviation is fine. Whatever you put in your menu should be legal here.
t	name x y	Translate the location of the object with name, <i>name</i>
s	name sx sy	Scale the named object (with respect to its definition) by <i>sx</i> and <i>sy</i>
r	name alpha	Rotate the named object (with respect to its definition) by <i>alpha</i> degrees

Comments can be included. They start with ‘#’ and continue to the end of the line. Blank lines should be allowed.

4. **Predefined scene.** If no input file is specified, or if the specified file name does not exist, the program should invoke code that displays a predefined scene that contains at least 2 instances of your predefined shapes..

**Point allocation**

- 10 Predefined scene with at least 2 instances of each shape.
- 15 Select an object from the scene as the current object and “highlight” it.
- 15 Translate. 5 points for non-interactive translation, 10 for interactive.
- 20 Scale. 5 points for non-interactive; 15 for interactive.
- 20 Rotate. 5 points for non-interactive; 15 for interactive.
- 15 Read a scene from a file

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.