CS4204 Computer Graphics Spring 2010
Project 1 – 2D Modeling Tool

Due Dates
Project 1 is due on Tuesday, 2/16/2010 11:59pm.

Introduction
You need to write an OpenGL program which can be used to model 2D Objects. The project will test your skills on 2D primitives, transformations, interactions and basic GUI design using OpenGL and GLUT.

You need also use your 2D modeling tool to model a nice 2D scene with as least ten 2D objects. You need to capture the screen of the result of your 2D scene and submit it as a JPG file.

This project is designed to be finished by **one person team** only.

Your 2D modeling tool should include all the following features described below.

Features of the program
Your program must contain all of the following features:

For drawing (35 points)
- Can draw triangles, rectangles, circles, polygons with arbitrary number of edges. You may have menu items include all kinds of shapes you want to draw. By selecting one of menu item, you can draw it out.
- When drawing triangles, and polygons, you need to specify the position of each vertex using mouse.
- When drawing rectangles, you only need to specify two vertices.
- When drawing circles, you need to specify the square that bounds the circle you are drawing.

For transforming (35 points)
- You should be able to select the primitives you draw on the screen. When selected, all the vertices of the primitive should be highlighted. (For circle, the bounding square should be highlighted.)
- You should be able to edit the position of each vertex of selected primitives using mouse. (When applied to circle and ellipse, it means the vertices for bounding square or rectangle).
- You should be able to translate, rotate, and scale the selected primitives. Be careful about rotate and scale, because these two operations need a pivot point.

For Coloring (**Optional, bonus feature**) (10 points)
- Your program should be able to change the filling color for the selected primitive.

What to Submit
Put your solution in one or more C++ source files. The main file (which includes function `main()`) should be named `project1.cpp`. Send all source files in a zip file to your TA via email tozammel@vt.edu. Please also include a description file, called "`descriptions.txt`" that describes how to use your program.

Also submit a JPG file, which shows the final result of your edited scene with 2D objects, as described in the introduction section.

Grading

Your overall grade will be based on the following:

- (70 points) Completion of the feature list.
- (10 points)How easy to use the GUI.
- (10 points) How complex and nice (in art sense) your final 2D scene is.
- (10 points) How good was your descriptions.txt file?

Hints and References

Please read carefully about GLUT tutorials here: http://www.lighthouse3d.com/opengl/glut/.

You can find the algorithms for circle drawing here:

Rendering a simple circle...

The solution for drawing a circle is simple: draw a series of connected lines of tangency, around the radius of the circle. If you use really short lines, and lots of them, it looks exactly like a "real" circle. The trick is, to use just enough lines to make a nice, round circle.. If you use too many lines, the rendering is slow. Use too few lines, and the rendering is faster, but the circle looks pixelated, or turns into an obvious polygon.

```
/* draw a circle from a bunch of short lines */
vectorY1=originY+radius;
vectorX1=originX;
glBegin(GL_LINE_STRIP);
for(angle=0.0f;angle<=(2.0f*3.14159);angle+=0.01f)
{
vectorX=originX+(radius*(float)sin((double)angle));
vectorY=originY+(radius*(float)cos((double)angle));
glVertex2d(vectorX1,vectorY1);
vectorY1=vectorY;
vectorX1=vectorX;
}
glEnd();
```

Rendering a solid circle...

Now, a similar problem occurs when trying to render a solid circle. We could just render a bunch of lines from the center of the circle to its edge to create a solid disc, but that would be way too slow. The faster way to do it, is to render the circle as a bunch of solid triangles sharing a common vertex at the center of the circle. Once again, the more triangles you use, the better the quality of the circle gets, and the fewer you use, the faster it will render.

Here is the code for a solid circle..

```
/* draw a solid disc from a bunch of triangles */

vectorY1=originY;
vectorX1=originX;
```

```
glBegin(GL_TRIANGLES);
for(i=0;i<=360;i++)
{
angle=(float)(((double)i)/57.29577957795135);
vectorX=originX+(radius*(float)sin((double)angle));
vectorY=originY+(radius*(float)cos((double)angle));
glVertex2d(originX,originY);
glVertex2d(vectorX1,vectorY1);
glVertex2d(vectorX,vectorY);
vectorY1=vectorY;
vectorX1=vectorX;
}
glEnd();
```