Due Monday, Sept. 30, 2009 (Demos after class)

This project will extend your existing application to support **user defined world coordinate systems**. All of your drawing functionality will remain the same, however, **your model will store all coordinate information in world coordinates**.

You will extend your application from project 1-2 as follows:

1. You will provide an option for the user to specify a world coordinate system, that will specify a 2D coordinate window: \((x_{\text{Min}}, x_{\text{Max}}, y_{\text{Min}}, y_{\text{Max}})\). \((x_{\text{Min}}, y_{\text{Min}})\) represents the lower left corner and \((x_{\text{Max}}, y_{\text{Max}})\) represents the upper right corner. Use a dialog box to input the values.

2. You will write 2 modules to convert device coordinates into world coordinates and from world to device coordinates. Both are rectangle to rectangle transformations (**refer to text, section 6-3**). Note that currently all model primitives are in device coordinates, and your device rectangle is based on the canvas size.

3. Objects drawn on the canvas are received by events in device coordinates. These are then converted into the current world coordinates. Thus, the entire object model is now retained in world coordinates.

4. During a redraw, objects from the model in world coordinates are transformed into device coordinates and then drawn by OpenGL.

**General Requirements**

- Implementation in C++ or Java.
- Extend the pull down menus to support the new drawing primitives and operations.
- All drawing must be done using OpenGL, interactively (mouse, keyboard)
- **Documentation**: Your program must be well documented - all functions must have some documentation.
- **Evaluation**: An interactive demo of the project by the due date (usually after class) in 335. Also turn in hardcopies of your sources