CS 446/446G Project 1: Interactive Interface and 2D Transformations

(due on the class on 02/27/17)

This project has two parts. In the first part, you will start to learn the basic of OpenGL programming, including

1. The basic of drawing OpenGL geometry primitives
2. Interaction with keyboard
3. Interaction with mouse
4. Simple transformation between two different coordinate systems

In the second part, you will learn

1. how to use popup GLUT menu
2. how to perform 2D transformation on a 2D object

1 Part I: Simple interactive drawing

Specifically, you will develop an OpenGL program that allow a user to draw a circle in a position of the OpenGL window by clicking the left button of a mouse. Furthermore, your program should allow the user to draw multiple circles. When the user is done with drawing, he/she can terminate the program by entering the 'q' or 'Q'.

Your code will need to maintain a data structure to record the position of each left-click of mouse. For simplicity, you can use a 2-dimensional array, and assume that the maximum number of left-click is 100. Each time, when there is a GLUT_LEFT_BUTTON event, your code should insert the current mouse position to the 2-dimensional array.

There are two different coordinate systems in this simple drawing. The mouse position is under the matrix coordinate system, and a vertex for drawing OpenGL geometry primitive is under the geometry coordinate system. So your code should transform the mouse position from matrix coordinate to geometry coordinate. For simplicity, you can assume the height of the window is fixed.

Additional requirement on CS446G

Graduate students should further allow users to erase a circle by clicking the right-button of a mouse inside a drawn circle. So when there is a GLUT_RIGHT_BUTTON event, your code should perform a linear retrieval of the array of the left-click points to test whether the right-click point is inside the circle of a certain left-click points. If there is such a point, it should be removed from the array, then the rest of elements should be moved forward to the hole. (A more efficient data structure is the list. If you are familiar with C programming, you are encouraged to use list data structure to record the left-click points.) Since it is possible that the drawn circles are overlapping and the right-click points are inside multiple circles, your code should search through the entire array and remove all "relevant" left-click points.

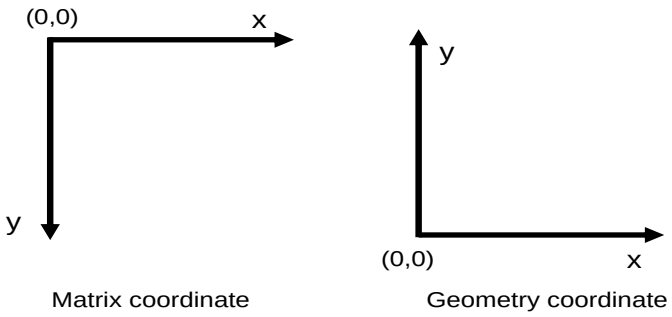

Figure 1: Two coordindate systems

2 Part II: 2D Transformation

You will develop an OpenGL program that allow a user to perform 2D affine transformation, including translation, scaling, rotation, on a 2D object (e.g., a solid square), by selecting an transformation item from the popup menu. In this project, you can have two-layer menu as follows:

- Translation
  - Horizontal increment
  - Horizontal decrement
  - Vertical increment
  - Vertical decrement
- Scaling
  - Horizontal enlarging
  - Horizontal shrinking
  - Vertical enlarging
  - Vertical shrinking
- Rotation
  - Clockwise
  - Counter-clockwise

Your code will need to maintain several geometry parameters of an object, including horizontal and vertical translation, orientation, and scaling factors. You code should include a function (called **menu**), in which you should update the geometry parameters of the object.

The object should be "complicated", i.e., a composition of at least 2 OpenGL geometry primitives.

The following parameters should be used for an individual 2D transformation:

- 5 pixels for horizontal increment/decrement

- 2 for horizontal enlarging, and 0.5 for horizontal shrinking
- 10° for a rotation.

Additional requirement on CS446G

1. CS446G students should explicitly construct a transformation matrix in order to perform a specified 2D transformation to the object. The transformation matrix is represented as a one-dimensional array in OpenGL that is used as a parameter for the OpenGL function glMultMatrix() to update the **current** transformation matrix.
2. CS446G students should also implement a simulation of moving the object. Specifically, the object is moved around a circle track clockwise and be back to its original position. So it is basically a task of performing a sequence of translations on the object.

On submission: Submit the source code and readme to blackboard. The source code should be well-documented, i.e., having good readability. The readme file should summarize: i) tasks that you completed, and ii) tasks that you did not complete.