CSCI 550 (Fall 2009)

Assignment #1

Handout: Tuesday, Sept. 15, 2009
Due: 3:00pm, Tuesday, Oct. 6, 2009

All assignments will be submitted through oncourse.
The non-programming part can be in any format such as Microsoft Word, PDF, plain text or scanned hand-writing.
The programming part should include: (1) all source code files; (2) the executable file (.exe); and (3) a README file containing necessary instructions, known bugs, and any other notes you would like me to read.

Part 1: non-programming problems (*50 pts*)

1. What is a "client-server" relationship? Give an example of a common client-server relationship (not a graphics example).

2. What is the GLUT reshape callback? Why is the reshape callback usually related to mouse handling?

3. Suppose an RGB raster system is to be designed using a 20-in by 15-in screen with a resolution of 100 pixels per inch in each direction. If we want to store 24 bits per pixel in the frame buffer:

   (a) How much storage (in bytes) do we need for the frame buffer ?

   (b) How many distinct colors can be represented using this frame buffer?

   (c) If a 100 Hz frame refreshing rate is required, what is the minimum memory access speed (bytes per second) for the frame buffer ?

4. Show two different ways of defining a cone surface using OpenGL primitives (show your solutions in the form of OpneGL commands).

5. Draw the graphical output if the following four points were given, in the that order, to the primitives (1) GL_LINES, (2) GL_LINE_LOOP, (3) GL_TRIANGLES, and (4) GL_TRIANGLESTRIP:

$$P_1 = (0, 0); P_2 = (0, 1); P_3 = (1, 1); P_4 = (1, 0)$$

Part 2: Programming problem: Interactive polyline drawing (*50 pts*)

In this project, you will write an interactive 2D polyline (line strip) drawing program. The requirements include:

1. Interactively define and draw the polyline, i.e. user will use the mouse to click points on the screen to define a sequence of vertices of the polyline, and the program should draw the current polyline every time a new vertex is defined.

2. Allow the user to enable (using menu or keyboard) the "rubberbanding" effect. When "rubberbanding" is enabled, the motion callback function will be used in defining each vertex, i.e. letting the user to drag the point (and the connected polyline edges) around.

3. Define a menu and corresponding keyboard events to execute and control the functionalities of your program. Required functionalities include: color, line thickness, rubberbanding, clear, and exit.

4. Allow window resizing. You are required to preserve the aspect ratio of your polyline, i.e. when the window is resized, the ratio of shape of the polyline should not change and the entire polyline is properly displayed within the window (i.e. no cutting).

5. Picking/editing/deletion. Your program needs to be able to pick a point on the existing polyline and move it to a new location (using rubberbanding) or delete the picked point.