Middle East Technical University
Department of Computer Engineering

CENG 477
Introduction to Computer Graphics

Fall 2007 - 2008
Warmup Homework

REGULATIONS

Due date: 17.10.2006 Wednesday 23:59

Late Submission: Late submission is not allowed.

Submission: Use cow.ceng.metu.edu.tr to submit your homework file. Each group will submit only one copy. ***Submission details will be announced in the newsgroup.***

Team: The homework has to be done by project groups of exactly 2 partners.

Evaluation: Your homework will be evaluated in a Linux environment. Before submission, you have to test your homework at one of the department's computers using Linux to have a common platform.

Cheating: In case of cheating, all parts involved (source(s) and receiver(s)) get zero.

Newsgroup: You **must** follow the newsgroup for discussions and possible updates.

MAIN THEME - YASE

In this warmup homework, you will implement YASE – Yet Another Simple Editor. YASE is used to create a set of points and then some geometric primitives are automatically drawn using these points.

GENERAL SPECIFICATIONS

- For **only** this homework, a **prototype** is provided. You can refer to this prototype for appearance and functionality of the homework. If you think that there is anything unclear or doubtful in the prototype, please ask the related issue in the course newsgroup.
- Create a window of size 640 * 480.
- You must initialize OpenGL using the following statements:

```
glViewport(0,0,640,480);
glMatrixMode(GL_PROJECTION);
glLoadIdentity();
glOrtho(-320,320,-240,240,-10,10);
glMatrixMode(GL_MODELVIEW);
glLoadIdentity();
```

- [Hint] As the OpenGL coordinate system stated above by glOrtho(…) and GLUT's coordinates for callback functions are different, for consistency you have to adjust *x* and *y* values coming from GLUT's callback functions:

  x = x - 320;
  y = 240 - y;
- You will prepare a user interface using GLUI in a standalone window. The user interface window must lie just at the right of the main window starting from the same height.
- Some functionality can be accomplished by more than one method (i.e. pressing a button in the user interface or pressing a key in the keyboard). In this warmup for such situations, you are expected to implement all such methods. Also keep in mind that GLUI user interface and keyboard keys must be used in parallel. In other words you also need to change the value in a checkbox if it is changed by pressing a key in the keyboard.
- All single characters stated inside parenthesis in the prototype define secondary methods for related functionalities (i.e. N means "New File"). These characters are case sensitive.
- [Hint] For rendering any text on the screen, you can use the *glutBitmapCharacter* method. The following example renders the character string to the position (*x*,*y*) on the screen.

```
void RenderText(int x, int y, char *string){
        int len, i;
        glRasterPos2f(x, y);
        len = (int) strlen(string);
        for (i = 0; i < len; i++)
                glutBitmapCharacter( GLUT_BITMAP_HELVETICA_18, string[i]);
}
```

- It is guaranteed that the main window will not be resized.
- Remember that for using the keyboard events defined below, you may need that your main window (but not the GLUI control window) must be the currently active window.
- Logging is a critical issue that you have to consider for a graphics program. For this warmup homework, details are given in the "Logging section".

FUNCTIONALITY

- At the top of the control window that you have created with GLUI, write project members' IDs.
- **Point Operations:**
  - **Add Point:** By pressing the left mouse button in the main window, a new point is added in the position of the mouse pointer. ([Be Careful] This event must be accomplished when the left mouse button is pressed not when it is released.)
  - **Delete Point:** When right mouse button is pressed, a popup menu is displayed. When "Delete Point" is selected, the last point in the point set will be deleted.
  - **Select and Move Point:** When right mouse button is pressed, a popup menu is displayed. When "Select and Move Point" is selected, this functionality will be active. Then whenever the middle mouse button is pressed on a point and the mouse pointer is dragged without releasing the middle mouse button, the point which is pressed will be selected and moved according to the mouse pointer.

        - To decide on which point is pressed, assume each point is just a square of size 12 pixels centered on the position of the point (Actually this is a bounding box definition). You just need to test whether the coordinates given by GLUT are inside this square or not.
    - [Hint] The popup menu is created by GLUT. You can find more information in the "Menu Management" section of the GLUT manual.

- **File Operations:**
    - **New File(N):** Clears the screen, discards the current set of points.
    - **Load File(L):** Loads the set of points which is stored at "points.txt"
    - **Save File(S):** Saves the set of points into "points.txt".
    - points.txt has the following format: (n is integer, others are float)

      <n: number of points>
      <x coordinate 1> <y coordinate 1> <point size 1>
      <x coordinate 2> <y coordinate 2> <point size 2>
      ….
      <x coordinate n> <y coordinate n> <point size n>

- **Drawing Operations:**
    - **Drawing Primitives:** According to the state of the radio buttons, one of the followings is used for drawing. The same functionality can be accomplished by 1, 2, 3 from keyboard.
        - Polygon (1)
        - Triangle Fan (2)
        - Triangle Strip (3)
    - **Drawing Mode:** According to state of the checkbox, following are displayed or not. The same functionality can be accomplished by 4, 5, 6 from keyboard.
        - Toggle Points (4)
        - Toggle Lines (5)
        - Toggle Geometry - Drawing Primitive (6)
    - **Change Color Schema (7):** When the related button is pressed, the colors of the elements in the display will be switched between coloring schemas. Details of the coloring schemas are given in the below table. The same functionality can be accomplished by 7 from keyboard. ([Hint] Use *glColor3f(red, green, blue);*)

Coloring Issue	**Color Schema 1 (Default)**			**Color Schema 2**		
Points and their labels	R: 0.3f	G: 0.3f	B: 0.3f	R: 0.0f	G: 0.0f	B: 0.7f
Lines	R: 0.4f	G: 0.0f	B: 0.9f	R: 1.0f	G: 0.3f	B: 0.0f
Geometric Primitives	R: 0.4f	G: 1.0f	B: 0.4f	R: 0.4f	G: 0.7f	B: 1.0f

- **Detailed Point Specifications:**
    - Number of points will not exceed 20.
    - Each point is generated by GL_POINTS in OpenGL. (Thus position of a point is its center.)

- Each point can have a different size.
- From the user interface, point size is can be adjusted by using a spinner.
- Point Size can be at least 8 and at most 12. Default point size is 10.
- [Hint] In the code, you need to use *glPointSize*.
- [Hint] You need to use *glEnable(GL_POINT_SMOOTH);* in order to have more smooth points. As OpenGL is a state machine, it is enough that you call this function only once in the most appropriate place; when initializing OpenGL.
- On top of each point you need to display the label of the point. Points are labeled by a single number which determined according to order of creation.
- For simplicity, assume that the points will never intersect.

- **Detailed Drawing Primitive Specifications:**
  - You will implement the drawing primitives by using GL_POLYGON, GL_TRIANGLE_FAN and GL_TRIANGLE_STRIP primitive types, respectively.
  - Assume that the current point set is always appropriate for the current drawing primitive. In other words, there is no violation on the OpenGL specifications of GL_POLYGON, GL_TRIANGLE_FAN and GL_TRIANGLE_STRIP and the input is error-free.
- **Detailed Line Specifications:**
  - You have to draw the lines according to the usual behavior of the current drawing primitive. For example for GL_POLYGON, you have to draw a line between each consecutive point including the line between the last point and the first point.
  - For lines, use *glLineWidth(3.0f);*
- When the main window is selected and the user presses the 'Q' in the keyboard, then your program will finish execution.

LOGGING

In order to debug more efficiently, you should better print **event logs** to the screen in your graphics programs. In order to have a little exercise, in this warmup homework you have to print the following to the screen (actually to the command prompt):

- When a point is added:

"Point <point label> added to x:<x coordinate as integer> y:< y coordinate as integer >"

- When a point is deleted:

"Point <point label> deleted!"

- When a point is selected:

"Selected Point: <point label>"

- When number of points limit reached:

"Number of points limit reached!"

- When no points left to delete:

"No points to delete!"