MyDraw - A Simple 2D Drawing Program

CS475/CS675: Computer Graphics - Assignment 1

Due Date: 30/8/2013

1 Introduction

You have to create a simple 2D drawing program that can draw lines and filled polygons. The following classes have to be implemented:

1. A class **pen_t** that implements a pen class.
2. A class **color_t** that implements color class.
3. A class **fill_t** that implements a fill class.
4. A class **point_t** that implements a point class.
5. A class **line_t** that implements a line class.
6. A class **polygon_t** that implements a polygon class.
7. A class **drawing_t** that implements a drawing class.
8. A class **canvas_t** that implement a canvas class.

2 Class Details

The classes mentioned above must have the following features -

1. The Pen Class, **pen_t**
   (a) A pen must have a size (how thick a line does it draw)
   (b) A pen must have a color (what color does it draw with)
   (c) A pen must have an eraser mode where it draws with the color of the background.
2. The Color Class, **color_t**
   (a) Must store the Red, Green and Blue components a color.
3. The Fill Class, **fill_t**

   (a) Must contain the current fill type and color.
   (b) Implement two kinds of fill - a solid fill, and a checkerboard (chessboard fill). For the checkerboard fill you should be able to specify the R,G,B of both the colors. Assume that the checks for the checkerboard are no bigger than 16x16 pixels in size.
   (c) The actual fill algorithm that you implement is your choice, but remember this is an interactive drawing program, so you can click inside a polygon or region of the drawing your want to fill.
   (d) Must have a draw method that implements the your fill algorithm.

4. The Point Class, **point_t**
   (a) Must contain the X and Y coordinates of a point.
   (b) Must have a draw method that plots the point at (X,Y) on the canvas. This has to use the underlying call from OpenGL.

5. The Line Class, **line_t**
   (a) Must contain two end points of a line.
   (b) Must contain color of the line.
   (c) Must have a draw method that draws the line. Must use the draw method from the point class and not OpenGL line drawing.
   (d) Implement the line drawing with the all integer Bresenham that works in all octants.

6. The Polygon Class, **polygon_t**
   (a) Must contain the vertices of the polygon.
   (b) Must contain the color of the polygon border.
   (c) Must contain a flag to indicate whether the polygon is filled or not.
   (d) If filled, it must contain the fill color.
   (e) Must have a draw method that draws the polygon. Must use the draw method from the line class for the boundary, and from the fill class to fill the polygon.

7. The Drawing Class, **drawing_t**
   (a) Must contain an array/list/vector of the lines and polygons that your drawing contains.
   (b) Must have a overall draw method that call the draw of all the contained elements.

8. The Canvas Class, **canvas_t**

(a) Must contain the current drawing.

(b) Must contain the size of the canvas (width x height).

(c) Must contain a background color for the canvas.

(d) Must contain a clear method that clears the canvas.

(e) Must contain a 2D array the size of your Canvas. Each array element corresponds to a pixel on your canvas and should be able to store a color. For any point you draw to the screen for a line or a polygon boundary, color the corresponding element in the array to the same color. Assume the left bottom corner of the window to be $(0, 0)$ with the positive X-axis going from left to right and the positive Y-axis going from bottom to top of the window. This array is needed to fill the polygons - write your fill algorithm so that it colors pixels in the array. To display the filled polygon you will directly display this array to the screen as explained below.

3 Saving and loading drawings

1. Add save and load methods to the **drawing_t** class that can save the list of lines and polygons to a text file.

2. You must save all information needed to recreate your drawing from the saved file - so all vertex, color, fill type and pen size information must be stored.

3. You are free to design your own file format to do this but it must be a text file format. Learn to use the C++ iostream and fstream methods to read files. Do not use C fscanf functions.

4 Drawing to be made for submission

You must make a 2D drawing of your hostel room from any point of view to finish the assignment.

5 Use of OpenGL and GLUT

GLUT is to be used to open your window, handle key and mouse events.

1. You must be able to open a window of a reasonable size - say 1024x768.

2. You must be able to handle the following keys and clicks in your application:

(a) ’N’ : Initialize a new canvas, including all elements of the 2D array to the background color. Assume that the size of canvas is equal to the size of your window (and is fixed). Take the background color for the canvas as input from the terminal or an initial config file.

(b) ’D’ : Initialize a new drawing.

(c) ’S/L’ : Save/Load drawing. For load, input filename on terminal.

(d) ’1’: Toggle Line drawing mode. Left clicking on the drawing line should draw a line between two successively clicked points, with the current pen.

(e) ’2’: Toggle Polygon drawing mode. Left clicking on the drawing line should draw a line between two successively clicked points, with the current pen.

(f) ’F’: Toggle Fill mode. Left clicking anywhere in the drawing now fills that bounded region with current fill color and type. When this mode is on, you will draw the 2D array to the screen. Read chapter 8 in the OpenGL programming guide on “Drawing Pixels, Bitmaps, Fonts, and Images” to figure out how to draw a array to the screen. Remember to set your coordinate system properly (see examples in the chapter mentioned above). When fill mode is off, directly draw only the lines and polygon boundaries to the screen using the corresponding draw functions.

(g) ’C’: If fill mode is active, pressing ’C’ must let you change the current fill type and colors, else it should let you change the attributes of the pen.

(h) ’Esc’: Exit the program.

You are allowed only **limited** OpenGL use in this assignment. You can use OpenGL for the following:

1. You will have to setup a 2D orthographic projection to setup your drawing - learn how to use the glOrtho function. For simplicity, set your coordinate frame to match the viewport size and assume integer coordinates. See the code for the Bresenham demo showed in class for example.

2. You will have to plot points in the draw method of the **point_t** class - use GL_POINTS here. Do **not** use any other OpenGL drawing functions.

3. Learn how to set the background (clear) color and line widths using OpenGL - you will need this to clear the canvas and set pen attributes. You will also use the glColor3f command to change drawing/fill color as necessary.

4. Understand the event driven GLUT programming model before you start and follow it.

Since there is no animation involved it is ok to ask for a single buffer in glut and use glFlush at the end of the display function.

AGAIN: **No** other OpenGL functionality is to be used. Do **not** directly draw lines or polygons to the screen - that defeats the purpose of the assignment.

6 Report

1. Make a webpage hosted on your "public_html" explaining what you did with appropriate screenshots for examples.

7 Marking

The assignment will be marked as follows:

1. Implementing the **pen_t** class correctly : 5 marks
2. Implementing the **color_t** class correctly : 5 marks
3. Implementing the **point_t** class correctly : 5 marks
4. Implementing the **canvas_t** class correctly : 15 marks
5. Implementing the **drawing_t** class correctly : 15 marks
6. Implementing the **line_t** class correctly : 15 marks
7. Implementing the **polygon_t** class correctly : 15 marks
8. Implementing the **fill_t** class correctly : 15 marks
9. Implementing the file reading and writing, loading and saving methods correctly : 15 marks
10. Implementing All the keyboard and mouse functionality : 10 marks
11. Making the Hostel Room Scene : 15 marks
12. Report + Viva during Demo : 5 + 15 marks
13. Total : 150 marks
14. Late submission will follow a policy of graceful degradation with a 25% penalty for each day's delay (i.e., zero marks if the assignment is more than three days late after the due date.)

15. It is **ok** to submit a partially done assignment if you cannot complete it. You will get partial credits.

16. Please do not plagiarize source code. If you borrow from anywhere please at the source citation in a comment in your source code and also acknowledge it in your report.

8 Submission Instructions

Your submission must be in the following format:

1. A Tar-Gzipped archive of the complete source code (and only source code). It should compile using a Makefile on any Ubuntu system.

2. Include a README file in the .tgz archive containing a link to the html report page on the assignment.

3. The exact mode of submission will be discussed later in a lecture.