4050/6050

Assignment 1

Basic Graphics Tool, Lines and Shapes

Description

In this assignment you will write a simple graphics program that can draw lines and shapes with different algorithms. Your program may use DDA or Midpoint to draw lines and will also draw ellipses (and circles for extra credit) using Midpoint. Further, your program will assemble shapes for rectangles and triangles by drawing lines from user input. The program will be an interactive tool that uses input from the mouse and keyboard to control the output graphics:

For lines, the (two) mouse inputs will represent the beginning and end of each line.

For rectangles, the (two) mouse inputs will represent two opposing corners for the box.

For triangles, the (three) mouse clicks will be interpreted as the vertices of the triangle.

For ellipses, the (two) points will represent the opposing corners of the bounding box that surrounds the ellipse. The ellipse will be axis-aligned with the primary coordinates (we will learn later about rotations to remove this restriction).

For circles (Extra Credit), the (two) mouse points will represent the center and the distance to the circle.

Your job is to scan convert the shapes, and you will write a rasterizing algorithm that discretizes all shapes into a high-quality pixel-based shape. Note, rectangles and triangles are built from lines (DDA or Midpoint) using your line tool, but the ellipses and circles must use Midpoint for the shape (do not use a poly-line approximation). You can draw several shapes on the screen at once, but will also allow a clear function, input mode described below.

To keep things fun, for color, there are two options. First, the simplest method is to pick the color of each shape at random. Or, you can gain extra credit by adding a color picker (on screen) that picks a color and then draws the next shape this color.

To differentiate between shape modes, a simple input (lower case key press) changes modes for the next shape input - for 'l' (lines), 'r' (rectangles), 't' (triangles), 'e' (ellipse), and 'c' (circles, optional) and the clear screen happens when the 'd' (delete) key is pressed.

Points are given as follows. Drawing line segments in any direction from two inputs (20pts). To draw the rectangles to the screen (10pts) and triangles (10pts) from their respective inputs. Drawing ellipses (30pts) should employ the algorithm you derived (10pts – assigned from paper assignment dur 9/12). Your program can keep drawing additional shapes and lines until the user clears the screen (10pts). You get up to 10 pts extra for adding color selection. And adding circles is additional 10pts EC.

Document your code and be neat (10 points). You may discuss the high-level concepts but you must write your own program, do not share code. Be cautious of your printed-out code (used for debugging) as well, if someone copies your "thrown out" program, you will be held responsible for cheating. Finally, your code must be written in C and/or C++ and run on the SoC Linux machines (e.g. those in McAdams 110) by typing "./assn1" at the shell command prompt (after compile).

Getting started

A starting program and makefile were provided for you.

The program can be compiled by typing "make" and opens a blank OpenGL window, showing the grid 0 to 500 in x and y with the origin in the lower left corner, click around on the grid and the last pixel touched is lit up. This program includes a fair amount of structure and helpful starting functions. Use this program to interpret mouse input, reading the keybindings described above, etc.

Also, IMPORTANTLY, the start program includes a function called **write_pixel**, you will use this to interface with OpenGL, meaning you should not make any explicit GL calls in the program on your own. All GL functionality has been taken care of in the start program. *You will not receive credit for using GL calls to draw lines or curves.*

Good Luck!