

Lab #4 – csc471

Scan converter – rasterizing lines

Today you need to write a program that simulates a scan converter – since we are doing this as a lab assignment (and not a real scan conversion program that would be done only in pixel coordinates) we will need to simulate the scan converter in an imperfect way. However, you should gain an understanding of how a basic line drawing algorithm works and get a chance to think about how scan conversion and linear interpolation works.

You should start with the code from the previous lab (which draws 3 lines – you do not need the circle drawing for this lab, but are free to leave it there). In addition to drawing the 3 lines using `GL_LINES`, your program will need to:

Draws the 3 line using the **parametric incremental line** drawing algorithm that we learned in class today by simulating scan conversion by drawing `GL_POINTS`. Just as with the previous program, the first mouse click (of a pair of clicks) will determine the first point on line. The second will be the second (final) point for this line. From lecture you should have the pseudo code for lines with a slope, $-1 \leq m \leq 1$, which you can use (with modifications to draw opengl points).

You will need to write additional code for lines with:

- $-\infty \leq m \leq -1$ and $1 \leq m \leq \infty$ (you will use the same code for both positive and negative infinity)

We will implement this in a somewhat imperfect way...

- get pixel coordinates from mouse
- convert those coordinates point to world coordinates
- draw the points using `GL_POINTS` in world coordinates (a real scan converter would operate only in pixel coordinates). You could instead use, `glRasterPos` and `glDrawPixels` if you prefer.

Add a toggle to compare “real” lines (i.e. those drawn using `GL_LINES`) for the same endpoints

- Use a keyboard event for the key “L” to toggle drawing the two types of lines

Add a different color to each endpoint of the line

- Color the intermediary points (between the two endpoints) appropriately using linear interpolation (you can use the incremental code discussed in class)

Extra credit:

- After all 3 lines are drawn – allow the user to click anywhere on the screen.
- Add a test to compute which side of the (last drawn) line the point falls on using the implicit equation for the line.

This lab is due two weeks from when it is released and it is worth **two lab credits**.