Computational Geometry Project 2

Alexander Richardson

October 2024

1 The Input to our Algorithm

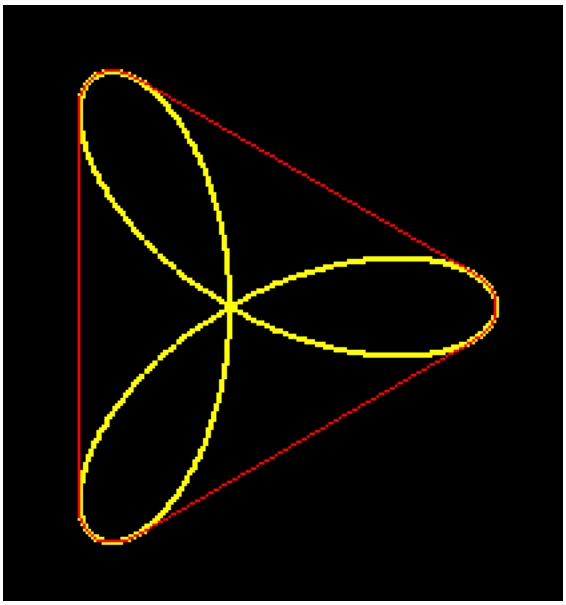
1.1 Degenerate Inputs

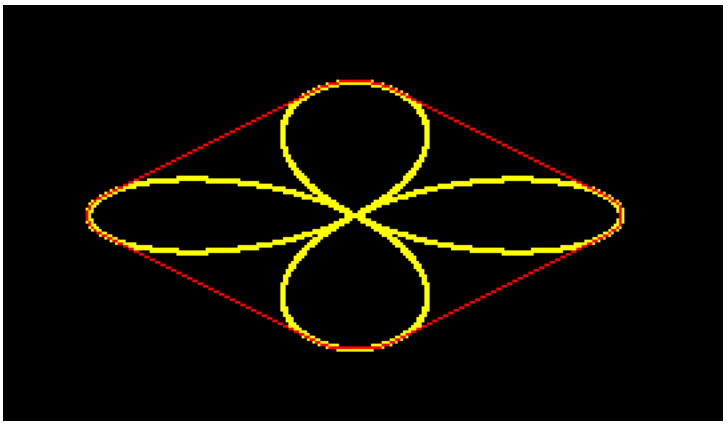
Two example degenerate inputs come to mind: a line and a vector of identical points. In the case of a vector of identical points, I put the first two on the hull by default and never add to it. In theory, my implementation shouldn't handle lines since a similar process to my handling of identical points should occur. However, in viewing the hull created by eye, it seems I am handling it properly due to the numerical instability of the left functions.

I handle collinearity in the sort function by ensuring that points that are closer to the bottom right are earlier in the list - this way we can ignore collinear points (using the left strictly function) after the closest one is put on the stack.

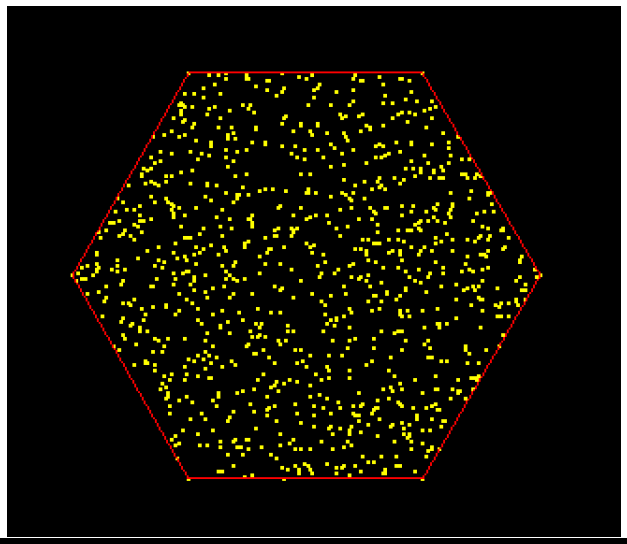
1.2 Created Initializers

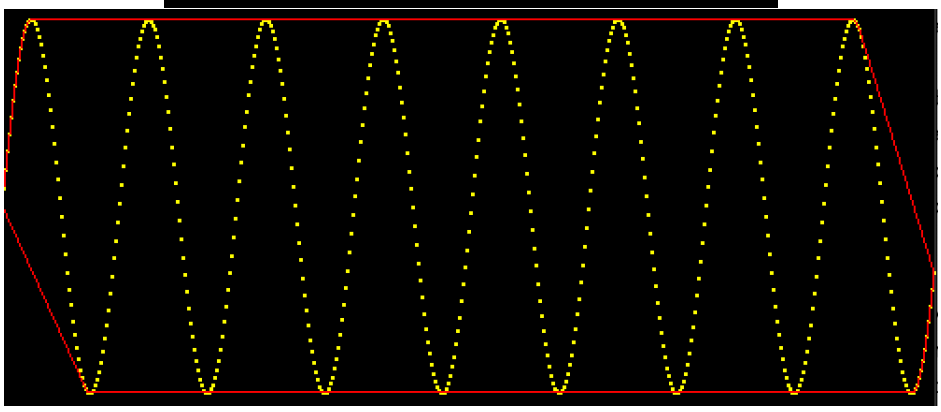
As a student in Topology class, I thought it would be fun to use the quadrefoil and trefoil depicted below:

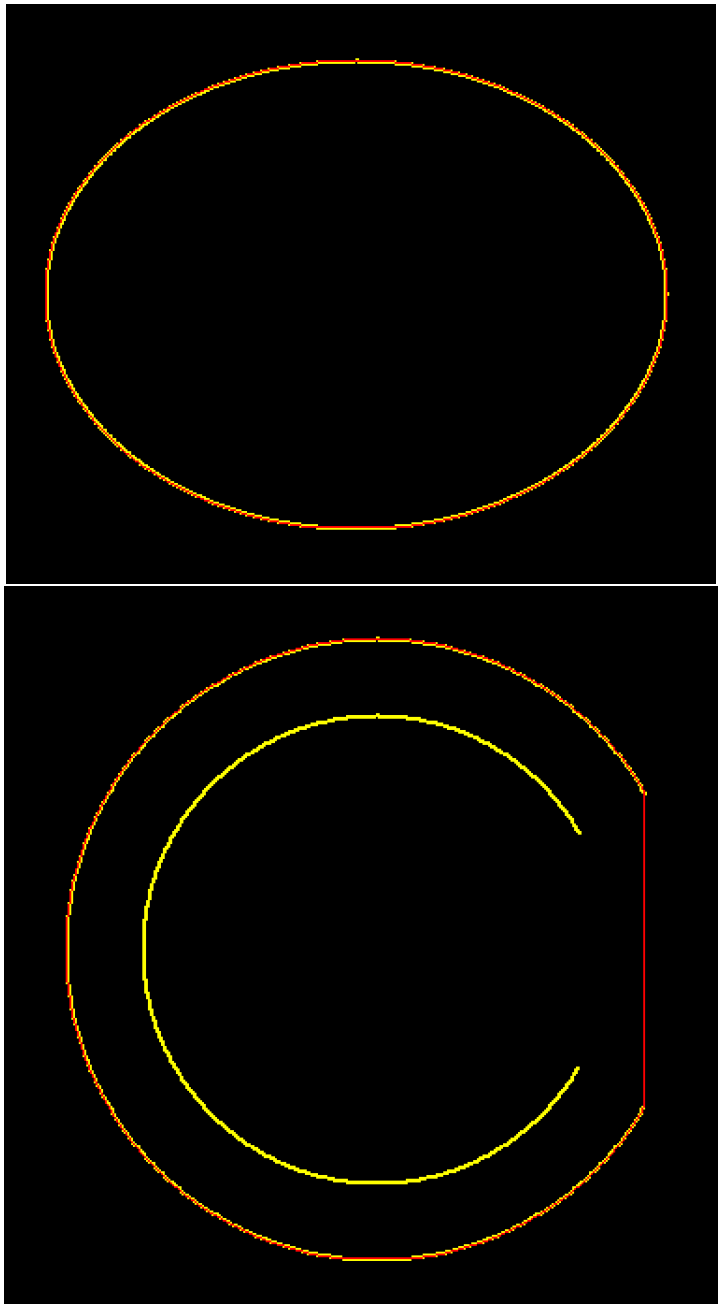


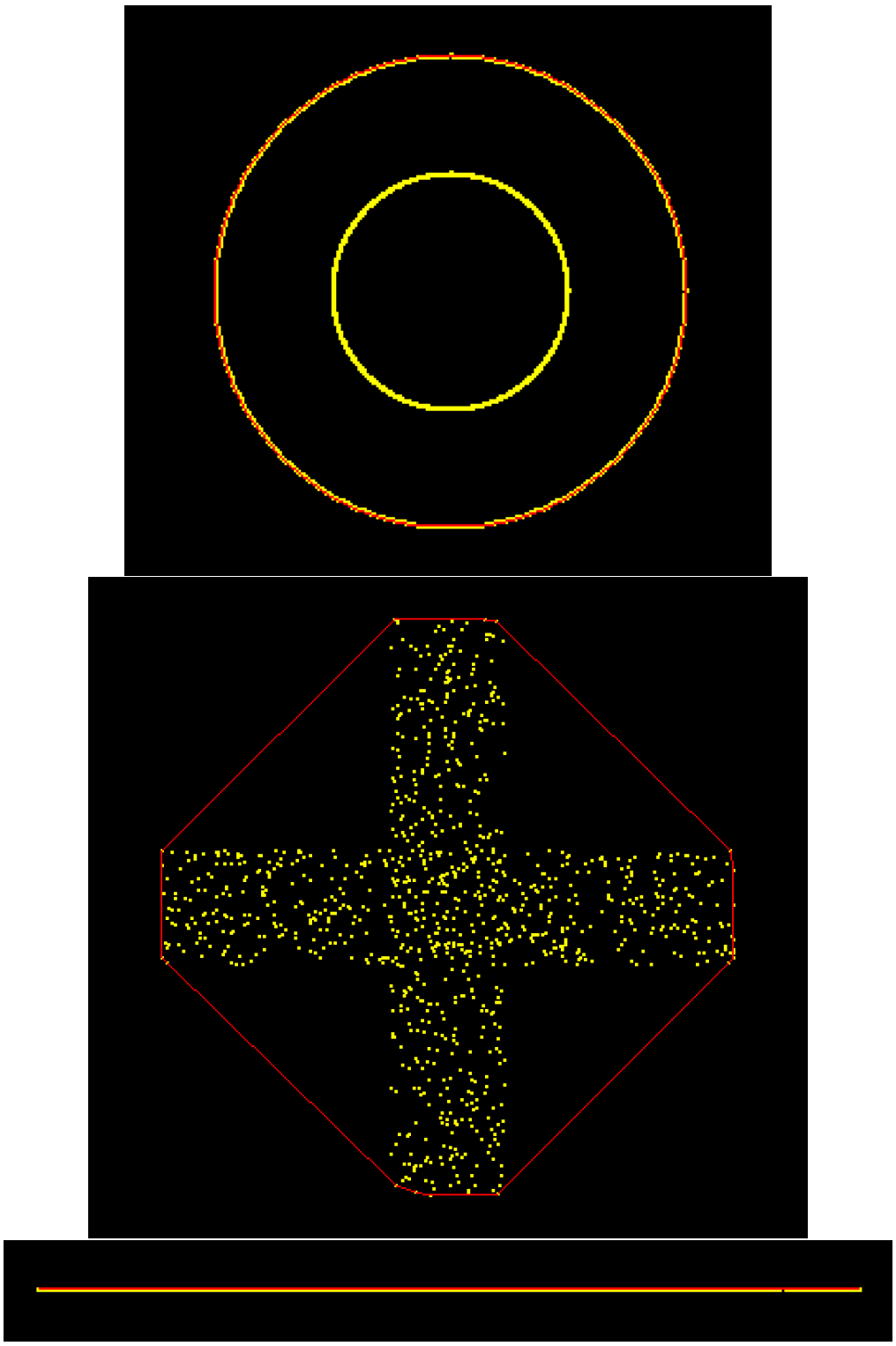


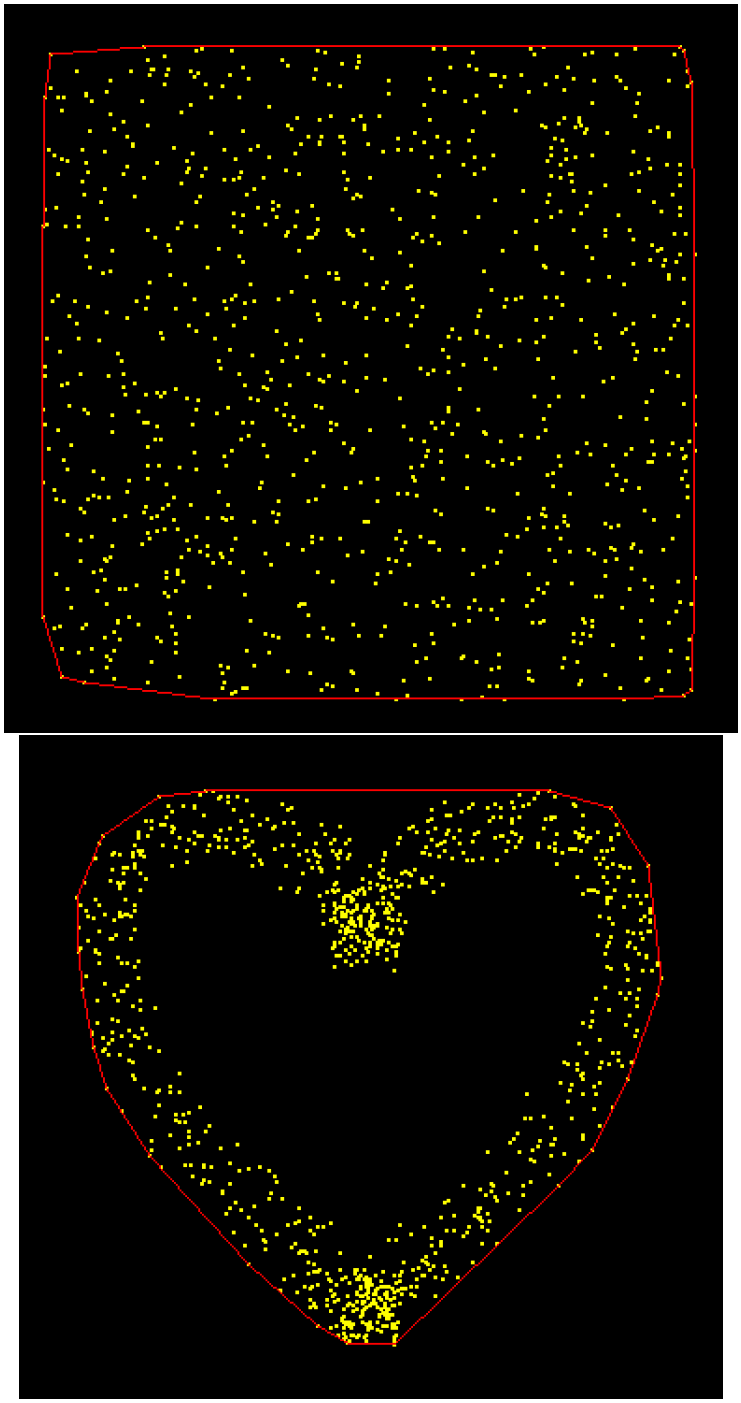
1.3 Other Initializers











2 Reflections

2.1 Time Spent + Reflection

I spent a good few hours on this. As Professor Toma knows, I spent bit of time trying to run the code and get my bearings with OpenGL. Once I got coding things became more fun.

Many of the helper functions were easy since we talked about the code in class. The main challenge was implementing an efficient sorting function. At first I implemented insertion sort by iterating through the

points, taking the radial angle, and then inserting it in the correct spot in the sorted array. However, I wanted a more elegant solution to this. Thus I did a bit of internet searching and found out about the auto function (Professor Toma also referenced this in class). I used this as a comparator for the built in sort function. However, it took a bit of mental heavy lifting to figure out how to deal with the degenerate cases when sorting. I am satisfied with the solution I came up with. The actual Graham Scan function was quite simple. Once sorted the algorithm really falls into place - all I needed was a while loop.

It was a little bit tedious copy and pasting our classmates initialization functions, updating the function declarations, and taking screenshots of more than 11 initializations. Overall the project did not feel tedious and I felt this was very doable given my current knowledge of C++.

2.2 Further Exploration

What about non-convex polygons? Is there any use to computing it? Suppose I want to find the shortest path to get to a point on the non-convex part of a heart - what would I do then?