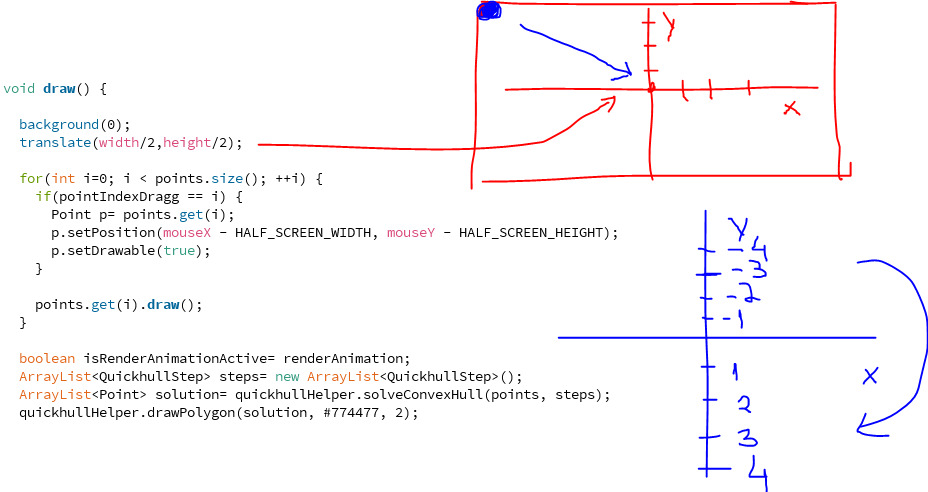
**COT 4521: Intro. To Computational Geometry (Fall 2020)**

**Final Project Report**

For this final project, a lot of trial and error was involved when testing comes in place. The Quick-Hull algorithm is easy when it is writing on paper but hard when programming comes in place. Especially because, there is more than one algorithm involved to solve quick-hull, which are: divide and conquer algorithm and extreme point algorithm.

The divide and conquer algorithm work as a recursive algorithm that breaks drown problem into two or more sub-problems to solve the problem in a simpler a faster way. I was not able to accomplish the recursion I still using divide and conquer but without calling the function recursively, because it was difficult to implement and at the same time, I take longer to solve. This is why I do create a stack of edges; I am using the stack of edges to replace the recursion. After the points are separated in the upper and lower hull by the extreme point algorithm. The edges in the upper hull are added to the stack to visited them later, this will continue until the stack is empty. The use of stack breaks down the problem in sub-problem like originally dividing and conquer does by using the recursive function but I used stack instead because it is simpler to implement.

One of the principal problems I experienced is working in processing due to it is the first time I work on it. I fail to accomplish the upper and lower hull in the right position, they are upside down. The x coordinates are good but the Y coordinates are upside-down, it is because of the way that processing manages the screen.

On the picture above the function translate() is called to place the origin in the middle of the screen. The problem is that in processing, by default, the upper left corner is at (0,0), I change that to be in the middle like the picture above explains. What happens is when I move the origin to the middle the Y coordinates is upside down compared to the normal cartesian coordinates. This means that the program treats the upper hull points like lower hull and lower hull points like upper hull.

In my final work you can see the upper hull points (Blue), and lower hull point (Red), the animation shows how the program solve the convex hull. In the end, I add a function that adds increment and decrement the point by pressing + and – sign on the keyboard in order to show how the animation change by incrementing or decrement points within the program.