**Basic 2d Shapes**

**Sample I/O:**

DDALine and MidpointLine: in inputs.js, you can edit the values that are in the javascript object to specify the starting and ending points of the line.

midpointCircle: in inputs.js, you can edit the radius of the circle that you’d like to draw

midpointEllipse: in inputs.js you can edit the values for the width and the height of the ellipse that will be drawn.

**Lessons learned:**

I think the most important lesson I learned from implementing these algorithms was just how much is done for us alright with graphics libraries like HTML Canvas, HTML DOM, SFML/other compiled libraries, etc. I feel a lot more forgiving for programs like Google Chrome and electron.js (for building desktop apps with html, css, and javascript) because I can understand why they need to eat up so many resources. I’m amazed at how accurate programs like the Adobe suite are.

Properly implementing the midpointLine (*The Bresenham Line-Drawing Algorithm*) was difficult because of all the cases we had to account for. It was time consuming but a I learned a lot about how to implement it. I found a great resource explaining the algoirithm and another one that that helped me understand the different conditions that had to be accounted for in order to implement the drawing properly for all conditions (positive slope, negative slope, horizontal line, and vertical line).

I’ve only worked with high level javascript frameworks before like React and Angular, so it was cool learning to use javascript by making <script> tags. I’ve used React the most which runs on JSX, so I’m usually writing HTML inside of my javascript instead of including javascript files in HTML.

**Issues faced:**

As mentioned above I had some difficulty figuring out that how I should account for all slope situations with the midpointLine.

**Remaining bugs:**

N/A, was able to get the assignment completed.

**Additional Functionality:**

I added an inputs javascript file for convenience. Simply edit the values in that file and reload the page to see the shapes change.

I guess it not really additional functionality, but I did the extra work to separate the javascript into another file to clean up the code a lot. I separated out a few redundant, verbose functions into cleaner wrapper functions. Thse were fillPixel() and contextFromId(). They’re the first two functions in shapes.js