Report

Logs:

We discussed which template we wanted to do and what functions we wanted to implement in it beforehand and set out with the target of completing at least one function per week.

DD/MM

01/04: We called each other and brainstormed what extensions we could implement.

04/04: Decided on what extensions we were capable of doing who wants to do which one.

07/04: Discussed the progress we made with the extensions we picked and worked together to properly integrate the standalone function Edward gave.

11/04: Omar is done with eraser function but it's too empty and Umar is working on the rectangle shape tool but is having issues with drawing multiple shapes.

14/04: We get on call on teams and shared our screens to see and fix the potential problems we're having with our collective functions.

19/04: The issues have been solved and we have added sliders to the eraser and brush functions.

24/04: The eraser, brush, rectangle, oval functions are completed.

29/04: Still trying to solve the issue of fully integrating the standalone function Edward gave into the application. Also, thought of a new function to add, an image import tool.

04/05: Found no native method in p5.js of taking an input. Trying through HTML but with no success.

05/05: Discuss the image import function with partner but neither of us can find a solution.

06/05: Started report and decided to do a dumb downed version of the image import tool.

08/05: Last tool added, now we're both just working on the report.

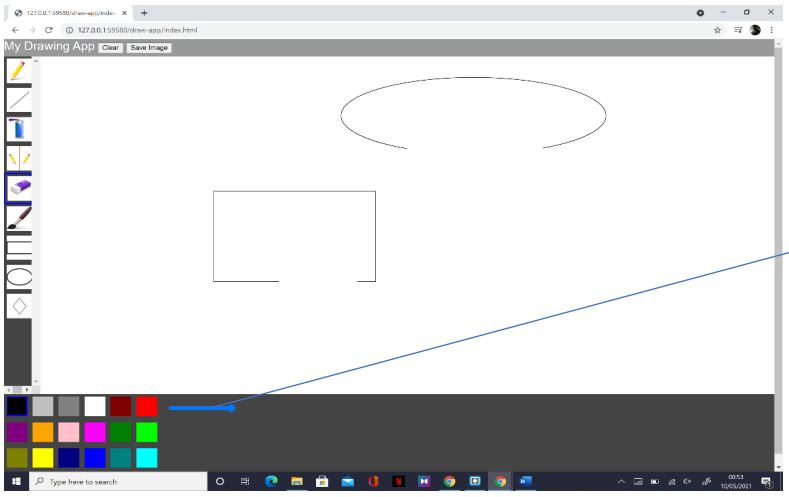
10/05: Report finished, just have to submit the project now.

Functions:

In our program, we added 7 functions. An eraser, paint brush, rectangle shape tool, oval shape tool, variable shape tool, image import tool and an image draw tool.

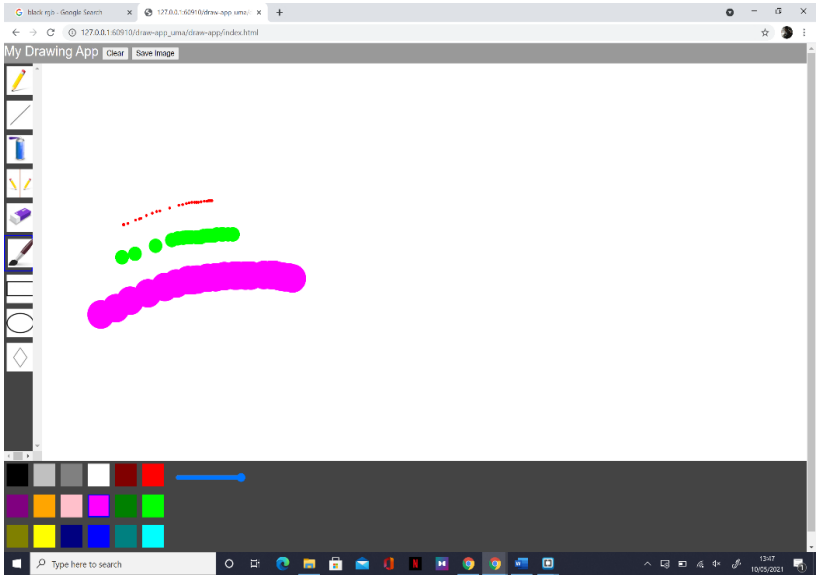
- Eraser (eraser.js)

The first function, an eraser. The eraser we implemented sets the pixels it passes over to white with a square for the head, with no stroke and its fill set to white. You can erase scribbles and images using it. We also implemented a slider by adding HTML through JavaScript. This slider allows you to set a value between a range of values for the size of the square by passing the value selected on the slider to the parameter used for size.



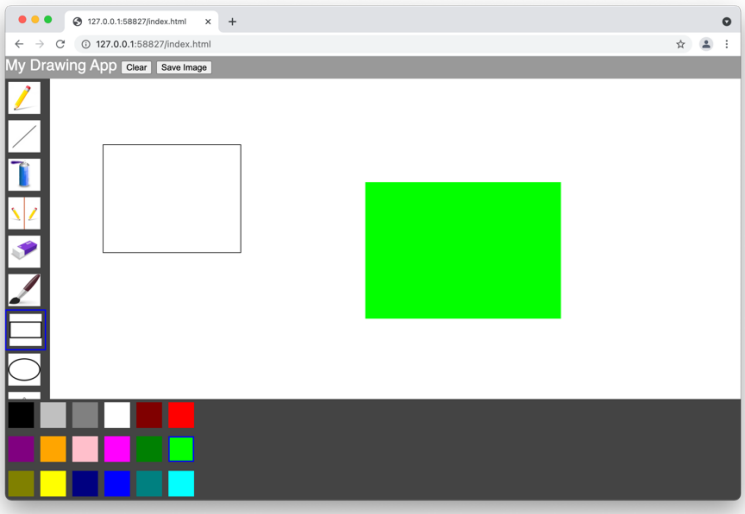
Above you can see the slider. You can also see some shapes that have been partially erased.

- Paint brush (brush.js)



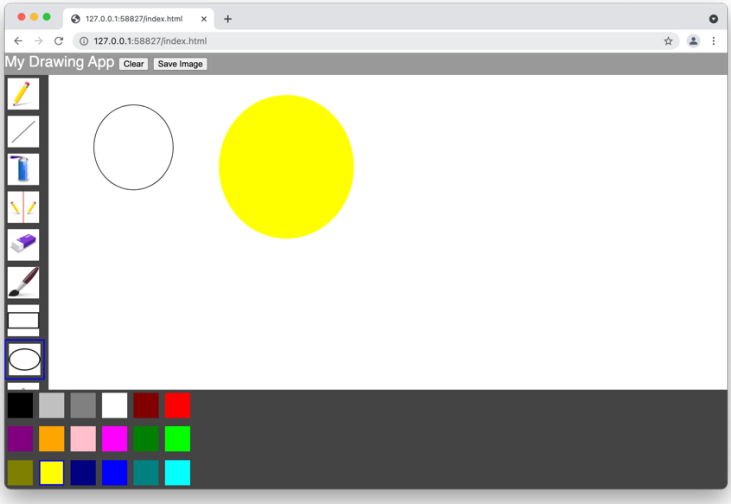
The second function, a paint brush. Similar to the eraser, it uses a slider to adjust its brush size. The paint brush uses all the colours provided in the colour palate. We used ellipses drawn without a stroke to mimic a brush tip. You can see in the screenshot, the paintbrush working with multiple colours in different stroke weights.

- Rectangle Shape (rectangle.js)



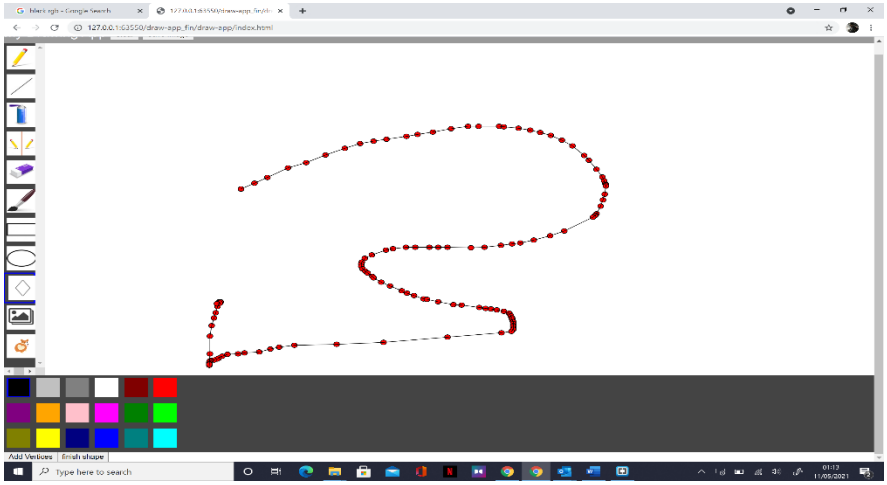
The third function, a rectangle shape tool. You can click on the canvas to draw or press and hold while moving the cursor to adjust the size of the rectangle being drawn. You can also pick a colour from the palate to draw a rectangle of solid colour.

- Oval shape tool (oval.js)



Fourth function, very similar to the third function however it draws ovals instead of rectangles. It provides all the same functionality as the rectangle tool.

- Variable Shape tool (shapeTool.js)

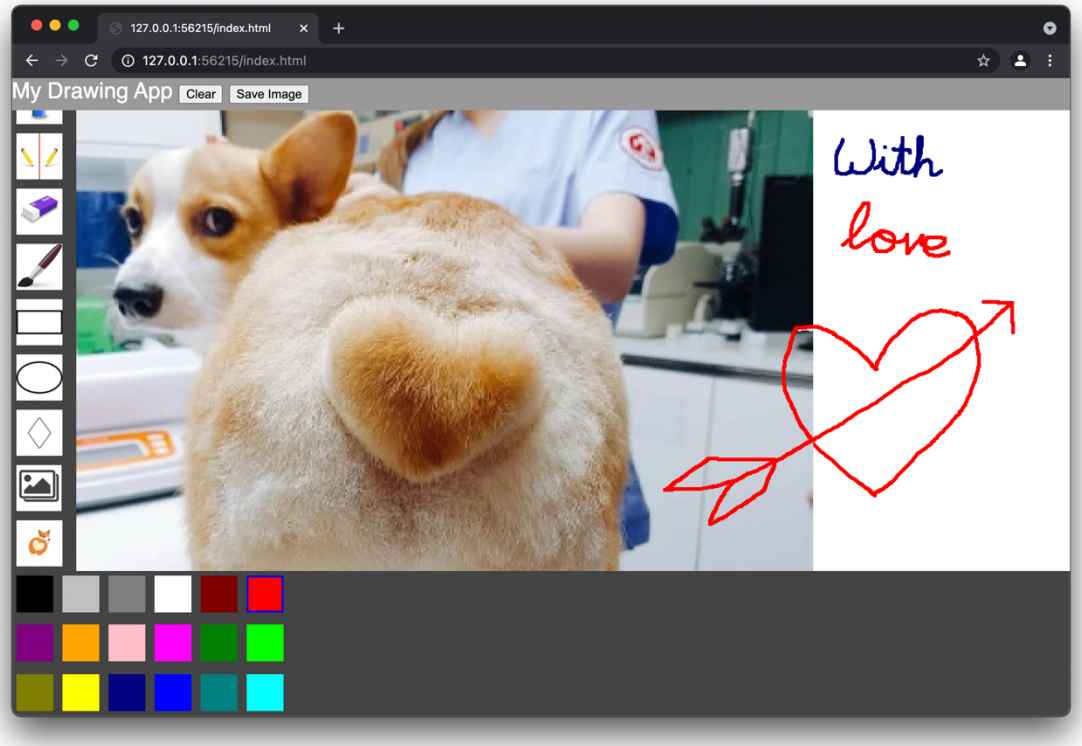


What Edward gave us as a stand-alone function to integrate into the application. You can draw any shape because of the attached strokes whose vertices can be adjusted to alter their shape.

- Import Image Tool (image.js)

This tool was supposed to allow the user to import any image onto the canvas to draw on. We couldn't get it to reliably work.

- Image Draw tool (corgiButt.js)



This was added, as a dumbed down version of the import image tool. It pulls up an image from assets to draw on using any of the other tools.

Challenges faced:

- Eraser (eraser.js)

Initially we tried to use the load and update pixels command to get the white colour to stick to the canvas but realised there was no need since it worked by setting the pixels to white where it passed without updating or loading pixels. The second problem, that it wouldn't erase unless you adjusted the slider first. Fixed that by giving the size parameter an initial value.

- Paint brush (brush.js)

We figured out how to accurately use the load and update pixel commands in this function. We couldn't get the brush to draw without picking a colour from the palate first. Also ran into the same problem in the eraser with the slider, it was fixed the same way as in eraser.

- Rectangle Shape tool (rectangle.js)

The issue was that it would just keep adjusting the last drawn shape. We fixed the issue by adding an else-if where we reset the drawing values back to their initial values and a loadPixels to save the last drawn shape.

- Oval Shape tool (oval.js)

Similar issues as rectangle shape tool that were resolved in a similar fashion.

- Variable Shape tool (shapeTool.js)

Two issues, firstly getting the edit and finish shape buttons to appear side by side in the options class. Secondly was how the strokes start acting strange to fill themselves once a colour is picked. We couldn't find a solution.

- Import Image Tool (image.js)

There was no way to accept an input using JavaScript alone so, we tried through HTML. We tried multiple variations of the input tag in index.html as well as different scripts in image.js for the html. We always ran into the same problem where it would take and acknowledge an input but would show the same error where it in the draw function it wanted an HTMLsvg/imgElement to draw and showed an issue with p5.min in the console. Ultimately, we couldn't find a method to overcome the error.

- Image Draw tool (corgiButt.js)

We could have used the image function in p5.js to draw this but we found a more unique method to draw the image instead. Otherwise, the function was too small and simple.

- General

We weren't able to integrate MousePressOnCanvas from shapeTool.js into the helper functions.

Planning and development:

We called each other to decide how are we going to approach the development of this project. We came up with ideas we both felt we could execute without issues.

To work more efficiently both of us split the tasks between ourselves and set to work on them. We took the functions we thought we were better suited to write. We contacted each other every day to make sure both of us are on track and to consult each other, in case one of us got stuck on a particular function or, what we could add to improve our functions.

During development one of us was struggling to implement a slider to adjust the size of both the paint brush and the eraser. The other explained and showed how to implement it by sharing his screen on teams and live coding it. Later on, in development one of us was having issues in drawing in the shape tools where it would just keep readjusting rather than drawing new shapes. The other was able to help him by explaining how the load and update pixels commands work and how values need to be reset at the end to separate them from previous values. This allowed us to learn and become capable of tackling problems togther that we couldn't alone.

We made sure to leave comments in the functions, the issues we were having, what one of us thought might be causing it or if we resolved something. Allowing us to better understand each other's work even without discussing the function to find out what's changed or causing issues.

Coding techniques:

We didn't use anything too complicated because that would make it harder for each other to work on the same functions since everyone doesn't have the same level of understanding. We used standard techniques such as constructors and self-referencing functions in those constructors. We made sure to use the same style of spacing, indentation, variable names and comments so that every function looks cohesive. We also tried to keep the code clean and well explained through comments so anyone can have an easy time understanding it. We used added HTML through JavaScript to the Dom as well as a little HTML in index.html. We also used getter functions in one constructor.

Self-Evaluation:

We learned that we have a lot left to learn. We're not entirely familiar with what p5.js can do or we wouldn't have to turn to HTML to implement things we couldn't in p5.js. Having been forced to turn to HTML made us realise how little we understand the DOM as well as what's possible through HTML and CSS. We could have solved a lot more like the stand-alone function Edward gave or written some script for improving the shape tools by giving them buttons to change from borders to solid shapes. We should have used more features in the console like the stepper features and isolated functions to see if they work by themselves. There is a lot to improve, things we aren't capable of yet. We have to practice more and work hard to expand our horizons ourselves.