Drawing Application

We used a google doc to co-ordinate our work. Setting goals for each week and noting which goals have been completed. We also added a change log to the google doc towards the end of the project where we can log minor changes made to the code so we know what changes the other person has made so we can avoid making the same change separately.

Progress slowed down significantly after starting on the blur and flood fill tools, as these posed large challenges with code efficiency that we couldn't entirely fix. It would have been better if we put those to the side to work on them sparingly, allowing us to focus on other aspects of project, like more helper functions.

Features

Rectangle / Ellipse:

Draws a rectangle or ellipse from the position where the mouse is clicked to where it is let go, and allows the user to toggle between solid colour and outline shapes.

This tools implementation is essentially the same as the line tool included in the template. It saves the canvas pixel state before drawing is begun and draws a preview of the shape according to the mouse position, then loads the saved canvas (removing the old preview) and draws a new preview. This continues until the mouse is released and the final shape is drawn.

This uses the 'populateOptions' function included in the template to allow for both outline and solid colour modes.

Stamp:

Prints a preset design at the point of mouse click, and allows the user to select between different designs.

Eraser:

Acts as a rubber, drawing a white line where the user clicks.

This is a variation of the freehand tool where the line drawn is locked as white and has increased width.

'PopulateOptions' and 'unselectTool' were used to change the fill and stroke when the tool is selected and then reset these values to default when a different tool is selected. Using this method meant that if the user switched colour while using the eraser the colour would change to that selected colour instead of white. This meant we had to add a line of code into to 'draw' function of the tool to change the fill to white in order to maintain proper functionality of this tool after the user changes colour.

Load button:

Creates a button on the top of the screen which opens file explorer. The user can then load an image of their choosing to the canvas.

This button is created in the html and the function is contained in the helper functions file.

There were issues during implementation due to browser security. The file path would return a fake path, so we had to use file readers instead of file path to load the image.

Star trail:

Draws stars of varying sizes periodically while the mouse is pressed.

This uses a variable outside the scope of the draw function as the value had to persist between multiple iterations of the draw function. This value constantly decrease while the mouse is pressed until it reaches zero, when a star is drawn and the interval variable is reset.

When a star is drawn a random number between 1 and 5 is generated and multiplies certain values in the stars co-ordinates to increase the size. Initially we were multiplying the wrong part of the co-ordinates which resulted in the stars having a random position instead of a random size.

Scissors:

Allows user to select an area of the canvas to be cleared, allowing for more precise erasing than the eraser tool.

This uses the freehand tool to indicate to the user the area they are selecting. Every time the freehand tool draws a line the mouse co-ordinates get added to an array. When the mouse is released we used ‘beginShape’ followed by a for loop adding a vertex to the shape for every point in the array and then ‘endShape’. This draws over the selected area of canvas in white.

This tool had a bug where the array persisted between multiple uses of the tool. Meaning the shape drawn would include all the selected points as vertexes as well as all the points selected every other time the tool was used prior. This was fixed by resetting the array after the shape is drawn.

Blur:

Blurs the affected pixels into each other, creating a smearing affect.

This tool takes the RGB values of pixels surrounding where the mouse is pressed to find the average colour and then draw an ellipse of that colour. It then repeats this multiple times around where the mouse is pressed.

This tool requires many steps to find the average colour of the pixels, resulting in it not being very efficient and running slowly. The blur affect works significantly better when the width of each ellipse is small and many ellipses are drawn. However, the amount of ellipse required to make the blur effect work as you’d expect a blur to work is takes a very long time to run. We couldn’t find a way to make this process run faster, so we had to settle on a balance between the blur’s efficiency and its effectiveness. In its

current state it draws 4 ellipses 20 pixels wide. It isn't immediately recognisable as a blur, but if you drag the mouse over an area a few times it looks blur like.

Spirograph:

Similar to the freehand tool included in the template, but repeats what is drawn at regular rotational intervals around the centre of the canvas.

This tool uses the slice method to copy the elements from 'positionArray' into 'previousPositionArray'. This allows us to change one array without altering the other. However, the elements of these arrays are arrays, so using slice to copy the entire array would return references to the inner arrays. This meant that in order to get a shallow copy of 'positionArray' we had to use the slice method for each inner array separately, using a for loop.

This equation was used to calculate the translations for the points:

$$x' = x\cos a - y\sin a \; y' = x\sin a + y\cos a$$

However, this equation had to be altered slightly because the origin of the canvas is in the top left corner instead of the middle and the y axis is reversed compared to standard co-ordinates.

Flood fill:

All pixels that are connected to the clicked pixel by an unbroken sequence of pixels of the same colour as the clicked pixel are changed to the desired colour.

A for loop wouldn't have worked for this tool because it needed to loop an unknown amount of times, so we used recursion. A while loop may have worked, but we thought this would be more complicated in this case than recursion. In order to prevent to recursion from being infinite we added the constraint that it returns if the pixel being checked is outside the height or width of the canvas. This constraint also prevents it the code from running when you click to change tools or colours as these lie beyond the bounds of canvas.

This tool checks the colour of each pixel surrounding the designated pixel and, if conditions are met, then checks the pixels surrounding those pixels. This is very inefficient because it means the same pixel gets checked multiple times. The complexity of this code is hyperbolic, so for larger areas it takes a very long time to compute, however it does work eventually.

26/02 - 05/03

Goals:

- Rectangle tool
- Ellipse tool

Complete:

- Rectangle tool
- Ellipse tool

05/02 - 12/03

Goals:

- Fill/Outline
- Stamp
- Eraser
- Load button

Complete:

- Fill/Outline
- Stamp
- Eraser
- Load button

12/03 - 19/03

Goals:

- Spirograph
- Scissors
- Star trail
- Blur tool

Complete:

- Spirograph
- Scissors
- Star trail
- Blur tool

19/03 - 22/03

Goals:

- Format current code and improve previous features usability

26/03 - 10/04

Goals:

- Flood fill
- Improve blur tool

Complete:

10/04 - 20/04

Goals:

- Improve flood fill
- Improve blur tool

Complete:

Changelog:

- Fixed the load image button.
- Improved the star and blur icons.
- Fixed an issue where the mirror draw tool would delete the last thing drawn by the previously selected tool for some tools.