Report

Implementations

Editable shape

This extension has been made into a constructor function. We placed the edit and finish shape button in the options area using the populateOptions function. The method to finish the shape is added to the unselectTool function, meaning the shape is also finished by changing tools.

Colour Palette

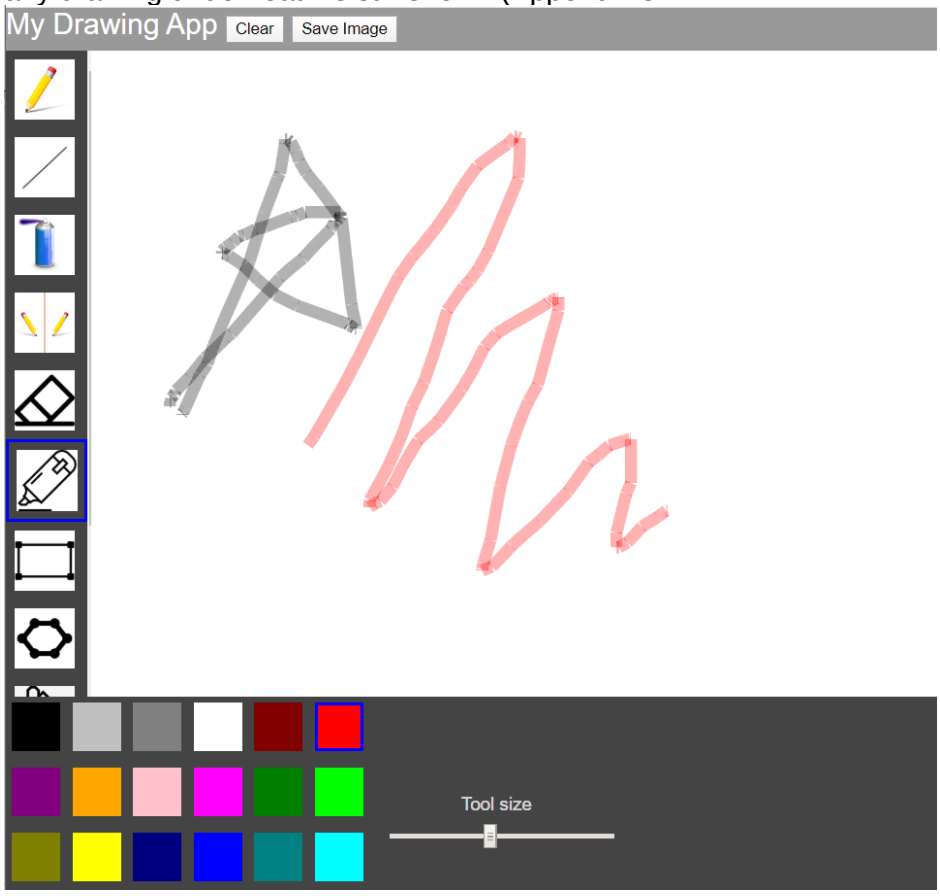
We added alternative colour values from the original colours for the highlighter and flood fill tool. The colour values are grouped together in an array in the colours array ((Appendix 2)). We made sure the border switched to the desired colour when changing colours on both these tools and made the colour change from the string value to the rgba value when using these tools and back when using other tools.

Freehand tool

We edited this tool to consist of three parts: an eraser, highlighter and pen. We added an if statements to check the name of the selected tool and altered the stroke, colour and stroke cap accordingly. The object is then called three times with the name of the tool and icon as parameters.

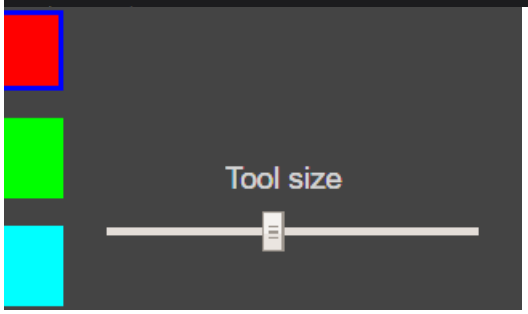
The eraser allows users to remove mistakes, it is controlled by the stroke slider but the stroke value is multiplied by six so the eraser is always bigger than the highest stroke option.

The highlighter tool sets the alpha value of the colour selected to 0.3 in the colour palette so any drawing underneath is still shown (Appendix 3

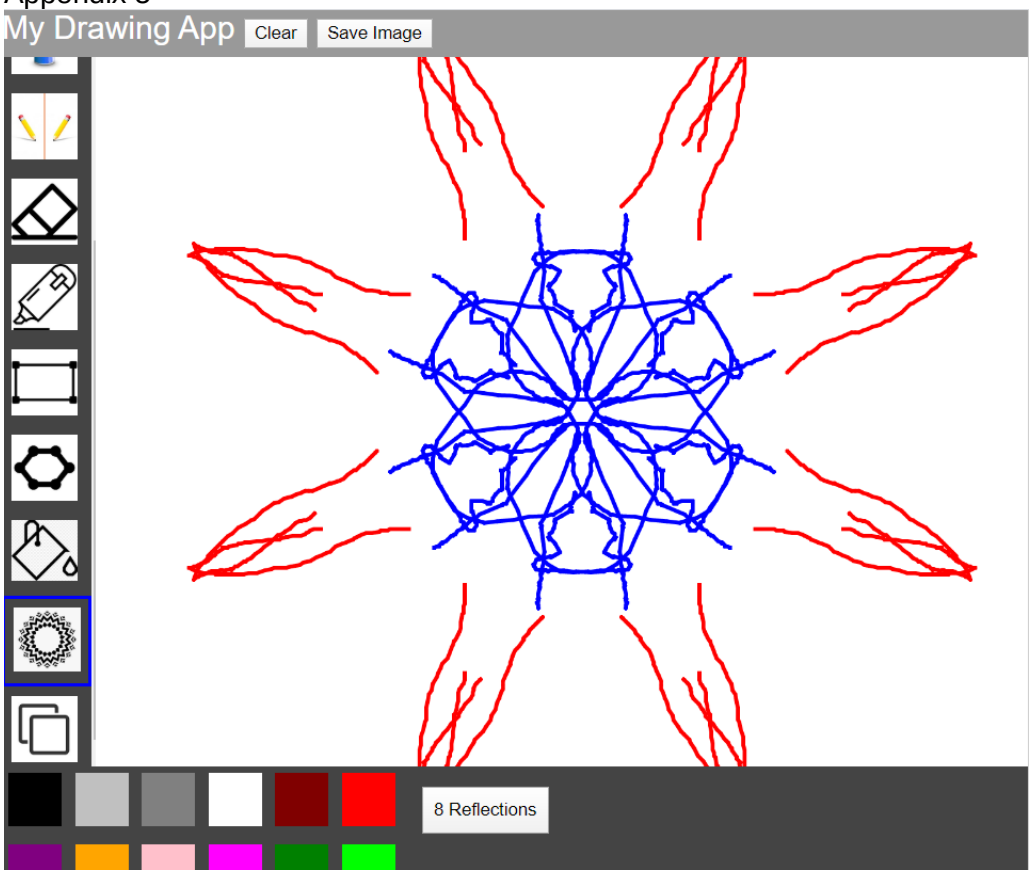


Appendix 4

```
22      //Adds the slider to the options area
23      this.slider = function(){
24          select(".options").child(slider);
25
26          //Created text above the slider
27          var text = createDiv("Tool size");
28          text.class("sliderText");
29          select(".options").child(text);
30      }
31
32      //Adjust the stroke size of drawing tools by using the slider
33      this.toolSize = function(){
34
35          var val = slider.value();
36
37          // The stroke size will be from 1-10 depending where the user changes the slider.
            If the eraser is selected, it will be x6 as big
38          if(toolbox.selectedTool.name == "eraser"){
39              strokeWeight(val * 6);
40          }
41
42          // If the spray can tool is selected, the size of each dot changes with the
            slider..
43          else if(toolbox.selectedTool.name == "sprayCanTool"){
44              strokeWeight(1 + (val - 1) * 0.16);
45          }
46
47          else if(toolbox.selectedTool.name == "highlighter"){
48          strokeWeight(val * 2);
49      }
50
51          //ALl other tools will use the default values of 1-10
52          else{
53              strokeWeight(val);
54          }
55      }
56
57      //Changes the cursor to a cross
58      this.cross = function(){
59          cursor(CROSS);
60      }
61
62      //Changes the cursor to an arrow
63      this.arrow = function() {
64          cursor(ARROW)
```



Appendix 5

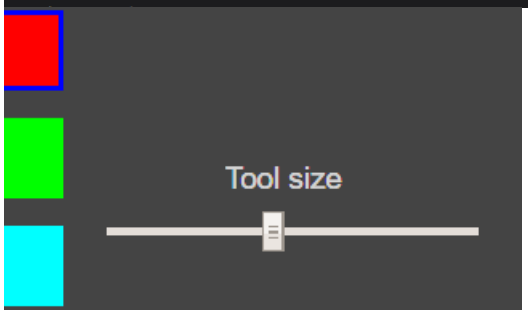


). The highlighter is also controlled by the stroke slider but the value selected is multiplied by two so the same stroke size is wide enough to cover any writing done using the pen.

Helper functions

We add the function which controls the tool size to this constructor. This function is called in the options area for all tools except flood fill (Appendix 4

```
22      //Adds the slider to the options area
23      this.slider = function(){
24          select(".options").child(slider);
25
26          //Created text above the slider
27          var text = createDiv("Tool size");
28          text.class("sliderText");
29          select(".options").child(text);
30      }
31
32      //Adjust the stroke size of drawing tools by using the slider
33      this.toolSize = function(){
34
35          var val = slider.value();
36
37          // The stroke size will be from 1-10 depending where the user changes the slider.
            If the eraser is selected, it will be x6 as big
38          if(toolbox.selectedTool.name == "eraser"){
39              strokeWeight(val * 6);
40          }
41
42          // If the spray can tool is selected, the size of each dot changes with the
            slider..
43          else if(toolbox.selectedTool.name == "sprayCanTool"){
44              strokeWeight(1 + (val - 1) * 0.16);
45          }
46
47          else if(toolbox.selectedTool.name == "highlighter"){
48          strokeWeight(val * 2);
49      }
50
51          //ALl other tools will use the default values of 1-10
52          else{
53              strokeWeight(val);
54          }
55      }
56
57      //Changes the cursor to a cross
58      this.cross = function(){
59          cursor(CROSS);
60      }
61
62      //Changes the cursor to an arrow
63      this.arrow = function() {
64          cursor(ARROW)
```

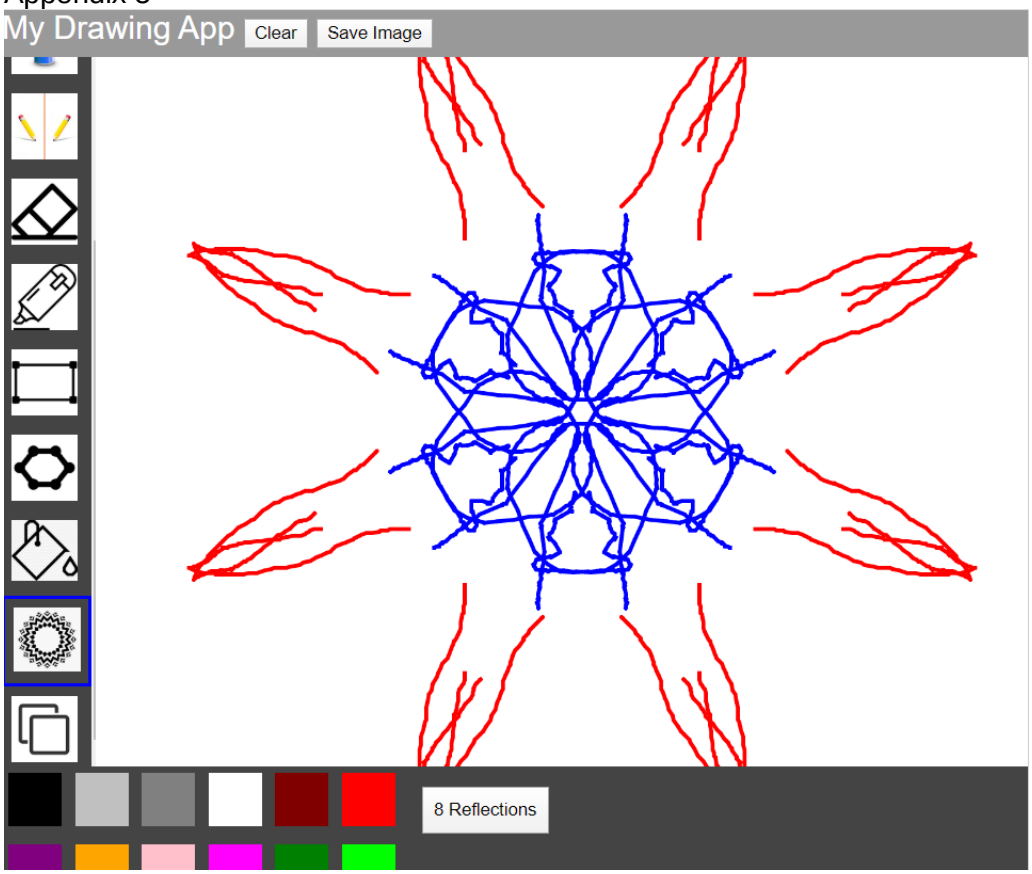


). An if statements is added to check the tool name selected, as some requires adjustments.

Kaleidoscope

We added this tool to allow the user to draw kaleidoscopes. The reflection point is the center of the screen. The tool automatically has eight reflections but there is an option button on the bottom menu to change it to six (

Appendix 5



).

Line or rect tool

We edited the original line tool to allow this function to draw lines or rectangles. If the tool name is “drawrect”, it adds a button allowing the user to choose between fill and no fill in the options area when drawing the rectangle (Appendix 6).

Cloning tool

This tool copies a selected area and redraws it in a new area using copy(). We implemented the mousePressed and keyPressed function to record the position of the mouse. When Ctrl and the mouse is pressed at the same time, the position saved will be where the original image starts. When the mouse is pressed again, it will record the mouse position of the new area. The cloning image will draw as long as the mouse button is held down. We later added a rectangle at the original image for the user to see the position of the image they are cloning (Appendix 7).

Flood fill

This tool fills an area of the same colour to another. We developed two versions: the first attempt (floodFill1.js) used recursion while the second (floodFill2.js) used a while loop and a queue.

The first method retrieved the colour value of the pixel using get(), which returned an array with the rgba value. In order to compare the colours, we had to add the rgba values of each colour to the colour palette and use a for loop to compare each index. The function calls recursively, changing the directions if the colour of the pixel is not the same as the new

colour or if the colour of the next pixel is the same as the originally retrieved colour. As this method only worked on smaller areas due to the maximum call stack being exceeded (Appendix 8), we decided to try another method.

The second methods enqueues the first pixel coordinates to the queue. While the queue is not empty, a variable is set as the head of the queue and then the queue is dequeued. We check if the colour of the variable pixel is the same as the original colour. If so, we set the colour to the new colour and enqueue the coordinates to the queue. This now works for large areas but takes much longer.

We have chosen to use the second method. Given more time, we would have liked to explore further ideas which improves the time for it to complete.

Planning and coordinating development

As each tool works independently, we both worked on different tools simultaneously, updating our log using Google Drive, so we knew what each other was working on. Each week we shared files and one of us merged them together so we could see each other's developments and work on the latest version of our app. We both kept change logs, recording which lines were added on pre existing files, making merging easier.

If one of us came across a problem, we shared our work so both of us could work on the problem at the same time, sharing which methods we have tried via WhatsApp.

Challenges faced

After using the eraser, when the user draws with any other tool, the colour remains white and did not change despite switching colours. We added a function to the colour palette, this.setColour(), which checks what the selected colour is and sets it to fill and stroke. We called this in the selectTool function in toolbox, so the colour selected is reset every time a different tool is selected.

Due to the rgba values being necessary for the highlighter tool, we had to solve how to change the colour value from the string value. When we got it to partially work, we encountered more problems. Once the highlighter tool was selected, the colour value would not update until we changed colours (Appendix 9). Additionally, when going from the highlighter to another tool, the colour reverts back to the colour selected before it was switched when using the highlighter (Appendix 10). This was because we wrote the code to convert the values in the colourClick function. As we had similar issues when implementing the eraser, we adopted the same solution and wrote a new function similar to the one in colourClick and called it in the selectTool function in toolbox (Appendix 1).

Self-Evaluation

We feel the app does not have the most intuitive user interface. Whilst the tool icons do make it evident which tool is which, we feel it would be more helpful having the name of the tool appear when hovering over their respective icon. Additionally, when using the tool size slider, there is no indication of what the tool size is. We could have implemented an outline of the tool size which scales with the slider. The eraser could also have an outline, as it is difficult to understand where it will start erasing when used.

Despite the colour palette taking the longest time to fix, it now functions correctly for all tools. Reflecting on the colour palette, because the flood fill tool needed the colour values in an array, we could have just made the colours as objects with the value as an array. With the highlighter needing a different alpha value, we could have pop() the original value and push() the highlighter’s value (

Appendix 12).Overall, we are highly satisfied with how the project went as all the tools and colours interact as expected. We collaborated well together to fix each other’s code and shared good practices.

Progress Log

Week	Task	What we did	Problems	Task achieved	Task not Achieved	Task to be completed
24/2	Create eraser tool	Added a constructor function which consisted of a white pen.	Once the eraser was selected and the pen was selected back, it would lose the colour, so the stroke would change back to the pen stroke but the colour would remain white. Added a function into the colourPalette function which we then called in the toolbox to check and update the colour for all the tools.	Created eraser tool	Getting the colour to revert back after using the eraser	Getting the colour to revert back after using the eraser by 8 March
	Create tool size function	Created a slider in setup function and created a constructor function which changed the stroke weight based on the slider value.	The strokes of the spray can also increases but the spread value did not, which made it look less of a spray can.	Tool size constructor function		Have the stroke weight for the spray can to remain at 1, regardless of the slider value by 8 March
2/3	Revert the colour back after using eraser	Created a function in the colour palette which checks the selected colour and sets it to fill and stroke. The function is called in the selectTool function in toolbox, so the colour selected is checked and set again every time a new tool is selected.		Colour now works after using the eraser		
	Have the stroke weight for the	Added an if statement in the tool size construction function which	Initially, the function was created and called after	Spray can stroke weight now		

	spray can to remain at 1	checks if toolbox.selectedTool.name == "sprayCanTool". If it did, set the stroke weight to 1.	new Toolbox but before toolbox.addTool in the setup function. Due to the tool size function checking if the selected tool name is the "sprayCanTool", it returned an error because the tool as not been created yet. Therefore the function was moved to the end after all the tools have been created.	remains at 1 regardless of the slider value.		
9/3	Complete extension tool - editable shapes	Followed instructions to create the extension. Created a constructor function and adapted the code. Made the canvas a parameter on the constructor function. Added the unselectTool function to allow an alternative way to finish the shape when the tool change.	Tried adding the edit and finish buttons to the options area but only managed to get one button in there. The buttons are now at the bottom of the page, and can be interacted with tools.	Adapted the extension to a constructor function	Placing the both buttons in the options area.	Place both buttons in the options area.
	Create draw rectangle tool	Created a simple constructor function for this tool.		Created draw rectangle tool.		
16/3	Create highlighter tool	Created the tool and replaced the colour values with the colour function to introduce a lower alpha value in the colour palette. E.g. color("rgba(0, 0, 0,"+aValue+")") with AValue = 0.3	Unable to figure out how to make the alpha value change from 0.3 for the highlighter to 1.0 for all other tools	Created highlighter tool	Working highlighter colours	console.log() the colour palette to find which function controls switching colour. Potentially introduce a new function to switch to the highlighter colours values

						and back to original values.
	Place both buttons in the options area for the editable shape tool	Referred to the mirror draw tool to understand how the button was added to the options area. Used the populateOptions function and then added 2 select(."options").html() for each button.	The only button to appear would be the finish button. The order was rearranged a few times and the second button would always replace the first.		Both buttons still not in the options area.	Understand why only the second button appears. Find a method to place both buttons in the options area.
23/3	console.log() the colour palette to find which function controls switching colour. Potentially introduce a new function to switch to the highlighter colours values and back to original values.	Reverted back to the original values but also added the highlighter colour values (color("rgba(0, 0, 0,"+aValue+")")) and placed both of them in an array inside the colours array. Found that the colour changes in the colourClick function. Created an if statement that if the tool selected is highlighter, transverse the colour array and if the current colour is the same as one of the colours in the array, swap it to the corrosponding rgba colour value.	- The colour value only changes when switching colours. When going from one tool to highlighter, the colour will be the same but the alpha value, to make it more transparent, is not changed until the colour changes. - When switching to any other tool, the colour reverts back to the colour before the colour change whilst on the highlighter. - The border highlighting the selected colour does not get removed when on highlighter.	Highlighter colour now works when switching colours		- Get the colour value to change when going on highlighter. - Get the colour to stay the same when switching to other tools. - Fix the selected colour border to remove the previous colour border when changing colours on highlighter tool.
	Understand why only the second button appears. Find a method to place both buttons in the options	Only used 1 select(."options").html() and placed both button tags within in.		Both the edit and finish shape buttons are now in the options area.		

	area.					
30/3	- Get the colour value to change when going on highlighter. - Get the colour to stay the same when switching to other tools. - Fix the selected colour border to remove the previous colour border when changing colours on highlighter tool.	- Using the same solution as the eraser, created a new function in the colour palette called highlighterSelect and copied the method in colourClick. The highlighterSelect function is then called in the toolbox. - Using the same solution, created new function called allToolCSelect and called it in toolbox. Used an if statement for the highlighterSelect function, if it the tool selected is highlighter call this function, else call the allToolCSelect function.	Unable to understand why the border does not get removed from the previous colours when on the highlighter tool. It continues to add borders everytime the colour switches.	Changes to the highlighter colour when selecting the highlighter tool and stays the same colour when changing to another tool	Removing the border on the colour palette when switching colours on highlighter tool.	Read comments in the colour palette to see where and when the border changes. Try console.log().
	Create flood fill tool	Searched for the flood fill algorithm. Started with retrieving the pixel colour using get(), which returns the value as [r, g, b, a]. This is assigned to a variable which represents the colour to match. Added the respective colour values in the same format to the colours array. The flood fill algorithm was followed with a for loop used to compare the colour values. The replacement colour was set as green.		Started the flood fill tool	Completion of the tool	Complete the tool
6/4	Fix the selected colour border to remove the previous colour border when	The border changes when the colour id is selected using jQuery and the id is created when the colour is loaded. Due to this, we created a a variable called		Whilst switching colours on highlighter, it now removes the old border.		

	changing colours on highlighter tool.	borderChange and made it false initially, and made it true when switching colours in the highlighter. We converted the highlighter value back to the original string value and if borderChange is true, use the same method to remove the old border.				
	Completing flood fill	Added recusion with the difference in directions. It will only run if the colour of the pixel is not the same as the new colour, or if the colour of the next pixel is the same as the originally retrieved colour.	The tool can only fill small to medium sized areas. If the area is too large, we get an error stating the max stack call was reached.	Completed flood fill tool.	Unable to fill large areas	Find out why it is unable to fill large areas and find ways not to get the max stack call error
13/4	Find out why the flood fill tool is unable to fill large areas and find ways not to get the max stack call error	Tried removing some of the recusive call but it now only fills some of the area. Tried replacing the set() function with point() instead, but it remains the same.			Still unable to fill large areas.	
	Change the cursor	Created a new constructor function to change the appearance of the cursor. Attempted to make an ellipse the size of the tool that scales with the slider value and placed it with the mouse at the centre. Called it in the draw function of the freehand tool to test.	As the function was called in the draw function, it continously draws the ellipse whether the mouse is pressed or not. We wanted it to show at the mouse position whithout it drawing on the canvas until the mouse is pressed. We tried using conditional statements, adding loadPixels()	Used the cursor(CROSS) function for the shape tools.	An ellipse the size of the tool to represent the brush size.	

			before, but it did not change.			
	No fill on the draw rect tool	Created a button in the options area with a function to switch between fill and no fill. The function follows similarily to the editable shape buttons.	When selecting no fill, then selecting a different colour, then selecting fill again and then drawing a shape, the stroke colour would be the new colour, but the fill would be the previous colour. Created a global variable call rectFill. Much like the highlighter and floodfill colour solution, in colourClick, we created an else if fillMode == true, rectFill and stroke = selected colour. Added to else rectFill and stroke also = selected colour.	The no fill option on draw rect tool functions correctly.		
20/4	Kaleidoscope	Created a constructor function which allows users to draw kaleidoscopes. Checked the p5 documentations to find a way to rotate around the center of the screen and found angleMode. Then used translate, push and pop to replicate any drawing onto all sections. Also found pmouseX and pmouseY on the documentation which made it easier to track any drawing.		Completed the kaleidoscope function.		
	Second flood fill tool	After talking to a tutor, the max stack call error is likely to be how	Due to the nature of the flood fill algorithm, the	A working flood fill tool which can now		

		javascript works rather than the code itself. We tried implementing the "forest fire" algorithm. The first half of the algorithm is the same, so the code only needed to be adapted a little. A queue is created and the first mouse click position is pushed to the queue. While the queue is not empty, assign a variable to the head and then shift it. Compare the colours to left, right, below, and above the variable, and if the colour is the same, replace the colour and push it to the queue.	time it takes for it to complete can be slow for large areas.	fill large areas.		
27/4	Clone stamp tool	Looked on p5 documentation to see if there was a function which can help and found get(). Considering we needed to press Ctrl and mouse click at same time, we implemented mouse/key press/released. Used an if statement that if both were pressed at the same time, record the mouse position. As mClick is still true, the condition is used again to record the new mouse position and distance away from the original mouse position. When the mouse is pressed, it will begin copying from the original image position. Made a rectangle the same size as the area being copied and made it show when the image is being copied.	The area being copied was initially set at 10x10 pixels as a test. We wanted the size to also scale with the slider, but 10x10 would already be the maximum value on the scale. The value was then using the arithmetic sequence.	Created a cloning tool		
	Scale the spray can tool with the	Tested different values for the spread and slider value. We didn't	On different slider values, the colour seems to be a	The spray can tool now scales with		

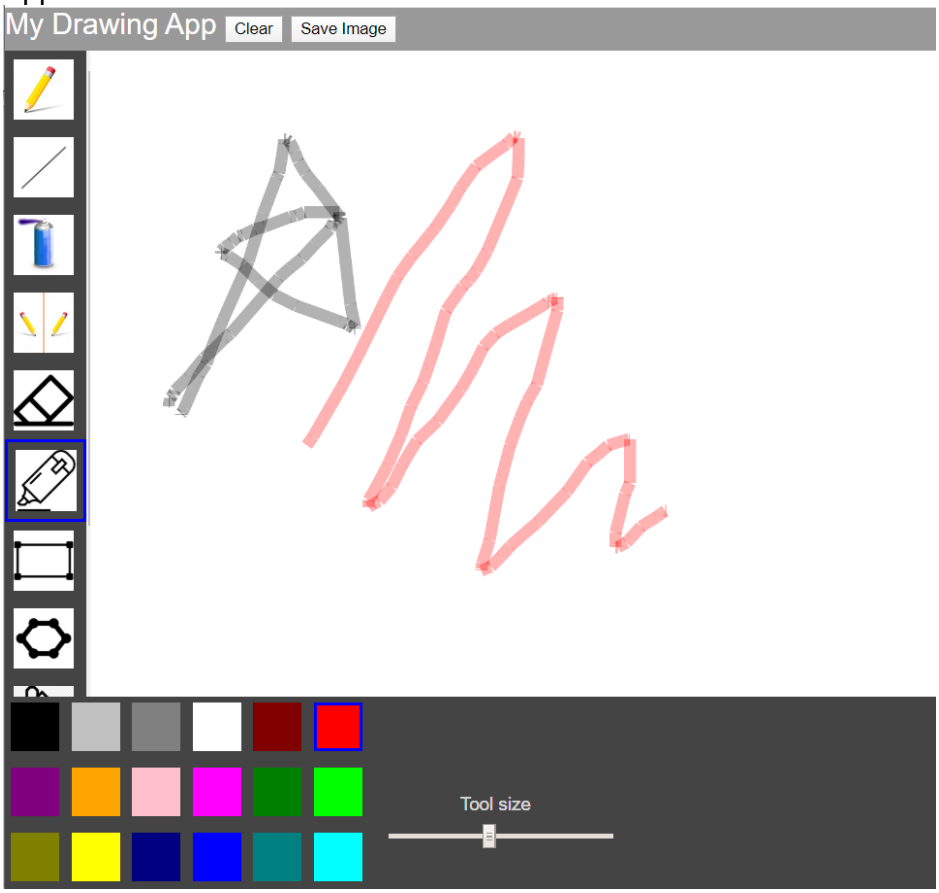
	slider	want the stroke size to be too big, so the arithmetic sequence was used in the toolSize function. Similiarly, we didn't want the spread to scale too high and used the sequence again.	little transparent. Not too sure why that is the case.	the slider.		
	Slider in options area	Added a class to the slider. In the populateOptions of each tool, added the slider as a child to the .options.	The slider would not appear on tools with buttons, when called at the begining of the populateOptions function. When place at the end of the function, the slider appears next to the button. Had to use CSS to place the slider at the bottom of the options area.	Slider is now in options area and not at the bottom		
4/5	Add text for the slider	Used createDiv() to have it say tool size. This text will be placed above the slider to allow users know what the slider's purpose is. Create a class and made it a child of .options. Rather than placing in the populateOptions function of ever tool, we made a function in the helperFunction with the slider and called it in every tool.		Text above the slider says "tool size"		
	Add instructions for clone tool	Similar to the text for the slider, but left the function in the clone tool as we are only using this once.		Text in the options area helping how to get started with the tool.		
	Remove constructor	Removed the constructor functions and place the methods in the		Removed constructor		

	function - cursor and tool size	helperFunction. Removed the slider function in every tool and updated it with the helperFunction		function - cursor and tool size		
	Removing global variables	The global variables were mostly used for the highlighter, flood fill, and draw rectangle tool. The global variables were assigned colour values and would be called in the tools. They have been removed and placed as property of the colour palette. In the tools, the global variables have replaced with colourP.selectedColour. We were also able to remove further redundant code in the colour palette which used the global variables, such as the border change for the highlighter tool.		No more non-essential global variables		
	Removed eraser and highlighter constructor function	As both of these tools are similar to the freehandtool, we added some if statements for if the name of the object was highlighter or eraser. 2 parameter were added to the function, name and icon picture. 2 new freehand objects are created in sketch, with appropriate parameters		Removed eraser and highlighter constructor function		
	Removed draw rect constructor function	Same idea as the freehandtool, except placed in the lineToTool. Added if statements with functions for if the tool is draw rect		Removed draw rect constructor function		

(Appendix 2)

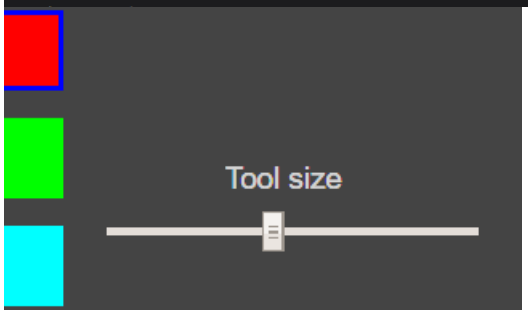
```
9
10      //colours[i][0] are colours for most tools
11      //colours[i][1] are colours specifically for the highlighter
12      //colours[i][2] are colours specifically for flood fill
13      this.colours = [
14          ["black",
15           color("rgba(0, 0, 0,"+aValue+")"),
16           [0, 0, 0, 255]],
17
18          ["silver",
19           color("rgba(192, 192, 192,"+aValue+")"),
20           [192, 192, 192, 255]],
21
22          ["grey",
23           color("rgba(128, 128, 128,"+aValue+")"),
24           [128, 128, 128, 255]],
25
26          ["white",
27           color("rgba(255, 255, 255,"+aValue+")"),
28           [255, 255, 255, 255]],
29
30          ["maroon",
31           color("rgba(128, 0, 0,"+aValue+")"),
32           [128, 0, 0, 255]],
33
34          ["red",
35           color("rgba(255, 0, 0,"+aValue+")"),
36           [255, 0, 0, 255]],
37
38          ["purple",
39           color("rgba(128, 0, 128,"+aValue+")"),
40           [128, 0, 128, 255]],
41
42          ["orange",
43           color("rgba(255, 165, 0,"+aValue+")"),
44           [255, 165, 0, 255]],
45
46          ["pink",
47           color("rgba(255, 192, 203,"+aValue+")"),
48           [255, 192, 203, 255]],
49
```

Appendix 3

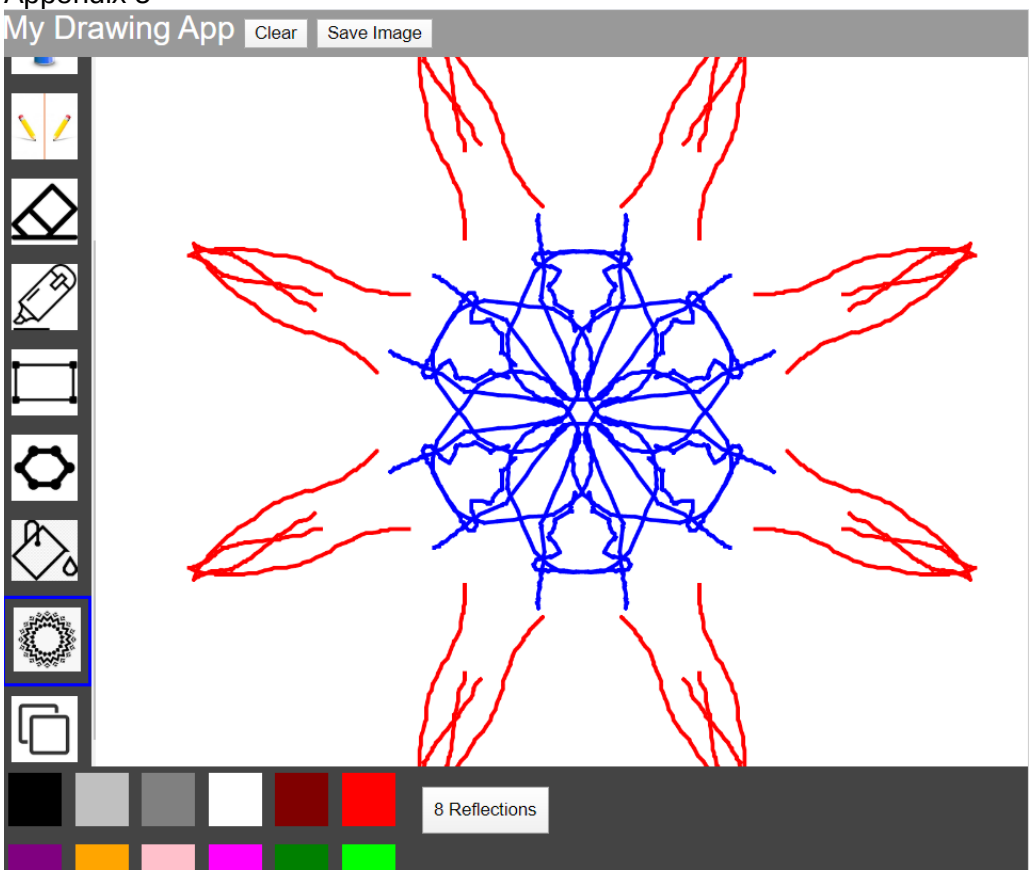


Appendix 4

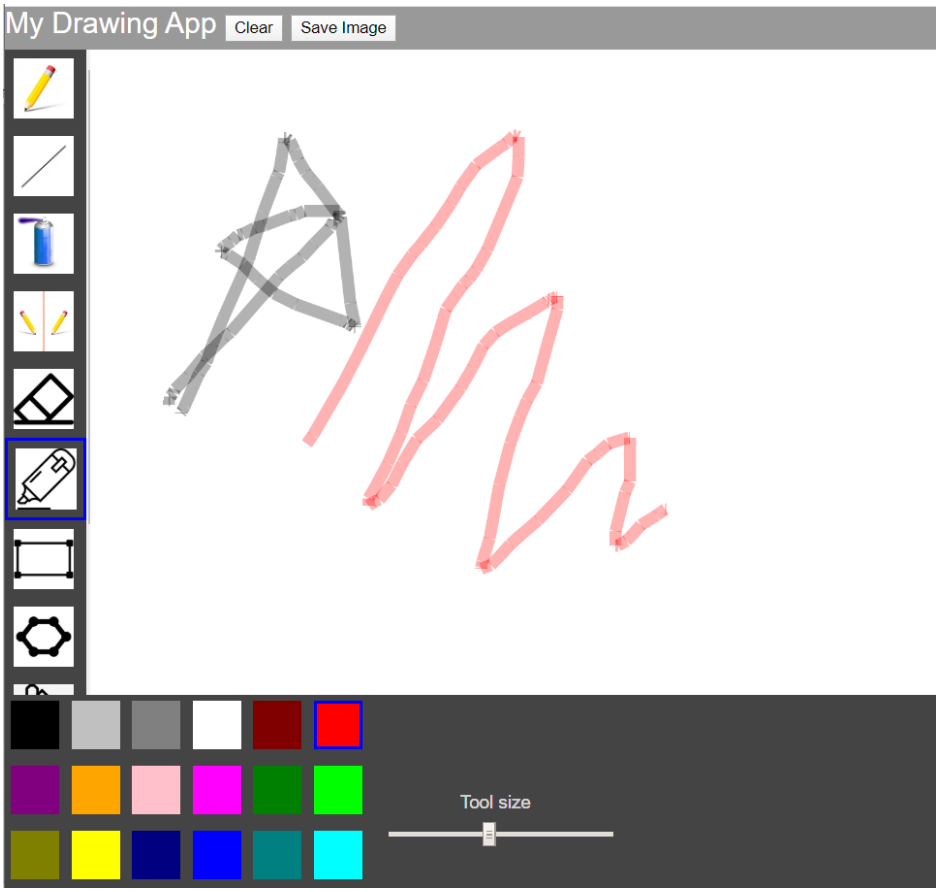
```
22      //Adds the slider to the options area
23      this.slider = function(){
24          select(".options").child(slider);
25
26          //Created text above the slider
27          var text = createDiv("Tool size");
28          text.class("sliderText");
29          select(".options").child(text);
30      }
31
32      //Adjust the stroke size of drawing tools by using the slider
33      this.toolSize = function(){
34
35          var val = slider.value();
36
37          // The stroke size will be from 1-10 depending where the user changes the slider.
            If the eraser is selected, it will be x6 as big
38          if(toolbox.selectedTool.name == "eraser"){
39              strokeWeight(val * 6);
40          }
41
42          // If the spray can tool is selected, the size of each dot changes with the
            slider..
43          else if(toolbox.selectedTool.name == "sprayCanTool"){
44              strokeWeight(1 + (val - 1) * 0.16);
45          }
46
47          else if(toolbox.selectedTool.name == "highlighter"){
48          strokeWeight(val * 2);
49      }
50
51          //ALl other tools will use the default values of 1-10
52          else{
53              strokeWeight(val);
54          }
55      }
56
57      //Changes the cursor to a cross
58      this.cross = function(){
59          cursor(CROSS);
60      }
61
62      //Changes the cursor to an arrow
63      this.arrow = function() {
64          cursor(ARROW)
```



Appendix 5

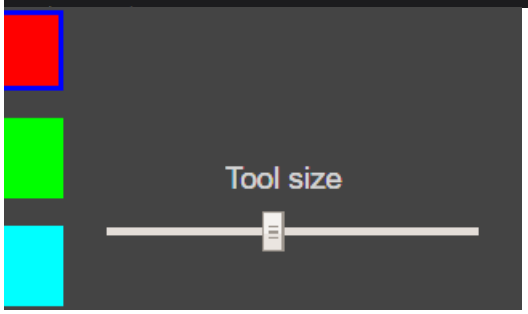


Appendix 3

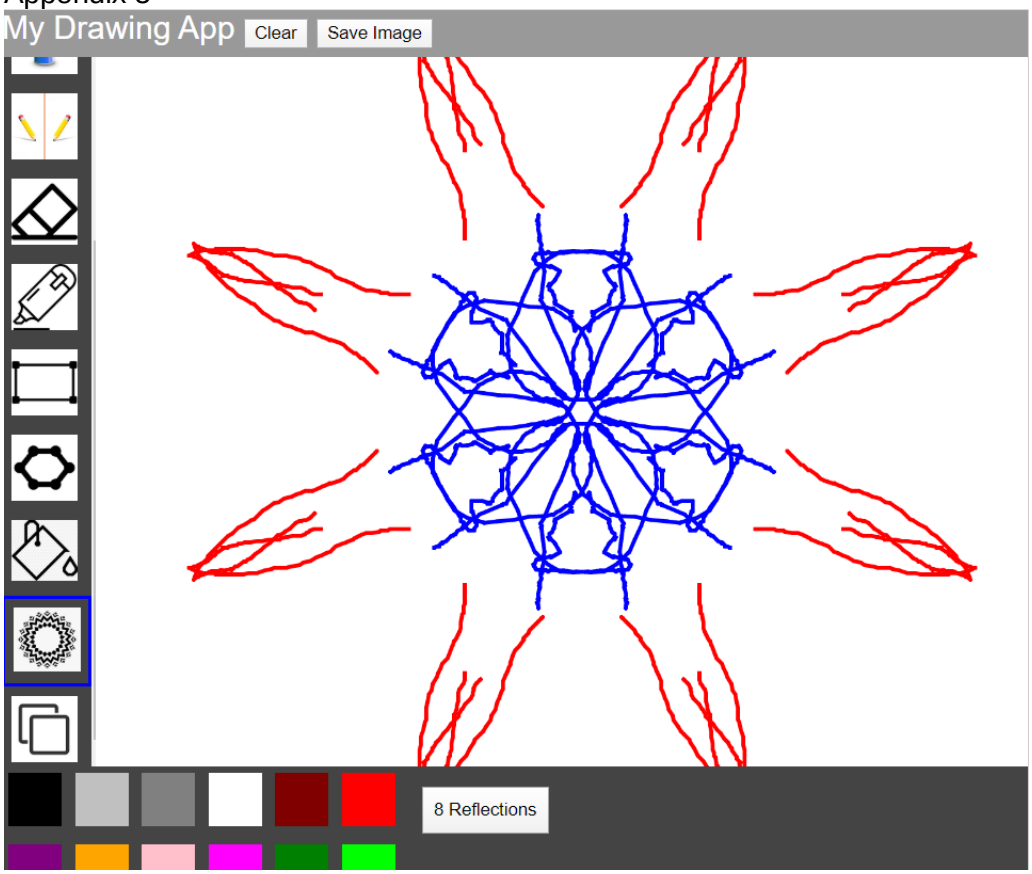


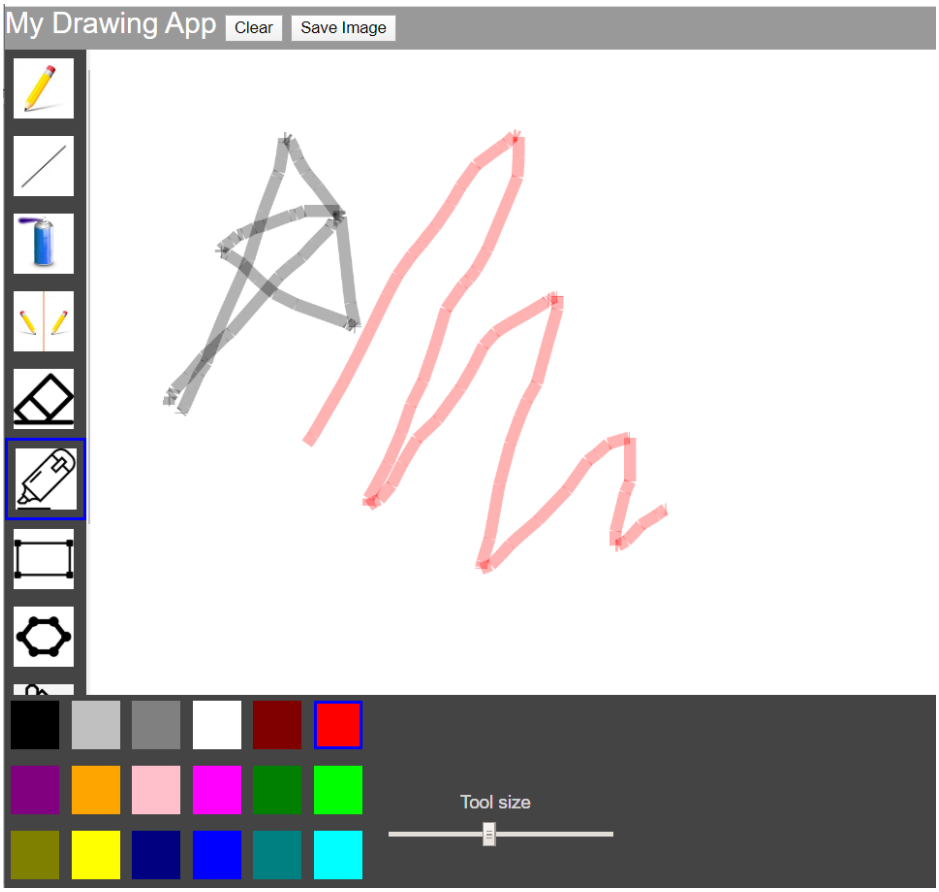
Appendix 4

```
22      //Adds the slider to the options area
23      this.slider = function(){
24          select(".options").child(slider);
25
26          //Created text above the slider
27          var text = createDiv("Tool size");
28          text.class("sliderText");
29          select(".options").child(text);
30      }
31
32      //Adjust the stroke size of drawing tools by using the slider
33      this.toolSize = function(){
34
35          var val = slider.value();
36
37          // The stroke size will be from 1-10 depending where the user changes the slider.
            If the eraser is selected, it will be x6 as big
38          if(toolbox.selectedTool.name == "eraser"){
39              strokeWeight(val * 6);
40          }
41
42          // If the spray can tool is selected, the size of each dot changes with the
            slider..
43          else if(toolbox.selectedTool.name == "sprayCanTool"){
44              strokeWeight(1 + (val - 1) * 0.16);
45          }
46
47          else if(toolbox.selectedTool.name == "highlighter"){
48          strokeWeight(val * 2);
49      }
50
51          //ALl other tools will use the default values of 1-10
52          else{
53              strokeWeight(val);
54          }
55      }
56
57      //Changes the cursor to a cross
58      this.cross = function(){
59          cursor(CROSS);
60      }
61
62      //Changes the cursor to an arrow
63      this.arrow = function() {
64          cursor(ARROW)
```



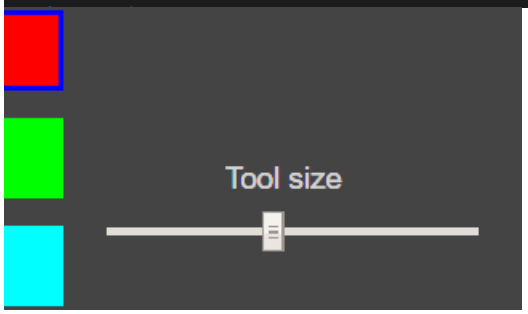
Appendix 5



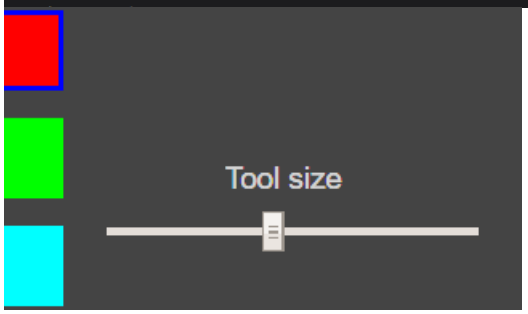


Appendix 4

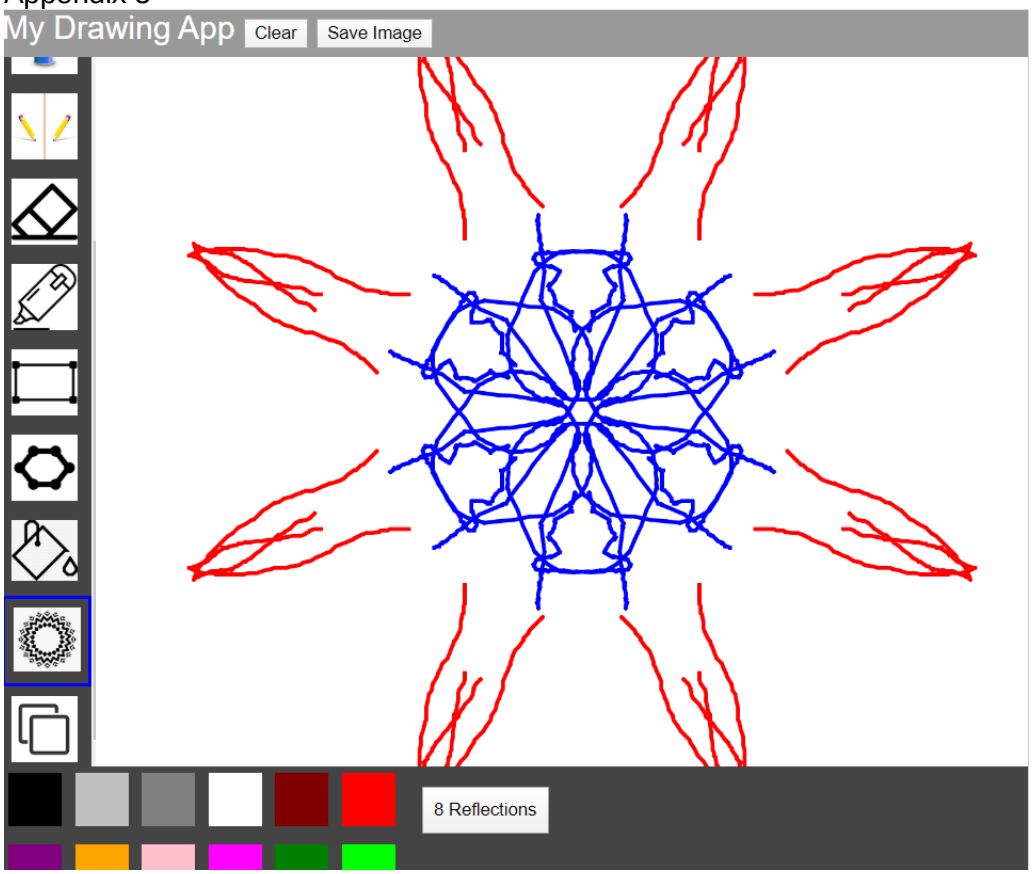
```
22      //Adds the slider to the options area
23      this.slider = function(){
24          select(".options").child(slider);
25
26          //Created text above the slider
27          var text = createDiv("Tool size");
28          text.class("sliderText");
29          select(".options").child(text);
30      }
31
32      //Adjust the stroke size of drawing tools by using the slider
33      this.toolSize = function(){
34
35          var val = slider.value();
36
37          // The stroke size will be from 1-10 depending where the user changes the slider.
            If the eraser is selected, it will be x6 as big
38          if(toolbox.selectedTool.name == "eraser"){
39              strokeWeight(val * 6);
40          }
41
42          // If the spray can tool is selected, the size of each dot changes with the
            slider..
43          else if(toolbox.selectedTool.name == "sprayCanTool"){
44              strokeWeight(1 + (val - 1) * 0.16);
45          }
46
47          else if(toolbox.selectedTool.name == "highlighter"){
48          strokeWeight(val * 2);
49      }
50
51          //ALl other tools will use the default values of 1-10
52          else{
53              strokeWeight(val);
54          }
55      }
56
57      //Changes the cursor to a cross
58      this.cross = function(){
59          cursor(CROSS);
60      }
61
62      //Changes the cursor to an arrow
63      this.arrow = function() {
64          cursor(ARROW)
```

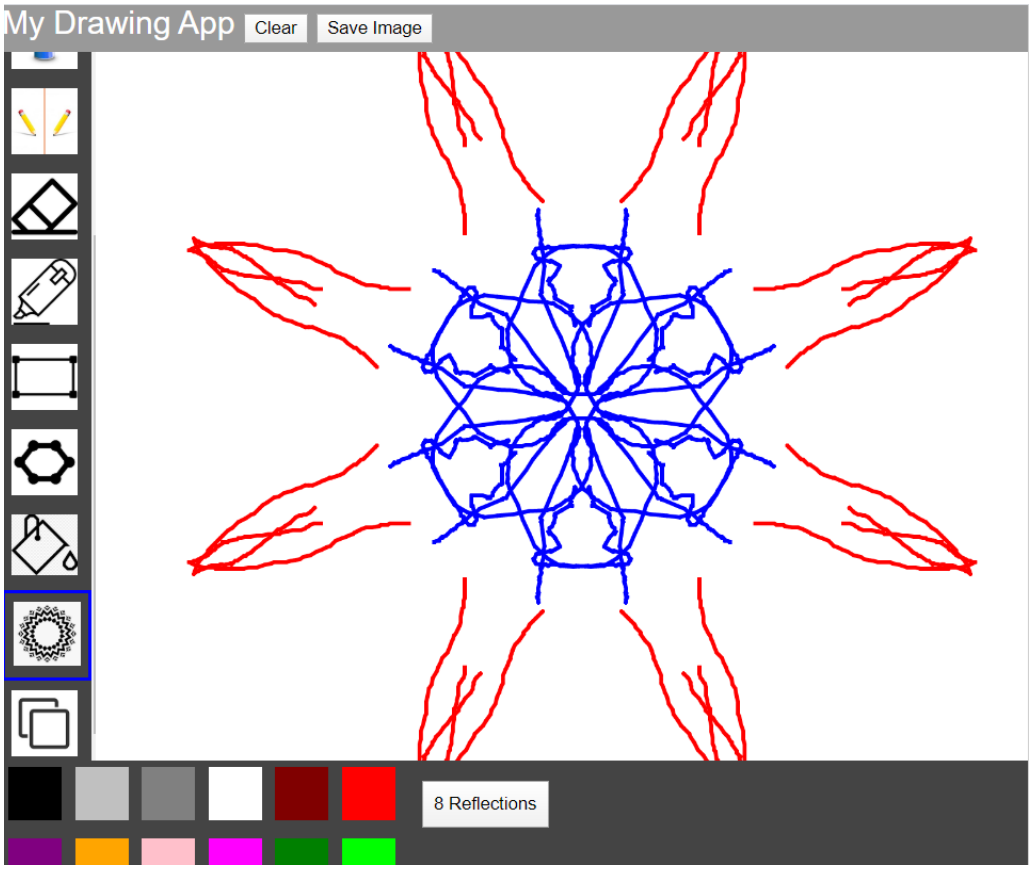


```
22      //Adds the slider to the options area
23      this.slider = function(){
24          select(".options").child(slider);
25
26          //Created text above the slider
27          var text = createDiv("Tool size");
28          text.class("sliderText");
29          select(".options").child(text);
30      }
31
32      //Adjust the stroke size of drawing tools by using the slider
33      this.toolSize = function(){
34
35          var val = slider.value();
36
37          // The stroke size will be from 1-10 depending where the user changes the slider.
            If the eraser is selected, it will be x6 as big
38          if(toolbox.selectedTool.name == "eraser"){
39              strokeWeight(val * 6);
40          }
41
42          // If the spray can tool is selected, the size of each dot changes with the
            slider..
43          else if(toolbox.selectedTool.name == "sprayCanTool"){
44              strokeWeight(1 + (val - 1) * 0.16);
45          }
46
47          else if(toolbox.selectedTool.name == "highlighter"){
48          strokeWeight(val * 2);
49      }
50
51          //ALl other tools will use the default values of 1-10
52          else{
53              strokeWeight(val);
54          }
55      }
56
57      //Changes the cursor to a cross
58      this.cross = function(){
59          cursor(CROSS);
60      }
61
62      //Changes the cursor to an arrow
63      this.arrow = function() {
64          cursor(ARROW)
```

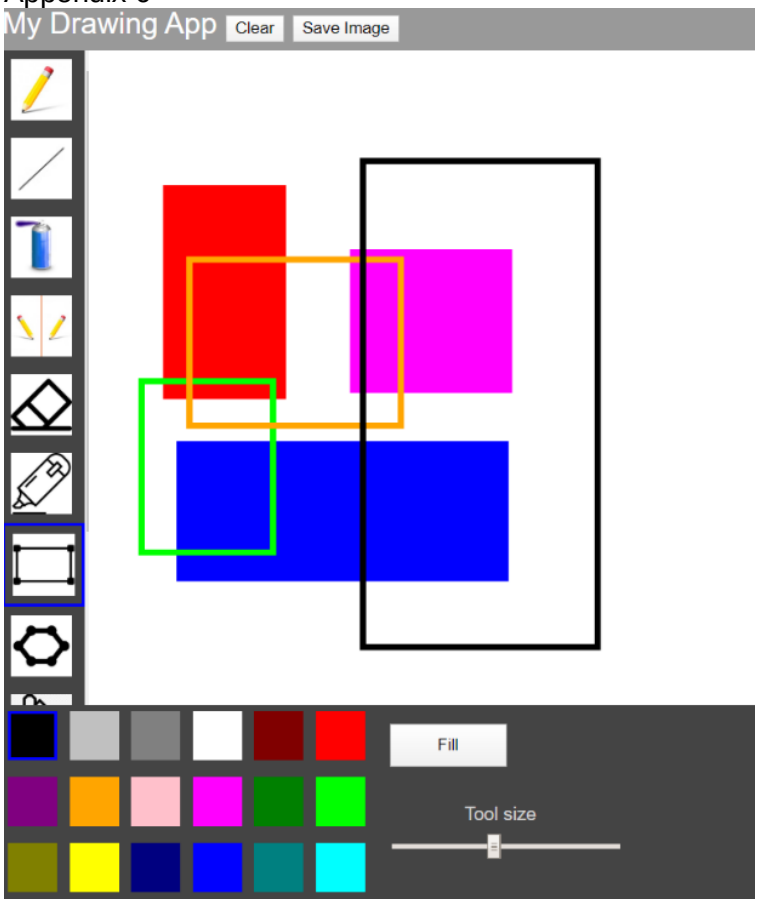


Appendix 5

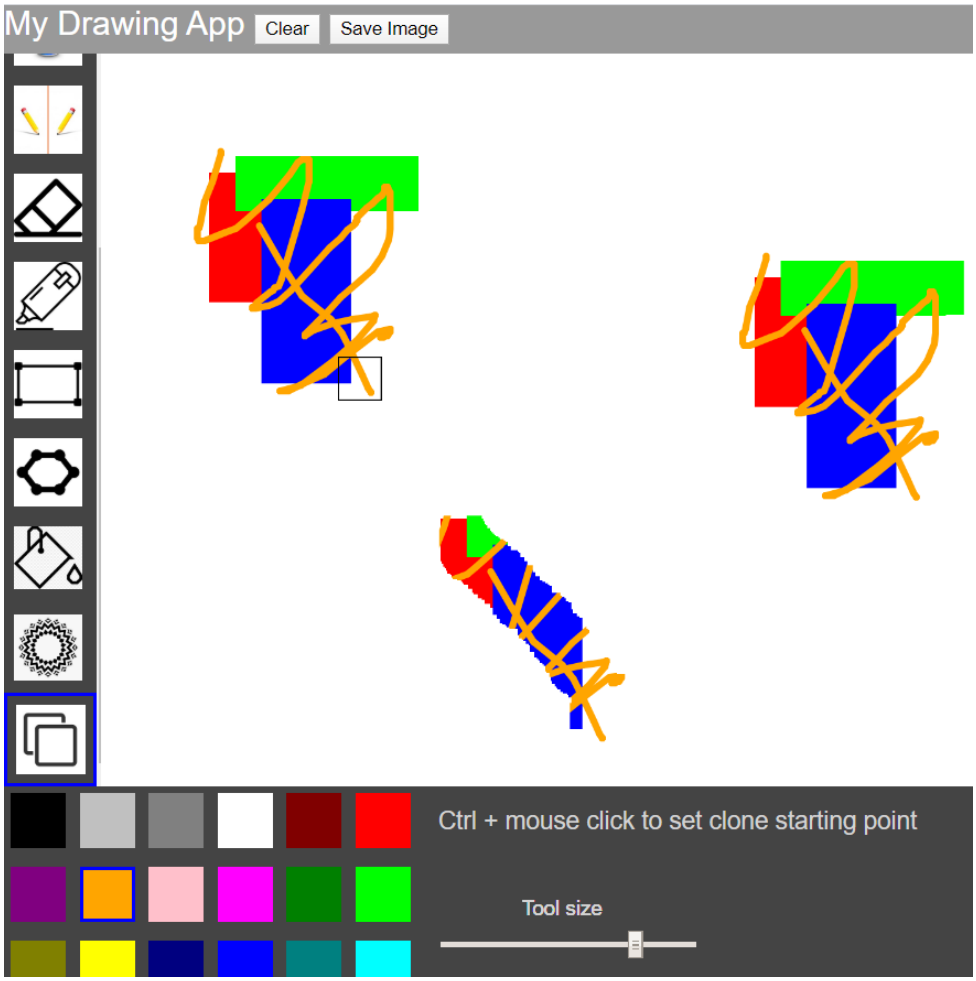




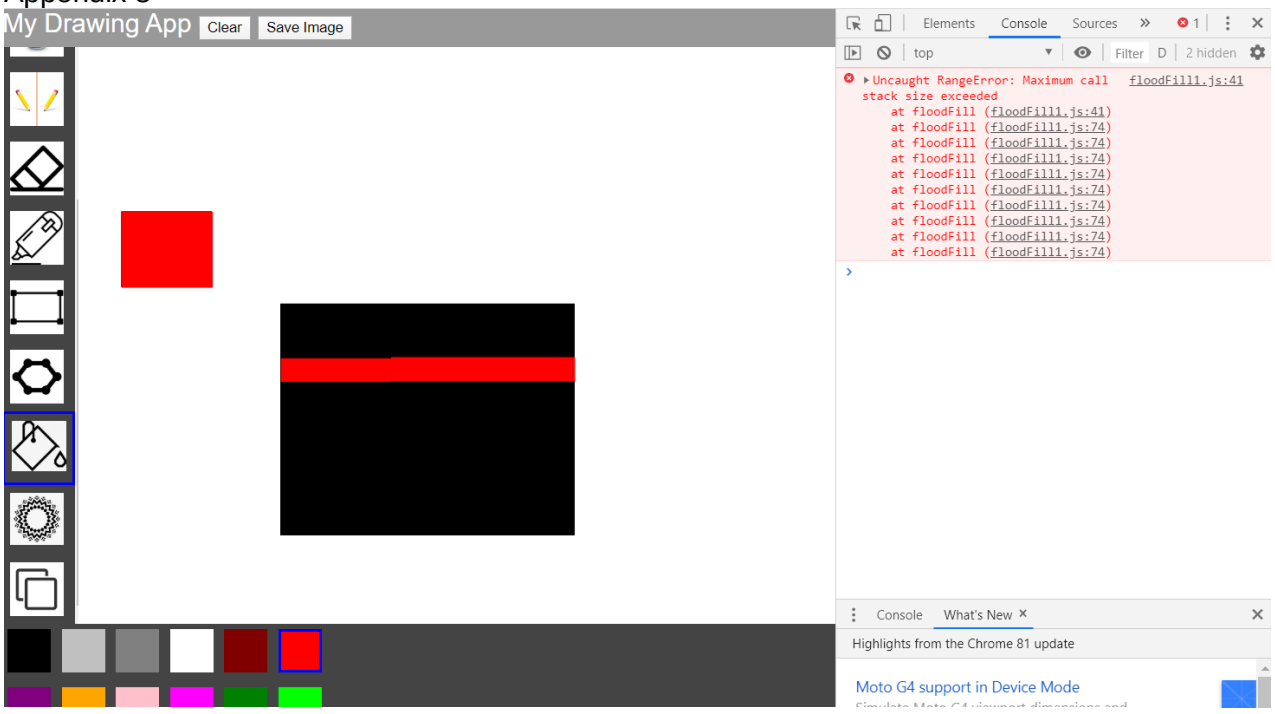
Appendix 6



Appendix 7


My Drawing App
Clear
Save Image
Ctrl + mouse click to set clone starting point
Tool size

Appendix 8


My Drawing App
Clear
Save Image
Elements
Console
Sources
Uncaught RangeError: Maximum call stack size exceeded
floodFill1.js:41
at floodFill (floodFill1.js:41)
at floodFill (floodFill1.js:74)
Console
What's New
Highlights from the Chrome 81 update
Moto G4 support in Device Mode

(Appendix 9

Appendix 10

Appendix 1

```
141     //Used to change the colour from colours[0] to colours[1] when the highlighter tool is
        selected
142     this.highlighterSelect = function(){
143         for(var i = 0; i < self.colours.length; i++){
144             if(self.selectedColour == self.colours[i][0]){
145                 highColour = self.colours[i][1]
146                 fill(highColour)
147                 stroke(highColour)
148             }
149         }
150     }
151
152     //Used to change the colour from colours[0] to colours[2] when the flood fill tool is
        selected
153     this.floodFillSelect = function(){
154         for(var i = 0; i < self.colours.length; i++){
155             if(self.selectedColour == self.colours[i][0]){
156                 var f = self.colours[i][2]
157                 self.fillToolColour = f
158             }
159         }
160     }
161
162     //Used to ensure that the colour is same when changing tools
163     this.allToolCSelect = function(){
164         fill(self.selectedColour);
165         stroke(self.selectedColour);
166     }
167
168     //added this so we can set the colour back to the selected colour after the eraser is
        used
169     this.setColour = function(c) {
170         self.selectedColour = c;
171         fill(c);
172         stroke(c);
173     }
174
```

Appendix 12

```
    var aAlpha = 255
    var hAlpha = 100;

    this.colours = {
        black: [0, 0, 0, aAlpha],
        silver: [192, 192, 192, aAlpha],
        grey: [128, 128, 128, aAlpha],
        white: [255, 255, 255, aAlpha],
        maroon: [128, 0, 0, aAlpha],
        red: [255, 0, 0, aAlpha],
        purple: [128, 0, 128, aAlpha],
        orange: [255, 165, 0, aAlpha],
        pink: [255, 192, 203, aAlpha],
        fuchsia: [255, 0, 255, aAlpha],
        green: [0, 128, 0, aAlpha],
        lime: [0, 255, 0, aAlpha],
        olive: [128, 128, 0, aAlpha],
        yellow: [255, 255, 0, aAlpha],
        navy: [0, 0, 128, aAlpha],
        blue: [0, 0, 255, aAlpha],
        teal: [0, 128, 128, aAlpha],
        aqua: [0, 255, 255, aAlpha]
    };


    var colourClick = function() {

        //remove the old border
        for(var i in self.colours){
            var current = select("#" + i + "Swatch");
            current.style("border", "0");
        }

        //get the new colour from the id of the clicked element
        var c = this.id().split("Swatch")[0];

//        If the highligter tool is selected, the the alpha value will change to 100.
        if(toolbox.selectedTool.name == "highlighter"){
            for(var i in self.colours){
                if(c == i){
                    var convert = self.colours[i]
                    convert.pop()
                    convert.push(hAlpha)
                    highColour = convert
                    fill(highColour)
                    stroke(highColour)
                }
            }
        }
//
```