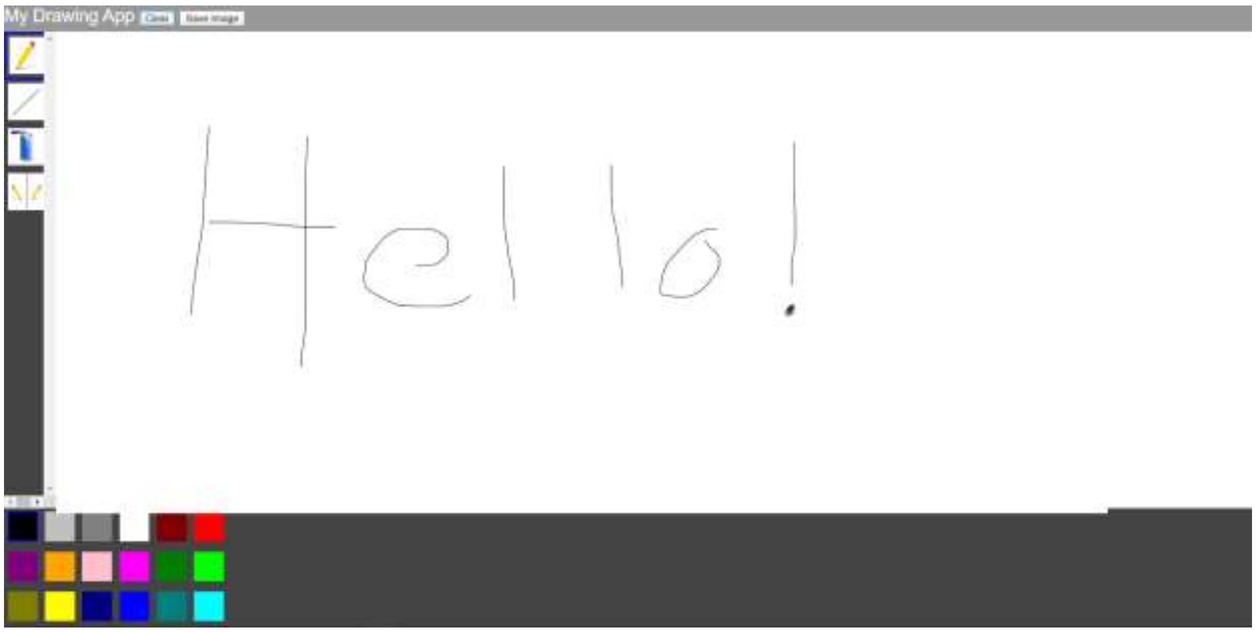
# Drawing Application



# Analysis

## Brief

The idea of this project is to create a multifunctional drawing application which can be used for various purposes such as tech-savvy creatives or casual freehand drawers.

## Computational Methods and Thinking

Using computational thinking we took our complex problems and broke it down into a series of small, more manageable problems which is known as decomposition. We split it up in sections such as first understanding how the code works and then actually developing features into it. We realised each of these smaller problems can then be looked at individually, we used pattern recognition to look for similarities among and within problems. For example, the line tool element of the program would be quite similar to the shapes tool which means we would be able to work on this much faster. Because of computational thinking, we created the first draft of our code quickly and efficiently.

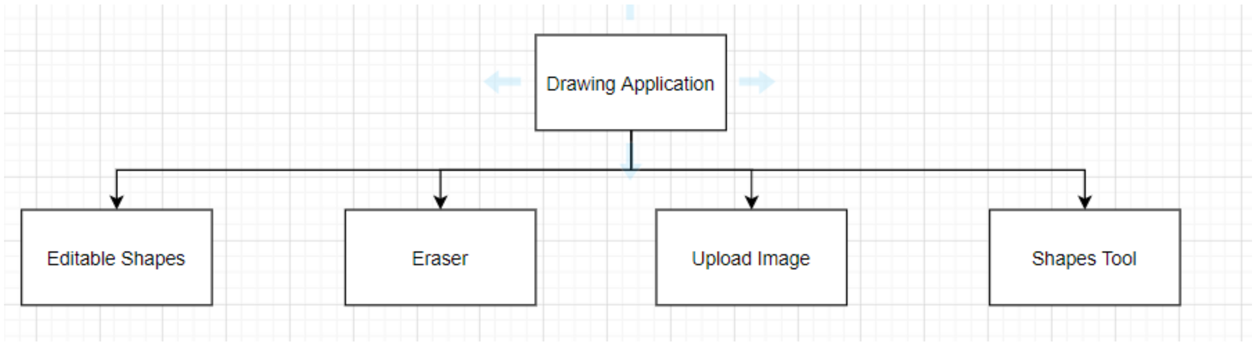
# Planning

Our overall aim is to create a simple but effective drawing application. We want it to consist of a range of tools that users can make use of, and at the same time, where each tool is completely bug free. We will also need to consider the technologies that we are familiar with and that are most suited to the job. In this scenario, the JavaScript library will be the most beneficial as what we will be using is p5.js. This is a JS client-side library for creating graphic and interactive experiences - perfect for what we want to develop.

Now that we have identified what we want to create and what we are going to create it with. We need to think of what features the application will include. We touched on some of them in the analysis section but will go into more here. We believe that all drawing applications should include features like erasers, shapes, text inputs and a range of colours. This is because these are just the basics that the target audience would require. However, it is unlikely that we will be able to build all of these tools but will aim to develop as much as we can. Tools like the eraser and shapes should be achievable as we can create them with our previous knowledge; whereas others will require extensive research. Our mission is to have at least 4-5 different tools in our application.

We now have an idea of what features we'll be building first in our program so now we can start coding. To find out which variables and functions we should write out first, we should break it into categories and decompose them.

## Decomposition



### Architecture for our Drawing Application

<u>Scenes:</u>

- **Editable Shapes**
- **Eraser**
- **Upload Image**
- **Shapes Tool**
- **Background Colour**
- **AddText Tool**
- **StickerTool**

<u>Objects:</u>

*These set an icon and name for the objects*

- **this.icon**
- **this.name**

*Gets called repeatedly*

- **this.draw**

*Functions to show and hide buttons when tool is pressed and unpressed*

- **this.populateOptions**
- **this.unselectTool**

<u>User interaction:</u>

*Detects mouse clicks*

- **mousePressOnCanvas**
- **mouseIsPressed**
- **mouseClicked**

*User enters text*

- **textInput**

*Loads image*

- **selectFile**

Here is our timeline for our Drawing Application, assuming we do 2-4 hours of work each week:

- Week 1: Understand the code given to us and develop first feature
- Week 2: Add in first feature into main code and start research for next one
- Week 3: Start to code second feature
- Week 4: Clean and maintain current development for peer review
- Week 5: Finish off second feature while also testing for any bugs
- Week 6: Move on to third feature with both research and development being done here
- Week 7: Finalise third feature and move on to next tool
- Week 8: Finish fourth feature
- Week 9: Polish everything up to this point, testing for any bugs and then fixing them
- Week 10: Finalising any issues and ensuring code is well structured and commented
- Week 11: Deadline for submission

This is just a rough estimate of how we believe our development will be. Some things may take longer than we expect while others that seem hard may take less. We aim to have at least a week buffer to ensure everything is satisfactory and up to scratch for the deadline.

## Development and Testing

### Approach to Testing

We will be developing the software using an iterative design methodology. This is where a product is tested and changed repeatedly at different stages of the design or development. The aim of this method is to eliminate usability issues before the application is launched. The software will be broken down into separate, smaller problems which are then much easier to solve on their own. After each iteration has been completed it will be tested so that eventually it will contain enough functionality that would result in the program being completed. By using this method, it will help show how we will test the program as it is being developed and then how we will test the entirety of the program at the very end of development. Our testing comes under our weekly logs - it will give an understanding of our progress and what our iterations are.

## Weekly Logs

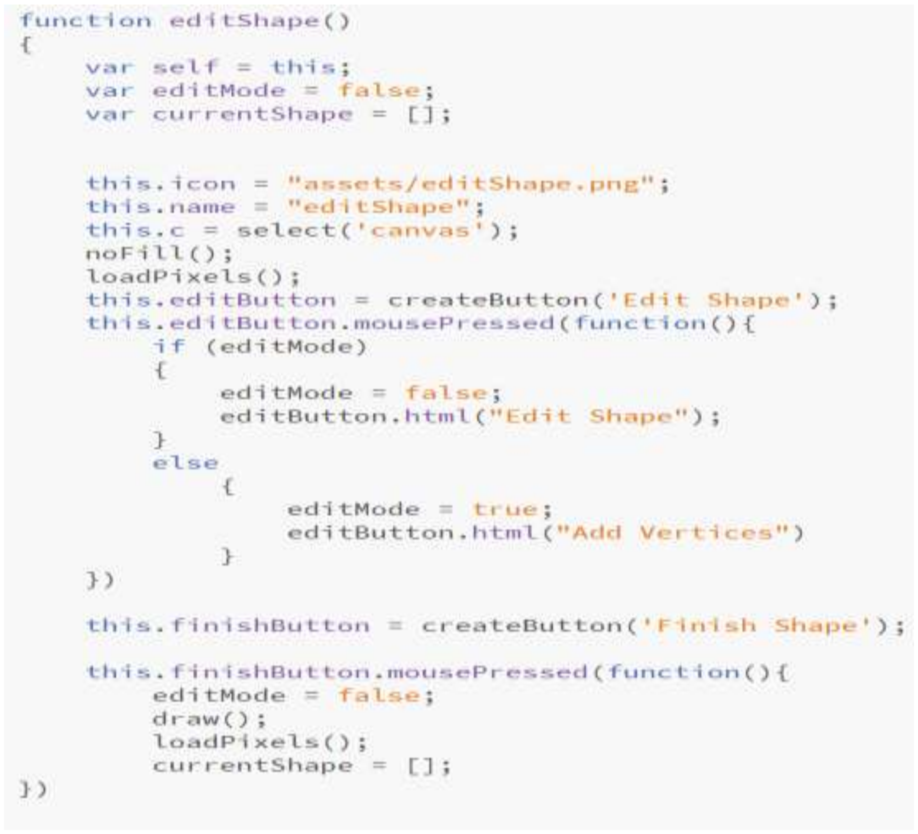
**Week 1 (24/02/2020)**

During the first week, our main aim was to fully understand the code. This is because we would then be able to implement extra features as well as develop existing ones in a much quicker and efficient way. Analysing how things work and why they are used is important so that we can progress much quicker with the tasks.

After this, we used the Editing Shapes video to create our first feature of the Drawing Application. However; this was just a standalone program, our next step was to actually embed it into the main application.

So far, we are satisfied with our progress in the first week. We've understood how the existing features are working with the main program and also created our next feature. Our target is now to implement that into the main application while also thinking of trying to extend that or add even more features.

- Coded Edited Shapes video
- Aim to next embed into main application and think of more features



```
function editShape()
{
    var self = this;
    var editMode = false;
    var currentShape = [];


    this.icon = "assets/editShape.png";
    this.name = "editShape";
    this.c = select('canvas');
    noFill();
    loadPixels();
    this.editButton = createButton('Edit Shape');
    this.editButton.mousePressed(function(){
        if (editMode)
        {
            editMode = false;
            editButton.html("Edit Shape");
        }
        else
            {
                editMode = true;
                editButton.html("Add Vertices")
            }
    })

    this.finishButton = createButton('Finish Shape');

    this.finishButton.mousePressed(function(){
        editMode = false;
        draw();
        loadPixels();
        currentShape = [];
})
```

**Week 2 (02/03/2020)**
In week 2, we have started to add in the Edited Shapes feature into the main application and are close to finalising it. We have encountered a few bugs such as when using this feature, the canvas wouldn't clear when using the button as well as when clicking off canvas, the drawing would follow the mouse when it shouldn't. So while we try to debug this, we also have researched on how to add an upload function to our application so that the users of the application are able to add images that they may want to work on. It was quite difficult implementing the Edited Shapes into the application as it required more functions and objects to make it work but in the end we got the majority of the tool to work with a few bugs to iron out.

- Finalising Edited Shapes feature into main application

- Few bugs:
  *canvas would not clear when using the button as well as when clicking off canvas - drawing would follow the mouse when it shouldn't*
- Researched upload image tool idea

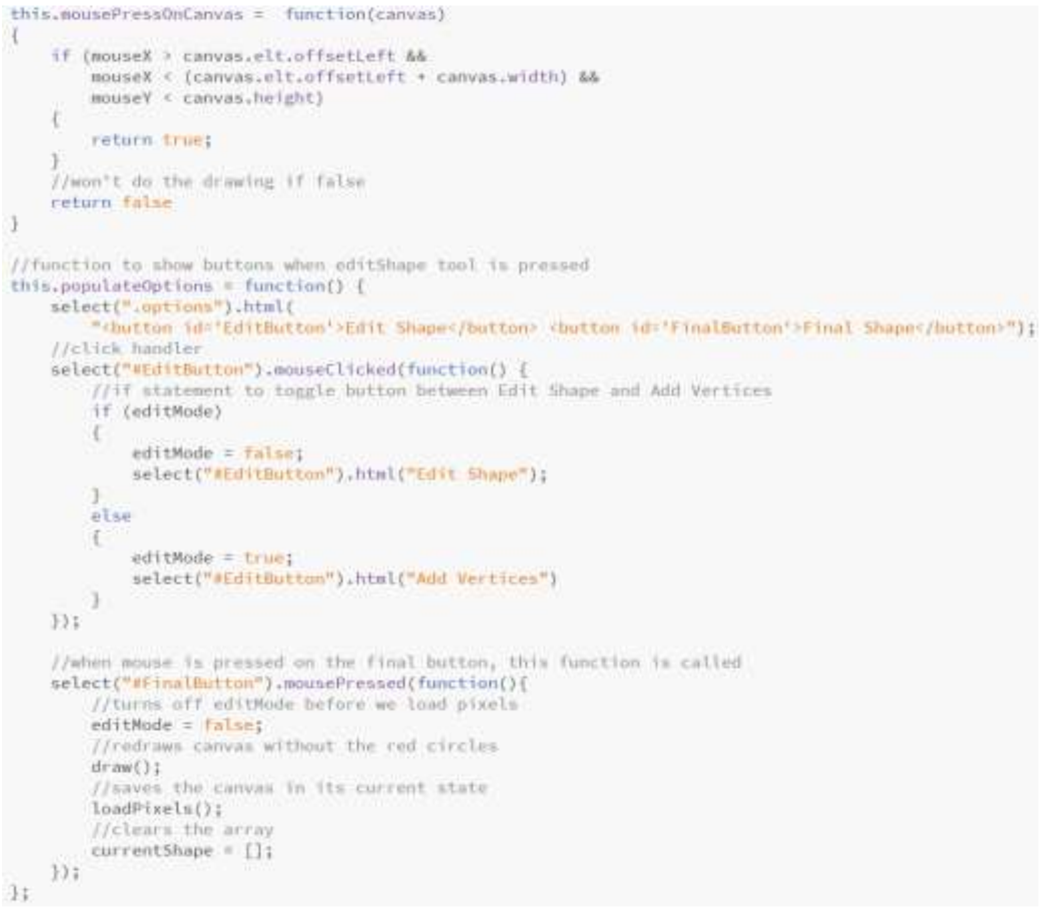
**<u>Week 3 (09/03/2020)</u>**

During week 3, we spent a lot of time trying to figure out how to fix these issues but when we fixed one thing, something else broke. With the help of the teaching assistants, we were able to fix these problems. Now with this complete, we moved on to coding the upload image tool. We thought of two methods of carrying this out. One was making use of the helperFunctions so that users can choose to upload images as default in the same position to where the Clear and Save Image buttons are. Or make it as a normal enhancement, to which the user will have to press the tool icon and then press another button to be able to open their File Explorer. To improve usability we went with method two but were unable to get the Choose File button to work. Despite being stuck, we are satisfied with our progress and hope to have at least two fully functional tools for next time.

- Week 2 bugs ironed out with help from teaching assistants
- Partially coded upload image tool
- Difficulty with getting Choose File button to work
- Hard coded Choose File button position - unsuitable for all screen resolutions

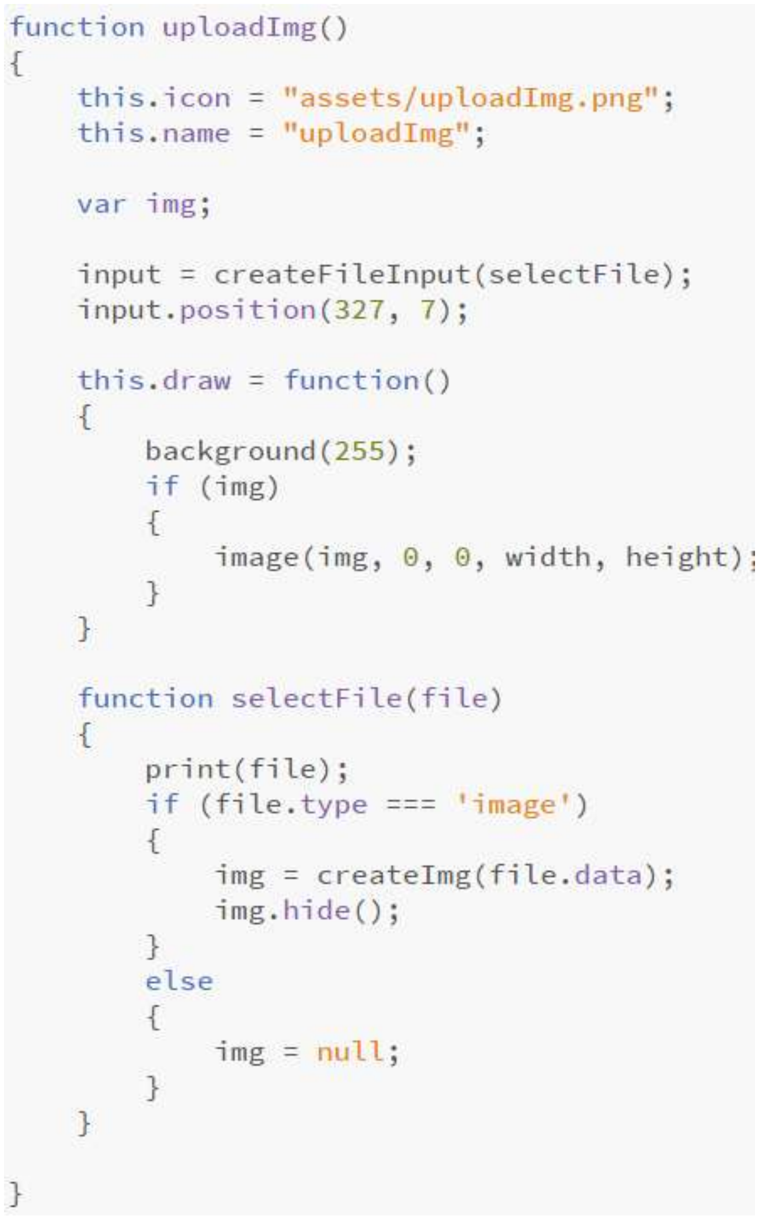
**How we fixed Edited Shapes bugs:**

We had to create something to detect if the mouse press is on the canvas or not while also altering the editMode through our populateOptions function.



```
this.mousePressOnCanvas =  function(canvas)
{
    if (mouseX > canvas.elt.offsetLeft &&
        mouseX < (canvas.elt.offsetLeft + canvas.width) &&
        mouseY < canvas.height)
    {
        return true;
    }
    //won't do the drawing if false
    return false
}

//function to show buttons when editShape tool is pressed
this.populateOptions = function() {
    select(".options").html(
        "<button id='EditButton'>Edit Shape</button> <button id='FinalButton'>Final Shape</button>");
    //click handler
    select("#EditButton").mouseClicked(function() {
        //if statement to toggle button between Edit Shape and Add Vertices
        if (editMode)
        {
            editMode = false;
            select("#EditButton").html("Edit Shape");
        }
        else
        {
            editMode = true;
            select("#EditButton").html("Add Vertices")
        }
    });

    //when mouse is pressed on the final button, this function is called
    select("#FinalButton").mousePressed(function(){
        //turns off editMode before we load pixels
        editMode = false;
        //redraws canvas without the red circles
        draw();
        //saves the canvas in its current state
        loadPixels();
        //clears the array
        currentShape = [];
    });
};
```

**Partially coded upload image tool:**

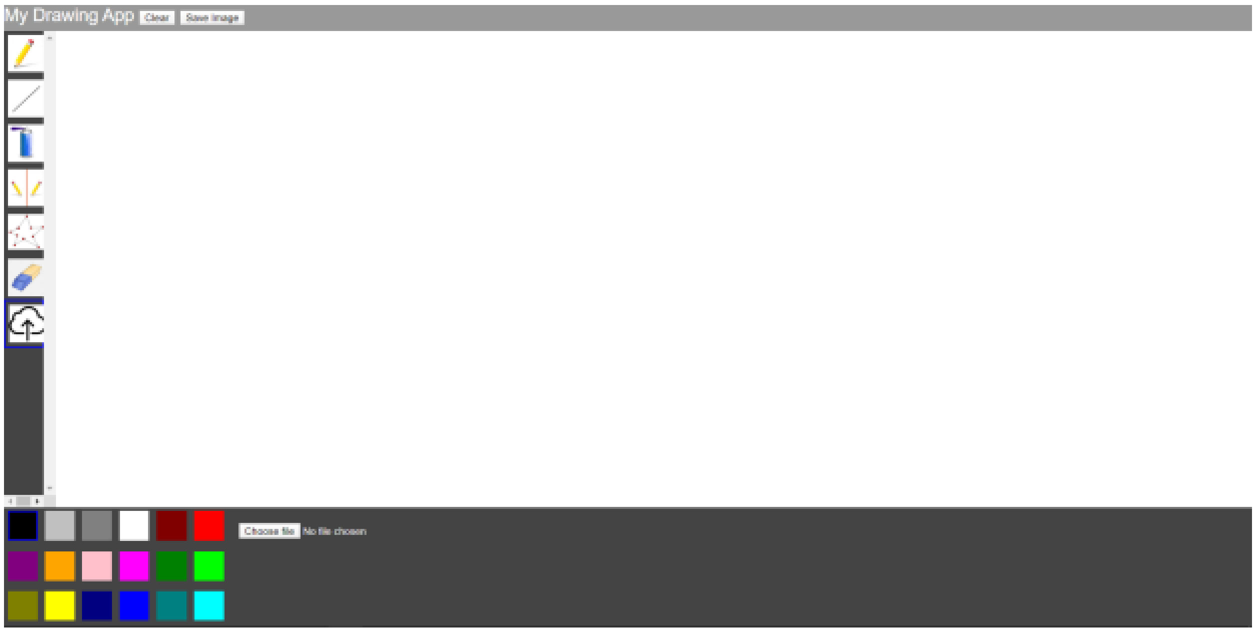


```
function uploadImg()
{
    this.icon = "assets/uploadImg.png";
    this.name = "uploadImg";

    var img;

    input = createFileInput(selectFile);
    input.position(327, 7);

    this.draw = function()
    {
        background(255);
        if (img)
        {
            image(img, 0, 0, width, height)
        }
    }

    function selectFile(file)
    {
        print(file);
        if (file.type === 'image')
        {
            img = createImg(file.data);
            img.hide();
        }
        else
        {
            img = null;
        }
    }

}
```

<u>**Week 4 (16/03/2020) - Work in progress peer review**</u>

In week 4, with our Work-in-progress peer review due in, we ensured that our code is well structured and is easy to read. We did this by adding clear and effective comments and splitting the code up into constructor functions that aid readability and reuse.

- Ensured that our code is well structured and is easy to read
- Added clear and effective comments and splitted the code up into constructor functions



**Week 5 (23/03/2020)**
Once our code was polished, we then wanted to add a shapes tool function which draws a rectangle at any size you want this is done by you pressing your mouse and dragging it across the canvas allowing you to draw the rectangle and we tried to add buttons where if you press it, it changes the shape it is drawing but we came across some bugs and were unable to implement it hence we would need to do more work and research on it.

- Added shapes tool function
- Tried implementing a button to change type of shape being drawn - came across multiple bugs
- Need to do more work and research on this tool

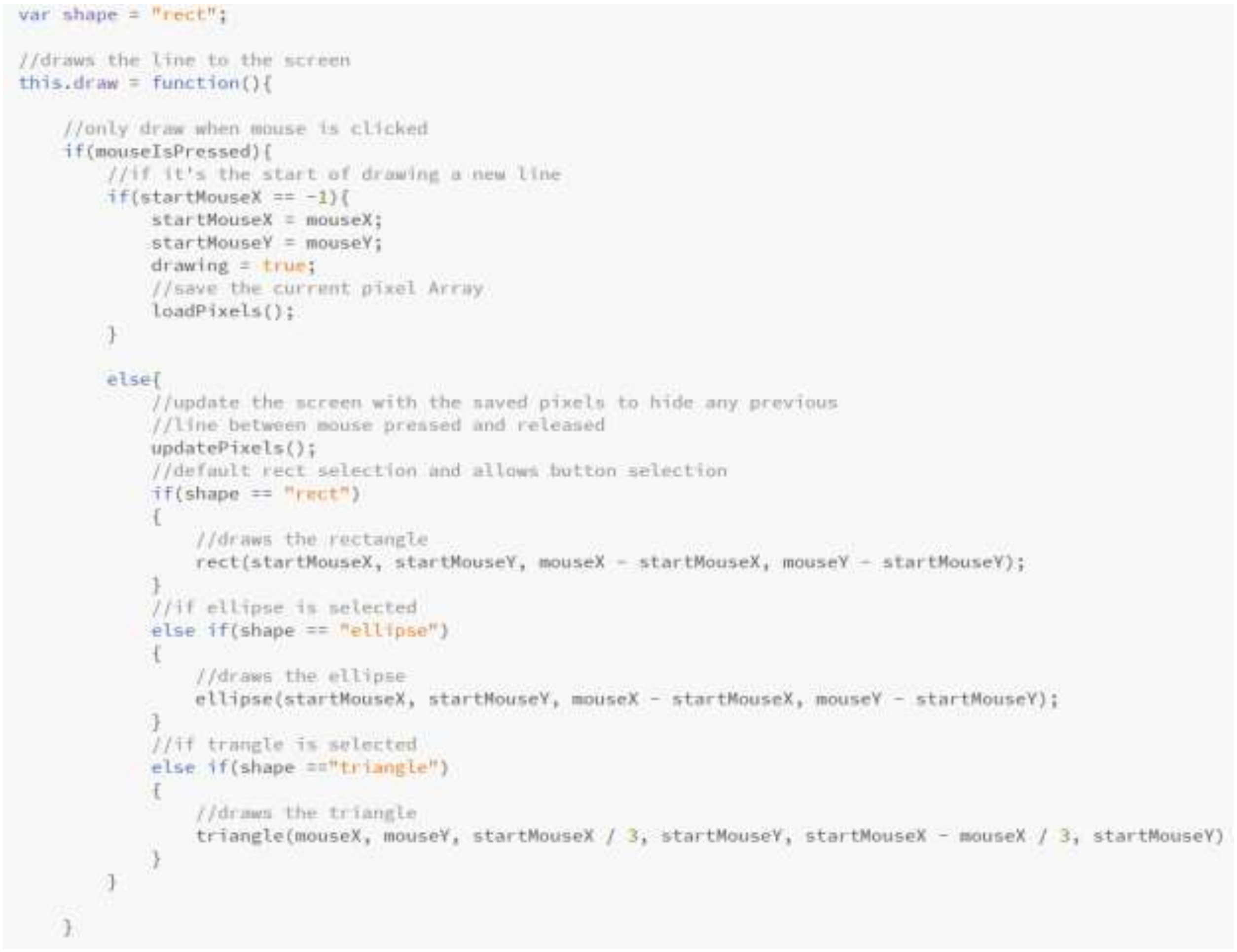
```
function shapesTool()
{
    this.icon = "assets/shapesTool.png";
    this.name = "shapes";

    var startMouseX = -1;
    var startMouseY = -1;
    var drawing = false;

    //draws the shape to the screen
    this.draw = function(){

        //only draw when mouse is clicked
        if(mouseIsPressed){
            //if it's the start of drawing a new shape
            if(startMouseX == -1){
                startMouseX = mouseX;
                startMouseY = mouseY;
                drawing = true;
                //save the current pixel Array
                loadPixels();
            }

            else{
                //update the screen with the saved pixels to hide any previous
                //shape between mouse pressed and released
                updatePixels();
                //draw the shape
                rect(startMouseX, startMouseY, mouseX - startMouseX, mouseY - startMouseY);
            }

        }

        else if(drawing){
            //save the pixels with the most recent shape and reset the
            //drawing bool and start locations
            loadPixels();
            drawing = false;
            startMouseX = -1;
            startMouseY = -1;
        }
    };

}
```

**Week 6 (30/03/2020)**

To help us accomplish what we want to do, we asked for guidance from the teaching assistants. To get the tool to draw out multiple different shapes, we had to add if statements for each shape so then the user could just draw the shapes by selecting them using the buttons next to the colour pallet.

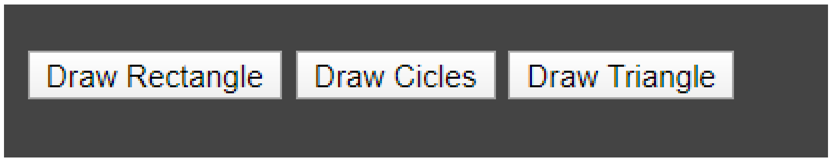
- Solved issues with help of teaching assistants - added if statements so then the user could just draw the shapes by selecting them using the buttons

**How we fixed Shapes Tool bugs:**



```
var shape = "rect";

//draws the line to the screen
this.draw = function(){

    //only draw when mouse is clicked
    if(mouseIsPressed){
        //if it's the start of drawing a new line
        if(startMouseX == -1){
            startMouseX = mouseX;
            startMouseY = mouseY;
            drawing = true;
            //save the current pixel Array
            loadPixels();
        }

        else{
            //update the screen with the saved pixels to hide any previous
            //line between mouse pressed and released
            updatePixels();
            //default rect selection and allows button selection
            if(shape == "rect")
            {
                //draws the rectangle
                rect(startMouseX, startMouseY, mouseX - startMouseX, mouseY - startMouseY);
            }
            //if ellipse is selected
            else if(shape == "ellipse")
            {
                //draws the ellipse
                ellipse(startMouseX, startMouseY, mouseX - startMouseX, mouseY - startMouseY);
            }
            //if trangle is selected
            else if(shape =="triangle")
            {
                //draws the triangle
                triangle(mouseX, mouseY, startMouseX / 3, startMouseY, startMouseX - mouseX / 3, startMouseY)
            }
        }

    }
```

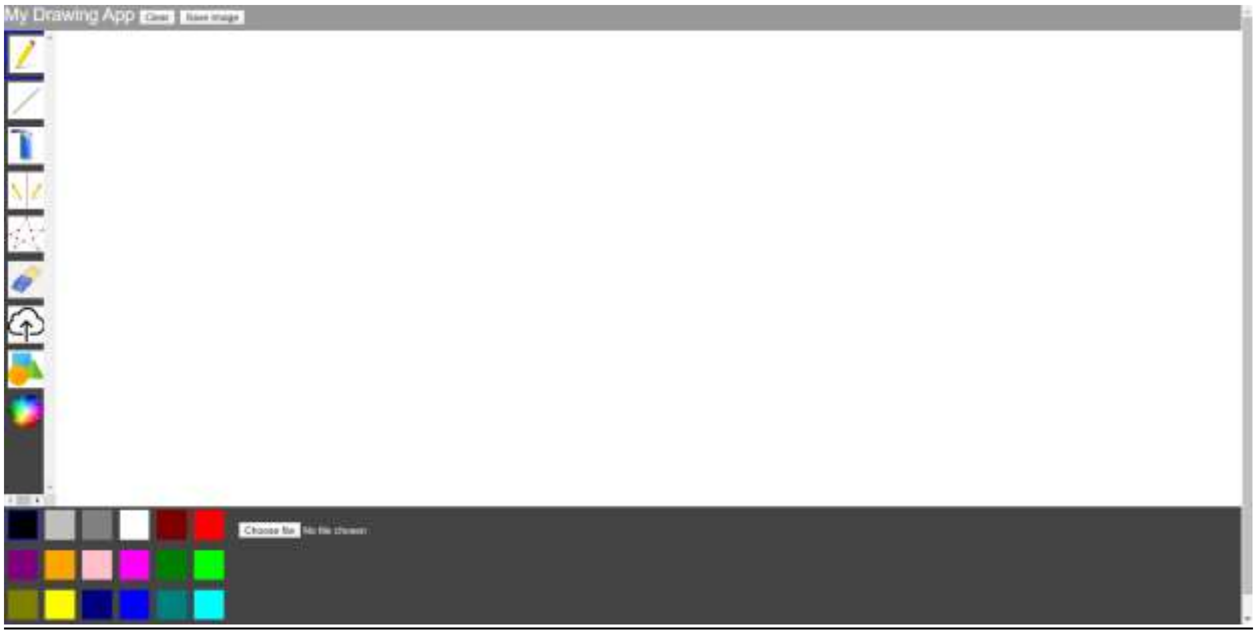
We created a variable and set it to rect as default then able to change it through the IF statements which would draw out depending on the buttons clicked.



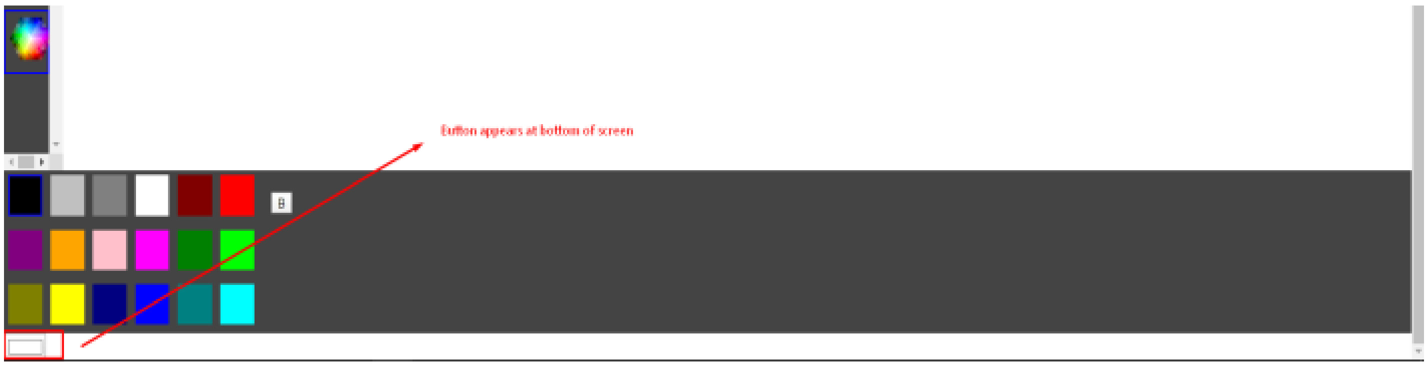
**<u>Week 7 (06/04/2020)</u>**

- 4 tools finalised up to this point with very few bugs to fix
- Close to finalising coding a background colour tool
- Could not get button in a suitable location for user

**Progress so far:**



**Background Tool bugs:**



## Week 8 (13/04/2020)

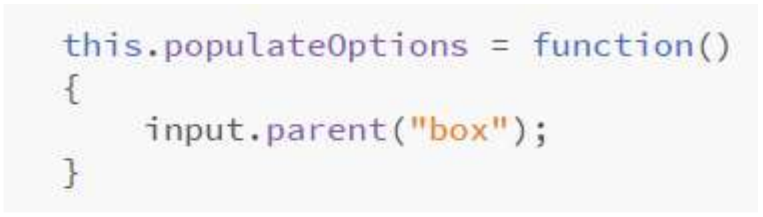
- Researched and started to code text input tool
- Finalised a text input tool

<u>**Week 9 (20/04/2020)**</u>

- Received help with previous tools - upload image and background colour tools
  *buttons did not previously work at all*
- Started to expand on eraser tool - lets user choose size options

**How we fixed upload and background tool bugs:**

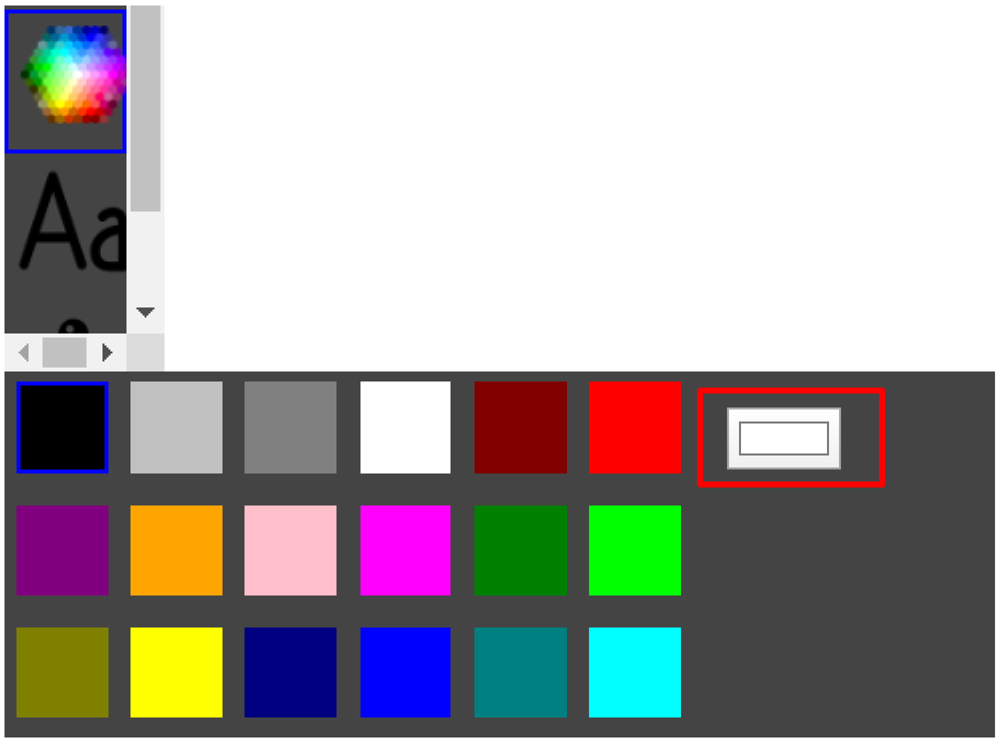
Since we hard coded the position of the button for Upload Image, which is completely unsuitable as users will be using the application on different resolutions so the button will appear in all sorts of positions, we fixed this by using HTML to  create a box model which would contain the button and only appear when the tool is pressed.



```
this.populateOptions = function()
{
    input.parent("box");
}
```

This positioned the button in the same place on every screen no matter the user's monitor resolution. This also fixed the functionality of the button since the functions would not match each other.

We used the exact same method for the Background Colour tool which enabled the button now appear next to the colour palette:


Aa

***Week 10 (27/04/2020)***

- Issues with size options - can only decrease size of eraser and not increase it
- Clear would not work on background colour tool

**Eraser Tool:**

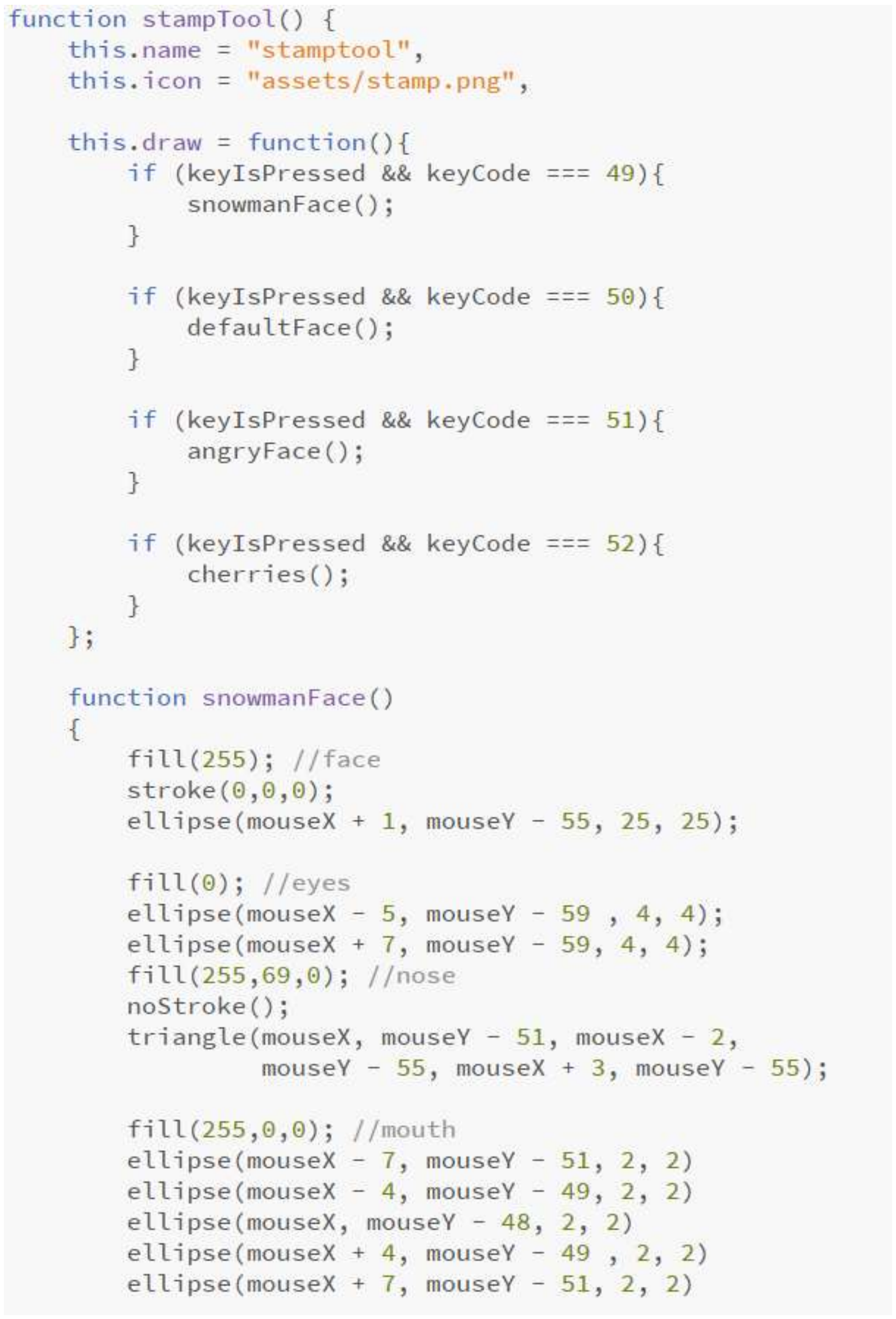
```
function eraser()
{
    //sets an icon and a name for the object
    this.icon = "assets/eraser.jpg";
    this.name = "eraser";
    var size = "ellipse";

    this.draw = function()
    {
        loadPixels();
        //if the mouse is pressed
        if(mouseIsPressed)
        {
        if(size == "ellipse")
        {
            //creates white circles to give erasing affect
            erase();
            noStroke();
            //only erases to where the mouse is with the size at 30
            ellipse(mouseX, mouseY, 30, 30);
            noErase();
        }
        else if(size == "bigellipse")
        {
            erase();
            //creates white circles to give erasing affect
            noStroke();
            //only erases to where the mouse is with the size at 45
            ellipse(mouseX, mouseY, 45, 45);
            noErase();
        }
        else if(size == "smallellipse")
        {
            erase();
            //creates white circles to give erasing affect
            noStroke();
            //only erases to where the mouse is with the size at 15
            ellipse(mouseX, mouseY, 15, 15);
            noErase();
        }
        }
    }
```

***<u>Week 11 (04/05/2020)</u>***

- Fixed eraser size issues - spelling of ids were not matching
- Fixed clear on background tool - created mouseClicked function outside of draw
- Researching new tool: Sticker
- Added a new tool: Sticker Tool
- Robustly tested application for bugs and ensured everything is commented and indented
- Changed cursor types for different tools.

**How we fixed background tool bugs:**
We asked for guidance from the teaching assistants and were able to fix it by creating a mouseClicked function outside of the draw function. This helped because we were changing the drawing state every frame which caused clear to never work.

**Sticker tool:**



```
function stampTool() {
    this.name = "stamptool",
    this.icon = "assets/stamp.png",

    this.draw = function(){
        if (keyIsPressed && keyCode === 49){
            snowmanFace();
        }

        if (keyIsPressed && keyCode === 50){
            defaultFace();
        }

        if (keyIsPressed && keyCode === 51){
            angryFace();
        }

        if (keyIsPressed && keyCode === 52){
            cherries();
        }
    };

    function snowmanFace()
    {
        fill(255); //face
        stroke(0,0,0);
        ellipse(mouseX + 1, mouseY - 55, 25, 25);

        fill(0); //eyes
        ellipse(mouseX - 5, mouseY - 59 , 4, 4);
        ellipse(mouseX + 7, mouseY - 59, 4, 4);
        fill(255,69,0); //nose
        noStroke();
        triangle(mouseX, mouseY - 51, mouseX - 2,
                 mouseY - 55, mouseX + 3, mouseY - 55);

        fill(255,0,0); //mouth
        ellipse(mouseX - 7, mouseY - 51, 2, 2)
        ellipse(mouseX - 4, mouseY - 49, 2, 2)
        ellipse(mouseX, mouseY - 48, 2, 2)
        ellipse(mouseX + 4, mouseY - 49 , 2, 2)
        ellipse(mouseX + 7, mouseY - 51, 2, 2)
```

We made use of our previous project to create sticker faces and objects for the user to utilise.

## Evaluation

Overall, we are quite satisfied with what we have achieved over the past weeks. Our aim was to create a minimum of 4 fully functional tools and in the end, we were able to create 7 - all of which are different and effective. Despite that, we are disappointed with some aspects of the application. Although this was never our overarching aim, it would have been great if we had got towards changing the visuals of the application as right now it is quite dull and boring to look at. Some GUI adjustments would be welcome for us and the users but right now the most integral element is just to get everything functioning correctly, so it felt great that we were able to do this. Additionally, replacing some of our tools with complex ones would be great but despite trying; we weren't able to do it in the end.

Whilst we did accomplish many things, it was never always easy to carry out. There were some external circumstances which caused our communication to drop a lot which had an effect on the things we were outputting and the pace we were outputting them at. But the most difficult part was just the coding in general. Trying to find the smallest bugs that caused the whole application to not work was extremely frustrating and although debugging through the console did help somewhat it was hard to keep trying to solve them.

For next time, things we would do differently would be to take more risks. We played it safe and tried to keep all the features simple but perhaps it would be best to code as many high level tools as possible - not only to get a higher mark but to also to help us in the future when we need it.

### Maintenance and Development

Perhaps a big benefit of this program is the length of the code. It is quite short and concise compared to many modern-day programs hence the execution times would be less lengthy.

If our audience would like to see some new features or add changes to the program, this can be easily done. Many of the lines have been commented so that it is easy to reference the function they want to improve on. Each function could be developed, tested and maintained very easily. This is made possible due to the programming being structured in a clear and simple way.

Future iterations could incorporate features such as the ability to blur things out, cut and crop and a spirograph tool.

### References

All references have been added into the code.