Drawing App Report

This report covers the following:

- the features implemented
- how you planned and coordinated development
- how you structured your code
- coding techniques used
- challenges faced
- self-evaluation: what would you do differently next time?
- progress log included as an appendix

Features implemented

Before implementing and developing the features for the extension, we had to decide which type of extension was the best for us to develop the features from our point of view; the drawing app, music visualiser or data visualiser. As a result, we have chosen the drawing app as our extension to add the necessary features. Then we have decided to add the following features:

- rectangle tool
- ellipse tool
- editable shape
- eraser

- ## Development planning and coordination

Throughout the development of this project, we also had to plan and coordinate this by using Discord and WhatsApp as our sources of communication with each other. We used these to our advantage in order to choose the best times to call each other and work together by reviewing each other's code and by supporting each other with potential solutions for the errors that we have encountered. We've used SMART as our technique to best plan the development of this project.

- Specific: specifying the features we have added and how to implement and structure these in the code to be readable
- Measurable: making sure that code was written within a period of time (for example; working on a function of a feature within 2 hours) and that we have chosen the right number of features that we should include in the drawing app.
- Achievable: by choosing the best number of features we had to ensure that the features are achieved within our scope and by the end of the deadline.
- Relevant: level of complexity chosen was relevant because it had to make sense to us in order to complete the project.
- Time-bound: setting deadlines for specific tasks throughout the development of the project for example (developing feature 1 in week 1, developing, feature 2 in week 2, coming back to feature 1 in week 3 to carry on completing it) and also using the best times to work as a group.

- **Coding techniques**

Throughout the development of this project, we have used loops, variables, array and conditionals. In terms of object orientation, we have used constructor functions in order to create the features, properties as a form of storage in the object related data and scoping to scope locally and globally variables by using object-oriented techniques. We have also used methods by using parameters and return. See screenshots for the coding techniques used.

editableShape tool – implementation of local variables and array

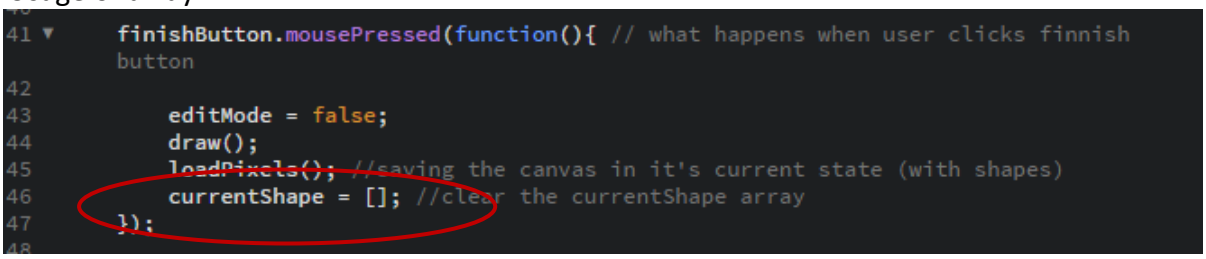
```
15 function EditableShape(){
16
17       //assigning variables
18       var editButton;
19       var finishButton;
20       var editMode = false;
21       var currentShape = [];
22
```

Usage of conditionals

```
    editButton.mousePressed(function(){ //what happens when user clicks edit button
        if(editMode){ //if edit mode isn't on
            editMode = false;
            editButton.html('edit shape');//update button's text so it says edit mode
        }else{
            editMode = true; //when editmode is on the button will say add vertices
            editButton.html('add vertices');
        };
    });
```

```
49  this.draw = function(){
50
51      updatePixels();
52      noFill();
53
54      if(mouseX>0 && mouseX<1832 && mouseY>0 && mouseY<773 && mouseIsPressed){ //if
        the mouse is on the drawing canvas and being clicked
55          if(!editMode){ //checks the edit mode to be off so we can add more lines
            to the shape
56              currentShape.push({ //saves the coordinates at that certain time of
                clicking on the canvas
57                  x: mouseX,
58                  y: mouseY
59              });
60      }else{ //if edit mode is on it will consider all the saved points of the
        shape
61
62              for(var i=0; i < currentShape.length; i++){
63                  if(dist(currentShape[i].x, currentShape[i].y, mouseX, mouseY) <
                    15){ //checks if the mouse is approaching any point, if so, it
                    will replace it's coordinates if the mouse drags it anywhere
64
65                      currentShape[i].x = mouseX;
66                      currentShape[i].y = mouseY;
67                  };
68              };
69          };
70      };
71      beginShape(); //will consider the currentShape array
72      for (var i =0; i < currentShape.length; i++){
73          vertex(currentShape[i].x, currentShape[i].y); //connects all the
            dots(coordinates saved in the array)
74          if(editMode){//if edit mode is on, it highlights the saved points so they
            can be moved
75              fill("green");
76              ellipse(currentShape[i].x, currentShape[i].y, 10);
77              noFill();
78          };
79      };
80      endShape();
81  };
82
83 };
```

Usage of array



```
41 ▼    finishButton.mousePressed(function(){ // what happens when user clicks finnish
        button
42
43          editMode = false;
44          draw();
45          loadPixels(); //saving the canvas in it's current state (with shapes)
46          currentShape = []; //clear the currentShape array
47      });
```

Eraser tool

Usage of for loops and local variables within the this.draw function.

```
/*
  1. basic circle of white color.
  2. adjustable size
  3. own cursor
  */
  var size = 30; // global variable for the size of the eraser

function keyTyped(){ //function for adjustable size of the eraser
    if(key === '+'){ // if you press "+" on your keyboard it would make the eraser
    bigger, "-" for making it smaller
        size = size + 10;
    }else if(key === '-'){
            size = size - 10;
```

Usage of local variables, constructors and conditionals.

```
function EraserTool() {
    this.name = "eraserTool"; //allocates a name to the tool
    this.icon = "assets/eraser.png"; //displays the icon for it

    var startMouseX = -1; //sets starting coordinates of the mouse
    var startMouseY = -1;
    var erasing = false; //sets erasing mode to false

    this.draw = function(){
        if(mouseIsPressed){
            if(startMouseX == -1){
                startMouseX = mouseX; // updates startMouseX/Y variables
                startMouseY = mouseY;
                erasing = true;
                //save the current pixel Array
                loadPixels();
            }else{
                noStroke(); //no outline to the eraser
                fill(255); //sets white color
                ellipse(mouseX, mouseY, size, size);//creates the white circle for
                erasing
            }

        }else{
            if(erasing){
            //save the pixels with the most recent modification and reset the
            //erasing boolean and start locations
            loadPixels();
            erasing = false;
            startMouseX = -1;
            startMouseY = -1;
```

drawCircles tool

Usage of parameters, conditionals and local variables.

```
function DrawCircles(){
    // assigns an icon and a name for the drawCircles tool
    this.icon = "assets/cerc.jpg";
    this.name = "drawCircles";
    this.draw = function(){

        // Call the variableEllipse() method and send it the parameters for the current
        mouse position and the previous mouse position
        variableEllipse(mouseX, mouseY, pmouseX, pmouseY);
    };

// The simple method variableEllipse() was created specifically for this program. It
calculates the speed of the mouse and draws a small ellipse if the mouse is moving
slowly and draws a large ellipse if the mouse is moving quickly
// abs returns a value of a number; in this case, it returns a speed value depending on
how fast the position of the cursor based on the speed of the mouse

    function variableEllipse(x, y, px, py){
        let speed = abs(x - px) + abs(y - py);
        stroke(random(0,255),(0,255),(0,255));
        stroke(speed);
        if(mouseIsPressed){
            ellipse(x, y, speed, speed);
        };
    };
};
```

drawSquares tool

Usage of parameters, conditionals and local variables.

```
function DrawSquares(){
    // assigns an icon and a name for the drawSquares tool
    this.icon = "assets/patrat.jpg";
    this.name = "drawSquares";
    this.draw = function(){
        // call the variableRect() method and send it the parameters for the current
        mouse position and the previous mouse position
        variableRect(mouseX, mouseY, pmouseX, pmouseY);
    };

// The simple method variableRect() was created specifically for this program. It
calculates the speed of the mouse and draws a small ellipse if the mouse is moving
slowly and draws a large ellipse if the mouse is moving quickly
// abs returns a value of a number; in this case, it returns a speed value depending on
how fast the position of the cursor based on the speed of the mouse

    function variableRect(x, y, px, py){
        let speed = abs(x - px) + abs(y - py);
        stroke(speed);
        if(mouseIsPressed){
        rect(x, y, speed, speed);
        };
    };
};
```

Program structure

To ensure that the drawing app was stable and usable for the user, we had to make sure that it runs correctly and that it works to our expectations. In addition, we ensured that the project was modular by saving code scripts into separated files for example; the eraser script saved in the eraser folder. In terms of the code structure, we have made comments to the code so it's readable and consistent. For example; using indentations in order for the code to be positioned which eventually helped us with keeping track with the progress of the development

Challenges faced

Multiple challenges have been faced throughout the implementation of the features in the program.

We have faced multiple errors when we first tried to implement the editable shape in the program for example; when referencing the editable shape script in the index file, it caused the other tools to disappear and glitch out; this was due to the fact that the code in the editable shape was not well structured and adapted to the program.

One was the editable shape tool that was inspired from the video tutorial from the VLE. In the tutorial, the code written for the editable shape was written separately from the files of the drawing app, so we had to work out a way to add it to the drawing app which was a challenge because we had to understand the code and look at how the other tools have been implemented into the program. The mousePressOnCanvas function has had many errors, regarding its variables. It wouldn't matter where I've placed the function or how I would initialise its variables, it still refused to work. At this point, we just made a series of conditionals to check the same thing the function would have: if the mouse is inside the drawing canvas. Setup function has had to be cancelled. We placed all the instructions inside of it into the main editableShape function.

The eraser tool needed a separate function to test for a specific button pressed on the keyboard, to change its size. After many hours spent researching different methods of doing this, the final form of the function is quite simple, yet very useful. Variables have been used and altered, in order to make the eraser expand and shrink at the user's will, just by pressing the '+' or '-' keys accordingly.

The drawCircle and drawSquare functions were not that complicated to build, but we still had some difficulties making sure they are drawing exactly what we need.

Self-evaluation

In our opinion, the final project we have put together is not the most consistent. We could have implemented many more tools, but we preferred to make few, better and as usable as possible tools. The editableShape tool is missing the add vertices feature. We tried to make it work but it was getting out of control. The editMode seemed to not be respected. The eraser tool is how we imagined it from the beginning, with a size modification feature. We intended to change the cursors of different tools but unfortunately we couldn't find the correct icons that the program would recognise. If we had more time to get this project done, more tools would have been implemented to our application.

Progress Log

George Voinescu(gvoin001)/Silviu Badica(sbadi001) FINAL PROJECT / group no. 96

Progress Log - 12 weeks total

week 1/12 - 28.01 - first lab on the drawing application

- got introduced to the application enviroment, how some of the tools work

- worked together as a team since the beggining, that's why we choosed to be in the same group

week 2/12 - 25.02 - final project introduction lab

- this was the moment we decided to continue out final project developing new tools for the drawing application

week 3/12 - 03.03 - initiated the development and implementation of the example extension, after watching the example extension video

week 4/12 - 10.03 - initiated the eraser tool, made a very basic but working tool. Minor bug when switching back to basic tools, the program won't draw again unless you choose a colour. At this point we had no clue how this could be fixed.

week 5/12 - peer review submission week - the example extension has been finished, but not fully implemented in the main application yet.

- the eraser tool was almost fully working, but didn't have the adjustable size feature. The bug when it won't draw when switching to other tools is still there.

week 6/12 - peer review feedback week - seeing and evaluating how other people have implemented different tools and coding techniques has made us figure out some minor bugs that we have encountered so far

week 7/12 - started implementing the example extension(editableShape) into the main application. First change was the setup function. We have cancelled the setup function, everything that was inside of it is now inside the editableShape function.

week 8/12 - The function mousePressOnCanvas was giving major errors. We spent loads of time trying to figure out how to make this function work. In the meantime, we decided to start adding a separate function to the eraser tool, one which would modify its size.

week 9/12 - Decided to drop the mousePressOnCanvas function totally, instead of it we applied some conditional statements that would do the same thing as the given function. Implemented the drawCircle and drawSquare tools to the main application, so far they are not perfectly joined together. Program would draw randomly.

week 10/12 - Eraser tool separate function for adjustable size is now fully working. We've had a hard time finding a specific function that would detect what button I'm pressing on my keyboard.

week 11/12 - Editable Shape is now fully working, except of the add vertices function. DrawCircle and drawSquare are fully working.

week 12/12 - As all the added tools were finnished, we spent a quite considerable amount of time commenting the code, making sure the code is very easy to read and organised. We started working on the report ducument recently, but we kept notes and observations along the way. The report is now just a matter of adding the remainder of previous notes.

Sources used as inspirations and as guides

Draw circles and draw squares tools: https://p5js.org/examples/drawing-patterns.html

Editable shape tool: http://igor.doc.gold.ac.uk/tutorials/videos/I2P_part2/exten/draw.mp4