Name – Rishi Sankhla
Student No. – **33724434**

Interactive Random Art Algorithm

I am building an interactive art platform, where all these paintings are random generated computer arts. On this platform artist can interact with these pre generated arts and can save their state. My motive was to play with the art through hand gestures and to save their state, so that user can build on the top of that. By this platform artists can play with other artist save paintings, which are generated on my application.

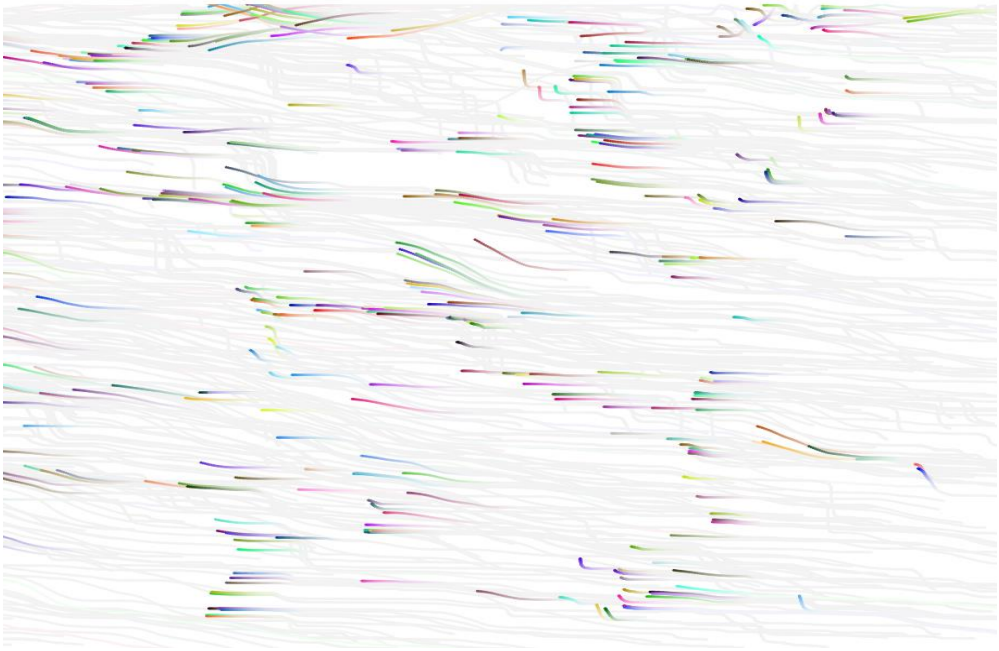
This application is build in javascript, and to run this application we need a browser and a webcam. I have used these javascript libraries to build this application:

- p5.min – this is the main library, as it contains all the pre build in javascript to for a drawing app.
- p5.dom.min – this library is used to connect our application with the HTML tags, it helps us to maintain an event based programming methodology.
- ml5.min – the main motive behind using this library is to build hand detection model, this library is build on the top of Tensorflow library, therefore it makes so much easier to build machine learning models through it.

To implement my whole project first I have build 5 random art algorithms (so that user can play with their parameters and can save their state) then one hand detection algorithm:

Art 1 is “sin-cos movement of circle”:

This art generated a random number of ellipse across the canvas with some random color value and then all the tiny-tiny ellipse points moves in a flow direction using trigonometry functions. These ellipse points and hits the border of the canvas then goes back in the opposite direction.



Here user can interact with two parameters, one is the size of circle and another one is the speed of their movement (this speed parameter is directly changing the sin-cos multiplier).

To build this pattern I have made one separate file naming art_1.js, this file contains one main constructor function and inside it there is a few public variable (so that can change over the time in sketch file) and two public functions, that is draw and move.

```
    this.draw = function(){
        this.size_e = slider8.value();
        this.moving_speed = slider7.value();
        push();
        noStroke();
        fill(this.r_color);
        ellipse(this.x,this.y,this.size_e,this.size_e);
        pop();
    }

    this.move = function(){

        var n1 = noise(this.x*this.moving_speed,this.y*this.moving_speed,frameCount*this.moving_speed);

        var a1 = TAU * n1;
        if(this.switch_gate_x == 0){
            this.x=this.x-(sin(a1)+this.moving_speed_new);
        }
        else if(this.switch_gate_x == 1){
            this.x=this.x+(sin(a1)+this.moving_speed_new);
        }
        if(this.x>width){
            this.switch_gate_x = 0;
        }
        else if(this.x<0){
            this.switch_gate_x = 1;
        }

        if(this.switch_gate_y == 0){
            this.y=this.y-(cos(a1)+this.moving_speed_new);
        }
        else if(this.switch_gate_y == 1){
            this.y=this.y+(cos(a1)+this.moving_speed_new);
        }
        if(this.y>height){
            this.switch_gate_y = 0;
        }
        else if(this.y<0){
            this.switch_gate_y = 1;
        }
    }
}

```

The above constructor function draws a single moving ellipse on the screen. At first draw function draws the ellipse accordingly then we call the move function to change the x and y value of a particular ellipse, in the move function I am using noise function of p5.js to generate the random movement value then it is multiplied by the trigonometry functions so that it has a curl flow in the movement. Then we are checking whether the ellipse hits our canvas border or not, if so then we change the direction.

In the sketch file, first we have made a global variable (which is an array), then we are adding multiple objects of this constructor function, which is ranging from 500 to 1500. And finally in the main draw loop in our sketch file, we call draw and move function of each element.

While building this pattern the major problem I faced was to build switch-gate system for the each ellipse, so that whenever it hits the canvas border we change the direction of it.

I have learnt to build this pattern from a YouTube video, checkout references.

Art 2 is “random square touch interaction”:

This generative art creates a random amount of squares on the screen and with random assigned color, x and y values, here we were able to accomplish overlapping between squares with the help of blendMode function. Half of the squares are having hard light blend and the other half squares are having normal blend.

Here, user can interact through 3 features –

- can change the position of square by random touch, means once we move over the squares, those squares will start changing their position randomly.
- Alpha value, user can also select the desired alpha value, so that he/she can play with the color brightness of our squares.
- Use can also play with the square number parameter, that is he/she can control the number of squares on the screen.

To build this pattern I have made one separate file naming art_2.js, this file contains one main constructor function and inside it there is a few public variable (so that can change over the time in sketch file) and two public functions, that is draw and move.

```
    this.draw = function(){
        this.alpha_r=slider10.value();

        push();

        noStroke();
        if(this.r<0.5){
            blendMode(HARD_LIGHT);
        }
        else{
            blendMode(BLEND);
        }

        fill(this.r_color[0],this.r_color[1],this.r_color[2],random(0,this.alpha_r));

        rect(this.x,this.y,this.h,this.w);
        pop();


    }

    this.move = function(x_m,y_m){
        var d = dist(this.x,this.y,x_m,y_m);
        if(d<200){
            this.x=this.x+random(-100,100);
            this.y=this.y+random(-100,100);
        }
        if(this.x<0 || this.x>(width+100)){
            this.x=random(0,width+100);
        }
        if(this.y<0 || this.y>(height+100)){
           this.y=random(0,height+100);
           }
    }
}
```

The above constructor function create a signle square on the screen, first we call the draw function and then the move function inside the main draw loop in oursketch file. Here in the draw function we are randomly using the blendMode function of p5.js to blend the colors of the rectangle, so that when they overlaps each other it creates colourful background pattern. Whereas, in the move function we are adgusting the position of that rectangle according to the user movements, and then I have written two if statements inside it, to check whether the square is inside the canvas or not, if not then we are randomly assigning the value.

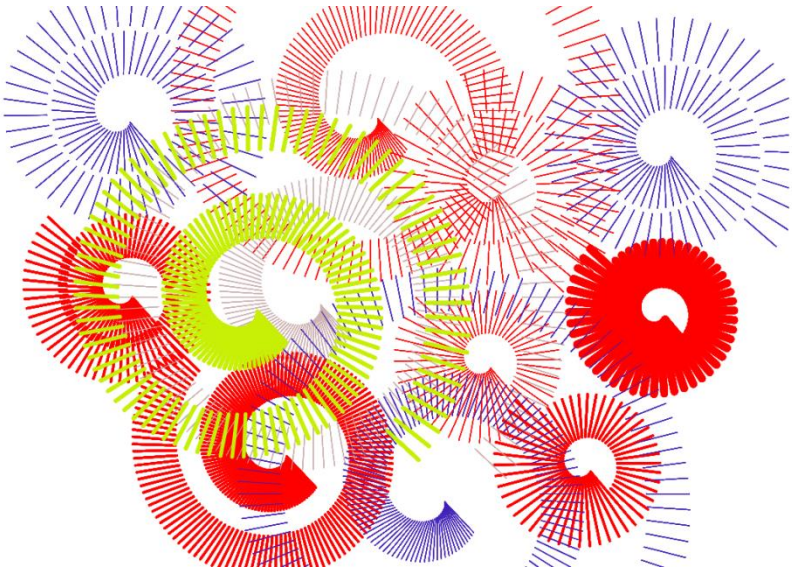
So to make multiple rectangles of random height and width, I have made one global array and then pushing multiple objects (of this function) in it, according to the user given value. Then inside the main draw loop I am iterating over the array and calling the appropriate functions.

I have learnt to build this pattern from a YouTube video, checkout references, the main thing I learnt is to play with blendMode function of p5.js.

Art 3 is “rotating line pattern”:

This art function allows user to draw a pre build rotating line pattern, user can print as many patterns as he wants and wherever he wants on the canvas. Here user can play with four parameters:

- User can adjust the stroke weight of each pattern and can decide how thick the line should be.
- He/she can also adjust the rotation value parameter, through this parameter we are calculate the pi multiplier of the rotate function.
- we can select the number of lines we wants in a particular pattern, so that we can adjust it according to our stroke weight size.
- And the last parameter is the color of each pattern, user can select a particular color through Color-Picker for any pattern on the screen.



To build this pattern I have made one separate file naming art_3.js, this file contains one main constructor function and inside it there is a few public variable (so that can change over the time in sketch file), one public functions, i.e draw and a private function, i.e single line.

Here our main constructor function art_3 generates one pattern shown above. So the logic behind it is, every time when user does some action we add new object of it in our global array and the x & y position of the new element is selected according to the use movement.

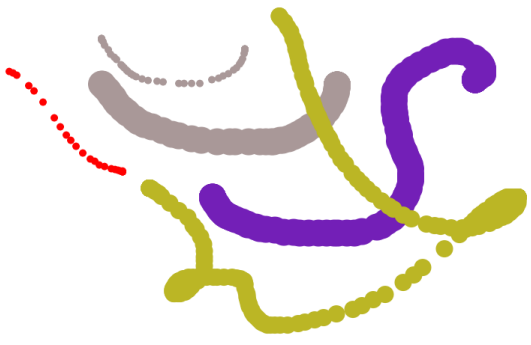
The concept to draw a single pattern is, we call our single_line private function inside the for loop in our draw function and passing all the necessary parameters to it, then single_line function first translate and rotate the canvas accordingly, and in the end it draws a particular line. And in the end we call the draw function of each pattern in our main draw loop, which is inside the sketch.js file.

```
//rotating line pattern

function art_3(){
    this.x=0;
    this.y=0;
    this.number_of_l = slider4.value();
    this.line_gap = slider2.value();
    this.line_dis = 10;
    this.line_h=50;
    this.strokeWeight = slider3.value();
    this.line_color = colorPicker1.color();

    function single_line(x_s,y_s,lin_h,d_n,
                         line_dis,s_w,line_color){
        push();
        strokeWeight(s_w);
        stroke(line_color);

        translate(x_s,y_s);
        rotate((2*PI)/d_n);
        line(line_dis,line_dis,lin_h,lin_h);
        pop();
    }
    this.draw = function(){

        push();
        for(var i=0;i<this.number_of_l;i++){

            single_line(this.x,this.y,this.line_h+(i*this.line_gap),
                        this.number_of_l/((i+1)+(i*this.line_gap)),
                        this.line_dis+(i*this.line_gap),
                        this.strokeWeight,this.line_color);
        }
        pop();

    }
}
```

This pattern was my own idea, so the major difficulty I faced here was to rotate value for each iteration, then later on I was able to figure out calculation behind it, that full circle is 2 * PI and we just have to divide it accordingly.

Art 4 is “canvas drawing pen”:

This art form allows user to draw over the screen, he/she can draw anything over other patterns, here user can just play with our colouring pen anywhere on the canvas. We need to adjust two things before using the pen, that is the color of the pen and the thickness of the pen.



To build this pattern I have made one separate file naming art_4.js, this file contains one main constructor function and inside it there is a few public variable (so that can change over the time in sketch file) and two public functions, that is draw and setup.

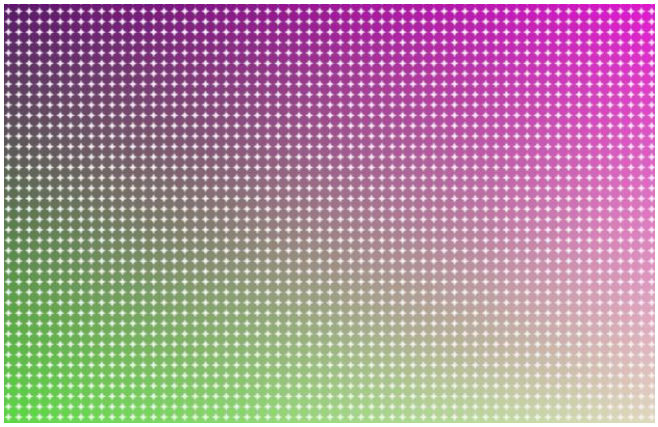
```
1  //canvas drawing pen
2
3  function art_4(){
4      this.el_size = slider1.value();
5      this.colorPicker;
6
7      this.setup = function(){
8          this.colorPicker = createColorPicker('#ff0000');
9          this.colorPicker.position(20, 640);
10     }
11
12
13     this.draw = function(x,y){
14         this.el_size = slider1.value();
15         push();
16         noStroke();
17         fill(this.colorPicker.color());
18         ellipse(x,y,this.el_size,this.el_size);
19         pop();
20     }
21 }
22
```

To program this functionality in our app, I have to remove the background in the main draw loop, so that it creates the effect of drawing. The working of this feature is quite straightforward, first we setup the color picker and all the other parameters, then we extract the user movement value to draw the ellipse in each frame. And in the end to integrate this constructor to our application, I have first created a global object of it in the sketch file, then called the setup function of it (so that color picker is up and running), then final in our main draw loop, we are calling this draw function.

Art 5 is “colourful doted background”:

This art form generates a grid format of ellipse across the canvas, to build a colourful pattern, in this pattern we can see the color is gradually change from the first ellipse to the last ellipse. Change is color is calculated according to x and y position of each ellipse. Here, user can play with three different types of parameter:

- If user hover over the grid pattern, he/she can see the change in the colour of each ellipse, which is again calculated gradually according the ellipse position.
- Use can change the ellipse size, so that he/she can select the density of these pixels.
- User can also shift between RGB values of fill function, and can pick which corner they want more lighter than the other.



To build this pattern I have made one separate file naming art_5.js, this file contains two constructor function, one to draw a single ellipse and one to draw the whole grid, these are art_5_mini and art_5 respectively. In art_5 function there is a few public variable (so that can change over the time in sketch file) and one public functions, that is draw. Whereas, in art_5_mini there are two public functions, i.e single_circle and cal_dis function.

```
function art_5(){

    this.size_c = slider5.value();
    this.color_trip = slider6.value();
    this.color_trip_array = [];
    this.num_c = round(width/this.size_c)*round(height/this.size_c);

    this.draw = function(m_x1,m_y1){
        this.size_c = slider5.value();
        this.color_trip = slider6.value();
        var c_array = [];
        for(var w=1;w<=this.num_c;w++){
            c_array.push(new art_5_mini());
        }
        for(var s=0;s<c_array.length;s++){
            c_array[s].select_number(s+1);
        }
        var c=0;
        for(var i=0;i<round(width/this.size_c);i++){

            var x = this.size_c*(i%round(width/this.size_c));
            var m1 = map(i,0,round(width/this.size_c),0,255);

            for(var p=0;p<round(height/this.size_c);p++){

                var y = this.size_c*(p%round(height/this.size_c));
                var m2 = map(p,0,round(height/this.size_c),0,255);
                var single_element = c_array[c];
                single_element.x=x;
                single_element.y=y;
                var d =single_element.cal_dis(m_x1,m_y1);
                var cal_di = sqrt(sq(width)+sq(height));
                var m3 = map(d,0,cal_di,0,255);
                this.color_trip_array = [];
                this.color_trip_array.push([m1,m2,m3]);
                this.color_trip_array.push([m1,m3,m2]);
```

```

function art_5_mini(){
    this.circle_number = 0;
    this.x=0;
    this.y=0;
    this.select_number = function(a){
        this.circle_number = a;
    }
    this.single_circle = function(size_c,fill_c){

        push();
        noStroke();
        fill(fill_c);
        ellipse(this.x,this.y,size_c,size_c);
        pop();
    }
    this.cal_dis = function(m_x,m_y){
        var d =dist(this.x,this.y,m_x,m_y);
        return d;
    }
}
```

Here I'm creating multiple object of art_5_mini function inside the art_5 function, then I'm assigning x and y values to each circle, and in the end I'm mapping the precise color value. The way it is achieved is by calculating the distance of each circle with respect to the user movement and then changing the RGB and alpha value of fill function accordingly, to create a gradual flow of colourful grid.

We are creating an object of the main constructor function (i.e. art_5) in our setck.js file, so that we can call its draw function inside our main draw loop.

In this pattern the major issue I was facing was the slowdown in frame rate, which I later on figured out that if the ellipse size is below 13 then it would slowdown the whole application, because in this pattern we are performing numerous amount of calculation for each ellipse, i.e. first drawing, calculate distance, calculating each RGB value for each ellipse etc.

Hand detection algorithm:

The motive of this algorithm is it detect the human hand gestures but I was not fully able to integrate it fully with my drawing app, I had accomplished detecting hand and finger tips but to configure it with all the patterns I need to write a separate class of motions and then change the art parameters accordingly. Therefore, I have added slider bars and colour buttons to interact with our generative arts.

To write this algorithm, first I am creating an object to capture our video and an object of ml5.handpose function (of ml5.js library), then passing the detected points and assigning to our global variable through hand_detection.on function, after that I am initialising button for our video. This all is done inside the setup function.

Next step is to write a constructor function to draw each hand detected point, as we can see below I have made key_point function, this function takes three parameters as an argument, i.e. x value, y value and the unique number, which is assigned iteratively in drawpoints function. Inside the key_point function there are three global variables and one global function, i.e. draw (which draws the ellipse and writes the number on top of it).

```
146
147     //video functions
148     video = createCapture(VIDEO);
149     video.size(640, 480);
150
151     hand_detection = ml5.handpose(video, modelReady);
152
153     hand_detection.on("hand", results => {
154     points_array = results;
155     });
156     video.hide();
157
158     video_state=false;
159     video_button = createButton("plain video");
160     video_button.position(560, 580);
161     video_button.mousePressed(for_vo);
```

```
1   //function to create single hand detection point
2
3   function key_point(x,y,s_n){
4       this.x=x;
5       this.y=y;
6       this.select_num=s_n;
7       this.draw = function(){
8           push();
9           fill(0, 255, 0);
10          noStroke();
11          ellipse(this.x, this.y, 15, 15);
12          push();
13          fill(255);
14          textSize(15);
15          text(this.select_num.toString(), this.x, this.y);
16          pop();
17          pop();
18      }
19  }
20
```

So here the main function is the drawpoints function, this function called inside our main draw loop in the sketch file, so that it keep on updating the point x and y value, with each frame. Here in this function there are nested for loops, where the main for loop iterate over the point_array, and inside it we extract the x and y value from each iteration to create a

new object of key_point class, then we iterate over the c_key_point_array to call the draw of each element according to the selected pattern.

Then in the end I have written one small function (modelReady) which prints in the console once the model is up running. If we refer back to the above screenshot, we can see inside the setup function we have passed modelReady function as an argument to the ml5.handpose object.

The major technical difficulty I faced with my application was, computer processing power and building the gesture class. As p5.js is not the most efficient library and I have build so many classes and their iterative objects, it takes a lot of time to perform all the computation and making my computer slower, hence it is taking extra time to reload the application and in the end also leading to slower frame rate. Now to integrate all the gestures with our generative art, I have to make more classes and objects of it. If I had more time then I would have done performance testing of each class to know its efficiency and would write gesture class accordingly.

```
function drawpoints() {
    for (var i = 0; i < points_array.length; i += 1) {

        var first_prediction = points_array[i];
        t_number = 1;
        var c_key_point_array = [];
        for (var j = 0; j < first_prediction.landmarks.length; j += 1) {
            var keypoint = first_prediction.landmarks[j];
            c_key_point_array.push(new key_point(keypoint[0], keypoint[1],t_number));
            t_number=t_number+1;
        }

        for(var d=0;d<c_key_point_array.length;d++){
            var s_e = c_key_point_array[d];

            if(state_change==5){
                if(s_e.select_num==9){
                    s_e.draw();
                }
                if(s_e.select_num==5){
                    s_e.draw();
                }
                var common_d1 = dist(c_key_point_array[8].x,c_key_point_array[8].y,
                                     c_key_point_array[4].x,c_key_point_array[4].y);
                var our_m1 = map(common_d1,13,170,10,50);

            }
            else{
                s_e.draw();
            }

        }

    }
}


function modelReady() {
  console.log("Model is working");
}
```

<u>evaluation and testing:</u>

if I evaluate this project as a one entire application, then I was expecting a full interactive art software, means where user can interact with all the generative arts through hand gestures and can save each pattern state on the top of each other. Whereas so far I was only able to accomplish 5 generative art algorithms and where user mostly need to interact through mouse and keyboard. So to finish the integration of hand gestures, I have to build a few more separate classes for it and then configure those classes accordingly.

In terms of technical aspects I was not expecting that multiple classes and objects will slow down the frame rate and would need this much of computing power to run in p5.js. however, I was also able to fix two classes, as through console testing I realised the length of few arrays were going to infinity, so I have set the maximum limit from them. In the performance testing I have concluded one more thing, both the algorithms (hand detection and random art generative) run very smoothly, if run separated.

To further test the application I going to collect user feedback on my application, in terms of performance, functionality, concept and technical aspects. The major questions in my questionnaire will be:

- Were you easily able to use the application?
- Do you find it interesting?
- How was the performance of the app, was it too slow?
- Were all the feature useful?
- Would you prefer proper hand gestures for the interaction?
- Is this app worth building art?
- Would you use it again?
- Would prefer to collaborate with other artist on this application in future?
- Was there any other similar application you have used, if so then what's the name?
- What feature should I add in it?

After asking all these question feedback and analysing app performance while user testing, I'll be able to know my project flaws, that what are the unnecessary features, is this concept going to work or not, what are the performance issues etc. and then I'll be able to conclude improvisation methods.

Apart from above question I am also going to test the overall performance while user is using the app, like I would note the amount of time he/she spends on the application, the amount of time application needs to reload the functions, the number of times each pattern is used and I might need to write performance testing function so that I can calculate the time each class takes to reload. I am also going to perform black box testing on each main class. This all will help me to improve the code stability and will make my app more efficient in terms of performance.

Completion of the project according to time line:

Particulars	Date	Finished
Background research	10-11-2022	✓
project specification report	14-11-2022	✓
project design	20-11-2022	✓
code part 1 of project (random art algorithm)	01-12-2022	✓
code part 2 of project (hand detection art algorithm)	10-12-2022	✓
design specification report	16-12-2022	✓
code part 3 of project (building complete working app)	01-01-2023	✓
collecting user experience	01-02-2023	
analysis of project	31-03-2023	✓
improvisation on project	01-04-2023	
poster presentation	02-05-2023	
collecting final reviews	04-05-2023	
evaluting final summary	12-05-2023	

References :

1. *I tried making Generative Art (p5js) (2022) YouTube. YouTube. Available at: https://www.youtube.com/watch?v=UsIF5r8rAvk (Accessed: March 31, 2023).*
2. *Easy Perlin Noise Flow Fields (2021) YouTube. YouTube. Available at: https://www.youtube.com/watch?v=sZBfLgfsvSk&t=1s (Accessed: March 31, 2023).*
3. *L6: Ml5.js serial (no date) Physical Computing. Available at: https://makeabilitylab.github.io/physcomp/communication/ml5js-serial.html (Accessed: March 31, 2023).*
4. *ml5js (no date) Ml5js/ML5-library: Friendly machine learning for the web! , GitHub. Available at: https://github.com/ml5js/ml5-library (Accessed: March 31, 2023).*
5. *Chen, V.P. (2021) Getting started with ml5.js-tutorial part I: Image classifier, Medium. AIxDESIGN. Available at: https://medium.com/aixdesign/getting-started-with-ml5-js-tutorial-part-i-image-classifier-6d437ec38045 (Accessed: March 31, 2023).*
6. *Shvembldr (2019) How to make your first generative art with p5.js, Medium. Medium. Available at: https://medium.com/@shvembldr/how-to-make-your-first-generative-art-with-p5-js-3f10afc07de2 (Accessed: March 31, 2023).*

7. *Beginner's Guide to Learning p5.js for generative art (no date) fxhash. Available at: https://www.fxhash.xyz/article/beginner's-guide-to-learning-p5.js-for-generative-art (Accessed: March 31, 2023).*
8. *Easywebsify (2022) Basic p5.js create processing art, Medium. Level Up Coding. Available at: https://levelup.gitconnected.com/basic-p5-js-create-processing-art-beca397607d7 (Accessed: March 31, 2023).*
9. *Making simple patterns in p5.js (2022) YouTube. YouTube. Available at: https://www.youtube.com/watch?v=ig0q6vfpD38 (Accessed: March 31, 2023).*
10. *Part 01 (no date) The aesthetic of generative coding. Available at: https://digitalideation.github.io/ba_222_gencg_h1801/notebooks/week01.html (Accessed: March 31, 2023).*
11. *Turning my body into a Controller with machine learning, ml5.js, and p5.js (2021) YouTube. YouTube. Available at: https://www.youtube.com/watch?v=96sWFP9CCkQ (Accessed: March 31, 2023).*
12. *Multiple hands detection in p5.js (2021) YouTube. YouTube. Available at: https://www.youtube.com/watch?v=BX8ibqq0MJU (Accessed: March 31, 2023).*
13. *Koerismo (no date) Koerismo/P5.buttons: A library that aims to make scripting buttons easier., GitHub. Available at: https://github.com/koerismo/p5.buttons (Accessed: March 31, 2023).*
14. *P5 noise function (no date) reference | p5.js. Available at: https://p5js.org/reference/#/p5/noise (Accessed: December 30, 2022).*
15. *L7: Handpose Serial (no date) | Physical Computing. Available at: https://makeabilitylab.github.io/physcomp/communication/handpose-serial.html (Accessed: December 30, 2022).*
16. *A friendly machine learning library for the web. (no date) | ml5. Available at: https://learn.ml5js.org/#/ (Accessed: December 30, 2022).*
17. *ml5 handpose algorithm. (no date) | ml5. Available at: https://learn.ml5js.org/#/reference/handpose?id=config (Accessed: December 30, 2022).*
18. *Hello! (no date) home | p5.js. Available at: https://p5js.org/ (Accessed: December 30, 2022).*
19. *Get started (no date) get started | p5.js. Available at: https://p5js.org/get-started/ (Accessed: December 30, 2022).*
20. *Libraries (no date) libraries | p5.js. Available at: https://p5js.org/libraries/ (Accessed: December 30, 2022).*

GitHub repository link :
https://github.com/rishisankhla/goldsmiths-final-project---interactive-random-art-algorithm.git