Interactive Random Art Algorithm

Student – Rishi Sankhla

Student Number - 33724434

Supervisor Name - Andy Lomas

Abstract:

The idea is to develop an interactive art platform, through which artist can play with pre generated computer art (which is possible by changing art parameters through his hand gestures.) and can also save their drawing state. For this platform all we need is a computer with a webcam. And my aim is to deliver a software written in P5.js.
Here user will come on this platform and through his hand movements he/she can start playing with the 2D random generated art. This whole project is mainly divided into two part, first one is the hand detection algorithm, and the second half is the random art algorithm.
With this platform artist can not only play with other artist save paintings which are generated on my application but can also build on the top of that.
As this is a new concept this could be useful to many industries, like art industry, education industry, computer industry and it could be a software for school children to play with the computer drawings etc.

Keywords:

Computer-art

Interactive

Random-algorithms

Hand-detection

Generative-art

Introduction:

As we know generative art industry and interactive art industry is still immerging and yet to develop. So I am here to build a software which is totally a new concept and can open many doors for upcoming computer artist. Though if we compare there are many drawing application in the market these days, but there are very few apps where user can interact with the art.

So here my concept is build an interactive art platform, through which artist can play around with the other artist drawings (these drawing are random generative art) using hand movements. This idea is very abstract and subjective, but can be implemented for many industries, like art industry, education industry, computer industry. This new concept could be useful for beginner artists to play with other art forms and can save the state of their drawings on the top of other artist drawings. Later on it can also be turned into new platform where artist can share their generative art with each other so that other artists can play with their drawings, or can also build on the top of that.

Objective:

- My main objective is to deliver one application through which artist can make art improvisation of random art, so at first I will write a random art generation algorithm, then on the top that I will be writing machine learning algorithm which will detect human hand gesture to change the art parameters (like structure, density, size etc).
- Second objective is to create one working mechanism between these two algorithms, i.e. through hand detection algorithm user can influence the parameters of random art algorithm.
- And the final objective is, adding extra features like different types of art, going from 2D art drawing to 3D art drawings, would also try to implement camera angle interaction feature, so that user can also interact with the camera angles.

Minimum viable product:
My MVP will be a full working software written in p5.js (which will contain two major algorithms, that is random art generation algorithm and hand movement detection algorithm), through which artist can change random art parameters using hand movements.
This report also delivers a final user review of my software along with the summrisation of my project, explaining what features were accomplished and how it was accomplished. This report will contain detail documentation, stating all the course of action undertaken to build this project.

I am hoping to learn more about random art generation algorithms and how machine learning to can lead to new art forms. This project is also going to explore the world of human computer interaction, as it might lead to era of artist, where artist can improvise on pre generated computer art.

Background (literature review):

There are many random pattern generating algorithms, which generate two dimension abstract art forms like maze patterns, graphical ellipses, image repetition pattern etc. However, there are very few programs which develop three dimension patterns, such graphics are either programmed by big graphic production companies (Walt Disney Studio Animation, Pixar Animation Studios, DreamWorks Animation, Sony Pictures Animation) or by individual artists.

Currently there are many tools in the market (like Patternizer, Patterninja, Mazeletter, Patternify Repper etc) to generate art patterns, but there few limitations to them:

- Firstly, most of these tools generate two dimension patterns, so there's still a big gap for the 3D art generators.
- Secondly, the human interaction with these tools is dependent on hardware interaction, like mouse movement or key press. And to generate any painting he/she needs to start drawing from scratch.
- Tool like patternninja is bound by limited functionalities, for example they need some initial image or sticker to start building the patterns.
- Most of tools only generate one, two, three different types of computer pattern.
- No tool has the ability to improvise on previously generated pattern.

P5.js is one of the most famous library in building computer patterns and graphics, it has all the in build functions to draw 2D figure. It also has other in build libraries like ml5.js (to write machine learning algorithms), p5.dimensions (to build 3D modelling in p5 environment), p5.easycam (to make simple 3d camera controls like zoom and rotate) etc. So these days there are many algorithms which we can write in javascript (using p5.js) to create computational patterns like Tessellate, Euclidian tilings, Edge tesselation, Penrose tiling etc. Checkout below references to explore pattern generation algorithms in p5.js, many of them also build the foundation for industrial level of graphical projects.

When it comes to writing machine learning code for my hand detection function, I will be using ml5.js library, this library is build on the top of Tensorflow library, it has all the in build function to detect body parts and movements. With the help of this library we can easily integrate neural network in p5.js to indentify artist hand through webcam.

Here first I am going to learn how hand is detected in p5.js using machine learning then how hand movement are captured to create some art work, as here the sole purpose would be to influence our pattern parameters using hand movements. Apart from that we can also change the camera angle with different hand movements.

When we talk about interactive art there are many different kinds of projects and approaches, as there is a whole new group of computational artist who generates art work based mathematical algorithms. The field of human

computer interaction is also playing a significant role in children entertainment and learning softwares. Few strategies to build interactive-art project which artists are using these:

- Strategy of game
- Strategy of instrument
- Strategy of rhizome
- Strategy of spectacle

If we look into the field of human computer interaction then there is so many application of interactive art project with different type of art forms, so if we specifically talk about webcam arts then there is few projects worth discussing:

- Kurt Caviezel took 4 million photographs through webcams to create a collective-art images.
- Joana Martins, Isabel Valverde, Todd Cochrane and Ana Moura Santos created an art form where users have to dance to generate art patterns.
- Marcin Wichrowski created a teaching augment reality tool for students to learning and create art.
- Stewart-Hamilton had build a tool to pain in 3D world using our webcam.
- Nevalainen Henaes, Axel Rehnberg and Robin created a tool to perform 3D modeling of artifacts using webcam.
- Aryan Misra build a algorithm to Make 2D painting using webcam, in this he used “neural style transfer” to transform the pre generated paintings.

These days interactive art and human computer interaction field is playing a crucial role in the industry, computer programmers not only consider these projects for better user experience but also look forward to solve some social problems, for example increasing children learning ability. And to create better user experience artist uses all different kind of approaches and methodology to build their art works, like visual effects engagement, sound effects engagement, engagement through touch effects etc.

Most of the interactive art projects relies heavily on artificial technology, and after the hype of neural network, artist use these technology for all the different purposes like face gesture, body movement, analysing user behaviour etc. they are also using AI to create NFT art works.

- Start from scratch – many of the above project starts from a blank canvas, where users have to start their art work from scratch.
- Two dimension interaction – most of the interactive art project are build up on 2D drawings, whereas these days computational artists looking forward for three dimension drawings.
- Relying on many equipments – the field of human computer interaction is so vast that artist need big and heavy equipments to generate interactive art, whereas my project will just need a webcam to start building random arts.
- No improvisation – all the art projects lack in one basic feature that is improvisation, there is no way for next artist to build on the top of other artist creations.

From the above analysis we can conclude that in field of interactive art, artists and programmers had already created random art generating algorithms, hand detection algorithms, and on the top of that there are many fully operating interactive art projects, but there is still a gap for two main features, first building 3D art works using hand movements, and second improvisation feature (where one artist can build on the top of previously generated art work).

My major motivation comes from:

- Andy Lomas (my supervisor) – Is a computational artist and mathematician, ha had worked on many industrial level projects (like matrix movie, avatar movie) and had also won the Emmy award.

- Jake Elwes – Is a media artist, he has been working on AI systems and machine learning art projects, and had exhibited his art work in museums and galleries internationally, including the ZKM, Karlsruhe, Germany.
- Memo Akten – is a computer scientist and multi disciplinary artist, he had completed his PhD from goldsmiths, university of London. His art work is also based on deep learning models, and had won Golden Nica Award, Prix Ars Electronica, Linz.

Methods:

Mainly I will be using two different types of methodology to build my project, the first one is action research, to solve the direct problem of art improvisation platform, and the second one is experimental to collect the user data iteratively.
The main purpose to use action research methodology is to deep dive in all the possible techniques for the artist, so that they can build their artwork on the top of other artist creations. Through this methodology I can directly start building the algorithm and can straightforwardly deliver a running p5.js software application, I need to study all the different types of algorithms like hand detection, art generation, camera angle detection, motion detection etc. For this methodology I need to do some basic research and can start programming the project.
The main reason to use experimental methodology is to collect user review of my software. This methodology will create iteration in the development cycle which will help me to improve my software
UI and performance over the time. I can also perform trial and error approach with this methodology. So here I will first build the beta version of the software then I am going to test it with the actual artists and after collecting their data, I will change the algorithm accordingly.
Though there is one more type of methodology I have used here is object oriented methodology, as we can see from above diagram I have written five random art generating algorithms and one hand point detection algo, which means creating their individual object, then calling them inside the sketch file accordingly.
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:

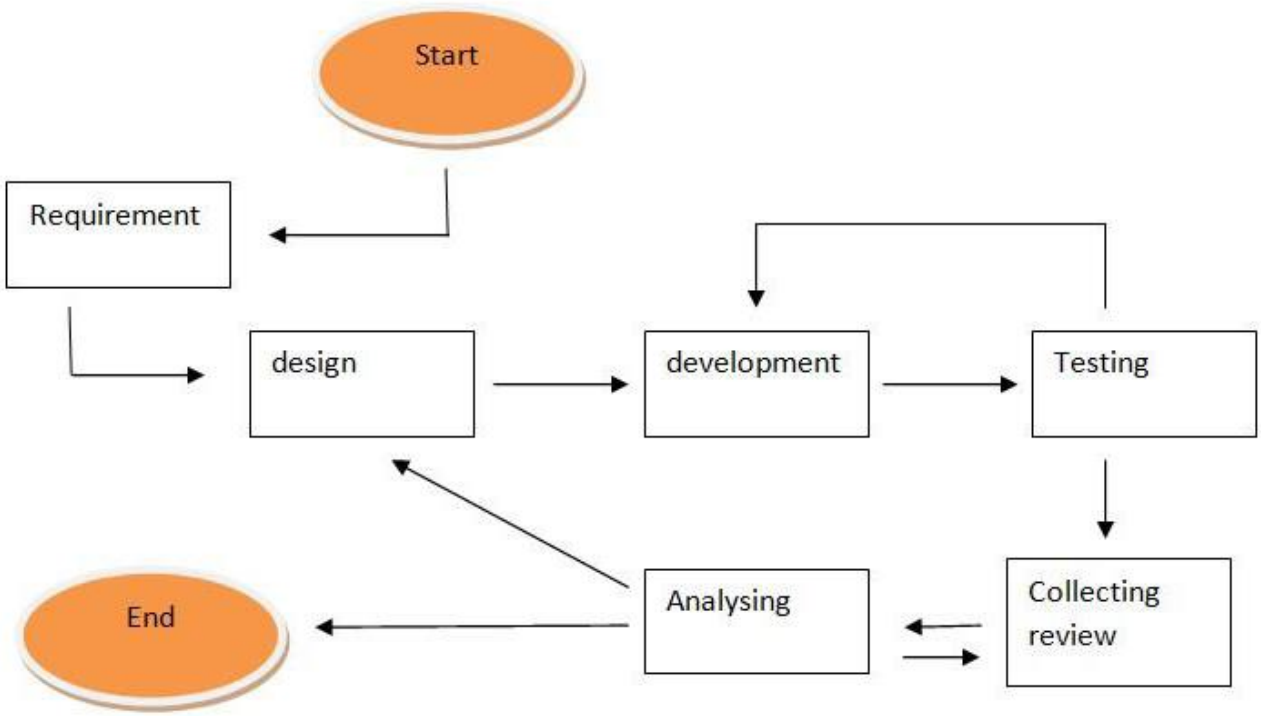
- p5.min – this is the main library, as it contains all the pre build in javascript to for a drawing app.
- p5.dom.min – this library is used to connect our application with the HTML tags, it helps us to maintain an event based programming methodology.
- ml5.min – the main motive behind using this library 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.

Type of resources required:

- Software side = on the programming side I will be using javascript language to create the full application, and main library will be p5.js (as it has all the inbuilt function to draw random art work) and ml5.js (the sole purpose of this library is to implement hand movement detection algorithm. It contains all the machine learning and deep learning model from Tensorflow, so that we can build on the top of that)
- Hardware side = for hardware requirement we will just need a working computer with a webcam. I am trying to build this application with the minimum hardware requirement so that srtist does not have rely on heavy equipments.
- Skill set = one would require a good skills in programming along with the some hands on knowledge of machine learning algorithms. We will also need some knowledge of mathematical random functions, and on the top of that one has to use agile approach to improvise the software. We also need to deep dive in different kind of research papers and topics, like interactive art, random art generators, human computer interaction.

Software life cycle according to my mythologies:



According to my methodology, here software development life cycle will start from writing the requirements, then building design for it, after design will write the code for it, then I am going to test my code, if there is any error I will write it again. Then I am going to collect user data and analyse the performance, if we need some improvement then we will repeat the cycle.

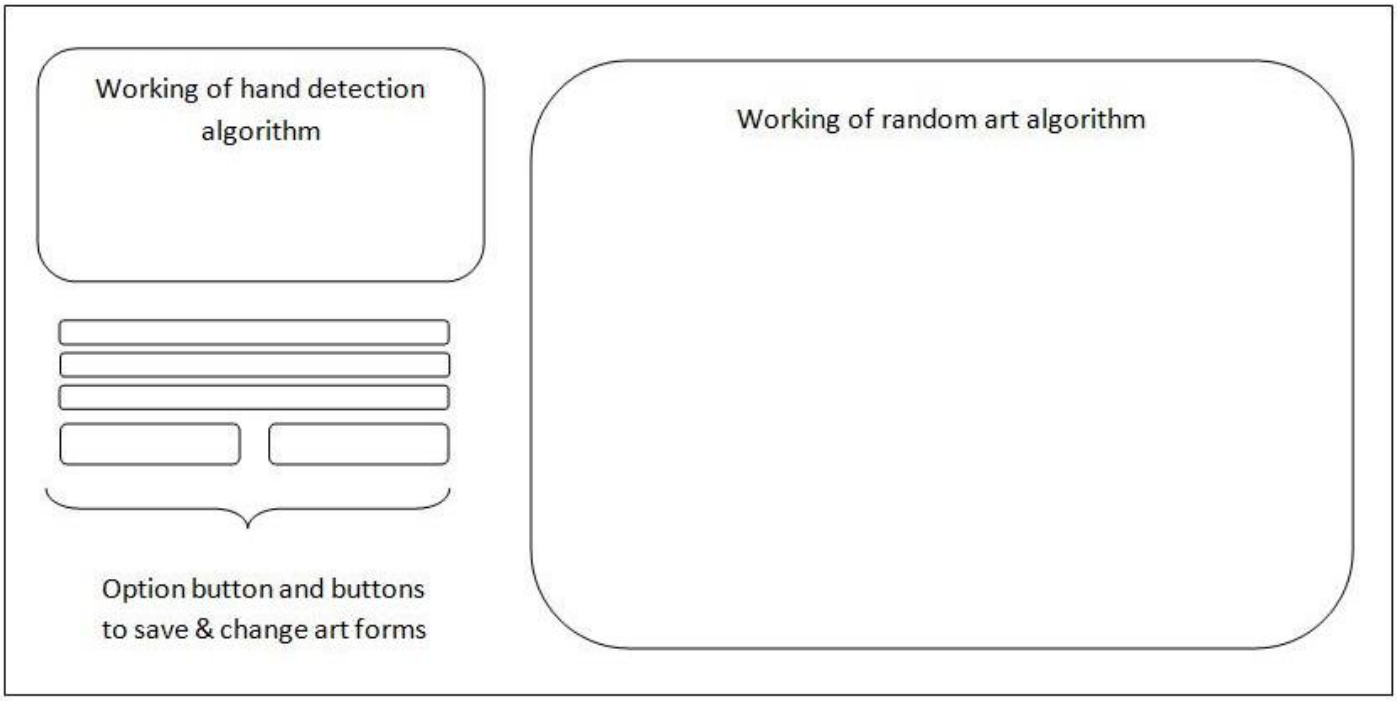
User requirement:

- Pre generated art uploading feature
- Selecting pre generated art forms
- Configuring art parameters with the hand detection algorithm
- Detecting hand through machine learning algorithm
- Saving the art image

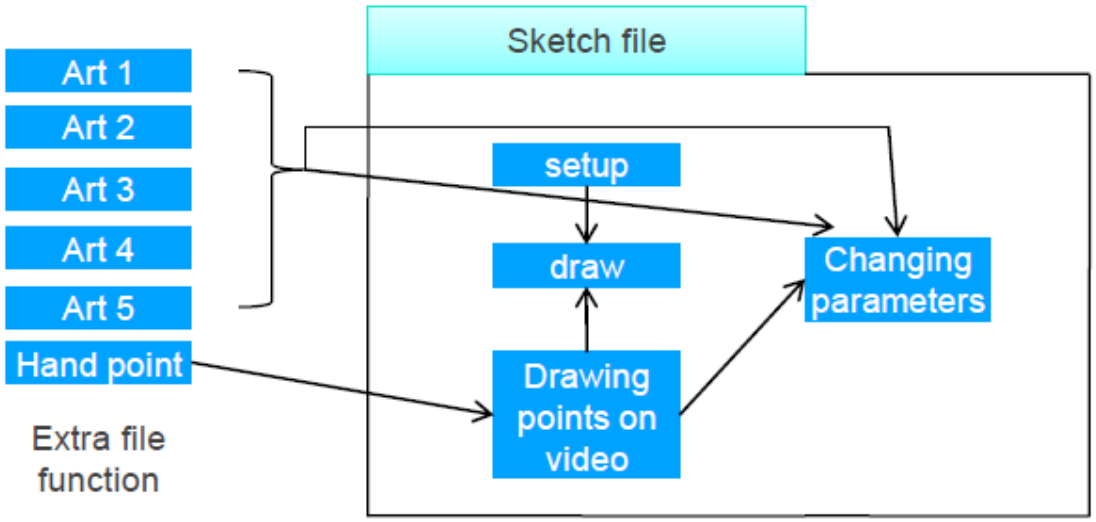
Functionality specification:

- Working of hand detection algorithm
- Working random art generating algorithm
- Having extra option buttons to change art form
- Integration of both the algorithms

Main home page wire frame:



Interaction of files and functions:



Iteration 1 of my software development-------

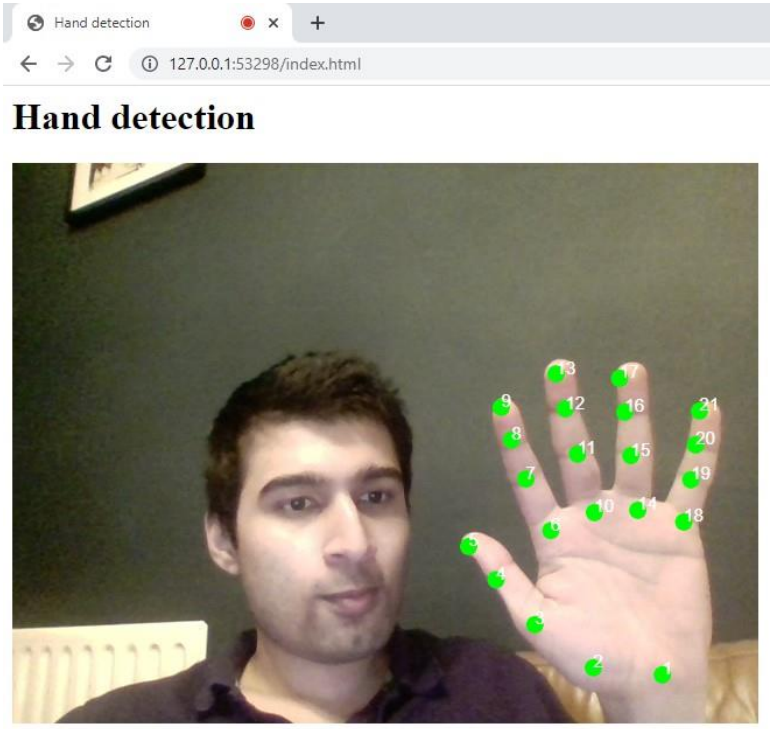
As my whole software relies on two main types of algorithm (hand detection and random art), I have build the initial sample version of both the algorithms as a proof of concept for this project, using javascript p5.js library (refer my GitHub repository).

When it comes to software side technology I will be using javascript most of the time, and the man two libraries will be p5.js and ml5.js, p5.js is the library to build art & drawings in javascript and it contains all the inbuilt function for it. This library draw the art through a html file in the browser. And ml5.js library is used to write the machine learning code for hand detection algorithm, It contains all the machine learning and deep learning model from Tensorflow, and everything I will be building on top of it.

If we talk about hardware side technology, all we need is a working computer with a webcam, I am planning to build this application with the minimum hardware requirement so that artist does not have rely on heavy equipments.

In addition I might use some third party software in future to add any extra feature, improve software performance, achieve flexibility, or to add more features. For example I can use Wekinator for hand detection algorithm, I can use virtual reality glasses to improve user interaction. I might use other p5.js libraries to add some extra functionality, like adding GUI buttons, building 3D art drawings.

Sample implementation of hand detection algorithm ----



The above screenshot is the working first version of our hand detection algorithm. To build this I took reference from ml5.js documentation and other GitHub repositories. This algorithm capture the each frame from webcam then mark all the points if there is any hand detected. As this is the basic layout of the hand detection algorithm, the working of it is quite straightforward, first we create an object of ml5.handpose (function from l5.js library), this object takes the captured video as an argument and the ".on" function returns an array which contains the x and y position of each point. Then inside the draw function I am calling drawpoints function in which I am plotting all the points on screen (refer below code screenshot).

The way I have achieved this is with the nested for loop, as each element of point array is another array and this array stores the final x & y position of each point. Here I am also assigning number to each point and drawing on the top of ellipse.

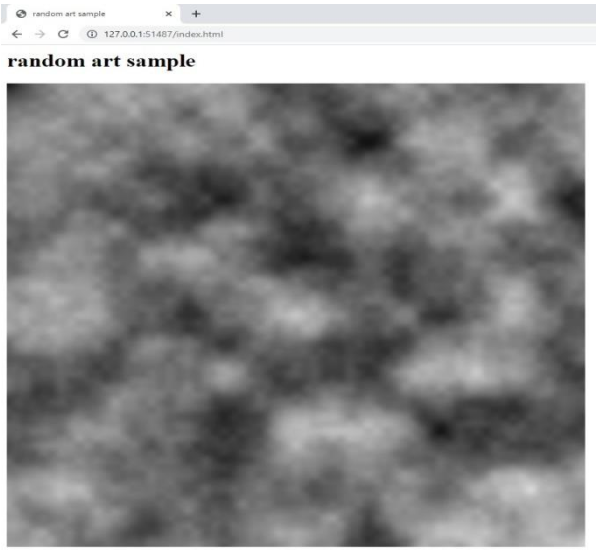
From this algorithm so far I have learn how to use the basic functions of ml5.js library, and how to plot each point in each frame. Later on I will be building lines between them and will measure the distance between them so that user can start interacting with it, and with the help of that we can calculate the distance and then configure it with the random art algorithm. I am also thinking to go beyond simple

hand detection algorithm, i.e. we can also consider the hand gestures and moves to rotate the camera angle for 3D art forms.

```
//working of hand detection algorithm
//took help from https://learn.ml5js.org/#/reference/handpose?id=config
//took help from https://learn.ml5js.org/#/
//took help from https://makeabilitylab.github.io/physcomp/communication/handpose-serial.html

var hand_detection;
var video;
var points_array = [];

function setup() {

  createCanvas(640, 480);
  video = createCapture(VIDEO);
  video.size(width, height);

  hand_detection = ml5.handpose(video, modelReady);

  hand_detection.on("hand", results => {
    points_array = results;
  });
  video.hide();
}

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

function draw() {
  image(video, 0, 0, width, height);

  drawpoints();
}
```

```

function drawpoints() {
  for (let i = 0; i < points_array.length; i += 1) {
    const first_prediction = points_array[i];
      t_number = 1;
    for (let j = 0; j < first_prediction.landmarks.length; j += 1) {
      const keypoint = first_prediction.landmarks[j];
      fill(0, 255, 0);
      noStroke();
      ellipse(keypoint[0], keypoint[1], 15, 15);
        push();
        fill(255);
        textSize(15);
        text(t_number.toString(), keypoint[0], keypoint[1]);
        pop();
        t_number=t_number+1;
    }
  }
}

```

Sample implementation of random art algorithm ----



There are many ways to make random art drawings, but here I have build the basic structure of my algorithm which generate this above art form. I took reference from p5.js documentation and their editor projects to understand the screen pixel concept. The way I have implemented this algorithm is, there is an inbuilt pixel array in p5.js which has 4 values for each pixel (i.e red, green, blue and alpha) and I am assigning random noise value from 0 to 255. Here noise function returns a random generated value between 0 to 1.

```
//took help from https://p5js.org/reference/#/p5/noise
//took help from https://editor.p5js.org/codingtrain/sketches/2_hBcOBrF

var add_val = 0.01;

function setup() {
  createCanvas(800, 800);
  pixelDensity(1);
}

function draw() {
  randomart();
}

function randomart() {
  var y_val = 0;
  loadPixels();
  for (let y = 0; y < height; y++) {
    var x_val = 0;
    for (let x = 0; x < width; x++) {
      var index = (x + y * width) * 4;
      var final_val = noise(x_val, y_val) * 255;
      pixels[index + 0] = final_val;
      pixels[index + 1] = final_val;
      pixels[index + 2] = final_val;
      pixels[index + 3] = 255;

      x_val += add_val;
    }
    y_val += add_val;
  }
  updatePixels();
}

```

First we have to load the pixel array in draw function then run nested for loop (with respect to the width and the height of canvas) to update each pixel value and in the end update the array by calling updatePixels() function.

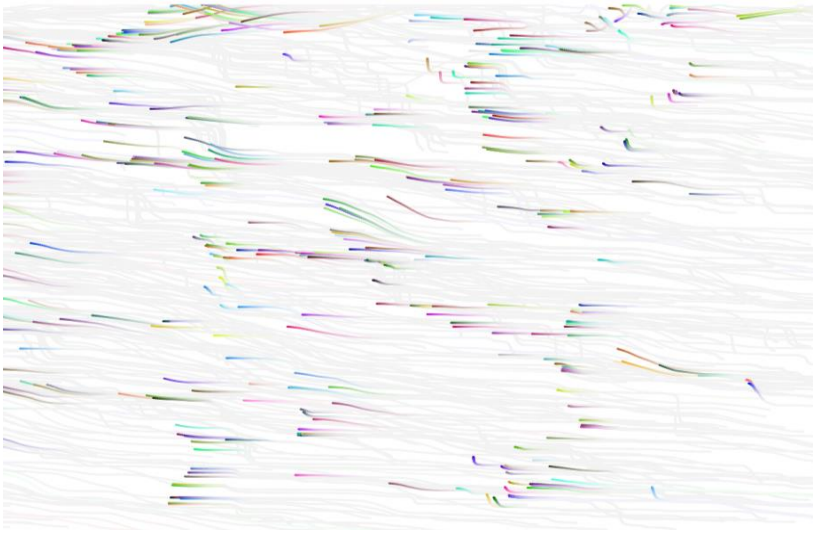
As this is just the basic structure of our algorithm, later on I will different shades and color into it, I will try to create complex patterns and then will finally integrate it with the hand detection algorithm so that user play with its parameters. So far I can say I have learn the concept of pixel array (how to access each row and what its elements) and the usage of noise function.

Here if we think about going beyond from random art generative algorithm, then we can think about creating 3D art forms, adding some extra functionality for the user like selecting between different drawings or uploading their own version of algorithms etc.

Iteration 2 of my software development-------

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.

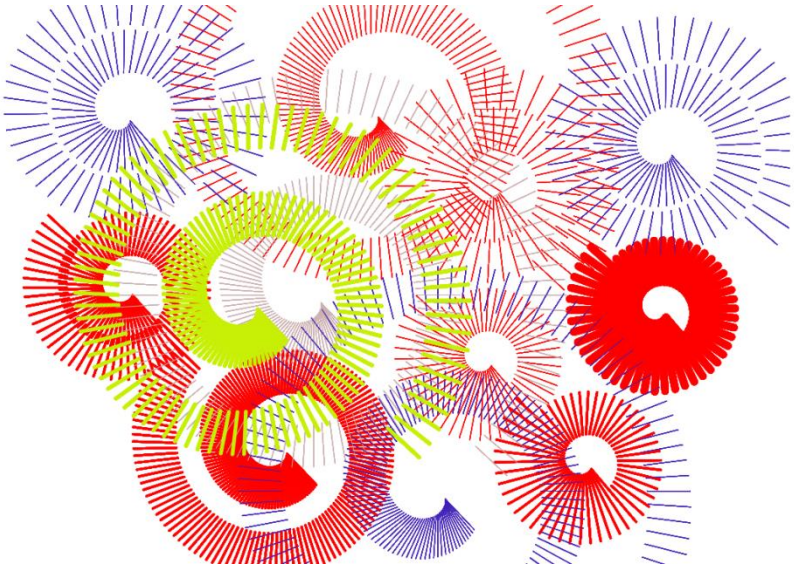
So to make multiple rectangles of random height and width, I have made one global array and then pushing multiple objects (of this function) in it, according to the user given value. Then inside the main draw loop I am iterating over the array and calling the appropriate functions.

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.

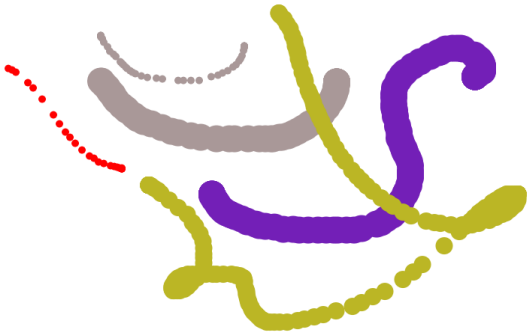
The concept to draw a single pattern is, we call our single_line private function inside the for loop in our draw function and passing all the necessary parameters to it, then single_line function first translate and rotate the canvas accordingly, and in the end it draws a particular line. And in the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

```
1  //rotating line pattern
2
3  function art_3(){
4      this.x=0;
5      this.y=0;
6      this.number_of_l = slider4.value();
7      this.line_gap = slider2.value();
8      this.line_dis = 10;
9      this.line_h=50;
10     this.strokeWeight = slider3.value();
11     this.line_color = colorPicker1.color();
12
13     function single_line(x_s,y_s,lin_h,d_n,
14                          line_dis,s_w,line_color){
15         push();
16         strokeWeight(s_w);
17         stroke(line_color);
18
19         translate(x_s,y_s);
20         rotate((2*PI)/d_n);
21         line(line_dis,line_dis,lin_h,lin_h);
22         pop();
23     }
24     this.draw = function(){
25
26         push();
27         for(var i=0;i<this.number_of_l;i++){
28
29             single_line(this.x,this.y,this.line_h+(i*this.line_gap),
30                         this.number_of_l/((i+1)+(i*this.line_gap)),
31                         this.line_dis+(i*this.line_gap),
32                         this.strokeWeight,this.line_color);
33         }
34         pop();
35
36     }
37 }
```

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.

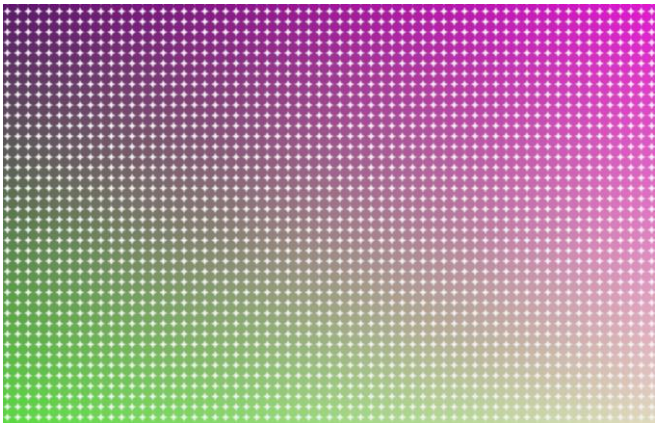
```
1  //canvas drawing pen
2
3  function art_4(){
4      this.el_size = slider1.value();
5      this.colorPicker;
6
7      this.setup = function(){
8          this.colorPicker = createColorPicker('#ff0000');
9          this.colorPicker.position(20, 640);
10     }
11
12
13     this.draw = function(x,y){
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;
    }
}
```

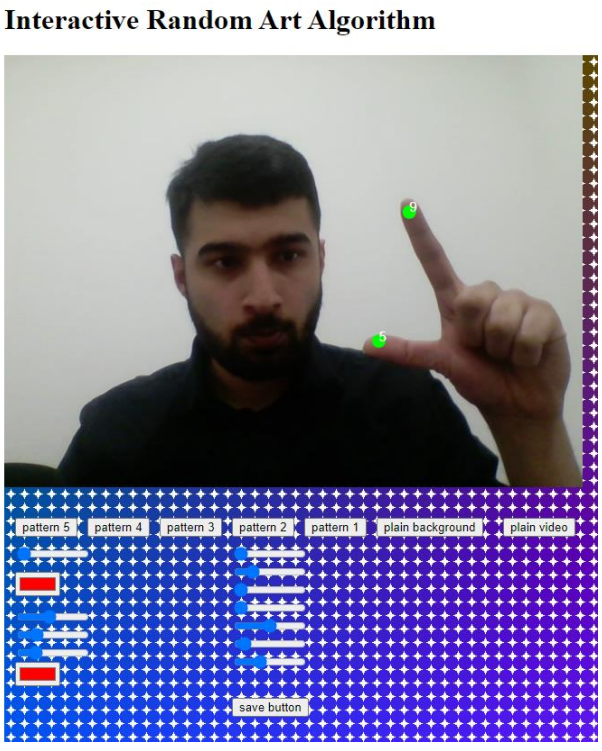
Here I'm creating multiple object of art_5_mini function inside the art_5 function, then I'm assigning x and y values to each circle, and in the end I'm mapping the precise color value. The way it is achieved is by calculating the distance of each circle with respect to the user movement and then changing 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");
}
```

Results:

If I compare it with my actual concept that is interactive art platform where user can play with the art and can save state of their interaction on top of other drawings, then my 5 art algorithms are working properly and so far I am able to detect all the points on hand, plus had also configured a few arts with hand movements, to play with all the parameters of all the arts I have also added GUI button and slider bar. In the end I have also added one save button to save the screenshot of each drawing.

Technical testing:

Here first I have done static testing, where I have checked the code thoroughly for any errors, and found that everything is properly working, i.e. there is no logical or syntax errors. Secondly I have performed black box testing, and found that all the classes were responding correctly. However, I was not able to figure out why a few classes take a bit more time to reload.

And had also done a brief user testing of this project (which is, in terms of performance, functionality, concept and technical aspects)

User testing results -------

User 1:

Software usefulness score = 9.5

required improvement score = 3.8

easy to use score = 7.4

overall software speed = 5.8

time spending on the software = 16 mins

overall software rating = 7

what do you like about the software?

Answer = this user liked that later on if this interactive art platform gets popular that there would be more drawing to play with.

Any improvement suggestion you would like to give?

Answer = this user found my software a bit slow in terms of frame rate.

Do you find it interesting? Yes/No

Answer = yes

Would you use it again? Yes/No

Answer = yes

Is this app worth playing ? Yes/No

Answer = yes

Would you recommend to your friend? Yes/No

Answer = yes

How this new concept could be useful?

Answer = this user said this concept could be useful for beginner artists to learn about random computer art.

User 2:

Software usefulness score = 5

required improvement score = 4.2

easy to use score = 9.1

overall software speed = 5.6

time spending on the software = 15 mins

overall software rating = 8

what do you like about the software?

Answer = this user liked that there were also GUI buttons to change the parameters, which made it easy to understand the working of our software.

Any improvement suggestion you would like to give?

Answer = this user suggested that there should be more hand gestures synchronization.

Do you find it interesting? Yes/No

Answer = yes

Would you use it again? Yes/No

Answer = no

Is this app worth playing ? Yes/No

Answer = yes

Would you recommend to your friend? Yes/No

Answer = no

How this new concept could be useful?

Answer = this user said it could be useful for children to play with interactive art.

User 3:

Software usefulness score = 5

required improvement score = 5.8

easy to use score = 8.1

overall software speed = 6.3

time spending on the software = 20 mins

overall software rating = 6.3

what do you like about the software?

Answer = this user liked that every art was different than the previous one.

Any improvement suggestion you would like to give?

Answer = this user said there should be an option to upload more algorithms too.

Do you find it interesting? Yes/No

Answer = yes

Would you use it again? Yes/No

Answer = no

Is this app worth playing ? Yes/No

Answer = yes

Would you recommend to your friend? Yes/No

Answer = yes

How this new concept could be useful?

Answer = this user said it could be useful for artist to present their interactive art.

So according to the user review I can say that user was able to use the application and was able to interact with all the art but found it hard time to interact through hand movements, mainly because not all the art was configured fully and because of the slow frame rate. I also realized that user also wanted to save the drawing on the top of other drawing, which technically I still have not figured it out. Other than that user enjoyed the overall app experience.

Discussion:

Here if we evaluate the concept, then I have realized that it could be more useful for other purposes rather than building drawing on it, according to the user feedbacks, in future if I add some other functionalities then it could be useful for artist to present their art, to share their art and it could also be used as interactive art app for children etc. This software could be a good starting point between generative art and interactive art, but it was not entirely able to cover the gap between them.

The questionnaire I gave it to users was mainly asking for two types of feedbacks, that is concept testing and performance testing. In performance testing I get know about the usefulness and usability of the software.

Average score according to user feedback:

- Software usefulness score = 6.5
- required improvement score = 4.6
- easy to use score = 8.2
- overall software speed = 5.9
- time spending on the software = 17 mins
- overall software rating = 7.1

What improvement user wants from in our app:

- Better graphical user interface
- User prefer high frame rate
- Better synchronisation of hand movements
- Overall user wants more functionality, like adding their own algorithms.

According to the concept testing and the data I have collected from user, I should have asked a few more question like:

- 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?
- 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?

The aim of my project was to deliver an application where artist and can interact with random generated art forms, but if I have to 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 simple hand movements, 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. And in the last I was also able to implement the save state system too, through which user can save their drawings.

In terms of technical aspects I was not expecting that multiple classes and objects will slow down the frame rate (for that in future I might need to write performance testing function so that I can calculate the time each class takes to reload.) 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 realized 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.

Limitations and future works:

Concept limitation:

I was trying to build a system where artist can not only interact with other artist drawings but can also build on top of that, whereas so far this application can only save the state of our drawing after the interaction. At present this concept could be useful for school children to play the art or it could be there first step or interaction with the art industry, but to implement this platform for professional artists there is a lot more to improve on it.

Technical limitations:

There are a few technical limitation I have faced here, first and the major one was the computing power, as p5.js is not the most effective and efficient library, computer takes a lot of time to process the calculation for all the classes and object. Second technical difficulty was the proper configuration of the hand movements, as can notice because of slow frame rate hand movements are not synchronized with all the arts accordingly.

Future work:

In future we can add one more feature through which user can also upload their new algorithm and can configure it with our hand detection algorithm. I can also try to fix some of the issues like:

- computation power by developing better class and object structure
- can try to improve the configuration of user hand movement
- can try to build a better GUI for the application
- can try to segregate art into separate pages

I would also like to configure hand gestures instead of hand movements, and 3D drawings instead of 2D drawing. This project could be useful to many industries so I am also interested to know how this software could be useful for education industry, because as we know children learn more from interaction rather than just reading books. So I might read a few more research paper in the field of “interactive art for education”.

Conclusion:

So far if I look at my milestones I can say I have completed my task on time as I have properly implemented all the main algorithms and functionalities. I have a lot more to add in future like adding new feature and better GUI. When it comes to management strategy I am basically following my software development lifecycle (I have mentioned my chart above) and my time line table (scroll down for it) to complete my project on time.

There is definitely a few elements which I feel like I could have made better (if I had a bit more time to get more user feedback) like could have tried better hand movements, could have build 3D art etc. But if I consider my overall project with the expected output then I can say it is not the best interactive project but it has a good potential to share a totally new concept in the computational art market. I also felt that I need to learn more about ml5.js (hand detection algorithms) and would also like to explore more complex generative algorithms.

Conceptual wise this project is really big and can be effective in many segments of interactive art industry, so the full usefulness of it is yet to explore. The main conceptual problem I have faced with this project was product market fit. The main reason for this is the scope of project, there are so many features we can add in this project, which had always raise the question that how complex it should be. To deal with such risk I adapted agile approach, so that I can build this entire software. During the development cycle I also kept on asking my friends, users and supervisor to suggest me how to improve on it and how to make it more useable.

In conclusion MVP is working and overall it was an enjoyable experience for users to try something totally new. Though there was a bit of performance issues which technically I am not able to figure it out, but I have learnt a lot throughout the journey, I learnt how to do research in a particular topic, I learnt how hand detection algorithms work, I learnt so many new libraries in p5.js and also learnt about generative art.

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	✓

GitHub repository link:

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

Bibliography:

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).*
21. *P5 noise function (no date) reference | p5.js. Available at: https://p5js.org/reference/#/p5/noise (Accessed: December 30, 2022).*
22. *P5.js web editor (no date) | P5.js web editor. Available at: https://editor.p5js.org/codingtrain/sketches/2_hBcOBrF (Accessed: December 30, 2022).*
23. *Wekinator software (no date) | Wekinator. Available at: http://www.wekinator.org/ (Accessed: December 30, 2022).*
24. *L7: Handpose Serial (no date) | Physical Computing. Available at: https://makeabilitylab.github.io/physcomp/communication/handpose-serial.html (Accessed: December 30, 2022).*
25. *A friendly machine learning library for the web. (no date) | ml5. Available at: https://learn.ml5js.org/#/ (Accessed: December 30, 2022).*
26. *ml5 handpose algorithm. (no date) | ml5. Available at: https://learn.ml5js.org/#/reference/handpose?id=config (Accessed: December 30, 2022).*
27. *Hello! (no date) home | p5.js. Available at: https://p5js.org/ (Accessed: December 30, 2022).*

28. *Get started (no date) get started | p5.js. Available at: https://p5js.org/get-started/ (Accessed: December 30, 2022).*
29. *Libraries (no date) libraries | p5.js. Available at: https://p5js.org/libraries/ (Accessed: December 30, 2022).*
30. *List of animation studios (2023) Wikipedia. Wikimedia Foundation. Available at: https://en.wikipedia.org/wiki/List_of_animation_studios (Accessed: May 5, 2023).*
31. *University, G.P.G.Q. et al. (2011) The wheatstone stereoscopic random line pair generator with fitness function for non-objective art: Proceedings of the 8th ACM Conference on Creativity and Cognition, ACM Conferences. Available at: https://dl.acm.org/doi/pdf/10.1145/2069618.2069728 (Accessed: May 5, 2023).*
32. *IEEE Xplore Full-text PDF: (no date). Available at: https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=7169508 (Accessed: May 5, 2023).*
33. *IEEE Xplore Full-text PDF: (no date). Available at: https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=7169508 (Accessed: May 5, 2023).*
34. *McCauley, J. (2020) The Best Free Pattern Generation Tools, Creative Bloq. Creative Bloq. Available at: https://www.creativebloq.com/features/free-pattern-generation-tools (Accessed: May 5, 2023).*
35. *Deconbatch (2022) Making a moire patterns with big holes (p5.js)., deconbatch's Land of 1000 Creative Codings. deconbatch's Land of 1000 Creative Codings. Available at: https://www.deconbatch.com/2022/07/punchmoire.html (Accessed: May 5, 2023).*
36. *Making simple patterns in p5.js (2022) YouTube. YouTube. Available at: https://www.youtube.com/watch?v=ig0q6vfpD38 (Accessed: May 5, 2023).*
37. *Geometric pattern with "nested for loops" in p5.js (2020) YouTube. YouTube. Available at: https://www.youtube.com/watch?v=EMIoUxLEYJ4 (Accessed: May 5, 2023).*
38. *Part 01 (no date) The aesthetic of generative coding. Available at: https://digitalideation.github.io/ba_222_gencg_h1801/notebooks/week01.html (Accessed: May 5, 2023).*
39. *Inconvergent, A.H. (2016) On generative algorithms: Introduction · inconvergent, inconvergent.net. Available at: https://inconvergent.net/generative/#introduction (Accessed: May 5, 2023).*
40. *(no date) Packt subscription. Available at: https://subscription.packtpub.com/book/data/9781838821739/4/ch04lvl1sec27/pose-detection-with-ml5js (Accessed: May 5, 2023).*
41. *A friendly machine learning library for the web. (no date) ml5. Available at: https://learn.ml5js.org/#/reference/handpose (Accessed: May 5, 2023).*
42. *L6: Ml5.js serial (no date) Physical Computing. Available at: https://makeabilitylab.github.io/physcomp/communication/ml5js-serial.html (Accessed: May 5, 2023).*
43. *L7: Handpose Serial (no date) Physical Computing. Available at: https://makeabilitylab.github.io/physcomp/communication/handpose-serial.html (Accessed: May 5, 2023).*
44. *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: May 5, 2023).*
45. *www.youtube.com. (n.d.). Multiple Hands Detection in p5.js. [online] Available at: https://www.youtube.com/watch?v=BX8ibqq0MJU [Accessed 5 May 2023].*
46. *Xixia Liu , Musen Liu. (n.d.). Design and Implementation of Human-Computer Interface for Participatory Art Video Development Platform Based on Interactive Non-linear Algorithm. [online] Available at: https://www.frontiersin.org/articles/10.3389/fpsyg.2021.725761/full.*
47. *Edmonds, E. (2014). Human Computer Interaction, Art and Experience. Springer series on cultural computing, pp.11–23. doi:https://doi.org/10.1007/978-3-319-04510-8_2.*
48. *TrendHunter.com. (n.d.). 72 Interactive Art Installations. [online] Available at: https://www.trendhunter.com/slideshow/interactive-art-installations.*
49. *Misra, A. (2019). Making Art With Your Webcam. [online] Medium. Available at: https://towardsdatascience.com/making-art-with-your-webcam-ac6d0f5504f4 [Accessed 5 May 2023].*
50. *Nevalainen Henaes, Axel and Rehnberg, R. (2021). 3D modelling with a webcam. [online] Chalmers.se. Available at: https://odr.chalmers.se/items/fabed1c3-7f89-4cc4-9d04-60db433dc23d [Accessed 5 May 2023].*

51. *Kluszczynski, RyszardW. (2010). Strategies of interactive art. Journal of Aesthetics & Culture, 2(1), p.5525. doi:https://doi.org/10.3402/jac.v2i0.5525.*
52. *Mallonee, L. (n.d.). This Guy Peers Through People's Open Webcams to Create Art. [online] Wired. Available at: https://www.wired.com/2016/05/kurt-caviezel-wallpaper/ [Accessed 5 May 2023].*
53. *Anon, (n.d.). A Webcam Interface for Somatic-Technological Dance Experiences. [online] Available at: https://sure.sunderland.ac.uk/id/eprint/13317/3/ISEA2016_Proceedings.pdf#page=338.*
54. *Wichrowski, M. (2013). Teaching augmented reality in practice. Proceedings of the International Conference on Multimedia, Interaction, Design and Innovation. doi:https://doi.org/10.1145/2500342.2500362.*
55. *Staffpublished, C.B. (2009). Paint in 3D via your webcam. [online] Creative Bloq. Available at: https://www.creativebloq.com/3d/paint-3d-your-webcam-11098974 [Accessed 5 May 2023].*
56. *Cho, O.-H. and Lee, W.-H. (2011). Analysis about Application of Learning System into Interactive Digital Art Implementation. [online] IEEE Xplore. doi:https://doi.org/10.1109/ISPAW.2011.72.*
57. *Chen, X. and Liu, F. (2022). Application of Deep Neural Networks and Human-Computer Interaction Technology in Art and Design Area. Wireless Communications and Mobile Computing, [online] 2022, pp.1–9. doi:https://doi.org/10.1155/2022/6898758.*
58. *Lavigne, S. (2023). p5.PatGrad. [online] GitHub. Available at: https://github.com/antiboredom/p5.patgrad [Accessed 5 May 2023].*
59. *SYM380 (2023). p5.pattern. [online] GitHub. Available at: https://github.com/SYM380/p5.pattern [Accessed 5 May 2023].*
60. *GitHub. (2023). freshfork/p5.EasyCam. [online] Available at: https://github.com/freshfork/p5.EasyCam [Accessed 5 May 2023].*
61. *Cook, C. (2023). p5.dimensions.js. [online] GitHub. Available at: https://github.com/Smilebags/p5.dimensions.js [Accessed 5 May 2023].*
62. *R, F. (2023). p5.3D. [online] GitHub. Available at: https://github.com/FreddieRa/p5.3D [Accessed 5 May 2023].*
63. *Ml5js.org. (2019). ml5js·Friendly Machine Learning For The Web. [online] Available at: https://ml5js.org/.*
64. *P5js.org. (2019). p5.js | home. [online] Available at: https://p5js.org/.*
65. *AIArtists.org. (n.d.). Generative Art: Best Examples, Tools & Artists (2020 GUIDE). [online] Available at: https://aiartists.org/generative-art-design.*
66. *www.jakeelwes.com. (n.d.). Jake Elwes - About. [online] Available at: https://www.jakeelwes.com/about.html [Accessed 5 May 2023].*
67. *www.andylomas.com. (n.d.). Andy Lomas about. [online] Available at: https://www.andylomas.com/about.html [Accessed 5 May 2023].*
68. *Memo Akten | Mehmet Selim Akten | The Mega Super Awesome Visuals Company. (n.d.). Info. [online] Available at: https://www.memo.tv/.*
69. *Felipe Duarte, E., Merkle, L.E. and C. Baranauskas, M.C. (2019). The Interface between Interactive Art and Human-Computer Interaction: Exploring Dialogue Genres and Evaluative Practices. Journal of Interactive Systems, 10, p.20. doi:https://doi.org/10.5753/jis.2019.551.*
70. *Edmonds, E., Turner, G. and Candy, L. (2004). Approaches to interactive art systems. Proceedings of the 2nd international conference on Computer graphics and interactive techniques in Austalasia and Southe East Asia - GRAPHITE '04. doi:https://doi.org/10.1145/988834.988854.*

71. *Wikipedia Contributors (2019). Interactive art. [online] Wikipedia. Available at: https://en.wikipedia.org/wiki/Interactive_art.*
72. *Hummels, C. and Pieter Jan Stappers (1998). Meaningful gestures for human computer interaction: beyond hand postures. IEEE International Conference on Automatic Face and Gesture Recognition. doi:https://doi.org/10.1109/afgr.1998.671012.*