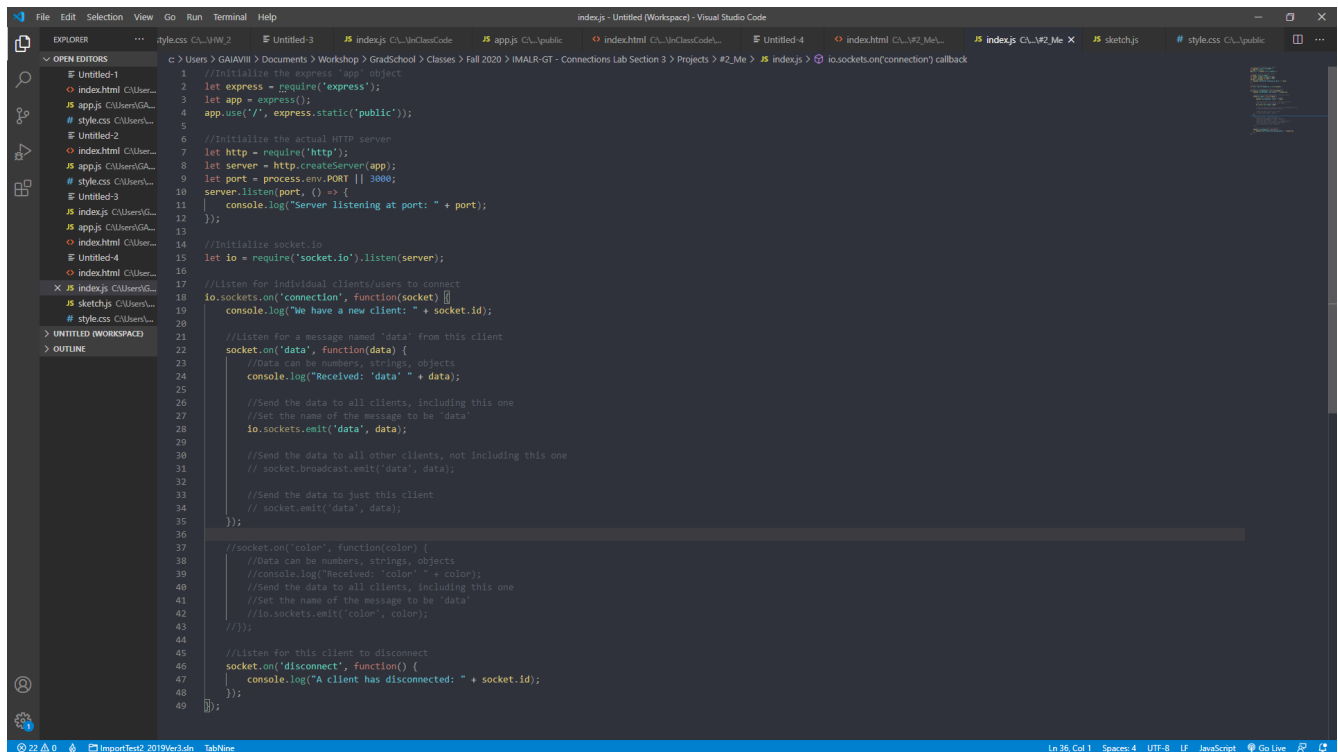


Chris Wray – Project 2 Overview

Project 2 started with the relatively simple idea of creating an experience that allowed the user to have an art therapy coloring experience, while also tapping into cross media capabilities. The original intent was to include calming music, and a metamorphic landscape along with the default drawing canvas that permitted the user to interact with another person in a parallel session. Mandalas are intended to be used as a means of both giving the user something to directly color onto, and something to help focus their mind during these exercises.

Index.js:

As I went along the coding process admittedly things became a bit more difficult. I had some issues with the expanded sockets color feature, and had to roll that back. I ended up using the normal sockets features that a number of drawing applications used in order to keep the experience simple, and reliable. Another way of implementing color in a calming manner was implemented in sketch.js, but it would be nice to go back and get both working together.



```
1 //Initialize the express app object
2 let express = require("express");
3 let app = express();
4 app.use('/', express.static('public'));
5
6 //Initialize the actual HTTP server
7 let http = require('http');
8 let server = http.createServer(app);
9 let port = process.env.PORT || 3000;
10 server.listen(port, () => {
11   console.log("Server listening at port: " + port);
12 });
13
14 //Initialize socket.io
15 let io = require('socket.io').listen(server);
16
17 //Listen for individual clients/users to connect
18 io.sockets.on('connection', function(socket) {
19   console.log("We have a new client: " + socket.id);
20
21   //Listen for a message named 'data' from this client
22   socket.on('data', function(data) {
23     //Data can be numbers, strings, objects
24     console.log("Received: 'data' " + data);
25
26     //Send the data to all clients, including this one
27     //Set the name of the message to be 'data'
28     io.sockets.emit('data', data);
29
30     //Send the data to all other clients, not including this one
31     // socket.broadcast.emit('data', data);
32
33     //Send the data to just this client
34     // socket.emit('data', data);
35   });
36
37   //socket.on('color', function(color) {
38     //Data can be numbers, strings, objects
39     //console.log("Received: 'color' " + color);
40     //Send the data to all clients, including this one
41     //Set the name of the message to be 'data'
42     //io.sockets.emit('color', color);
43   });
44
45   //Listen for this client to disconnect
46   socket.on('disconnect', function() {
47     console.log("A client has disconnected: " + socket.id);
48   });
49 });
```

Sketch.js:

Coloring was replicated by creating a paintbrush in p5 that was capable of rotating through the full color spectrum as the user paints upon the canvas. The “**colormode**” functionality was used for this task, and an “**if**” statement was also used to cycle through “**hue**” in order to wrap up this coloring code.

```
40 //let color_picker = document.getElementById("pickcolor");
41 let color_picker = document.getElementById("pickcolor");
42 //let color_btn = select("#color-btn");
43 let color_btn = document.getElementById("color-btn");
44 //let color_holder = select("#color-holder");
45 let colorStat = color_picker;
46
47 color_btn.addEventListener("click", function() {
48 | let color = color_picker.value();
49 })
50
51 //Sends color data to the server
52 socket.emit('color', color);
53 }*/
54
55 // Adding a mousePressed listener to the stroke width button
56 stroke-button.addEventListener("click", function () {
57 | let width = parseInt(stroke_width_picker.value());
58 | if (width > 0) strokeWidth = width;
59 });
60
61 function mousePressed(pos) {
62 | //stroke(drawPos(pos));
63 | drawPos(pos);
64 }
65
66 function drawPos(pos) {
67 | //Drawing of p5 objects happens down here
68 | // Allows user to draw a colored, rainbow brush by default
69 | if (hue >= 360) {
70 | | hue = 0;
71 | } else {
72 | | hue++;
73 | }
74
75 | //Selects full spectrum color mode
76 | colorMode(HSL, 360);
77 | noStroke();
78
79 | // Incorporates rainbow brush into fill function
80 | fill(hue, 200, 200);
81 | //stroke(pos.x, 0, pos.y);
82 | strokeWeight(strokeWidth);
83 | ellipse(pos.x, pos.y, 10, 10);
84 | //ellipse(pos.x-5, pos.y-5);
85 | //ellipse(pos.x*-1+600, pos.y-5);
86 | //ellipse(pos.x*-1+600, pos.y*-1+400);
87 | //ellipse(pos.x-5, pos.y*-1+400);
88 }
89
90 //Listen for confirmation of connection
91 socket.on('connect', function () {
92 | console.log("Connected");
93 });
94
95 function setup() {
96 | //Create a p5 canvas based on the size of the active users window width and height
97 | createCanvas(windowWidth, windowHeight);
98 | //createCanvas(800, 600);
99
100 | //Clear out the background of this new canvas to white
101 | background(255);
102
103 | //listen for messages named 'data' from the server
104 | socket.on('data', function (obj) {
105 | | console.log(obj);
106 | | //colorChoice(obj);
107 | | drawPos(obj); //This section is the invocation that "draws"
108 | | //button.onhold = mousePressed(obj);
109 | });
110 }
111
112 function mouseMoved() {
113 | //Stores mouse Coordinates in var
114 | let mousePos = { x: mouseX, y: mouseY, px: pmouseX, py: pmouseY };
115
116 | //Sends mouse position object to the server
117 | socket.emit('data', mousePos);
118 }
119
120 /*function colorChoice() {
121 | //Stores mouse Coordinates in var
122 | //let color_picker = select("#pickcolor");
123 | let color_picker = document.getElementById("pickcolor");
124 | //let color_btn = select("#color-btn");
125 | let color_btn = document.getElementById("color-btn");
126 | //let color_holder = select("#color-holder");
127 | let colorStat = color_picker;
128
129 | color_btn.addEventListener("click", function() {
130 | | let color = color_picker.value();
131 | })
132
133 | //Sends color data to the server
134 | socket.emit('color', color);
135 }*/
```

The mouse functionality was expanded to allow for more precision in the mouse variable with “mousePos”, and “ellipse” was also used as the default draw shape.

Index.html & style.css:

There was another feature that was partially implemented, and that is the creation of 5 icons that would allow the user to place a Mandala over the canvas layer.



This brings up the main pain point of this implementation of the concept thus far, and that is the usage of p5 for the creation of the canvas for drawing. As everything has to be routed through p5 seemingly to add/extend features this proved to be difficult. The “Mandala” feature is intended to be added, but due to the issues that I’ve had with p5 it is not unfortunately.

```
1 <!DOCTYPE html>
2 <html lang="en">
3
4 <head>
5   <meta charset="UTF-8">
6   <meta name="viewport" content="width=device-width, initial-scale=1.0">
7   <script src="index.js"></script>
8   <!--<script src="app.js"></script-->
9   <link href="style.css" rel="stylesheet">
10  </link>
11  <title>Drawing with Sockets and p5</title>
12 </head>
13
14 <body>
15   <!--<p>Choose color (# hex)</p>
16   <input type="text" name="custom_color" placeholder="#FFFFFF" id="pickcolor" class="call-picker" />
17   <div id="color-holder" class="color-holder call-picker"></div>
18   <button id="color-btn">Change color</button>-->
19   <br/>
20
21   <div class="strokes">
22     <p>Choose stroke width</p>
23     <input type="text" name="stroke_width" placeholder="4" id="stroke-width-picker" class="stroke_width_picker" />
24     <button id="stroke-button">Change stroke width</button>
25   </div>
26
27   <div class="mandalas">
28     <!--<p>Choose mandala type</p> -->
29     <div class="shape diamond"></div>
30     <div class="shape circle"></div>
31     <div class="shape square"></div>
32     <div class="shape happy"></div>
33     <div class="shape grand"></div>
34   </div>
35
36   <script type="text/javascript" src="/socket.io/socket.io.js"></script>
37   <script src="https://cdnjs.cloudflare.com/ajax/libs/p5.js/1.1.9/p5.js"></script>
38   <script src="https://cdnjs.cloudflare.com/ajax/libs/p5.js/1.1.9/addons/p5.sound.min.js"></script>
39   <script src="sketch.js"></script>
40 </body>
41
42 </html>
```

5 “Div” tags were created to display these and the CSS was linked to the physical pictures to display the icons for the Mandalas as displayed below.

```

22 .mandalas {
23   position: relative;
24   align-items: flex-start;
25 }
26
27 .shape {
28   display: inline-block;
29   height: 48px;
30   width: 48px;
31 }
32
33 .shape.diamond {
34   width: 46px;
35   height: 44px;
36   background: url("/images/Mandalal.gif");
37   background-position: center;
38 }
39
40 .shape.circle {
41   width: 46px;
42   height: 44px;
43   background-image: url("/images/Mandala2.gif");
44   background-position: center;
45 }
46
47 .shape.square {
48   width: 46px;
49   height: 44px;
50   background-image: url("/images/Mandala3.png");
51   background-position: center;
52 }
53
54 .shape.happy {
55   width: 46px;
56   height: 44px;
57   background-image: url("/images/Mandala4.png");
58   background-position: center;
59 }
60
61 .shape.grand {
62   width: 46px;
63   height: 44px;
64   background-image: url("/images/Mandala5.png");
65   background-position: center;

```

I was not quite sure how to get these linked correctly, but this was due to the forthcoming linkages to get the Mandalas functioning, but like most other features I don't know if I can get p5 to play nice with the features I want to implement.

```

1  body {
2    background-image: url("https://media.giphy.com/media/4a7sWl1NZoRwYmH3p/source.gif");
3    background-position: center;
4    font-family: Arial, Helvetica, sans-serif;
5    font-size: 25px;
6    letter-spacing: 2.5px;
7    font-weight: 500;
8    color: #F8B4F4;
9    text-shadow: -1px 1px 0 #000;
10 }
11

```

Finally, the background was not implemented programmatically, but was instead a linked GIF that was being used as a background. I may push for various psychedelic options, but this worked out because this type of graphic would be difficult to implement in code and the gif is also not particularly resource intensive.

Conclusion:

In a number of ways this felt like a failure because so many features that I wanted to implement did not actually work out as planned. This was of benefit when considering they led to the implementation of the background and multicolored paintbrush. I feel that I tend to learn more from these failures and appreciate the chance to learn more about making p5, sockets, and other related technologies work together. Ideally, I would like to refactor this concept into something more fully fleshed out, and no matter if this involves p5 or html technologies I look forward to the additional lessons to learn through the journey!

