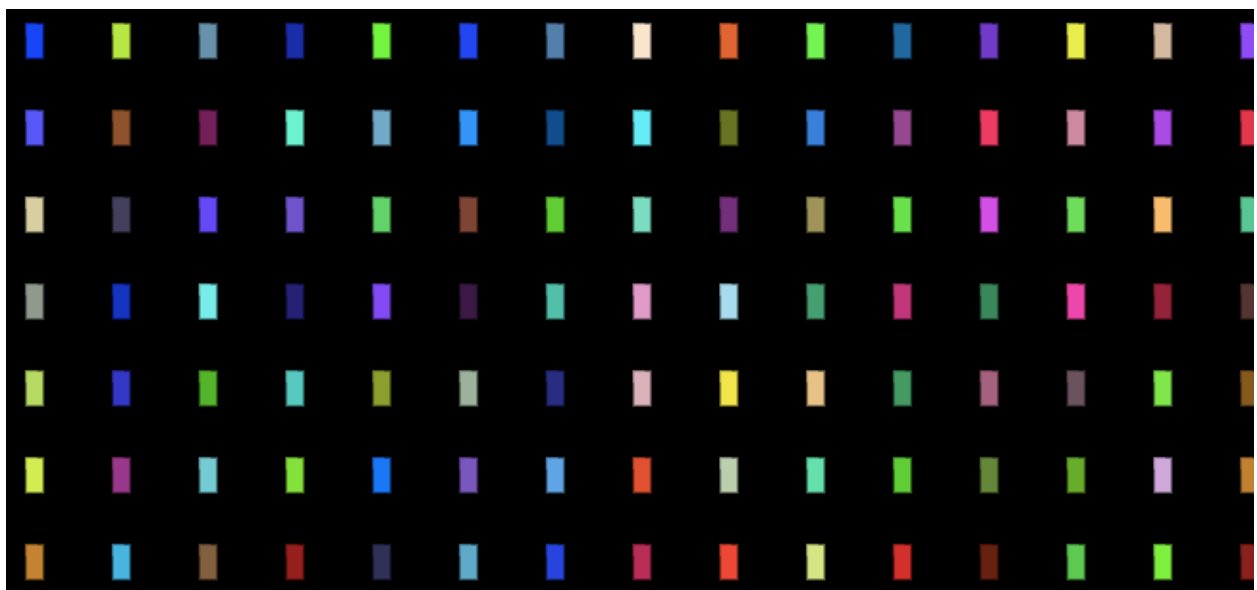


## Midterm Reflection/Documentation:

For this project, I was initially inspired by the movie Bird Box. Solely because of how the movie speaks about mental health in such a subtle yet obvious way. The movie is revolved around a group of characters who all have to remain blinded folded as there is some sort creature who if they look at in the eye, they will immediately kill themselves and those around them. As the movie progresses you begin to realize that the only creature that exists, is the so called “demon” that lives in your head, which are your worst thoughts about yourself and your surroundings, it focuses on how you are your biggest critique and no one is going to be as harsh on you as you are on yourself, thus they show how the biggest monster in your head is you. That was basically the synopsis of the movie and it had me wondering about how much this is related to real life scenario of suicide.

When we think of suicide, you envision one person who is so deep into their mental illness that had to unfortunately take their own lives, yet we do not tend think about the mass of people that going through the exact same thing, instead, the majority think we’re alone in this world. According to WHO, in every forty seconds there is someone taking their own lives. This was the main fact I wanted to incorporate into my project, however as I thought about it, I realized I did not just want to show whom it affects or how many people it affects but the process for an individual to reach the point of having suicidal thoughts and committing suicide.



```

3 function createCell( xOff, yOff, x, y, speed, unit ){
4
5   let cell = {
6
7     xOff: xOff,
8     yOff: yOff,
9     x: x,
10    y: y,
11    speed: speed,
12    unit: unit,
13    xDir: 10,
14    yDir: 10,
15    lifespan: 400,
16    update:update,
17    draw:display
18  }
19
20  return cell
21
22
23
24
25
26
27 function update() {
28   this.x = this.x + this.speed * this.xDir;
29   if (this.x >= this.unit || this.x <= 1) {
30     this.xDir *= -1;
31     this.x = this.x + 1 * this.xDir;
32     this.y = this.y + 1 * this.yDir;
33   }
34   if (this.y >= this.unit || this.y <= 0) {
35     this.yDir *= -1;
36     this.y = this.y + 1 * this.yDir;
37   }
38 }
39
40 function display() {
41   console.log("draw")
42   this.lifespan =
43     fill(random(255),random(255),random(255),this.lifespan);
44   // console.log(this.xOff)
45   rect(this.xOff + this.x, this.yOff + this.y, 6, 6);
46   rect(this.xOff + this.x, this.yOff + this.y, 10, 20);
47 }
48
49
50 let unit = 50;

```

```

50 let unit = 50;
51 let count;
52 let mods = [];
53 let highCount, wideCount
54
55 function setup() {
56   createCanvas(720, 360);
57   noStroke();
58   wideCount = width / unit;
59   highCount = height / unit;
60   count = wideCount * highCount;
61
62   let index = 0;
63   for (let y = 0; y < highCount; y++) {
64
65     let col = []
66     for (let x = 0; x < wideCount; x++) {
67
68       let mod = createCell(
69         x * unit,
70         y * unit,
71         unit / 4,
72         unit / 4,
73         random(0.01, 0.02),
74         unit
75       )
76       col.push(mod)
77     }
78     mods.push(col)
79   }
80 }
81
82 function draw() {
83   background(0);
84
85   for (let y = 0; y < highCount; y++) {
86     for (let x = 0; x < wideCount; x++) {
87       // console.log(mods[y][x])
88       let mod = mods[y][x]
89       let neighBoor = mods[y+1][x]
90       if (mod){
91         fill(random(255))
92         // ellipse(mod.x+mod.xOff,mod.y+mod.yOff,10,10)
93         // mod.update()
94         mod.draw()
95       } else {
96         console.log(x,y)
97       }
98     }
99   }

```

Code is below:

```

66   for (let x = 0; x < wideCount; x++) {
67
68     let mod = createCell(
69       x * unit,
70       y * unit,
71       unit / 4,
72       unit / 4,
73       random(0.01, 0.02),
74       unit
75     )
76     col.push(mod)
77   }
78   mods.push(col)
79 }
80
81
82 function draw() {
83   background(0);
84
85   for (let y = 0; y < highCount; y++) {
86     for (let x = 0; x < wideCount; x++) {
87       // console.log(mods[y][x])
88       let mod = mods[y][x]
89       let neighBoor = mods[y+1][x]
90       if (mod){
91         fill(random(255))
92         // ellipse(mod.x+mod.xOff,mod.y+mod.yOff,10,10)
93         // mod.update()
94         mod.draw()
95       } else {
96         console.log(x,y)
97       }
98     }
99     // console.log(mod)
100
101     // fill(255,255,0)
102
103     // ellipse(mod.x,mod.y,5,5)
104
105   }
106 }
107
108 // for (let i = 0; i < count; i++) {
109 //   mods[i].update();
110 //   mods[i].draw();
111 // }
112
113 // source: https://p5js.org/examples/arrays-objects.html

```

The image above shows my first sketch, which represents my first idea of having a building complex, and when the clock hits forty seconds, a light switches off. However,

in this case, all the windows begin to fade into black which represents the process of being engulfed by a mental illness to the extent of suicide.

After this sketch, I was introduced to the Game of Life by John Horton Conway. Which basically is a zero-player game which have the rules of:

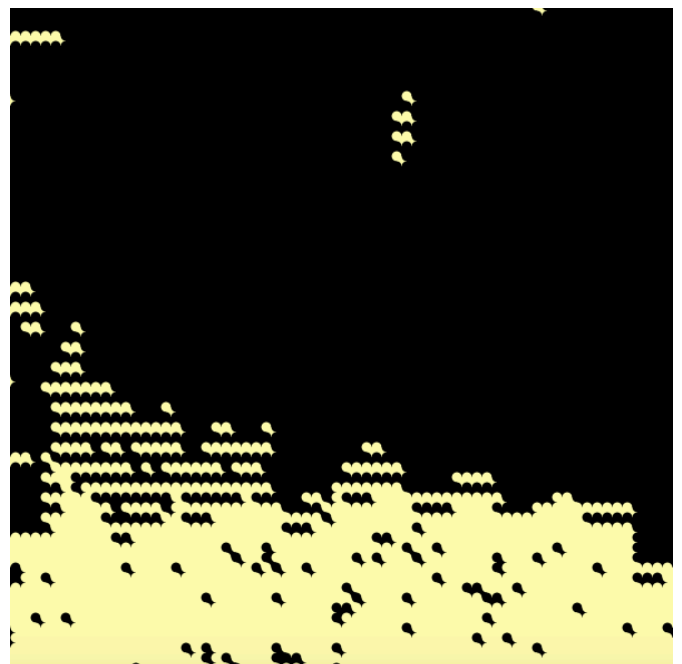
1. Any cell with fewer than two live neighbors dies, by loneliness.
2. Any cell with two or three live neighbors lives continues to the next generation.

However I altered the rules based on my scenario to be:

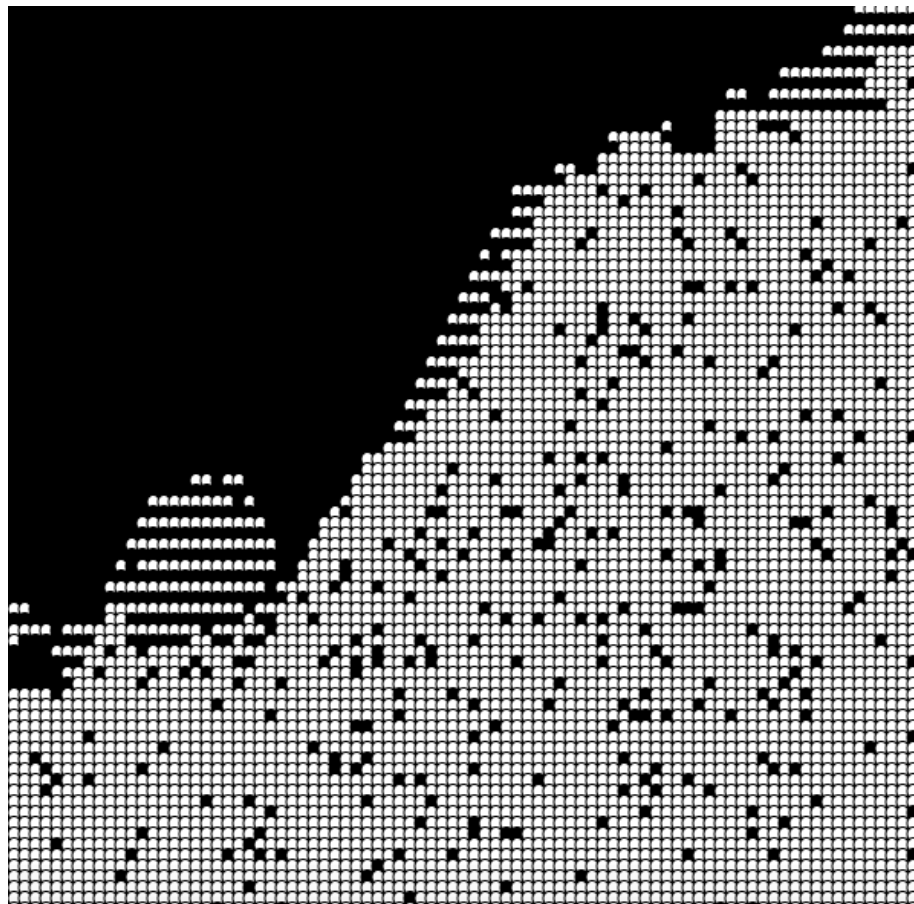
1. Any cell with fewer than one or two neighbors dies but from suicide, they will be engulfed with their depression which in the case of my sketches are the black cells, those who live will receive a new neighbor.
2. Black cells will eat up those who are live, none will survive as it is inevitable.
3. A wave of black cells will kill the remaining cells alive, those who survive are either lonely or have two to three neighbors to accompany them. This represents people who have overcome their mental illnesses.



Example of Rule 1



Example of Rule 2



Example of Rule 3

Code for Rule 3:

```

1 function make2DArray (cols, rows){
2   let array = []
3
4   for (var i = 0; i < cols; i++){
5     let row = []
6
7     for (var j = 0; j < rows; j++){
8       let obj = {
9         x:i,
10        y:j,
11        state: random(0.05)
12      }
13
14      row.push(obj)
15    }
16
17    array.push(row)
18  }
19
20  // console.log(array[4][3])
21
22  // let i = 3
23  // let j = 3
24
25  // let cell = array[i][j]
26  // n2 = array[i+1][j]
27  // n1 = array[i-1][j]
28  // let arr = new Array(cols);
29  // for (let i = 0; i < arr.length; i++) {
30  //   arr[i] = new Array(rows);
31  // }
32
33  return array
34 }
35
36 let grid;
37 let cols;
38 let rows;
39 let resolution = 7;
40
41
42
43
44
45
46
47
48 let resolution = 7;
49
50
51 function setup(){
52   createCanvas(600,600);
53   cols = width / resolution;
54   rows = height / resolution;
55   grid = make2DArray(cols, rows);
56
57 }
58
59
60
61 function draw(){
62   background(0)
63
64   grid = est()
65   display()
66
67 }
68
69
70
71 function display(){
72
73   for (var x = 0; x < grid.length; x++){
74
75     let row = grid[x]
76
77     for (var y = 0; y < row.length; y++){
78
79       let cell = grid[x][y]
80
81       fill(255)
82       stroke(0)
83
84       if (cell.state){
85         fill(0)
86       }
87
88       ellipse(cell.x*resolution, cell.y*resolution, 8, 9)
89
90     }
91
92   }
93
94 }
95

```

```

94     }
95   }
96 }
97 }
98 }
99 }
100
101 function eat(){
102   let newGrid = []
103
104   for (var i = 0; i < grid.length; i++){
105     let newRow = []
106     let row = grid[i]
107
108     for (var j = 0; j < row.length; j++){
109       let score = 0
110       let cell = grid[i][j]
111
112       let a = constrain(i-1, 0, grid.length);
113       let b = constrain(i+1, 0, grid.length);
114       let c = constrain(j-1, 0, row.length);
115       let d = constrain(j+1, 0, row.length);
116
117       // let n1 = grid[a][c]
118       // let n2 = grid[a][j]
119       let n3 = grid[a][d]
120       // console.log(n1)
121
122       let n4 = grid[i][c]
123       let n6 = grid[i][d]
124
125       let n7 = grid[a][j]
126       let n8 = grid[a][c]
127       // let n9 = grid[b][d]
128
129       let neighbours = [n7,n8,n3,n4,n6]
130
131       for (var r = 2; r < neighbours.length; r++){
132         if (neighbours[r]){
133           score += neighbours[r].state
134         }
135       }
136
137       let obj = {
138         x:i,
139         y:j,
140         state: (random()>0.5)
141       }
142     }
143     newRow.push(obj)
144   }
145   newGrid.push(newRow)
146 }

```

```

140   }
141   // console.log(score,a,b,c,d)
142
143   let obj = {
144     x:i,
145     y:j,
146     state: (random()>0.5)
147   }
148   newRow.push(obj)
149 }
150 newGrid.push(newRow)
151 }
152 return newGrid
153 }
154 // function draw(){
155 }

```

## Code for Rule 1:

```

1 function make2DArray (cols, rows){
2   let array = []
3
4   for (var i = 0; i < cols; i++){
5     let row = []
6
7     for (var j = 0; j < rows; j++){
8       let obj = {
9         x:i,
10        y:j,
11        state: (random()>0.5)
12      }
13      row.push(obj)
14    }
15    array.push(row)
16  }
17  // console.log(array[4][3])
18  // let i = 3
19  // let j = 3
20
21  // let cell = array[i][j]
22  // n2 = array[i+1][j]
23  // n1 = array[i-1][j]
24  // let arr = new Array(cols);
25  // for (let i = 0; i < arr.length; i++) {
26  //   arr[i] = new Array(rows);
27  // }
28  return array
29 }
30
31 let grid;
32 let cols;
33 let rows;
34 let resolution = 6;
35
36 let resolution = 6;
37
38 function setup(){
39   createCanvas(600,600);
40   cols = width / resolution;
41   rows = height / resolution;
42   grid = make2DArray(cols, rows);
43 }
44
45 function draw(){
46   background(255)
47
48   grid = eat()
49   display()
50 }
51
52 function display(){
53   for (var x = 0; x < grid.length; x++){
54     let row = grid[x]
55
56     for (var y = 0; y < row.length; y++){
57       let cell = grid[x][y]
58
59       fill(255)
60       strokeWeight(0)
61
62       if (cell.state){
63         fill(0)
64       }
65
66       rect(cell.x*resolution,cell.y*resolution,resolution,resolution)
67     }
68   }
69 }

```

```

94     }
95   }
96 }
97
98 }
99
100 function eat(){
101
102   let newGrid = []
103
104   for (var i = 0; i < grid.length; i++){
105
106     let newRow = []
107     let row = grid[i]
108
109     for (var j = 0; j < row.length; j++){
110
111       let score = 0
112       let cell = grid[i][j]
113
114       let a = constrain(i-1, 0, grid.length);
115       let b = constrain(i+1, 1, grid.length);
116       let c = constrain(j-1, 2, row.length);
117       let d = constrain(j+1, 4, row.length);
118
119
120
121       let n1 = grid[a][c]
122       let n2 = grid[a][j]
123       let n3 = grid[a][d]
124       // console.log(n1)
125
126       let n4 = grid[i][c]
127       let n6 = grid[i][d]
128
129       //let n7 = grid[a][j]
130       let n8 = grid[a][c]
131       //let n9 = grid[b][d]
132
133       let neighbours = [n1,n2,n3,n4,n8]
134
135       for (var r = 3; r < neighbours.length; r++){
136         if (neighbours[r]){
137           score += neighbours[r].state
138         }
139       }
140     }

```

Code for Rule 2:

```

1 function make2DArray (cols, rows){
2
3   let array = []
4
5   for (var i = 0; i < cols; i++){
6
7     let row = []
8
9     for (var j = 0; j < rows; j++){
10
11       let obj = {
12         x:i,
13         y:j,
14         state: (random()>0.5)
15       }
16
17       row.push(obj)
18
19     }
20     array.push(row)
21   }
22
23   // console.log(array[4][3])
24
25   // let i = 3
26   // let j = 3
27
28   // let cell = array[i][j]
29   // n2 = array[i+1][j]
30   // n1 = array[i-1][j]
31   // let arr = new Array(cols);
32   // for (let i =0; i < arr.length; i++) {
33   //   arr [i] = new Array(rows);
34   // }
35
36   return array
37 }
38
39 let grid;
40 let cols;
41 let rows;
42 let resolution = 9;

```

```

51 function setup(){
52   createCanvas(600,600);
53   cols = width / resolution;
54   rows = height / resolution;
55   grid = make2DArray(cols, rows);
56
57
58 }
59
60 function draw(){
61   background(255)
62
63   grid = eat()
64   display()
65
66
67
68 }
69
70 function display(){
71
72   for (var x = 0; x < grid.length; x++){
73
74     let row = grid[x]
75
76     for (var y = 0; y < row.length; y++){
77
78       let cell = grid[x][y]
79
80       fill(255,255,174)
81       stroke(0)
82       strokeWeight(0)
83
84       if (cell.state){
85         fill(0)
86       }
87
88       rect(cell.x*resolution,cell.y*resolution,resolution,resolution)
89       ellipse(cell.x*resolution,cell.y*resolution,resolution,resolution)
90
91     }
92
93   }
94
95 }
96
97
98

```

```
141 // console.log(score,y,j,y);
142
143     let obj = {
144
145         x:i,
146         y:j,
147
148     }
149     if (score < 2){
150         obj.state = true
151     } else if (random()<0.5) {
152         obj.state = true
153
154     } else {
155
156         obj.state = false
157
158     }
159
160     newRow.push(obj)
161
162
163
164 }
165 newGrid.push(newRow)
166
167 }
168
169 return newGrid
170
171 }
172 // function draw(){
173 //     background(0);
174
```

## References:

World Health Organization. (2019). Suicide across the world (2016). [online] Available at: [https://www.who.int/mental\\_health/prevention/suicide/suicideprevent/en/](https://www.who.int/mental_health/prevention/suicide/suicideprevent/en/) [Accessed 20 Oct. 2019].