

# P5.js Body as Interface

Rodrigo Carvalho (@VISIOPHONE\_)  
Processing Community Day Porto 2024

[CODE COLLECTION AT P5JS EDITOR](#)

# P5JS (Processing / javascript / web / browser)

P5JS web.editor: [editor.p5js.org](https://editor.p5js.org)

The screenshot shows the p5.js web editor interface. At the top, there's a pink header bar with the p5 logo. Below it is a toolbar with icons for play, stop, auto-refresh, and account information. The main area has tabs for 'sketch.js' and 'Preview'. The code editor on the left contains the following P5.js code:

```
1 function setup() {
2   createCanvas(windowWidth, windowHeight);
3 }
4
5 function draw() {
6   background(220);
7   fill(0);
8   ellipse(windowWidth / 2, windowHeight / 2, 200, 200);
9 }
```

The preview window on the right shows a large black circle centered on a light gray background. At the bottom left is a 'Console' panel with a 'Clear' button.

Account  
Collections  
Sketches

Hello, visiophone! | My Account ▾

FILE NAME

The screenshot shows the p5.js web editor interface. At the top, there's a navigation bar with 'p5\*' and links for File, Edit, Sketch, and Help. Below the navigation is a toolbar with a play button, a square button, an auto-refresh checkbox, and the text 'Holy stetson'. To the right of the toolbar is a user dropdown with 'Hello, visiophone!' and 'My Account'.

The main area is divided into several sections:

- FILES**: A sidebar on the left containing a large arrow pointing right, followed by 'sketch.js', 'index.html', and 'style.css'.
- WEB**: A section below the files sidebar.
- CODE EDITOR**: A large central area showing the following JavaScript code:

```
// variable to store circles positions X/Y
var posX=400;
var posY=400;

var bSize=100;

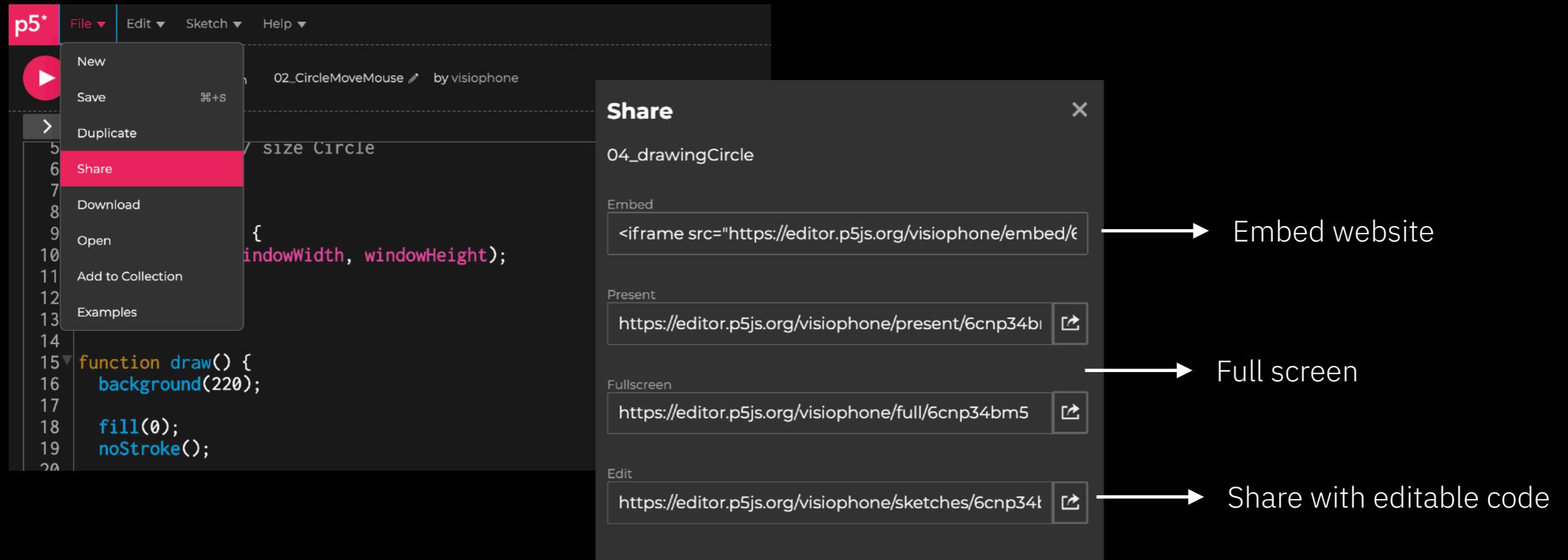
function setup() {
  createCanvas(windowWidth, windowHeight);
}

function draw() {
  background(220);
  fill(0);
  ellipse(width/2,height/2,bSize,bSize);
}
```

- PREVIEW**: A large white canvas on the right side where the output of the sketch is displayed. It currently shows a single black circle centered at the coordinates (400, 400).
- CONSOLE**: A small dark panel at the bottom left with a 'Console' tab and a 'Clear' dropdown.

CONSOLE / PRINT

PREVIEW



p5js / Artes Digitais		
<a href="#">exemplos p5js</a>  <a href="#">Collection by visiophone</a>		
<a href="#">Share</a>  <a href="#">Add Sketch</a>		
Name	Date Added	Owner
01_Circle+Mouse	Sep 29, 2020 2:29 PM	visiophone 
02_CircleMoveMouse	Sep 29, 2020 2:30 PM	visiophone 
03_drawingCircle	Sep 29, 2020 2:41 PM	visiophone 

→ COLLECTIONS

## DIFFERENCES P5JS - PROCESSING

[HTTPS://GITHUB.COM/PROCESSING/P5.JS/WIKI/PROCESSING-TRANSITION](https://github.com/processing/p5.js/wiki/Processing-Transition)

```
1 // variable to store circles positions X/Y
2 var posX=400;
3 var posY=400; ←
4
5 var bSize=100;
6
7 ▼ function setup() { ←
8   createCanvas(windowWidth, windowHeight); ←
9 }
10
11 ▼ function draw() { ←
12   background(220);
13
14   fill(0);
15   ellipse(width/2,height/2,bSize,bSize);
16
17   |
18 }
```

```
float posY=400;
float bSize=100;

void setup() {
  size(displayWidth, displayHeight);
}

void draw() {
  background(220);

  fill(0);
  ellipse(width/2,height/2,bSize,bSize);

}
```

# p5.js

a cheat sheet  
for beginners!

## program structure

```
//runs once when program starts
function setup(){
    createCanvas(800,600);
}

//run continuously after setup
function draw(){
    //rendering loop
}
```

## system variables

**windowWidth / windowHeight**

width / height of window

**width / height**

width / height of canvas

**mouseX / mouseY**

current horizontal / vertical  
mouse position

## non-visual feedback

**print()**

report data to the output console

## color

```
fill(120) gray: 0-255
fill(100,125,255) r, g, b: 0-255
fill(255, 0, 0, 50) r, g, b, alpha
fill('red') color string
fill('#ccc') 3-digit hex
fill('#222222') 6-digit hex fill
color(0, 0, 255) p5.Color object
```

## 2d primitives

**line(x1, y1, x2, y2)**

**ellipse(x1, y1, width, height)**

**rect(x1, y1, width, height)**

**arc(x1, y1, width, height, start, stop)**

**beginShape();**

**vertex(x1, y1);**

**vertex(x2, y2);**

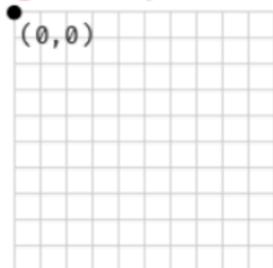
**vertex(x3, y3);**

//add more vertex

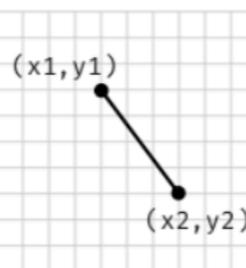
**endShape(CLOSE);**

**text("string", x, y, boxwidth, boxheight)**

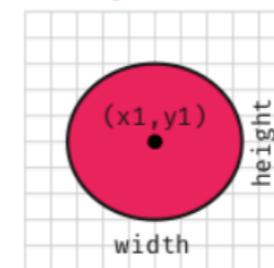
**grid system**



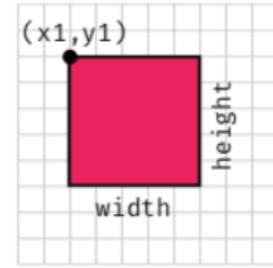
**line()**



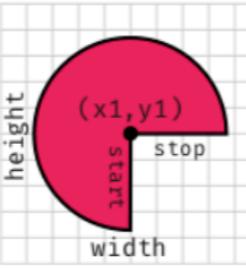
**ellipse()**



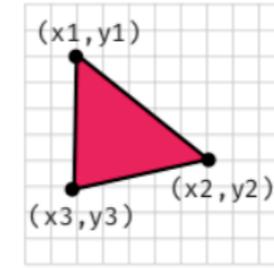
**rect()**



**arc()**



**vertex()**



**math**

+ - / \*

**random(low,high)**

**map(value, in1, in2, out1, out2)**

map a value from input to output range

## attributes

**background(color)**

set the background color

**fill(color)**

set the fill color

**noFill()**

disables fill

**stroke(color)**

set the stroke color

**strokeWeight(weight)**

set the stroke's width

**noStroke()**

disables stroke

**ellipseMode(MODE)**

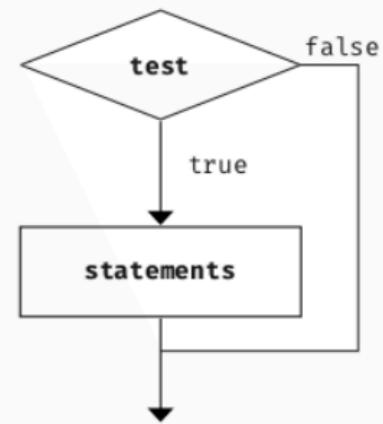
**rectMode(MODE)**

CENTER, CORNER

**textSize(pixels)**

## if/then logic

```
if(test){  
    statements  
}
```



**== equal to**

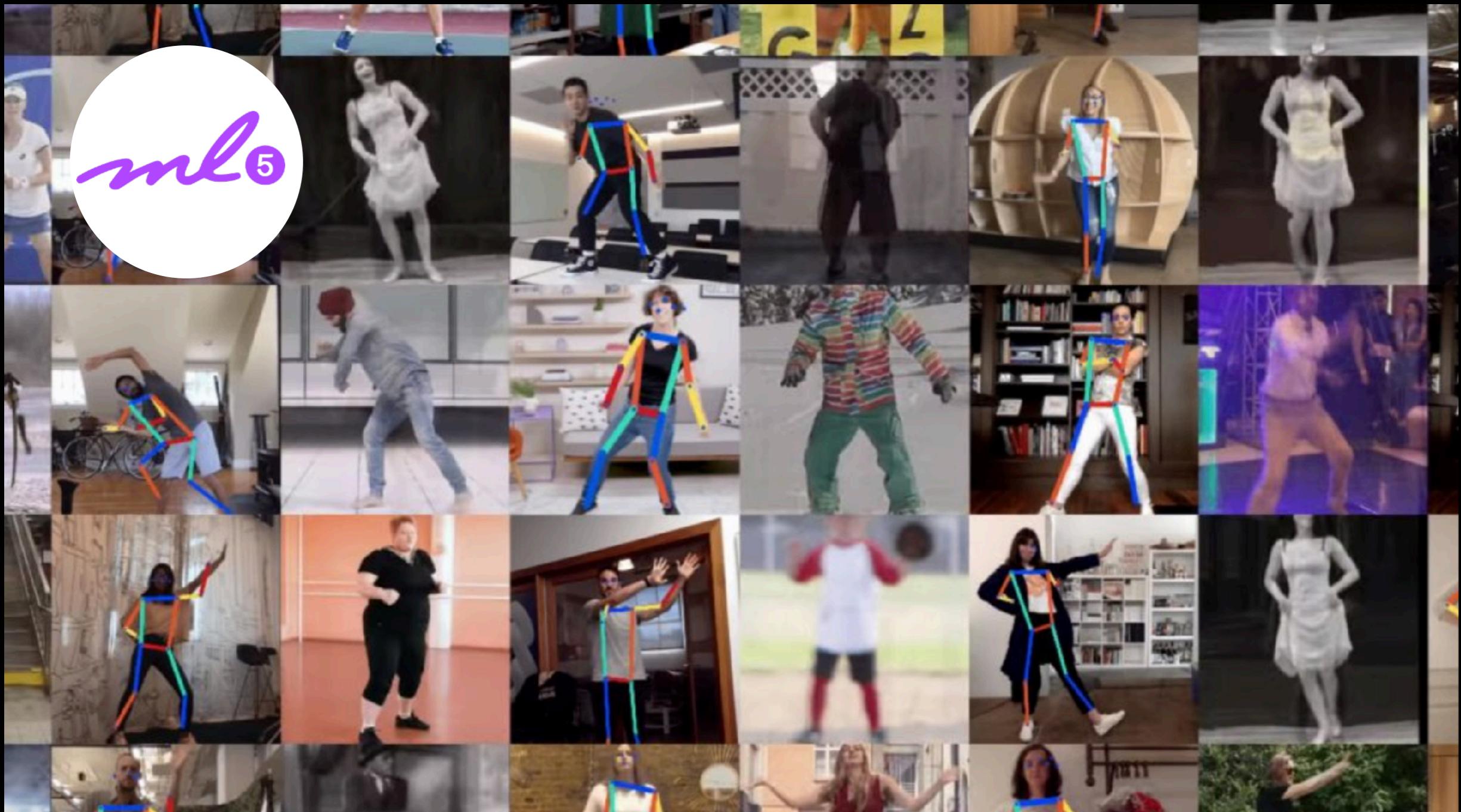
**!= not equal**

**> greater than**

**< less than**

**>= greater than or equal**

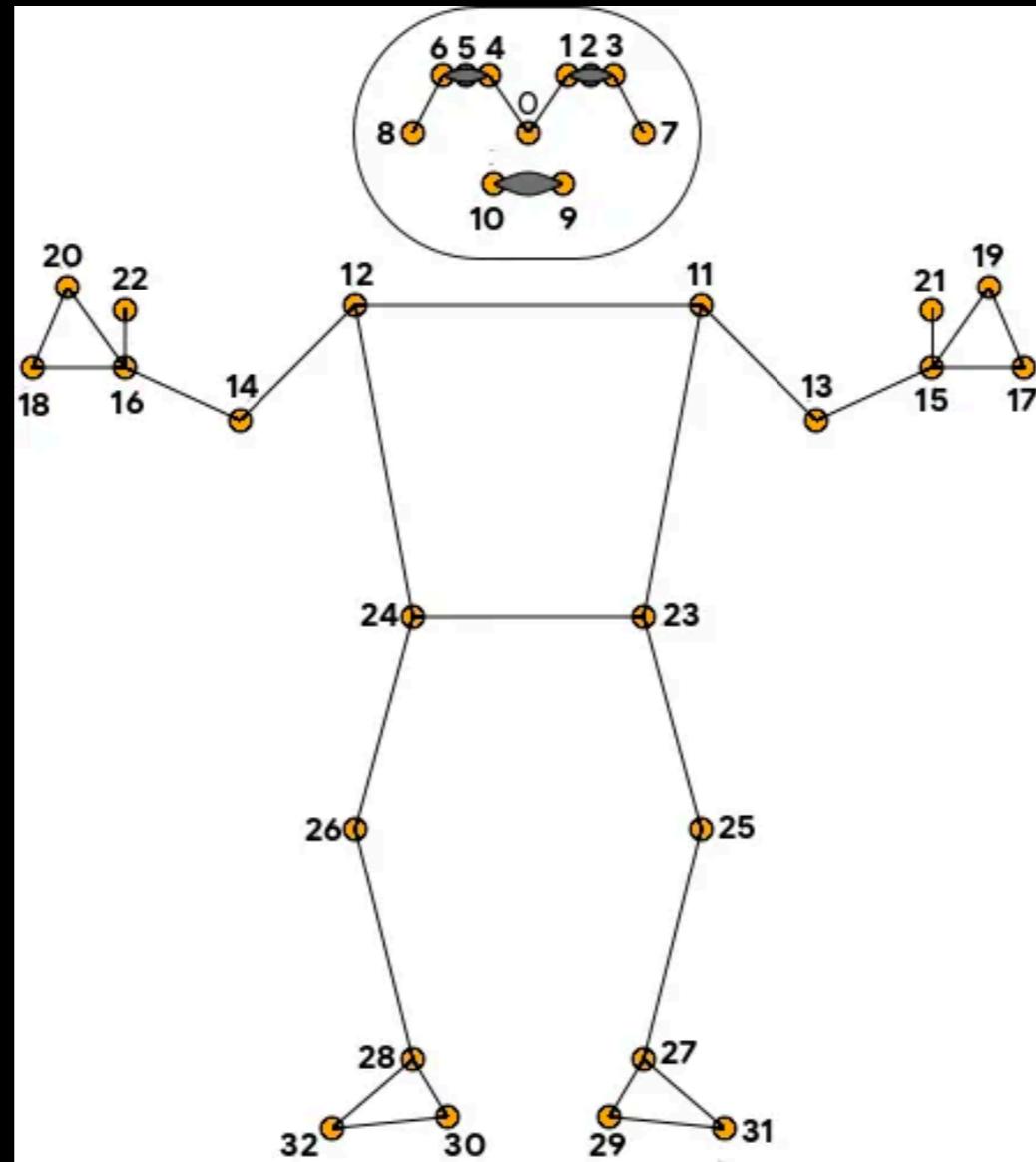
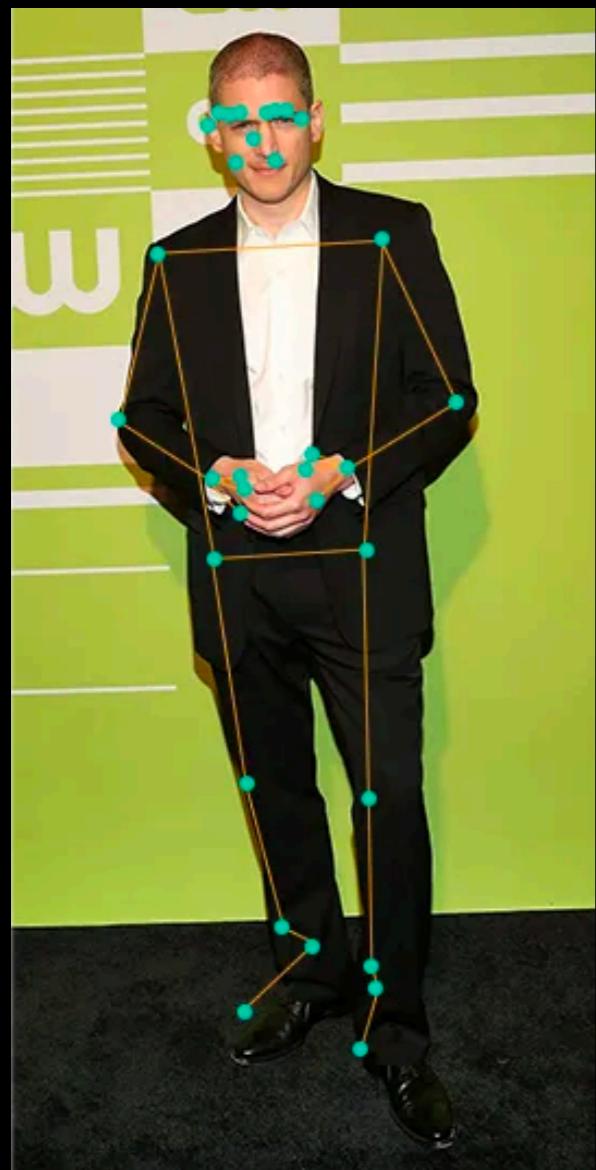
**<= less than or equal**



ML5 | a library/platform that provides access to machine learning algorithms and models in the browser

<https://ml5js.org/> Examples ML5 p5js-Editor

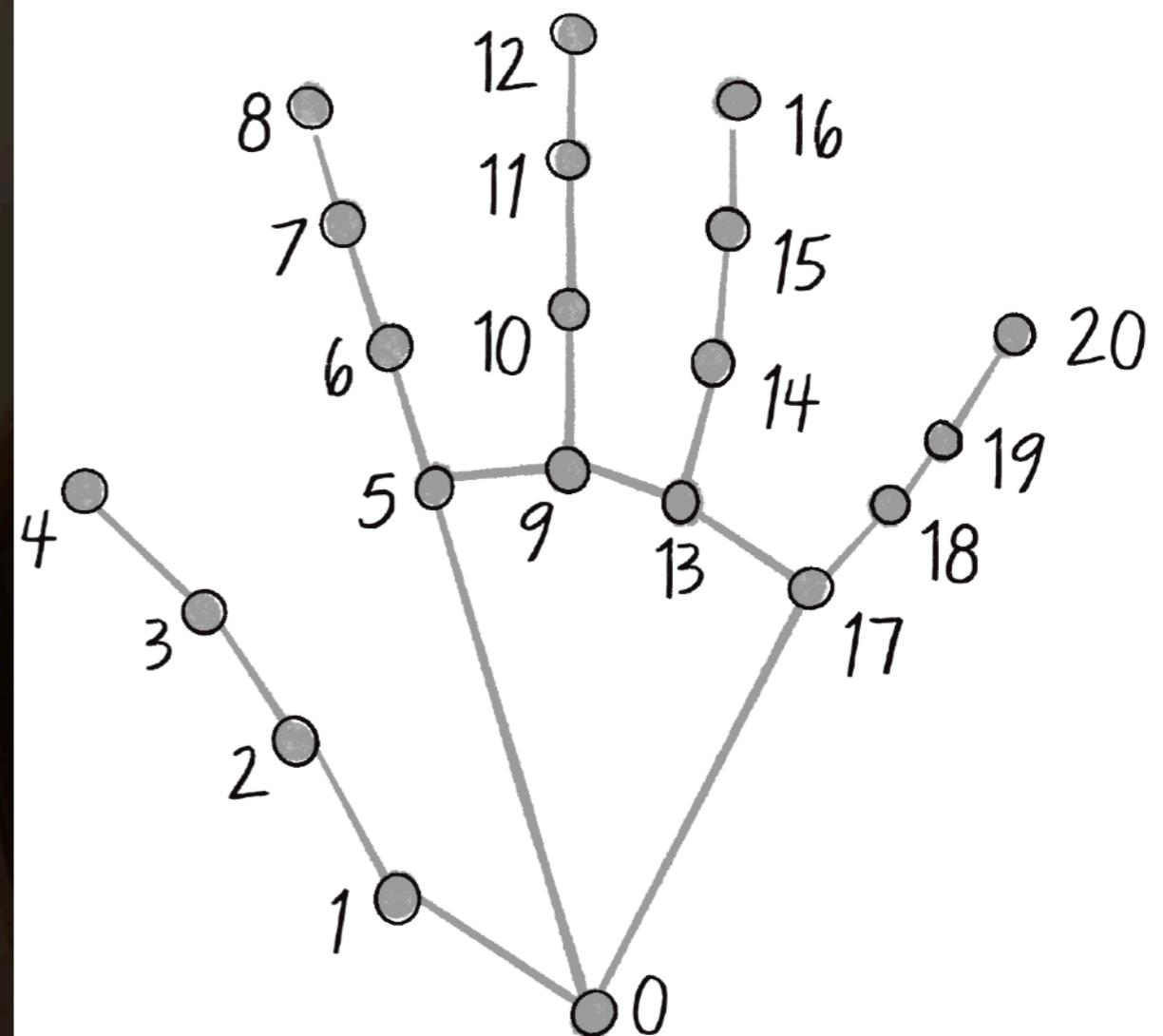
BLAZEPOSE : Machine-learning model for Real-time Human Pose Estimation.



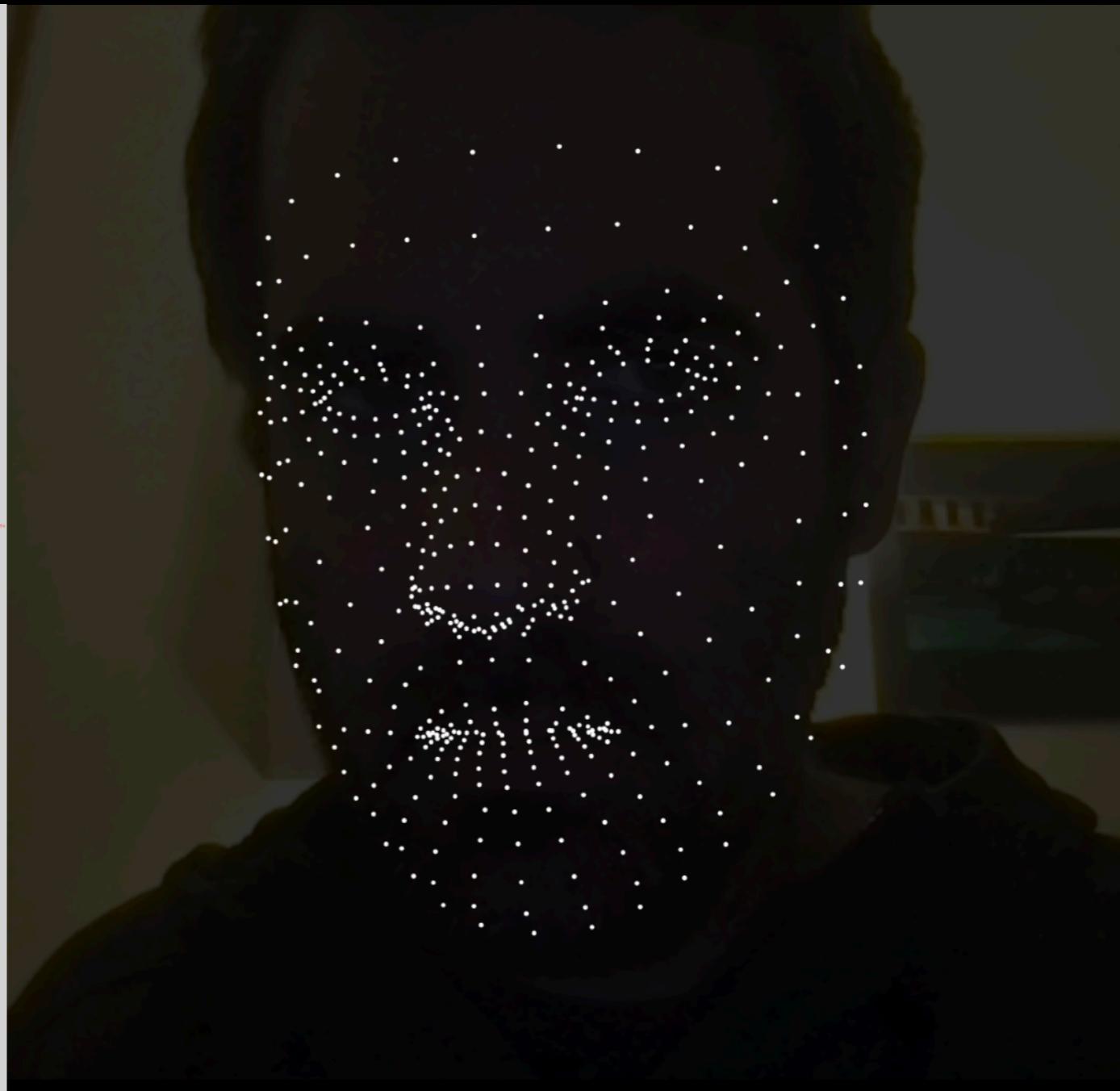
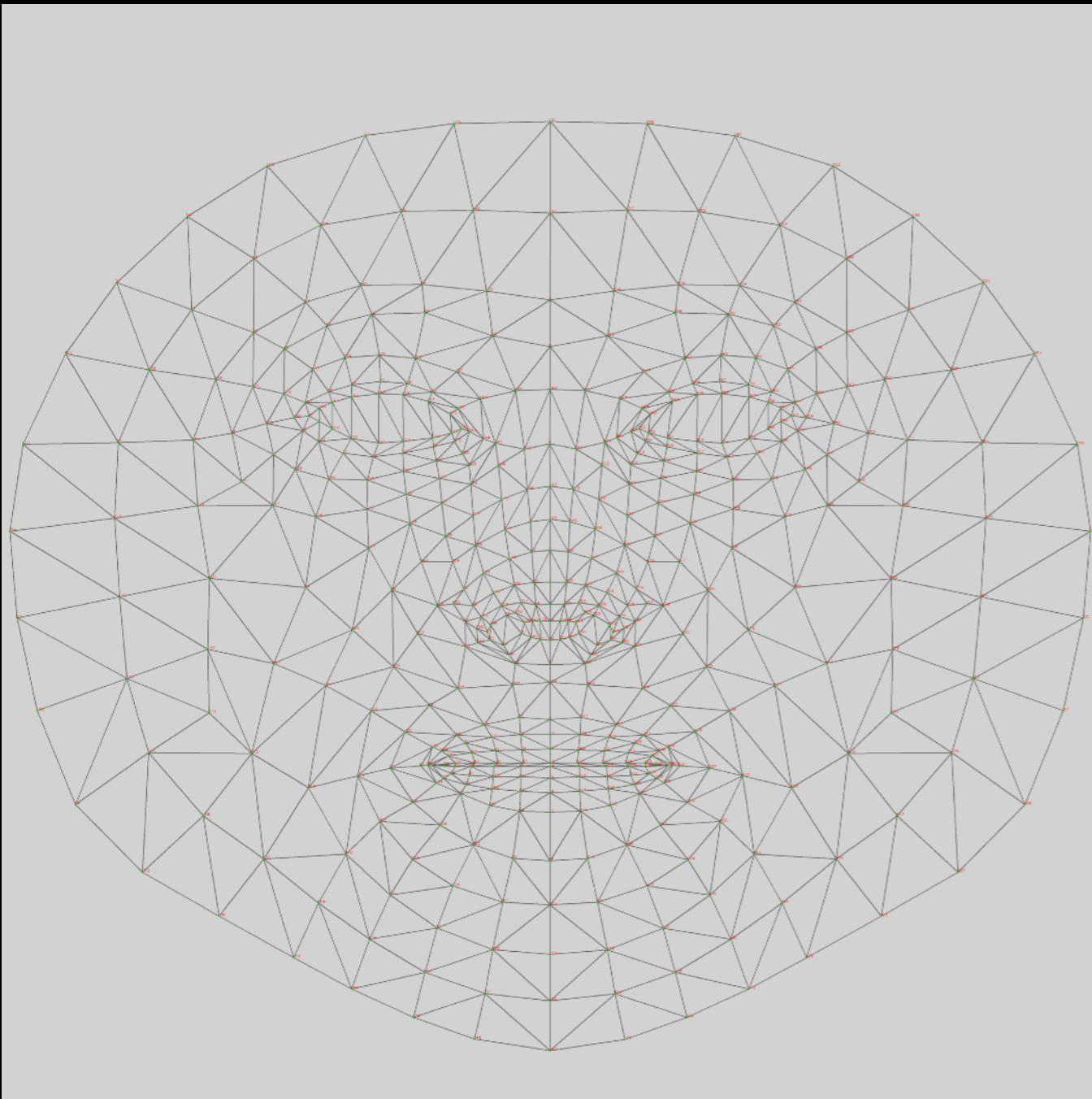
### Landmarks

- |                    |                      |
|--------------------|----------------------|
| 0. Nose            | 17. Left_pinky       |
| 1. Left_eye_inner  | 18. Right_pinky      |
| 2. Left_eye        | 19. Left_index       |
| 3. Left_eye_outer  | 20. Right_index      |
| 4. Right_eye_inner | 21. Left_thumb       |
| 5. Right_eye       | 22. Right_thumb      |
| 6. Right_eye_outer | 23. Left_hip         |
| 7. Left_ear        | 24. Right_hip        |
| 8. Right_ear       | 25. Left_knee        |
| 9. Left_mouth      | 26. Right_knee       |
| 10. Right_mouth    | 27. Left_ankle       |
| 11. Left_shoulder  | 28. Right_ankle      |
| 12. Right_shoulder | 29. Left_heel        |
| 13. Left_elbow     | 30. Right_heel       |
| 14. Right_elbow    | 31. Left_foot_index  |
| 15. Left_wrist     | 32. Right_foot_index |
| 16. Right_wrist    |                      |

# HANDPOSE : Machine-learning model for hand & finger tracking



# FACEMESH : Machine-learning model for facial landmark detection



## HANDPOSE : starting from ml5 handpose keypoints example

p5\* File ▾ Edit ▾ Sketch ▾ Help ▾ English ▾ Hello, visiophone! ▾

Auto-refresh 01\_Handpose by visiophone

sketch.js • Saved: 1 minute ago Preview

```
14
15  function setup() {
16    createCanvas(640, 480);
17    // Create the webcam video and hide it
18    video = createCapture(VIDEO);
19    video.size(width, height);
20    video.hide();
21
22    // start detecting hands from the webcam video
23    handpose.detectStart(video, gotHands);
24
25  }
26
27  function draw() {
28
29    ////////////// Flipped webcam
30    translate(width, 0); // move canvas to the right
31    scale(-1.0, 1.0);
32    image(video, 0, 0, width, height); // Draw the webcam video
33    ///////////////////
34
35    // Draw all the tracked hand points
36    for (let i = 0; i < hands.length; i++) {
37      let hand = hands[i];
38      for (let j = 0; j < hand.keypoints.length; j++) {
39        let keypoint = hand.keypoints[j];
40        fill(0, 255, 0);
41        noStroke();
42        circle(keypoint.x, keypoint.y, 10);
43      }
44    }
45
46    // Callback function for when handpose outputs data
47    function gotHands(results) {
48      // save the output to the hands variable
49      hands = results;
50    }
51  }
```



Mirroring the image

Loops through the hands' keypoints and draws a circle on each point

## HANDPOSE : lines connecting keypoints

p5\*

File ▾ Edit ▾ Sketch ▾ Help ▾

English ▾ Hello, visiophone! ▾

Auto-refresh 01\_Handpose\_LinesConnecting\_Keypoints by visiophone

sketch.js •

Saved: 2 minutes ago

Preview

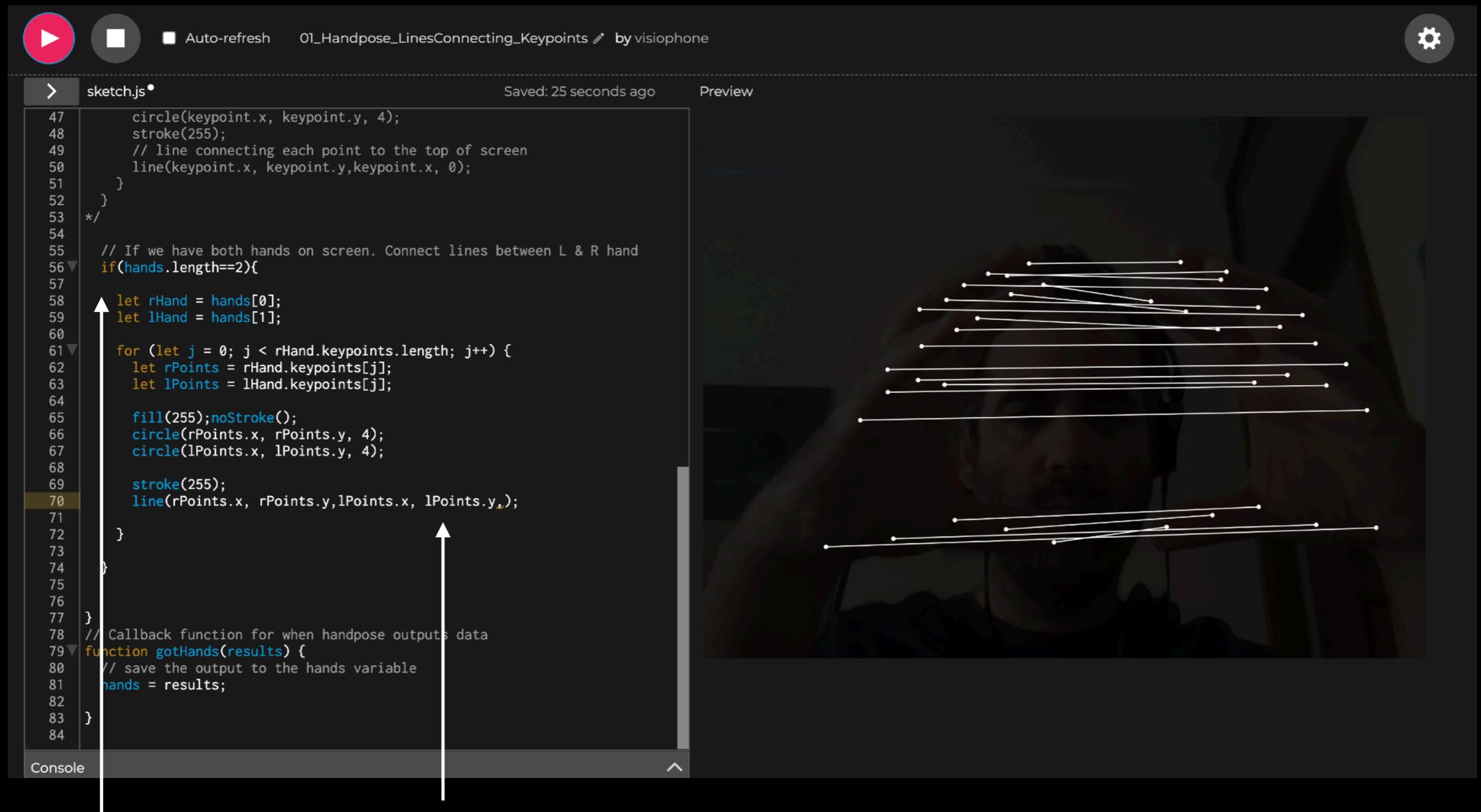
```
19 | video.size(width, height);
20 | video.hide();
21 |
22 | // start detecting hands from the webcam video
23 | handpose.detectStart(video, gotHands);
24 |
25 | }
26 |
27 |> function draw() {
28 |
29 |     ////////////// Flipped webcam
30 |     translate(width, 0); // move canvas to the right
31 |     scale(-1.0, 1.0);
32 |     image(video, 0, 0, width, height); // Draw the webcam video
33 |     // image(video, width-width/4, 0, width/4, height/4); //video feed in the
34 |     // screen corner
35 |     fill(0,200);noStroke(); // darkening Keypoints
36 |     rect(0,0,width,height);
37 |     /////////////////////
38 |
39 |     // Draw all the tracked hand points
40 |     for (let i = 0; i < hands.length; i++) {
41 |         let hand = hands[i];
42 |         for (let j = 0; j < hand.keypoints.length; j++) {
43 |             let keypoint = hand.keypoints[j];
44 |             fill(255);
45 |             noStroke();
46 |             circle(keypoint.x, keypoint.y, 4);
47 |             stroke(255);
48 |             // line connecting each point to the top of screen
49 |             line(keypoint.x, keypoint.y, keypoint.x, 0);
50 |         }
51 |     }
52 |
53 |/* 
54 | // If we have both hands on screen. Connect lines between L & R hand
55 | if(hands.length==2){
56 | */

```

Console

Lines connecting each point to the top of screen

## HANDPOSE : lines connecting keypoints



```
sketch.js • Saved: 25 seconds ago Preview
47   circle(keypoint.x, keypoint.y, 4);
48   stroke(255);
49   // line connecting each point to the top of screen
50   line(keypoint.x, keypoint.y, keypoint.x, 0);
51 }
52 */
53
54
55 // If we have both hands on screen. Connect lines between L & R hand
56 if(hands.length==2){
57   let rHand = hands[0];
58   let lHand = hands[1];
59
60   for (let j = 0; j < rHand.keypoints.length; j++) {
61     let rPoints = rHand.keypoints[j];
62     let lPoints = lHand.keypoints[j];
63
64     fill(255);noStroke();
65     circle(rPoints.x, rPoints.y, 4);
66     circle(lPoints.x, lPoints.y, 4);
67
68     stroke(255);
69     line(rPoints.x, rPoints.y,lPoints.x, lPoints.y);
70   }
71 }
72
73
74
75
76
77
78 // Callback function for when handpose outputs data
79 function gotHands(results) {
80   // save the output to the hands variable
81   hands = results;
82
83 }
```

Check if we have two hands on screen

Lines connecting one hand to the other

HANDPOSE : hands data

21 Keypoints (x,y)      1 Hand detected  
21 Keypoints3D (x,y,z)

Independent Hand parts  
2D & 3D points

```
> sketch.js • Saved: 7 minutes ago
55 // Callback function for when handpose outputs data
56 function gotHands(results) {
57   // save the output to the hands variable
58   hands = results;
59   console.log(hands);
60 }
61
```

Console Clear ▾

- ▶ (1) *Object*
- ▼ (1) [Object]
- ▼ 0: Object
  - keypoints: Array(21)
  - keypoints3D: Array(21)
    - score: 0.961761474609375
    - handedness: "Right"
  - ▼ wrist: Object
    - x: 310.5135917663574
    - y: 532.8429794311523
    - x3D: 0.017497681081295013
    - y3D: 0.0825820192694664
    - z3D: 0.056304931640625
  - thumb\_cmc: Object
  - thumb\_mcp: Object
  - thumb\_ip: Object
  - thumb\_tip: Object
  - index\_finger\_mcp: Object
  - index\_finger\_pip: Object
  - index\_finger\_dip: Object
  - index\_finger\_tip: Object
  - middle\_finger\_mcp: Object
  - middle\_finger\_pip: Object
  - middle\_finger\_dip: Object
  - middle\_finger\_tip: Object
  - ring\_finger\_mcp: Object
  - ring\_finger\_pip: Object
  - ring\_finger\_dip: Object
  - ring\_finger\_tip: Object
  - pinky\_finger\_mcp: Object
  - pinky\_finger\_pip: Object
  - pinky\_finger\_dip: Object
  - pinky\_finger\_tip: Object

## HANDPOSE : fingers ID

5 fingers tips + wrist

The screenshot shows a Processing.js environment with the following details:

- Title:** sketch.js
- File:** 02\_Handpose\_ by visophone
- Status:** Auto-refresh, Saved: 15 seconds ago
- Code:** The code uses the Handpose library to detect hands. It defines variables for the wrist and finger tips (h0 to h5) and then checks distances to determine if fingers are up. For example, it checks if the distance between the index tip and wrist is greater than 0.17 to consider it 'UP'. The code also includes text output for the distance and finger labels.
- Preview:** A dark background image of a hand with colored dots at the tips of the fingers and the thumb. The fingers are labeled: INDEX (red), MIDDLE (green), RING (blue), PINKY (yellow), and THUMB (cyan).

Checking if finger is UP or DOWN: distance between tip and wrist. If the distance is bigger that 0.17 -> FINGER UP

## HANDPOSE : ml5 handpose parts example

Calling specific finger parts ( tip of index & thumb)

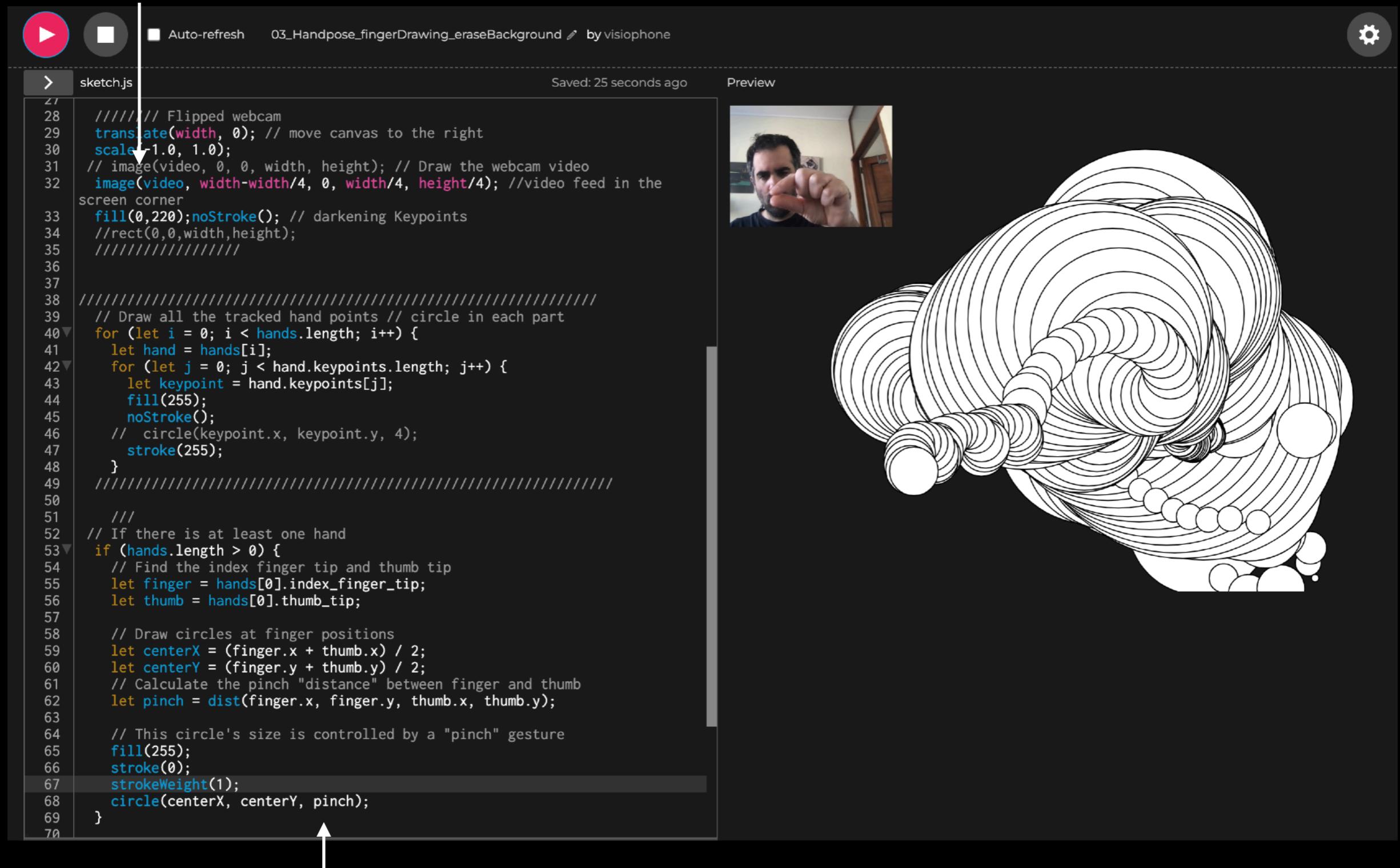
```
sketch.js
8 let hands = [];
9
10 // A variable to track a pinch between thumb and index
11 let pinch = 0;
12
13 function preload() {
14   // Load the handpose model.
15   handpose = ml5.handpose();
16 }
17
18 function setup() {
19   createCanvas(640, 480);
20   // Create the webcam video and hide it
21   video = createCapture('VIDEO');
22   video.size(width, height);
23   video.hide();
24   // start detecting hands from the webcam video
25   handpose.detectStart(video, gotHands);
26 }
27
28 function draw() {
29   // Draw the webcam video
30   image(video, 0, 0, width, height);
31
32   // If there is at least one hand
33   if (hands.length > 0) {
34     // Find the index finger tip and thumb tip
35     let finger = hands[0].index_finger_tip;
36     let thumb = hands[0].thumb_tip;
37
38     // Draw circles at finger positions
39     let centerX = (finger.x + thumb.x) / 2;
40     let centerY = (finger.y + thumb.y) / 2;
41     // Calculate the pinch "distance" between finger and thumb
42     let pinch = dist(finger.x, finger.y, thumb.x, thumb.y);
43
44     // This circle's size is controlled by a "pinch" gesture
45     fill(0, 255, 0, 200);
46     stroke(0);
47     strokeWeight(2);
48     circle(centerX, centerY, pinch);
49   }
50 }
51
52 // Callback function for when handpose outputs data
53 function gotHands(results) {
54   // save the output to the hands variable
55   hands = results;
56   // console.log(hands);
57 }
```

Distance between booth fingers

Finding the middle point between booth fingers

## HANDPOSE : Drawing with the fingers (erase\_background)

Webcam video only in the corner of the screen. No background color



sketch.js

```
27 ////////////// Flipped webcam
28 translate(width, 0); // move canvas to the right
29 scale(-1.0, 1.0);
30 // image(video, 0, 0, width, height); // Draw the webcam video
31 image(video, width-width/4, 0, width/4, height/4); //video feed in the
screen corner
32 fill(0,220);noStroke(); // darkening Keypoints
33 //rect(0,0,width,height);
34 ///////////////////
35
36
37
38 /////////////////////////////////
39 // Draw all the tracked hand points // circle in each part
40 for (let i = 0; i < hands.length; i++) {
41   let hand = hands[i];
42   for (let j = 0; j < hand.keypoints.length; j++) {
43     let keypoint = hand.keypoints[j];
44     fill(255);
45     noStroke();
46     // circle(keypoint.x, keypoint.y, 4);
47     stroke(255);
48   }
49 /////////////////////
50
51 /**
52 // If there is at least one hand
53 if (hands.length > 0) {
54   // Find the index finger tip and thumb tip
55   let finger = hands[0].index_finger_tip;
56   let thumb = hands[0].thumb_tip;
57
58   // Draw circles at finger positions
59   let centerX = (finger.x + thumb.x) / 2;
60   let centerY = (finger.y + thumb.y) / 2;
61   // Calculate the pinch "distance" between finger and thumb
62   let pinch = dist(finger.x, finger.y, thumb.x, thumb.y);
63
64   // This circle's size is controlled by a "pinch" gesture
65   fill(255);
66   stroke(0);
67   strokeWeight(1);
68   circle(centerX, centerY, pinch);
69 }
70
```

Drawing circles

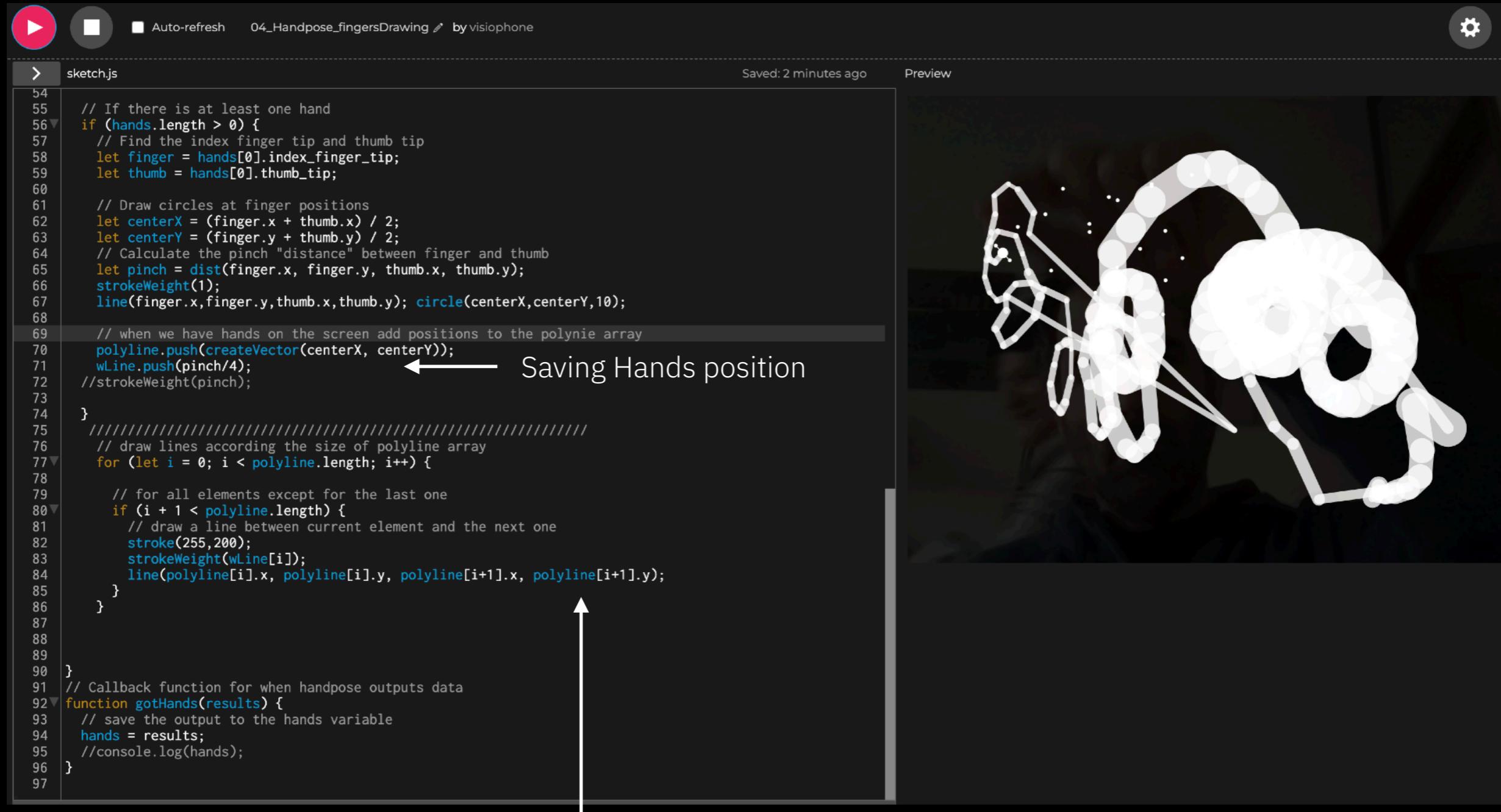
## HANDPOSE : Drawing with the fingers (saving points on array)

```
12 let polyline = []; // array to store the drawing points
13 let wLine = []; // array to store line's weight
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54 // If there is at least one hand
55 if (hands.length > 0) {
56   // Find the index finger tip and thumb tip
57   let finger = hands[0].index_finger_tip;
58   let thumb = hands[0].thumb_tip;
59
60   // Draw circles at finger positions
61   let centerX = (finger.x + thumb.x) / 2;
62   let centerY = (finger.y + thumb.y) / 2;
63   // Calculate the pinch "distance" between finger and thumb
64   let pinch = dist(finger.x, finger.y, thumb.x, thumb.y);
65   strokeWeight(1);
66   line(finger.x, finger.y, thumb.x, thumb.y); circle(centerX, centerY, 10);
67
68   // when we have hands on the screen add positions to the polyline array
69   polyline.push(createVector(centerX, centerY));
70   wLine.push(pinch/4);
71   //strokeWeight(pinch);
72 }
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
```

← Adding on the top

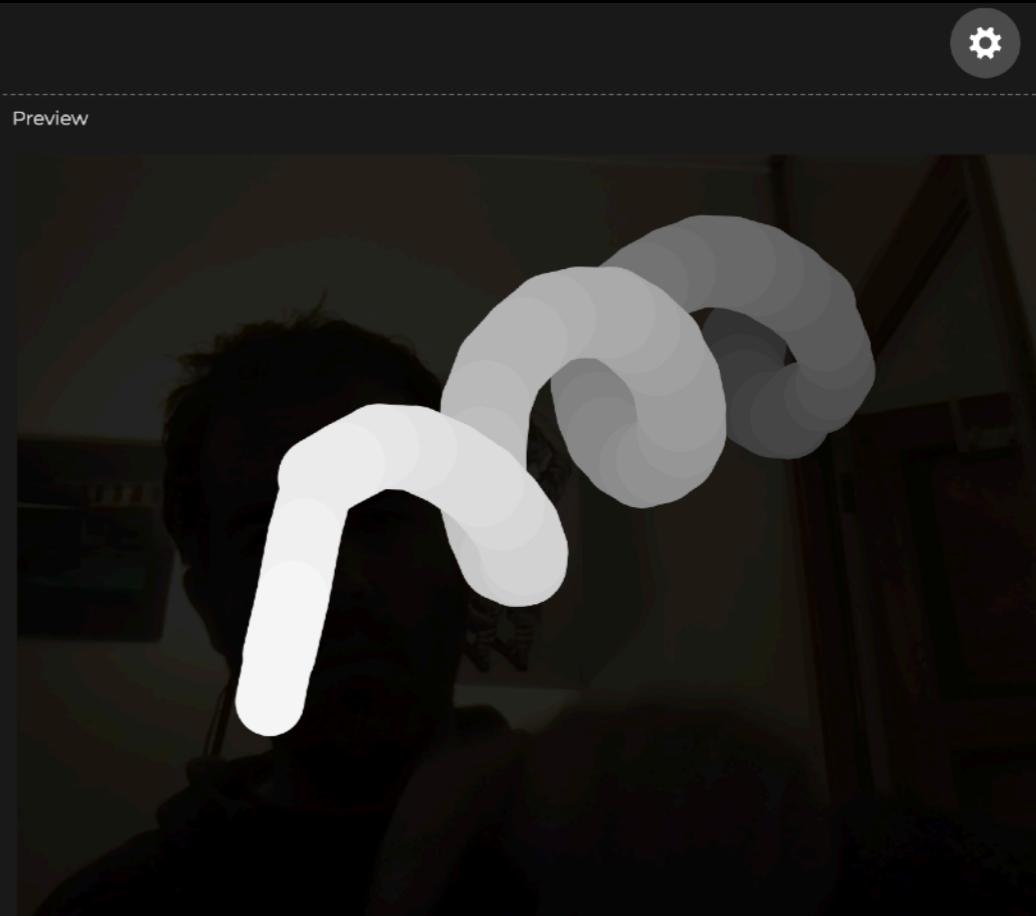
← Saving Hands position

↑ Drawing the line



## HANDPOSE : Drawing with the fingers (saving points on array)

Smoothing the position values, so the drawing is more fluid



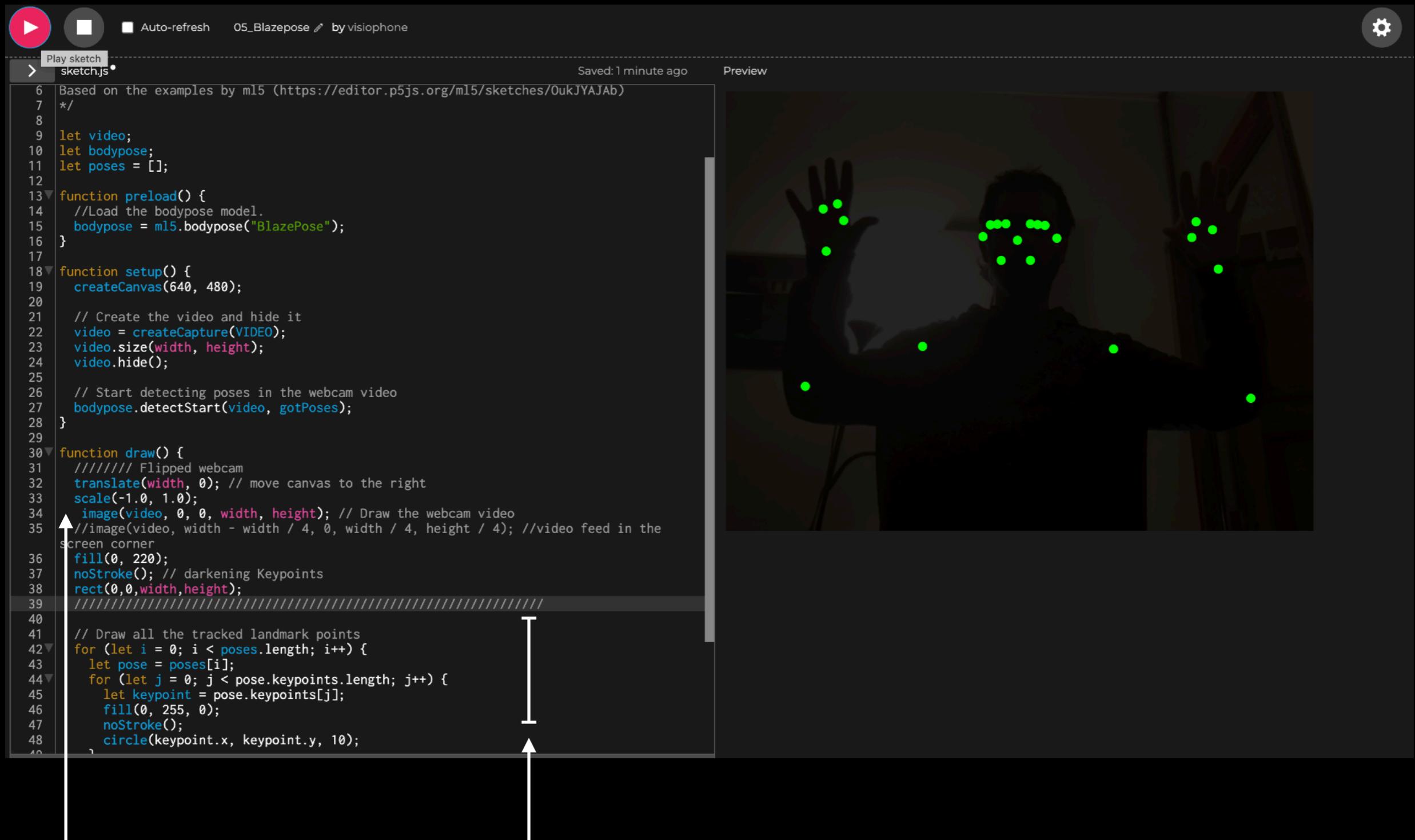
The screenshot shows a Processing.js sketch titled "04\_Handpose\_fingersDrawing" by visiophone. The code in the sketch.js file is as follows:

```
sketch.js
59 // If there is at least one hand
60 if (hands.length > 0) {
61     // Find the index finger tip and thumb tip
62     let finger = hands[0].index_finger_tip;
63     let thumb = hands[0].thumb_tip;
64
65     // Draw circles at finger positions
66     let centerX = (finger.x + thumb.x) / 2;
67     let centerY = (finger.y + thumb.y) / 2;
68     // Calculate the pinch "distance" between finger and thumb
69     let pinch = dist(finger.x, finger.y, thumb.x, thumb.y);
70
71     // smoothing values
72     posX+=(centerX-posX)*0.2;
73     posY+=(centerY-posY)*0.2;
74     wLinee+=(pinch/2-wLinee)*0.2;
75
76     strokeWeight(1); // line connecting the fingers
77     line(finger.x,finger.y,posX,posY); line(thumb.x,thumb.y,posX,posY);
78
79     // when we have hands on the screen add positions to the polyline array
80     polyline.push(createVector(posX, posY));
81     wLine.push(wLinee);
82
83 }
84
85 // draw lines according the size of polyline array
86 for (let i = 0; i < polyline.length; i++) {
87
88     // for all elements except for the last one
89     if (i + 1 < polyline.length) {
90         // draw a line between current element and the next one
91         stroke(map(i,0,polyline.length,50,255));
92         strokeWeight(wLine[i]);
93         line(polyline[i].x, polyline[i].y, polyline[i+1].x, polyline[i+1].y);
94     }
95
96 }
97
98 // Settingn a limite to the size of the line
99 if(polyline.length>60) {polyline.shift();wLine.shift();}
100
101
102
103
```

A vertical double-headed arrow points from the line "if(polyline.length>60) {polyline.shift();wLine.shift();}" in the code up to the preview window, indicating its function. The preview window shows a person's hand skeleton with a thick white line connecting the fingers and a series of smaller grey circles following the line's path.

Line Size limite. When it reaches the limit the first point is erased

## BLAZEPOSE : starting from ml5 Blazepose keypoints example



The screenshot shows the p5.js code editor with a sketch titled "05\_Blazepose" by visiophone. The code is based on examples from ml5. The preview window shows a person's face and hands with green circular keypoints overlaid, indicating detected landmarks. The code uses the ml5.bodypose library to detect poses in a video feed.

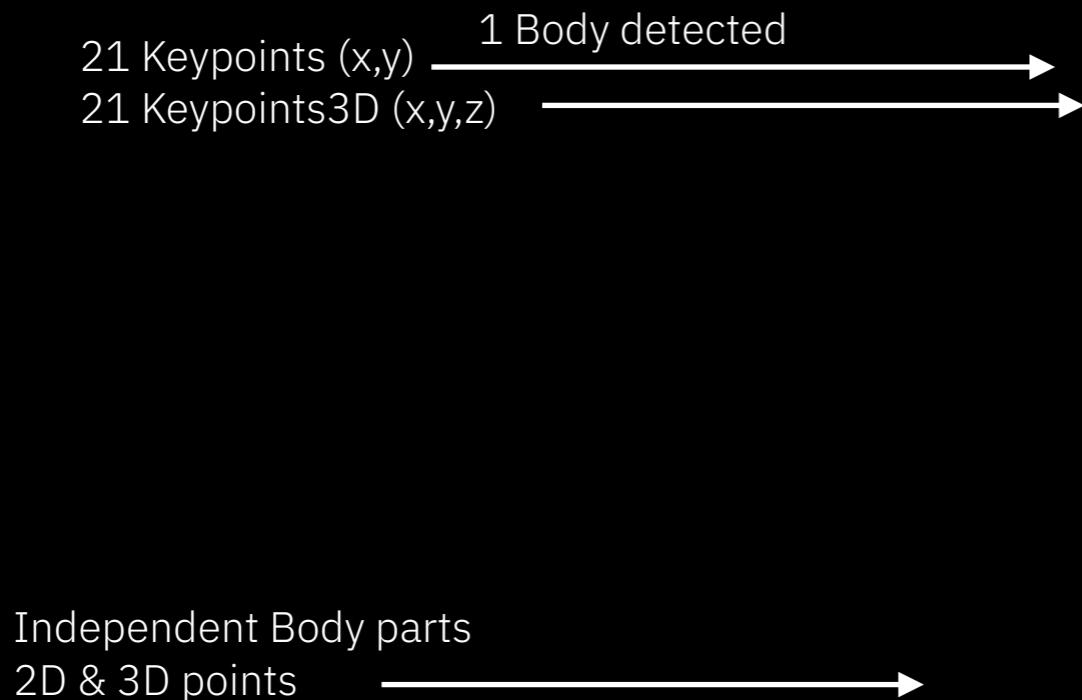
```
Play sketch sketch.js
Based on the examples by ml5 (https://editor.p5js.org/ml5/sketches/0ukJYAJAb)
Auto-refresh 05_Blazepose by visiophone
Saved: 1 minute ago Preview

6 Based on the examples by ml5 (https://editor.p5js.org/ml5/sketches/0ukJYAJAb)
7 */
8
9 let video;
10 let bodypose;
11 let poses = [];
12
13 function preload() {
14   //Load the bodypose model.
15   bodypose = ml5.bodypose("BlazePose");
16 }
17
18 function setup() {
19   createCanvas(640, 480);
20
21   // Create the video and hide it
22   video = createCapture(VIDEO);
23   video.size(width, height);
24   video.hide();
25
26   // Start detecting poses in the webcam video
27   bodypose.detectStart(video, gotPoses);
28 }
29
30 function draw() {
31   ////////////// Flipped webcam
32   translate(width, 0); // move canvas to the right
33   scale(-1.0, 1.0);
34   image(video, 0, 0, width, height); // Draw the webcam video
35   //image(video, width - width / 4, 0, width / 4, height / 4); //video feed in the
36   //screen corner
37   fill(0, 220);
38   noStroke(); // darkening Keypoints
39   rect(0,0,width,height);
40
41   // Draw all the tracked landmark points
42   for (let i = 0; i < poses.length; i++) {
43     let pose = poses[i];
44     for (let j = 0; j < pose.keypoints.length; j++) {
45       let keypoint = pose.keypoints[j];
46       fill(0, 255, 0);
47       noStroke();
48       circle(keypoint.x, keypoint.y, 10);
49     }
50   }
51 }
```

Mirroring & darkening the image

Loops through the body' keypoints and draws a circle on each point

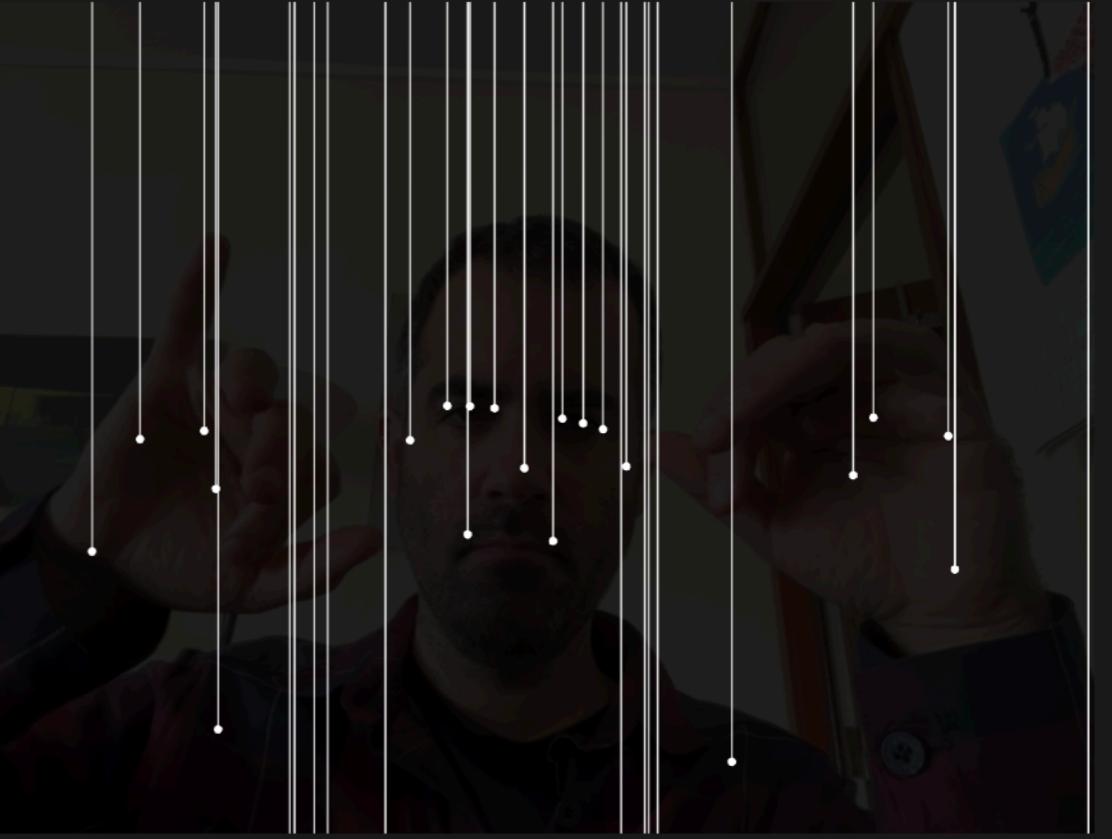
## BLAZEPOSE : body data



Console

```
    ▼ 0: Object
      ► keypoints: Array(33)
      ► keypoints3D: Array(33)
      ► nose: Object
      ► left_eye_inner: Object
      ► left_eye: Object
      ► left_eye_outer: Object
      ► right_eye_inner: Object
      ► right_eye: Object
      ► right_eye_outer: Object
      ► left_ear: Object
      ► right_ear: Object
      ► mouth_left: Object
      ► mouth_right: Object
      ► left_shoulder: Object
      ► right_shoulder: Object
      ► left_elbow: Object
      ► right_elbow: Object
      ► left_wrist: Object
      ► right_wrist: Object
      ► left_pinky: Object
      ► right_pinky: Object
      ► left_index: Object
      ► right_index: Object
      ► left_thumb: Object
      ► right_thumb: Object
      ► left_hip: Object
      ► right_hip: Object
      ► left_knee: Object
      ► right_knee: Object
      ► left_ankle: Object
      ► right_ankle: Object
      ► left_heel: Object
      ► right_heel: Object
      ► left_foot_index: Object
      ► right_foot_index: Object
```

## BLAZEPOSE : lines connecting keypoints



The image shows a Processing.js sketch window. At the top left are play and stop buttons. To the right of the buttons are checkboxes for 'Auto-refresh' and the sketch title '05\_BlaePose\_keypoints\_Lines' by visiophone. Below the title, it says 'Saved: 3 minutes ago'. On the right side of the window is a preview area showing a person's face with numerous small black dots representing keypoints and thin white lines connecting them to the top edge of the frame. The main area contains the sketch's code:

```
sketch.js
27 bodypose.detectStart(video, poses),
28 }
29
30 function draw() {
31     // Flipped webcam
32     translate(width, 0); // move canvas to the right
33     scale(-1.0, 1.0);
34     image(video, 0, 0, width, height); // Draw the webcam video
35     //image(video, width - width / 4, 0, width / 4, height / 4);
36     //video feed in the screen corner
37     fill(0, 220);
38     noStroke(); // darkening Keypoints
39     rect(0,0,width,height);
40
41
42     // Draw all the tracked landmark points
43     for (let i = 0; i < poses.length; i++) {
44         let pose = poses[i];
45         for (let j = 0; j < pose.keypoints.length; j++) {
46             let keypoint = pose.keypoints[j];
47             fill(255);
48             noStroke();
49             circle(keypoint.x, keypoint.y, 5);
50             stroke(255);
51             line(keypoint.x, keypoint.y, keypoint.x, -100);
52         }
53     }
54 }
```

Lines connecting each point to the top of screen

## BLAZEPOSE : eyes

Calling specific body parts ( right & left eyes)

The screenshot shows a Processing.js environment with the following details:

- Sketch Information:** sketch.js, 06\_Blazepose\_eyes by visiophone, Auto-refresh, Saved: 14 days ago.
- Code Preview:** A dark image of a man's face with two large white circles drawn over his eyes, representing the processed eye positions.
- Code Content:** The sketch.js file contains the following code:

```
48 if (poses.length > 0) {  
49     //smooth values to drag the inner eyes circle  
50     leftX+=(poses[0].left_eye.x-leftX)*0.1;  
51     leftY+=(poses[0].left_eye.y-leftY)*0.1;  
52     rightX+=(poses[0].right_eye.x-rightX)*0.1;  
53     rightY+=(poses[0].right_eye.y-rightY)*0.1;  
54     stroke(255);  
55     //line connecting realtime eyes position to the smoother position  
56     line(poses[0].left_eye.x,poses[0].left_eye.y,leftX,leftY);  
57     line(poses[0].right_eye.x,poses[0].right_eye.y,rightX,rightY);  
58     fill(255);  
59     circle(poses[0].left_eye.x,poses[0].left_eye.y,60); // eye's  
60     circle(poses[0].right_eye.x,poses[0].right_eye.y,60);  
61     fill(0);  
62     circle(leftX,leftY,40); // inner eyes circle  
63     circle(rightX,rightY,40);  
64 }  
65 }  
66 }  
67 }  
68 }  
69 }  
70 }  
71 }  
72 }  
73 }  
74 }  
75 // Callback function for when bodypose outputs data
```

Drawing a circle on the eyes' position

## BLAZEPOSE : mask

Fixed randomSeed for deterministic randomness

Calling specific body parts ( right & left eyes, nose, mouth)



The code demonstrates how to draw specific body parts (eyes and nose) from a BlazePose detection. It uses the `randomSeed` function to ensure deterministic randomness.

```
sketch.js
48 | randomSeed(seed); // fix the random number
49 |
50 |
51 if (poses.length > 0) {
52
53   fill(255); stroke(0); strokeWeight(1);
54   // Eyes External circle. Circle size is random
55   leftEye=int(random(15,150));
56   rightEye=int(random(15,150));
57   circle(poses[0].left_eye.x,poses[0].left_eye.y,leftEye);
58
59   circle(poses[0].right_eye.x,poses[0].right_eye.y,rightEye);
60
61   // Eyes Inner circle
62   fill(0);
63   circle(poses[0].left_eye.x,poses[0].left_eye.y,10);
64   circle(poses[0].right_eye.x,poses[0].right_eye.y,10);
65
66   // mouth
67   stroke(255); strokeCap(SQUARE);
68   mouth=int(random(1,30));
69   strokeWeight(mouth);
70
71   line(poses[0].mouth_left.x,poses[0].mouth_left.y,poses[0]
72 .mouth_right.x,poses[0].mouth_right.y);
73
74   // nose
75   fill(255,0,0);noStroke();
76   nose=int(random(15,70));
77   circle(poses[0].nose.x,poses[0].nose.y,nose);
78
79 }
80
81
82
83
84
85 function mouseReleased() {
86
87   seed=int(random(millis()));
88   print("RandomSeed: "+seed);
89 }
90
```

Drawing circles and lines with random sizes

Mouse click changes randomSeed

## BLAZEPOSE : mask PNG

Places images (png with no background) on specific body parts

The screenshot shows a Processing.js sketch titled "08\_Blazepose\_Mask\_pngs" by visiophone. The sketch.js code is as follows:

```
sketch.js
49 /////////////////
50 randomSeed(seed); // fix the random number
51
52 if (poses.length > 0) {
53   imageMode(CENTER); // position image from the center
54
55   randLeft=int(random(5)); // randomize image
56   randRight=int(random(5));
57   leftEye=int(random(15,150)); // randomize size
58   rightEye=int(random(15,150));
59   // insert the random image in the eye's position
60
61   image(pic[randLeft],poses[0].left_eye.x,poses[0].left_eye.y,leftEye,leftEye);
62
63   image(pic[randRight],poses[0].right_eye.x,poses[0].right_eye.y,rightEye,rightEye);
64
65
66
67 // Callback function for when bodypose outputs data
68 function gotPoses(results) {
```

Console output:

```
RandomSeed: 18130
RandomSeed: 11606
RandomSeed: 11580
```

At the bottom right, there is a block of code for the preload() function:

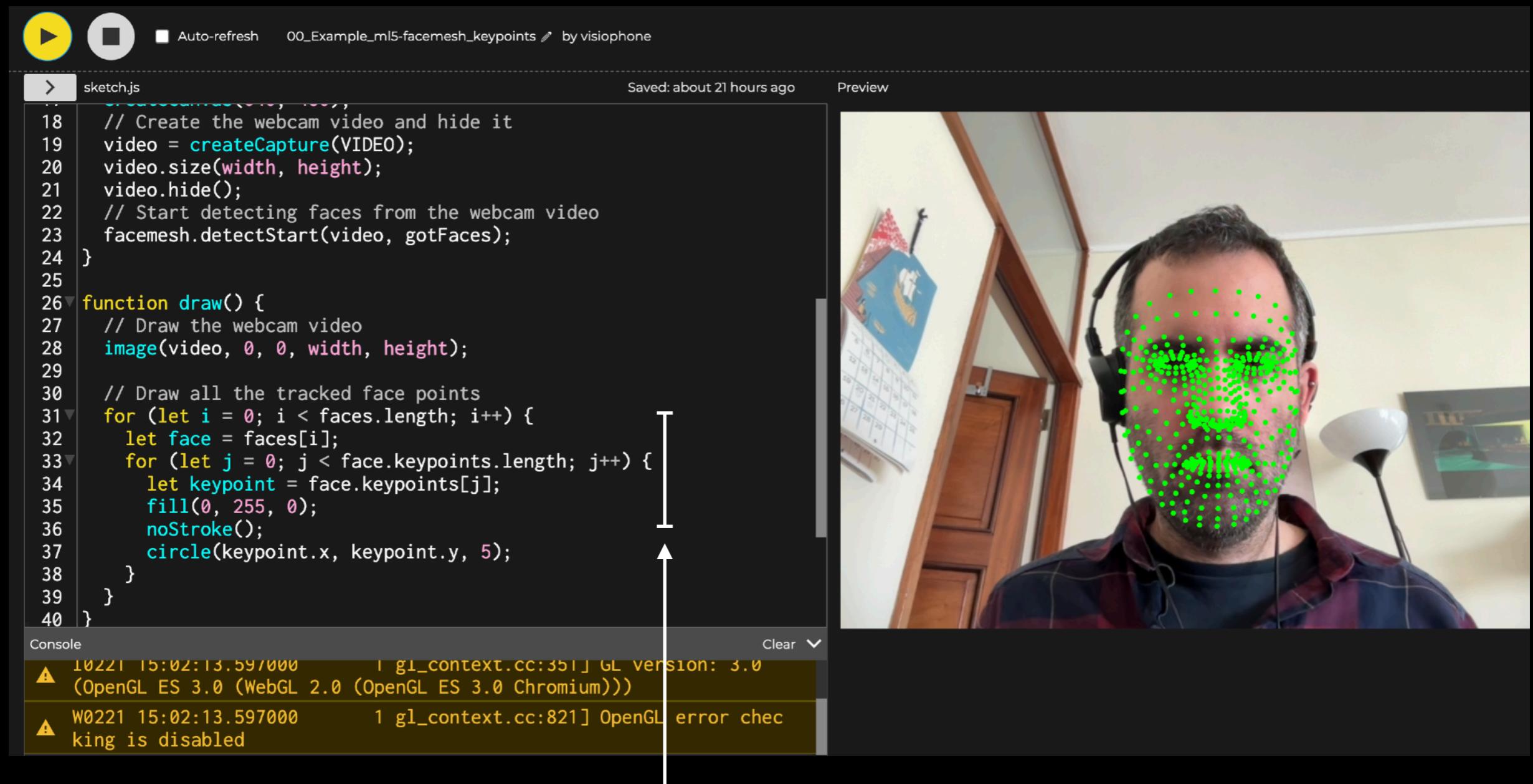
```
function preload() {
  bodypose = ml5.bodypose("BlazePose"); //Load the bodypose model.
  pic[0] = loadImage('pic/eye0.png');
  pic[1] = loadImage('pic/eye1.png');
  pic[2] = loadImage('pic/eye2.png');
  pic[3] = loadImage('pic/eye3.png');
  pic[4] = loadImage('pic/eye4.png');
```

A vertical line with arrows points from the code at line 59 down to the image of the man's face, and another arrow points up from the code at line 21 to the same image.

Image number and size are randomised

Images are pre-loaded into pic[] array

## FACEMESH : ml5\_facemesh keypoints example



The screenshot shows a code editor interface with a dark theme. At the top, there are play and stop buttons, an "Auto-refresh" checkbox, and the file name "00\_Example\_ml5-facemesh\_keypoints" by visiophone. Below the editor area, the status bar shows "Saved: about 21 hours ago". The main code area is titled "sketch.js" and contains the following JavaScript code:

```
18 // Create the webcam video and hide it
19 video = createCapture(VIDEO);
20 video.size(width, height);
21 video.hide();
22 // Start detecting faces from the webcam video
23 facemesh.detectStart(video, gotFaces);
24 }
25
26 function draw() {
27   // Draw the webcam video
28   image(video, 0, 0, width, height);
29
30   // Draw all the tracked face points
31   for (let i = 0; i < faces.length; i++) {
32     let face = faces[i];
33     for (let j = 0; j < face.keypoints.length; j++) {
34       let keypoint = face.keypoints[j];
35       fill(0, 255, 0);
36       noStroke();
37       circle(keypoint.x, keypoint.y, 5);
38     }
39   }
40 }
```

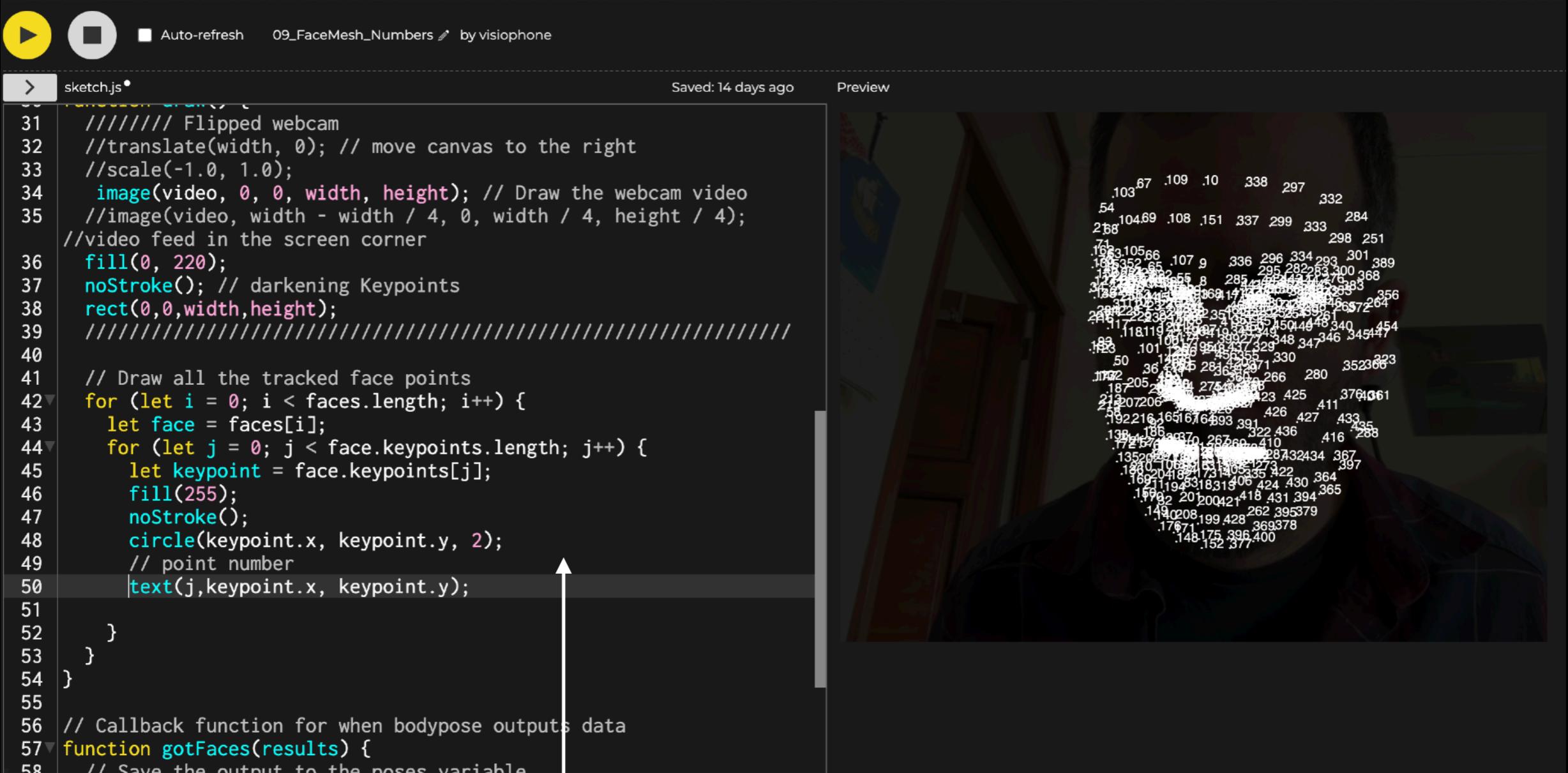
The "Console" tab at the bottom shows the following log output:

```
10221 15:02:13.597000 [gl_context.cc:351] GL version: 3.0
  (OpenGL ES 3.0 (WebGL 2.0 (OpenGL ES 3.0 Chromium)))
W0221 15:02:13.597000 [gl_context.cc:821] OpenGL error check
  king is disabled
```

A vertical arrow points from the explanatory text below to the "for (let j = 0; j < face.keypoints.length; j++) {" line in the code.

Loops through face's keypoints and draws a circle on each point

## FACE MESH : points numbers ID



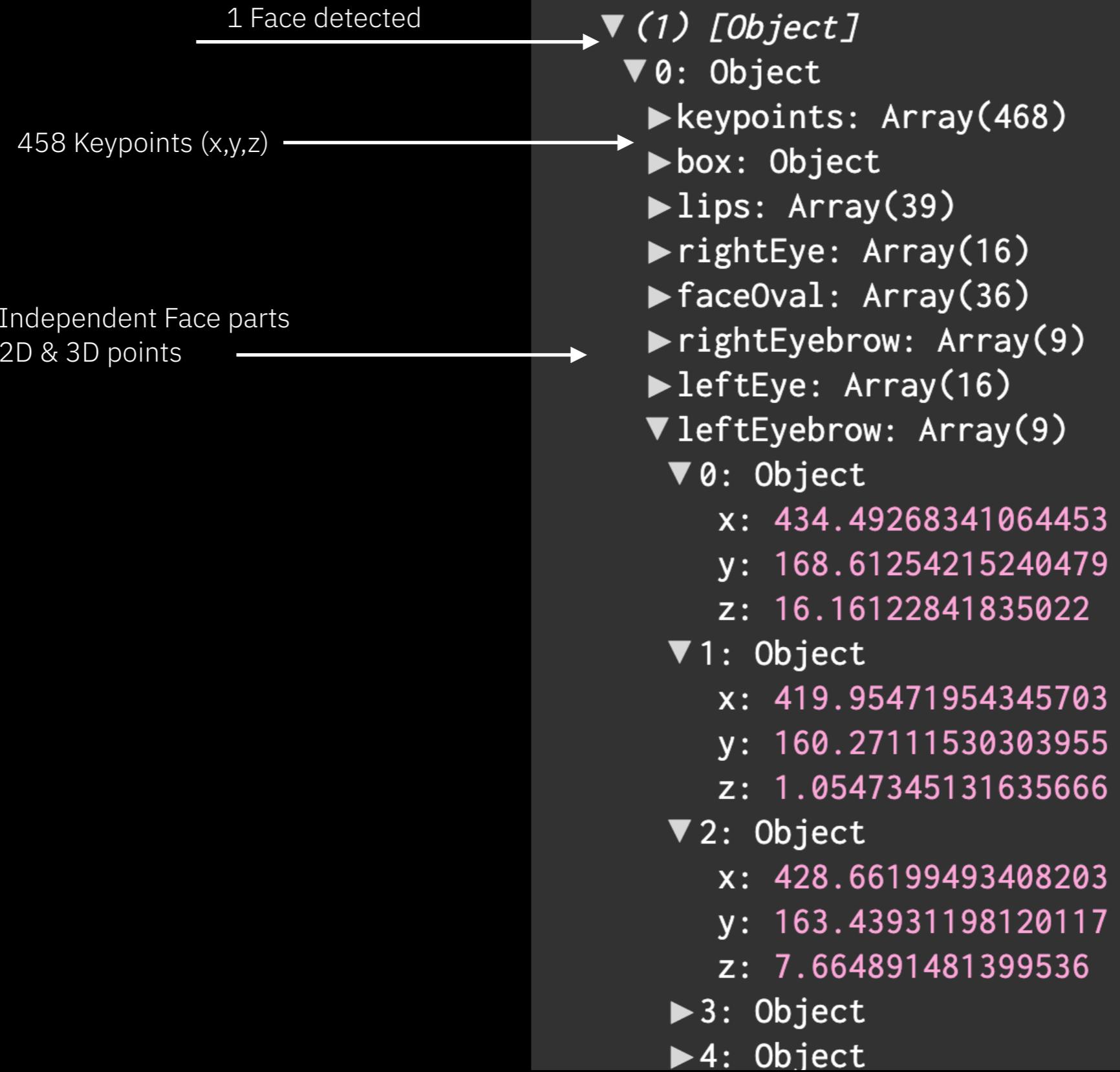
The image shows a Processing.js sketch window. At the top left are play and stop buttons, followed by an 'Auto-refresh' checkbox and the sketch title '09\_FaceMesh\_Numbers' by visiophone. To the right are 'Saved: 14 days ago' and a 'Preview' button. The code editor on the left contains the following script:

```
31 // Flipped webcam
32 //translate(width, 0); // move canvas to the right
33 //scale(-1.0, 1.0);
34 image(video, 0, 0, width, height); // Draw the webcam video
35 //image(video, width - width / 4, 0, width / 4, height / 4);
36 //video feed in the screen corner
37 fill(0, 220);
38 noStroke(); // darkening Keypoints
39 rect(0,0,width,height);
40 // Draw all the tracked face points
41 for (let i = 0; i < faces.length; i++) {
42   let face = faces[i];
43   for (let j = 0; j < face.keypoints.length; j++) {
44     let keypoint = face.keypoints[j];
45     fill(255);
46     noStroke();
47     circle(keypoint.x, keypoint.y, 2);
48     // point number
49     text(j,keypoint.x, keypoint.y);
50   }
51 }
52 }
53 }
54 }
55
56 // Callback function for when bodypose outputs data
57 function gotFaces(results) {
58   // Save the output to the poses variable
```

A vertical white arrow points upwards from the explanatory text below to the line 'text(j,keypoint.x, keypoint.y);' in the code.

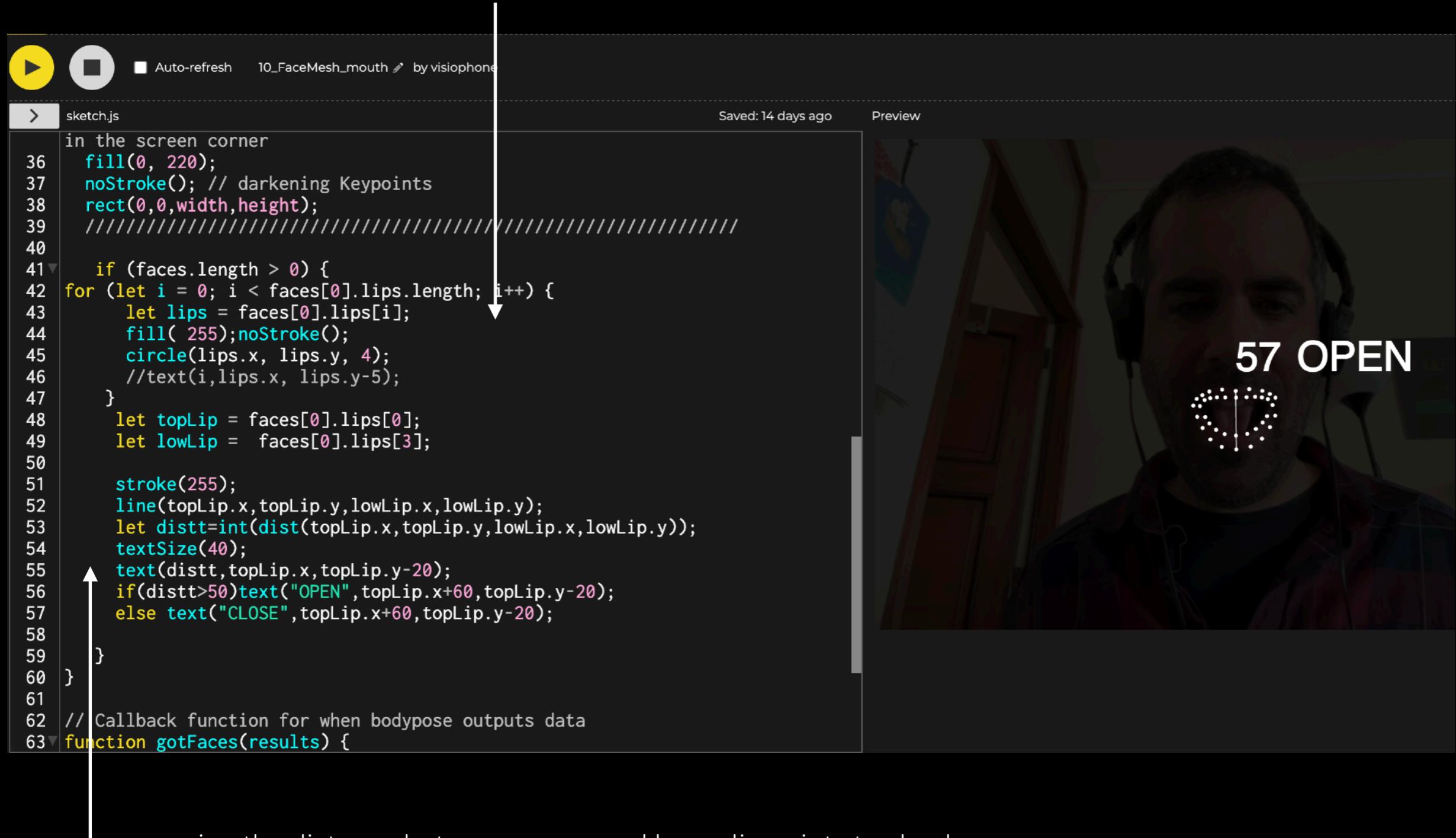
Loops through face's keypoints and writes point number ID

FACE MESH : FaceData



## FACE MESH : mouth points

Calling specific body parts (lips)

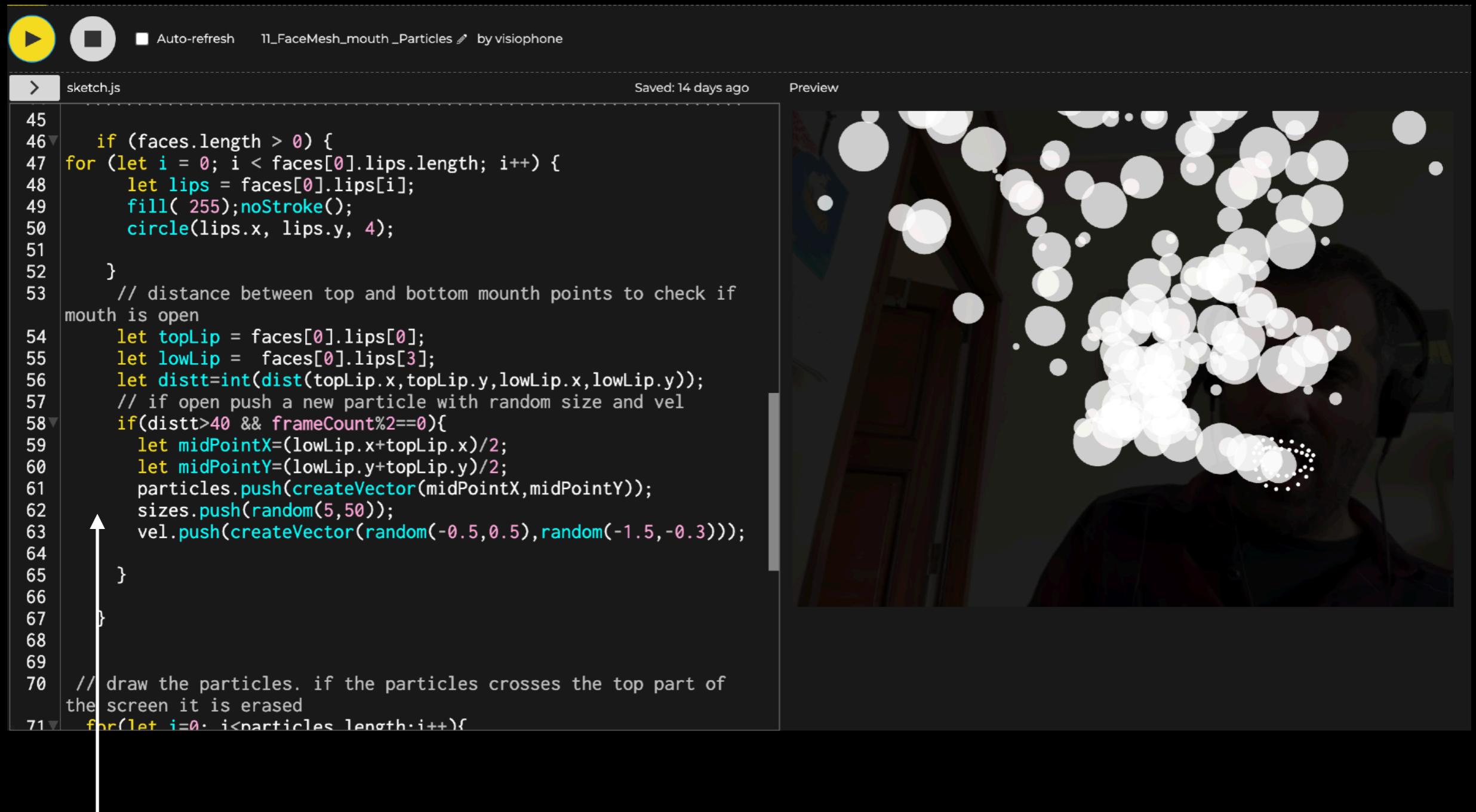


The screenshot shows a Processing.js sketch window. At the top, there are play and stop buttons, an 'Auto-refresh' checkbox, and the title '10\_FaceMesh\_mouth' by visiophone. To the right, it says 'Saved: 14 days ago' and 'Preview'. The preview image shows a man's face with a grid of points and the text '57 OPEN' overlaid. The code in the sketch.js file is as follows:

```
in the screen corner
36 fill(0, 220);
37 noStroke(); // darkening Keypoints
38 rect(0,0,width,height);
39 /////////////////////////////////
40
41 if (faces.length > 0) {
42     for (let i = 0; i < faces[0].lips.length; i++) {
43         let lips = faces[0].lips[i];
44         fill( 255);noStroke();
45         circle(lips.x, lips.y, 4);
46         //text(i,lips.x, lips.y-5);
47     }
48     let topLip = faces[0].lips[0];
49     let lowLip = faces[0].lips[3];
50
51     stroke(255);
52     line(topLip.x,topLip.y,lowLip.x,lowLip.y);
53     let distt=int(dist(topLip.x,topLip.y,lowLip.x,lowLip.y));
54     textSize(40);
55     text(distt,topLip.x,topLip.y-20);
56     if(distt>50)text("OPEN",topLip.x+60,topLip.y-20);
57     else text("CLOSE",topLip.x+60,topLip.y-20);
58
59 }
60 }
61
62 // Callback function for when bodypose outputs data
63 function gotFaces(results) {
```

measuring the distance between upper and lower lip points to check if mouth is open or closed

## FACEMESH : mouth points

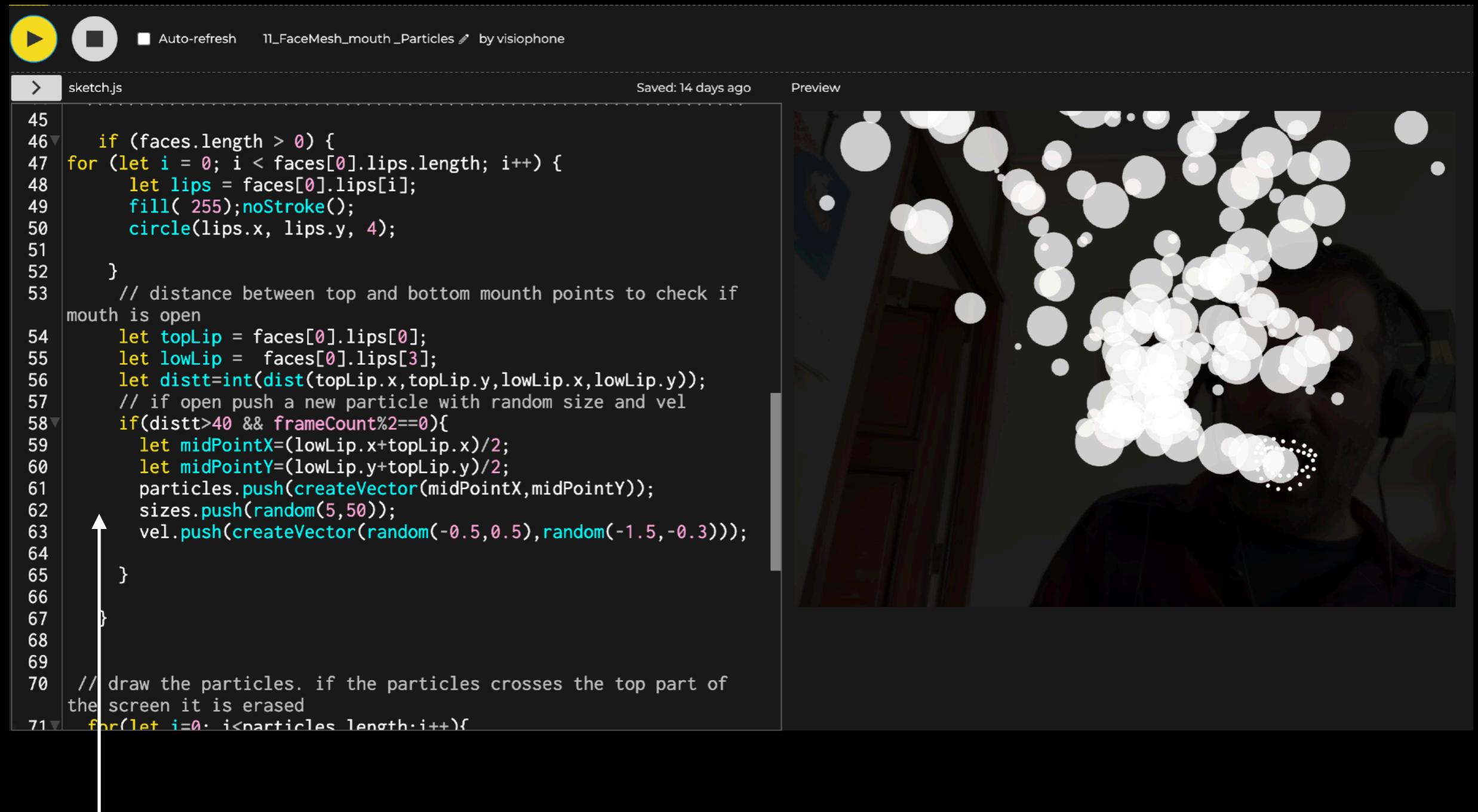


The image shows a Processing.js sketch window. At the top left are play and stop buttons, and an 'Auto-refresh' checkbox. The title bar says '11\_FaceMesh\_mouth\_Particles' by visiophone. The status bar indicates 'Saved: 14 days ago' and 'Preview'. The code editor on the left contains the following JavaScript code:

```
45 if (faces.length > 0) {
46   for (let i = 0; i < faces[0].lips.length; i++) {
47     let lips = faces[0].lips[i];
48     fill( 255);noStroke();
49     circle(lips.x, lips.y, 4);
50   }
51   // distance between top and bottom mounth points to check if
52   // mouth is open
53   let topLip = faces[0].lips[0];
54   let lowLip = faces[0].lips[3];
55   let distt=int(dist(topLip.x,topLip.y,lowLip.x,lowLip.y));
56   // if open push a new particle with random size and vel
57   if(distt>40 && frameCount%2==0){
58     let midPointX=(lowLip.x+topLip.x)/2;
59     let midPointY=(lowLip.y+topLip.y)/2;
60     particles.push(createVector(midPointX,midPointY));
61     sizes.push(random(5,50));
62     vel.push(createVector(random(-0.5,0.5),random(-1.5,-0.3)));
63   }
64 }
65
66 }
67 }
68 }
69 }
70 // draw the particles. if the particles crosses the top part of
71 // the screen it is erased
72 for(let i=0; i<particles.length;i++){
```

If mouth is open (distance between upper & lower lip bigger than 40) then create particles

## FACEMESH : mouth particles



The image shows a Processing.js sketch window. At the top left are play and stop buttons. Next to them is an 'Auto-refresh' checkbox and the file name '11\_FaceMesh\_mouth\_Particles' by 'visiophone'. To the right are 'Saved: 14 days ago' and 'Preview' buttons. The preview window on the right shows a person's face with numerous white circular particles appearing around the mouth area. The main code editor window contains the following JavaScript code:

```
45 if (faces.length > 0) {  
46   for (let i = 0; i < faces[0].lips.length; i++) {  
47     let lips = faces[0].lips[i];  
48     fill( 255);noStroke();  
49     circle(lips.x, lips.y, 4);  
50   }  
51   // distance between top and bottom mounth points to check if  
52   // mouth is open  
53   let topLip = faces[0].lips[0];  
54   let lowLip = faces[0].lips[3];  
55   let distt=int(dist(topLip.x,topLip.y,lowLip.x,lowLip.y));  
56   // if open push a new particle with random size and vel  
57   if(distt>40 && frameCount%2==0){  
58     let midPointX=(lowLip.x+topLip.x)/2;  
59     let midPointY=(lowLip.y+topLip.y)/2;  
60     particles.push(createVector(midPointX,midPointY));  
61     sizes.push(random(5,50));  
62     vel.push(createVector(random(-0.5,0.5),random(-1.5,-0.3)));  
63   }  
64 }  
65 }  
66 }  
67 }  
68 }  
69 }  
70 // draw the particles. if the particles crosses the top part of  
71 // the screen it is erased  
72 for(let i=0; i<particles.length;i++){
```

If mouth is open (distance between upper & lower lip bigger than 40) then create particles

## FACEMESH : mouth Oscillator

Mouth Opening changes oscillator frequency

The screenshot shows a Processing sketch titled "12\_FaceMesh\_mouth\_SOUND" by visiophone. The code editor displays the following P5.js code:

```
sketch.js*
13 // Sound
14 let osc, fft;
15
16 function preload() {
17   // Load the facemesh model
18   facemesh = ml5.facemesh();
19 }
20
21 function setup() {
22   createCanvas(640, 480);
23   // Create the webcam video and hide it
24   video = createCapture(VIDEO);
25   video.size(width, height);
26   video.hide();
27   // Start detecting faces from the webcam video
28   facemesh.detectStart(video, gotFaces);
29
30   // SOUND
31   osc = new p5.TriOsc(); // set frequency and type
32   osc.amp(0.5);
33   fft = new p5.FFT();
34   osc.start();
35 }
36
37 function gotFaces(error, faces) {
38   if (error) {
39     console.log(error);
40     return;
41   }
42
43   let leftLip = faces[0].lips[17];
44   let rightLip = faces[0].lips[34];
45
46   circle(leftLip.x, leftLip.y, 10);
47   circle(rightLip.x, rightLip.y, 10);
48   stroke(255); strokeWeight(1);
49   line(leftLip.x, leftLip.y, rightLip.x, rightLip.y);
50
51   // mouth opening/close changes oscillator's frequency
52   let freqq = map(distt, 20, 70, 20, 2880);
53   osc.freq(freqq);
54
55   // DRAW WAVE
56
57   fill(255);
58   let waveform = fft.waveform(); // analyze the waveform
59   beginShape();
60   strokeWeight(3);
61   let offSetX=100; //make the wave a little bigger than the mouth
62   for (let i = 0; i < waveform.length; i++) {
63     let x = map(i, 0, waveform.length, leftLip.x-offSetX,
64
65     // OpenGL status bar
66     351] GL Version: 3.0
67     351] OpenGL error chec
68
69   
```

The code uses the FaceMesh library to detect faces and the FFT library to analyze audio. It draws a line between the left and right lips and a waveform at the mouth position. The mouth opening changes the frequency of an oscillator, which is then used to modulate the waveform.

Annotations in the image:

- A callout points to the line `stroke(255); strokeWeight(1);` with the text "Draws audiowave at mouth position".
- A callout points to the line `osc.start();` with the text "P5Sound Library. Start Oscillator".
- A callout points to the OpenGL status bar with the text "OpenGL status bar".