

*Algorithmic
Art
Workbook #14
Pre-trained
AI models*



Table of Contents

MIT Licence	4
Workbook #14 Quick Content Summary	5
Module H Unit #3 handPose pretrained AI	7
The index.html file	8
Sketch H3.1 our starting sketch	9
Sketch H3.2 video	10
Sketch H3.3 putting you on the canvas	11
Sketch H3.4 flipping the video	12
Sketch H3.5 the handPose model	13
Sketch H3.6 start detecting	14
Sketch H3.7 callback	15
Sketch H3.8 an array of hands	17
Sketch H3.9 the keypoints	18
Sketch H3.10 mirroring the yellow dots	20
Sketch H3.11 reference keypoints	22
Sketch H3.12 index and thumb	24
Sketch H3.13 circle size	26
Sketch H3.14 adding some coloured color	28
Sketch H3.15 checking the distance	30
Sketch H3.16 slimmed down sketch	32
Sketch H3.17 many hands make light work	34
Sketch H3.18 the vertex	36
Module H Unit #4 handPose emoji	39
Sketch H4.1 index.html	40
Sketch H4.2 starting sketch	41
Sketch H4.3 hand of hands	42
Sketch H4.4 emitters	44
Sketch H4.5 particles	46
Sketch H4.6 movement	48
Sketch H4.7 let's go running	51
Sketch H4.8 is it alive?	55
Sketch H4.9 tweak	59
Module H Unit #5 faceMesh	64
Sketch H5.1 index.html	65
Sketch H5.2 our starting sketch	66
Sketch H5.3 points of view	67
Sketch H5.4 step back	69
Sketch H5.5 drawing outer lips	71
Sketch H5.6 and the inner lips	73

Sketch H5.7 lip sync	76
Sketch H5.8 now the nose	78
Blowing Bubbles	80
Sketch H5.9 adding the emitter	81
Sketch H5.10 adding the particle	84
Sketch H5.11 now the bubbles	88
Sketch H5.12 negative gravity	93
A face mask	98
Sketch H5.13 back to basics	99
Sketch H5.14 triangles	100
Sketch H5.15 showing the triangles	101
Sketch H5.16 adding a mask	103
Sketch H5.17 texture mapping	106
Sketch H5.18 mapping uv coordinates	108



MIT Licence

Copyright ©2026 Warren George

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NON-INFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

Brief summary:

Permissions (what you can do)

Commercial use
Modification
Distribution
Private use

Limitations (what is not covered)

Liability
Warranty



Workbook #14 Quick Content Summary

This continues the theme of AI and in particular using pre-trained models. In the last example, we trained our own very simple model. We can adopt models that have already been trained for specific purposes. The good thing is that ml5.js has already built them into the library, so we don't need to mess around and go looking for them.

If you are interested in AI, I will be bringing out another series of workbooks that go into a little bit more depth. Currently, they are free to download on my website www.ElegantAI.org, but I am intending to publish them at some time in the future; we will see.

Algorithmic Art

Module H

Unit #3

pretrained handPose

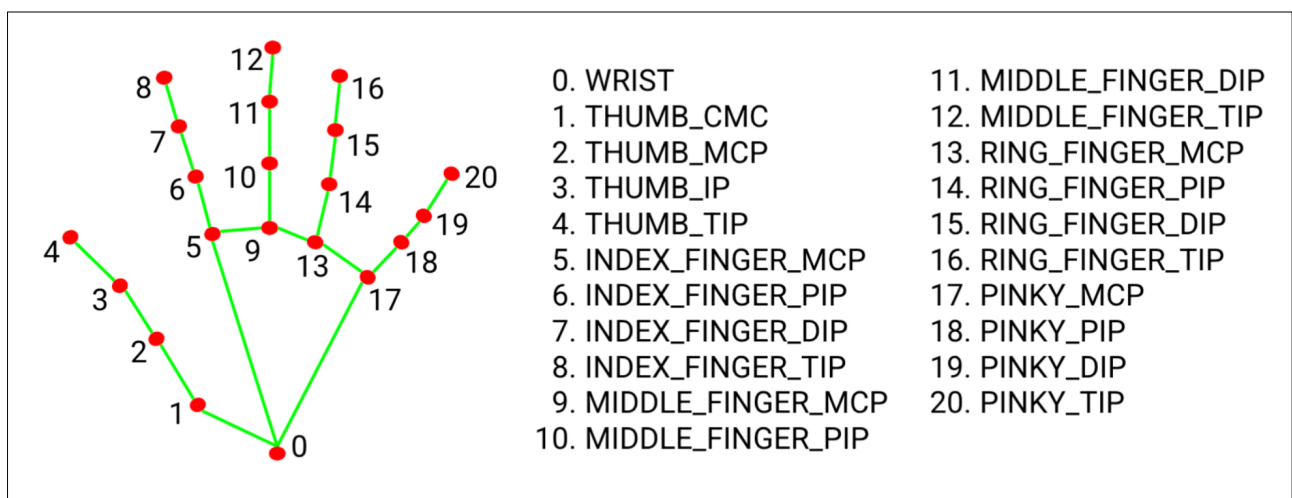


Module H Unit #3 handPose pretrained AI

The **handPose** model is another pre-trained model that is part of the ml5.js library. We can get all the data points of your hands and use some of them to control the size and position of a circle.

The **handPose** model can detect **21** keypoints on a hand (see **figure 1** below). It can detect both hands at the same time and determine the left from the right hand. Also, the keypoints are given as **3D** co-ordinates, but we will focus on the **2D** co-ordinates for now.

Figure 1: all 21 keypoints for handPose





The index.html file

Adding the ml5.js file to the index.html file.

```
<!DOCTYPE html>
<html lang="en"><head>
  <script src="https://cdn.jsdelivr.net/npm/p5@2.3.2/lib/p5.js"></script>
  <script src="https://unpkg.com/ml5@1/dist/ml5.min.js"></script>
  <link rel="stylesheet" type="text/css" href="style.css">
  <meta charset="utf-8">
</head>
<body>
  <main>
  </main>
  <script src="sketch.js"></script>
</body></html>
```

Notes

Insert the ml5.js library if you haven't carried over from the previous unit.



Sketch H3.1 our starting sketch

The process is similar to the previous unit, but we will still go through it step by step.

```
function setup()
{
  createCanvas(640, 480)
}

function draw()
{
  background(220)
}
```

Notes

The usual suspect.



Sketch H3.2 video

Getting the video feed from the webcam.

```
let video

function setup()
{
  createCanvas(640, 480)
  video = createCapture(VIDEO)
}

function draw()
{
  background(220)
}
```

Notes

You should get a grey canvas and underneath it, a video image of you.



Sketch H3.3 putting you on the canvas

We hide the video stream and use the `image()` function. Notice that we include the width and height, not entirely necessary in this case, but if you wanted to match the canvas dimensions, this is the way to go.

```
let video

function setup()
{
  createCanvas(640, 480)
  video = createCapture(VIDEO)
  video.hide()
}

function draw()
{
  background(220)
  image(video, 0, 0, width, height)
}
```

Notes

Now you should have a video image of yourself where the canvas was, but if you lift up your hand, it is not mirrored.



Sketch H3.4 flipping the video

We can flip the video using a function in p5.js.

```
let video

function setup()
{
  createCanvas(640, 480)
  video = createCapture(VIDEO, {flipped: true})
  video.hide()
}

function draw()
{
  background(220)
  image(video, 0, 0, width, height)
}
```

Notes

You will now have a mirror image from the camera.



Sketch H3.5 the handPose model

There is a pre-trained model within ml5.js, and all we need to do is call it. We will preload it before we try to use it; otherwise, we may get error messages.

! It takes a few seconds longer, be patient.

```
let video
let handPose

async function setup()
{
  ml5.setBackend("webgl")
  handPose = await ml5.handPose()
  createCanvas(640, 480)
  video = createCapture(VIDEO, {flipped: true})
  video.hide()
}

function draw()
{
  background(220)
  image(video, 0, 0, width, height)
}
```

Notes

Nothing will happen when you hold your hand up to the camera because we haven't connected the two together.



Sketch H3.6 start detecting

We use a function called `detectStart()`, and this will connect the model to the video feed.

! Don't run it yet.

```
let video
let handPose

async function setup()
{
  ml5.setBackend("webgl")
  handPose = await ml5.handPose()
  createCanvas(640, 480)
  video = createCapture(VIDEO, {flipped: true})
  video.hide()
  handPose.detectStart(video)
}

function draw()
{
  background(220)
  image(video, 0, 0, width, height)
}
```

Notes

! If you run this now, you will get an error message.

It will tell you that you need a callback function. This callback function will hold the data.



Sketch H3.7 callback

Let's create a callback function called `gotHands()` and `console.log()` the **results** as we did before.

```
let video
let handPose

async function setup()
{
  ml5.setBackend("webgl")
  handPose = await ml5.handPose()
  createCanvas(640, 480)
  video = createCapture(VIDEO, {flipped: true})
  video.hide()
  handPose.detectStart(video, gotHands)
}

function draw()
{
  background(220)
  image(video, 0, 0, width, height)
}

function gotHands(results)
{
  console.log(results)
}
```

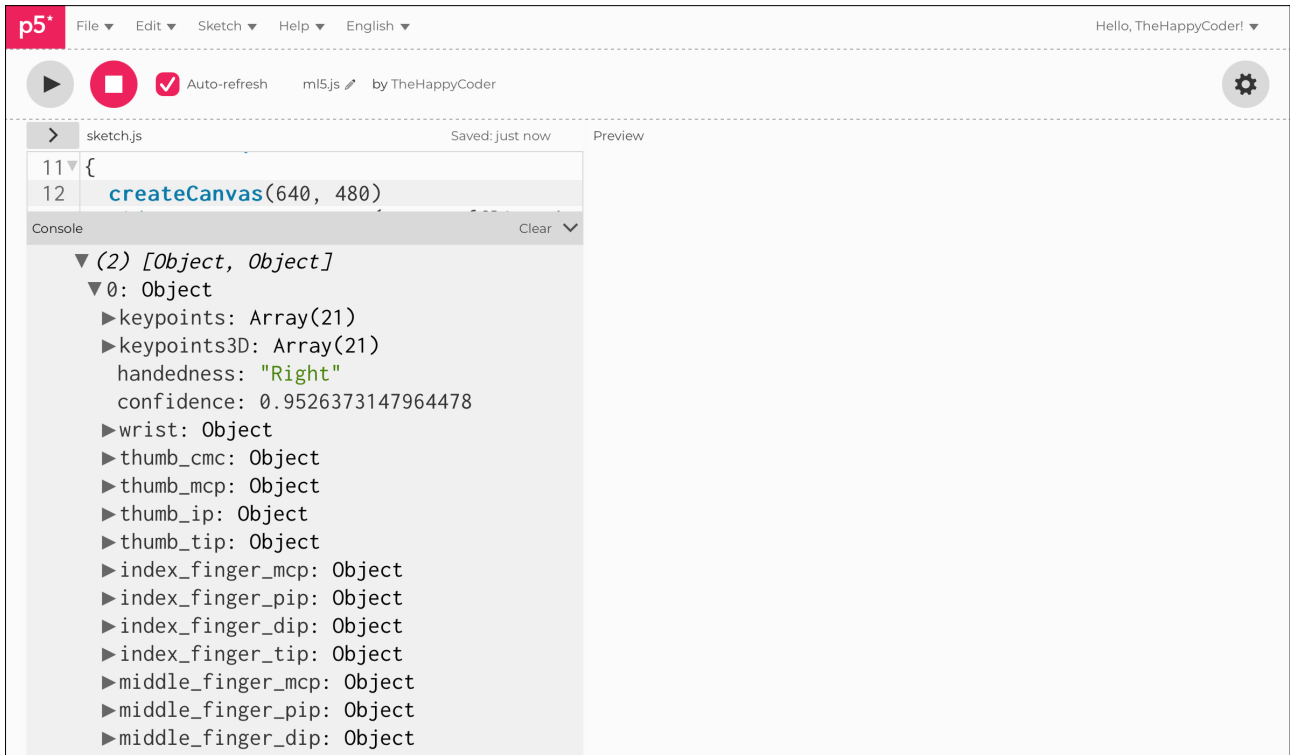
Notes

You will get an empty array when it doesn't see any hands, one array object when you hold up one hand, and two array objects when it can see two hands. If you pause the programme, you can examine the content of the arrays. You will notice how many points there are and the labelling of your fingers, plus other parts of your hand. You have **21 keypoints** for each hand.

Challenge

Highly recommend that you explore the results in the console.

Figure H3.7





Sketch H3.8 an array of hands

We need to store this data and access the array of your hands.

```
let video
let handPose
let hands = []

async function setup()
{
  ml5.setBackend("webgl")
  handPose = await ml5.handPose()
  createCanvas(640, 480)
  video = createCapture(VIDEO, {flipped: true})
  video.hide()
  handPose.detectStart(video, gotHands)
}

function draw()
{
  background(220)
  image(video, 0, 0, width, height)
}

function gotHands(results)
{
  hands = results
}
```

Notes

We create an empty array to put the data in and fill it with the results in the callback function.



Sketch H3.9 the keypoints

We want to cycle through all the **keypoints** in the hands array and draw them. For this, we use a nested loop.

```
let video
let handPose
let hands = []

async function setup()
{
  ml5.setBackend("webgl")
  handPose = await ml5.handPose()
  createCanvas(640, 480)
  video = createCapture(VIDEO, {flipped: true})
  video.hide()
  handPose.detectStart(video, gotHands)
}

function draw()
{
  background(220)
  image(video, 0, 0, width, height)
  for (let i = 0; i < hands.length; i++)
  {
    for (let j = 0; j < hands[i].keypoints.length; j++)
    {
      let keypoint = hands[i].keypoints[j]
      fill('yellow')
      circle(keypoint.x, keypoint.y, 10)
    }
  }
}

function gotHands(results)
{
  hands = results
}
```

Notes

Notice that the yellow dots are not mirrored!



Sketch H3.10 mirroring the yellow dots

We can add the **flipped** is **true**. Also, we will remove the video of you and replace it with a dark red background. Remove the **//** code.

```
let video
let handPose
let hands = []

async function setup()
{
  ml5.setBackend("webgl")
  handPose = await ml5.handPose({flipped: true})
  createCanvas(640, 480)
  video = createCapture(VIDEO, {flipped: true})
  video.hide()
  handPose.detectStart(video, gotHands)
}

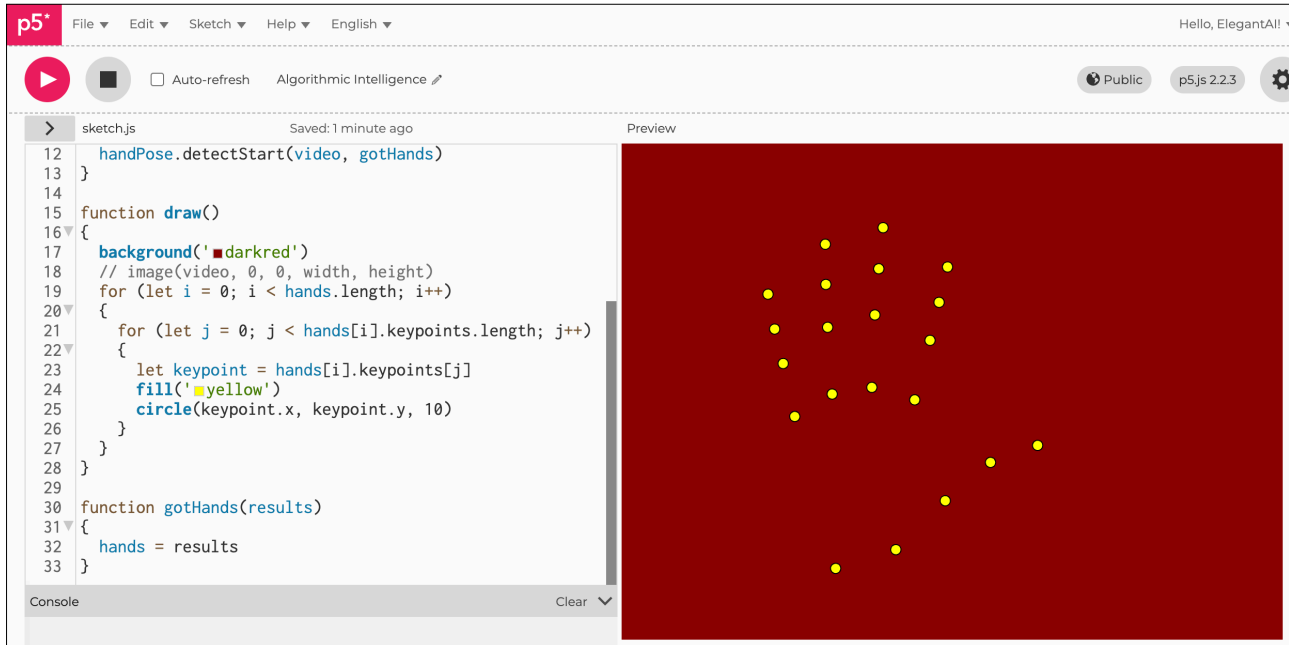
function draw()
{
  background('darkred')
  // image(video, 0, 0, width, height)
  for (let i = 0; i < hands.length; i++)
  {
    for (let j = 0; j < hands[i].keypoints.length; j++)
    {
      let keypoint = hands[i].keypoints[j]
      fill('yellow')
      circle(keypoint.x, keypoint.y, 10)
    }
  }
}

function gotHands(results)
{
  hands = results
}
```

Notes

The points are mirrored on both hands.

Figure H3.10





Sketch H3.11 reference keypoints

Instead of points, let's get the index reference for each point, because then we can use specific **keypoints**.

```
let video
let handPose
let hands = []

async function setup()
{
  ml5.setBackend("webgl")
  handPose = await ml5.handPose({flipped: true})
  createCanvas(640, 480)
  video = createCapture(VIDEO, {flipped: true})
  video.hide()
  handPose.detectStart(video, gotHands)
}

function draw()
{
  background('darkred')
  for (let i = 0; i < hands.length; i++)
  {
    for (let j = 0; j < hands[i].keypoints.length; j++)
    {
      let keypoint = hands[i].keypoints[j]
      fill('yellow')
      // circle(keypoint.x, keypoint.y, 10)
      text(i, keypoint.x -10, keypoint.y)
      text(j, keypoint.x, keypoint.y)
    }
  }
}

function gotHands(results)
{
  hands = results
}
```

Notes

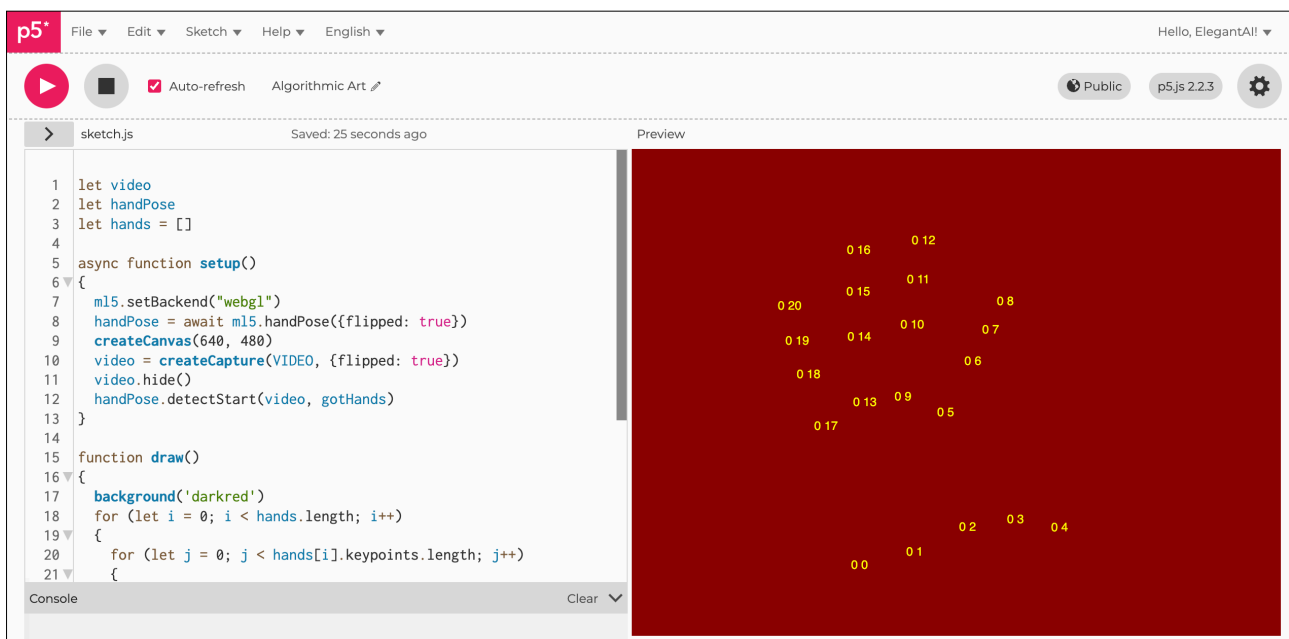
You will notice that when you hold up two hands, you get a **1** for the left hand followed by the reference number on that hand, and a **0** for the right hand followed by the corresponding reference number on the other hand.

We want the index finger and thumb (below are the reference numbers):

Index finger: keypoint **8**

Thumb: keypoint **4**

Figure H3.11





Sketch H3.12 index and thumb

Just to prove we have the right keypoint reference, we can write the words onto the canvas. We are only interested in one hand at a time. We can remove the commented-out nested loops.

```
let video
let handPose
let hands = []

async function setup()
{
  ml5.setBackend("webgl")
  handPose = await ml5.handPose({flipped: true})
  createCanvas(640, 480)
  video = createCapture(VIDEO, {flipped: true})
  video.hide()
  handPose.detectStart(video, gotHands)
}

function draw()
{
  background('darkred')

  if (hands.length > 0)
  {
    fill('white')
    noStroke()
    let index = hands[0].keypoints[8]
    let thumb = hands[0].keypoints[4]
    text('index', index.x, index.y)
    text('thumb', thumb.x, thumb.y)
  }
  // for (let i = 0; i < hands.length; i++)
  // {
  //   for (let j = 0; j < hands[i].keypoints.length; j++)
  //   {
  //     let keypoint = hands[i].keypoints[j]
  //     fill('yellow')
  //     // circle(keypoint.x, keypoint.y, 10)
  //   }
  // }
}
```

```

//    text(i, keypoint.x -10, keypoint.y)
//    text(j, keypoint.x, keypoint.y)
//  }
// }

}

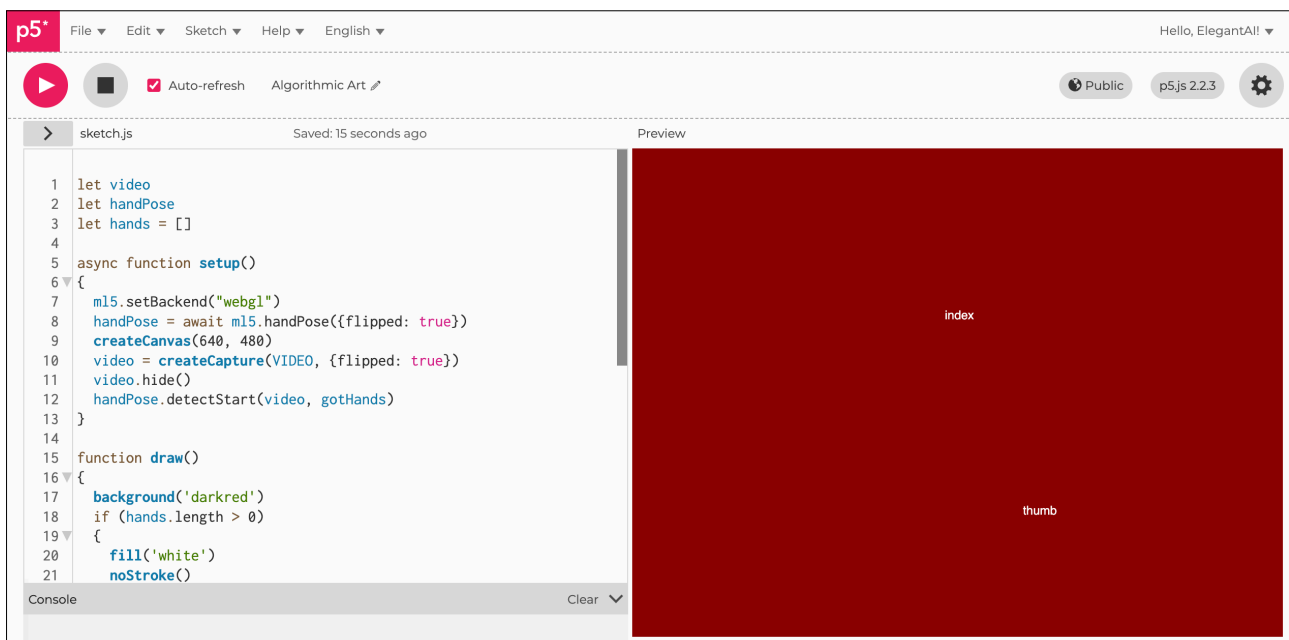
function gotHands(results)
{
  hands = results
}

```

Notes

You should have the index finger and thumb correctly identified using either hand.

Figure H3.12





Sketch H3.13 circle size

Instead of writing text, we will draw a circle that changes depending on the distance (d) between those two fingers.

! We have completely removed the nested loops as we don't need that information anymore. Also remove `//` commented code.

```
let video
let handPose
let hands = []

async function setup()
{
  ml5.setBackend("webgl")
  handPose = await ml5.handPose({flipped: true})
  createCanvas(640, 480)
  video = createCapture(VIDEO, {flipped: true})
  video.hide()
  handPose.detectStart(video, gotHands)
}

function draw()
{
  background('darkred')
  if (hands.length > 0)
  {
    // fill('white')
    // noStroke()
    noFill()
    strokeWeight(5)
    stroke('yellow')

    let index = hands[0].keypoints[8]
    let thumb = hands[0].keypoints[4]

    // text('index', index.x, index.y)
    // text('thumb', thumb.x, thumb.y)
    let d = dist(index.x, index.y, thumb.x, thumb.y)
    circle(index.x, index.y, d)
  }
}
```

```
function gotHands(results)
{
  hands = results
}
```

Notes

What you get is the circle changing size as you open and close your thumb and index finger. However, I have left the centre on the index finger for now. We want it halfway between the index finger and the thumb. You may not really be able to see it with the image commented out, but the circle is entered around the index finger.

Challenges

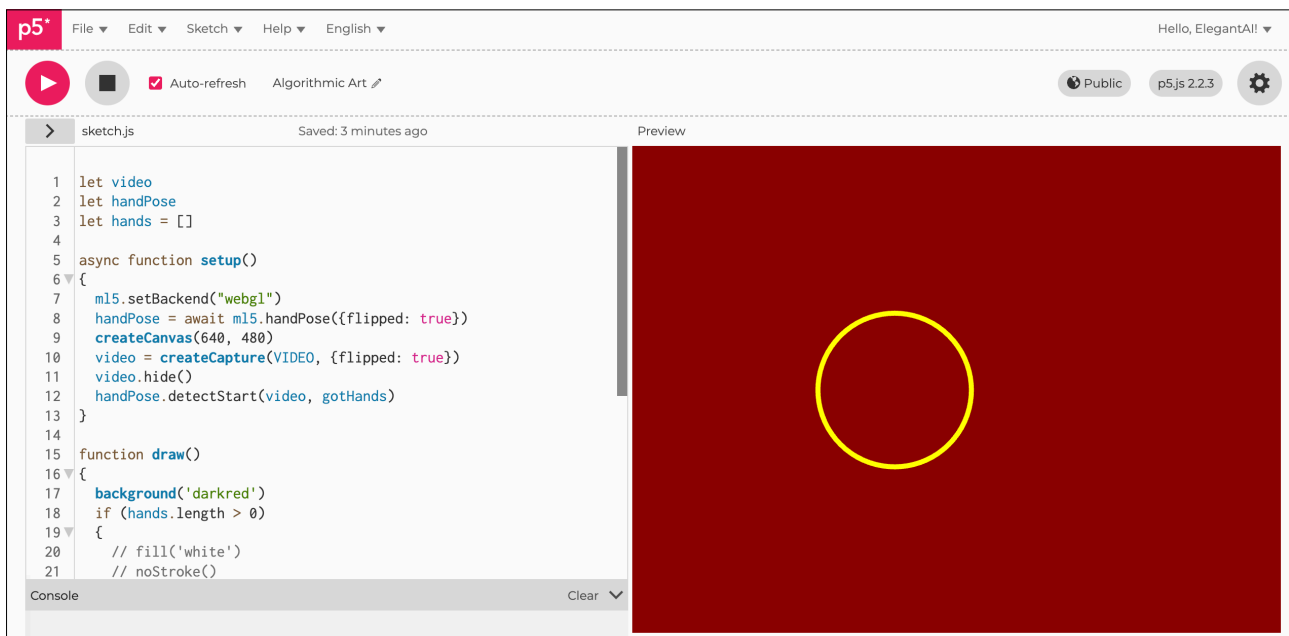
1. Uncomment the line of code to show the video feed.
2. You could try other fingers.

Code Explanation

```
let d = dist(index.x, index.y,
thumb.x, thumb.y)
```

Measures the distance between the x, y co-ordinates of the index finger with the x, y co-ordinates of the thumb

Figure H3.13





Sketch H3.14 adding some coloured color

Now we have the centre of the circle between our thumb and index finger.

```
let video
let handPose
let hands = []
let colour

async function setup()
{
  ml5.setBackend("webgl")
  handPose = await ml5.handPose({flipped: true})
  createCanvas(640, 480)
  video = createCapture(VIDEO, {flipped: true})
  video.hide()
  handPose.detectStart(video, gotHands)
  colour = color(255, 255, 0)
}

function draw()
{
  background('darkred')
  if (hands.length > 0)
  {
    noFill()
    strokeWeight(5)
    stroke(colour)

    let index = hands[0].keypoints[8]
    let thumb = hands[0].keypoints[4]
    let d = dist(index.x, index.y, thumb.x, thumb.y)
    circle(index.x, index.y, d)
  }
}

function gotHands(results)
{
  hands = results
}
```

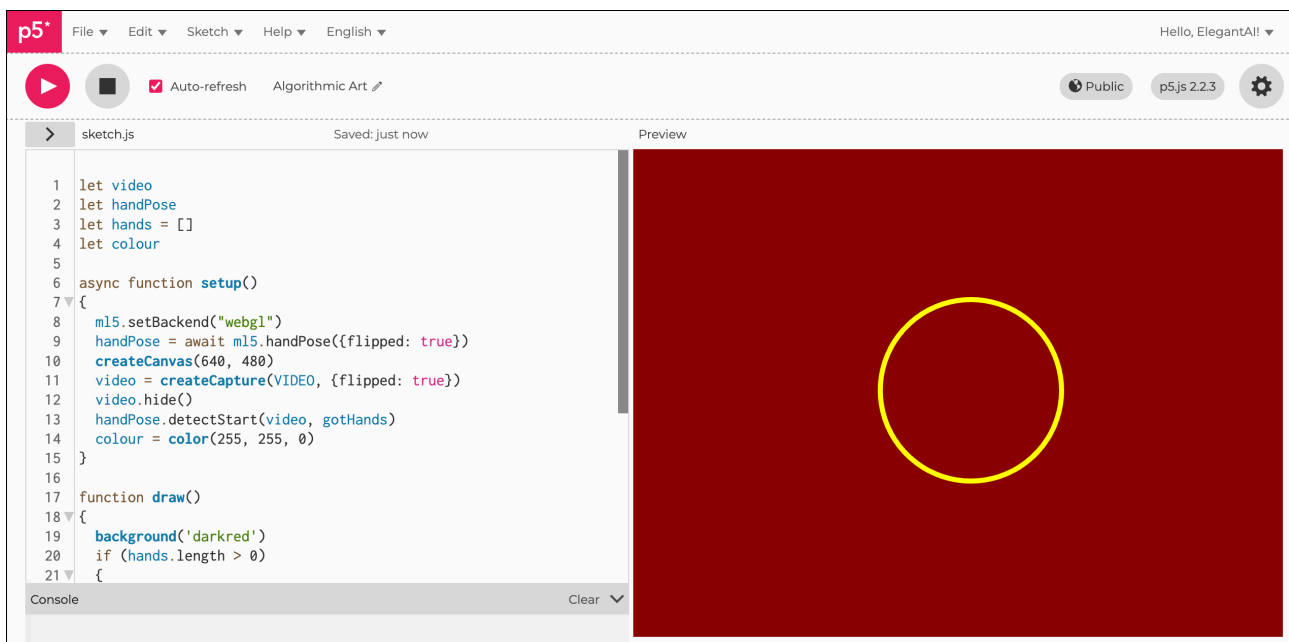
Notes

Just make a few tweaks, which doesn't do much as of yet. The colour variable holds the `color()` function values. We can change these at will.

Challenge

How about drawing a square, rectangle, or ellipse?

Figure H3.14





Sketch H3.15 checking the distance

Now that we have a distance between the thumb and index finger, we can change the colour of the circle to a random colour. Every time you close your index finger and thumb, the circle should change colour.

```
let video
let handPose
let hands = []
let colour

async function setup()
{
  ml5.setBackend("webgl")
  handPose = await ml5.handPose({flipped: true})
  createCanvas(640, 480)
  video = createCapture(VIDEO, {flipped: true})
  video.hide()
  handPose.detectStart(video, gotHands)
  colour = color(255, 255, 0)
}

function draw()
{
  background('darkred')
  if (hands.length > 0)
  {
    noFill()
    strokeWeight(5)
    stroke(colour)
    let index = hands[0].keypoints[8]
    let thumb = hands[0].keypoints[4]
    let d = dist(index.x, index.y, thumb.x, thumb.y)
    circle(index.x, index.y, d)

    if (d < 50)
    {
      colour = color(random(255), random(255), random(255))
    }
  }
  else
```

```

    {
      stroke(colour)
    }
  }
}

function gotHands(results)
{
  hands = results
}

```

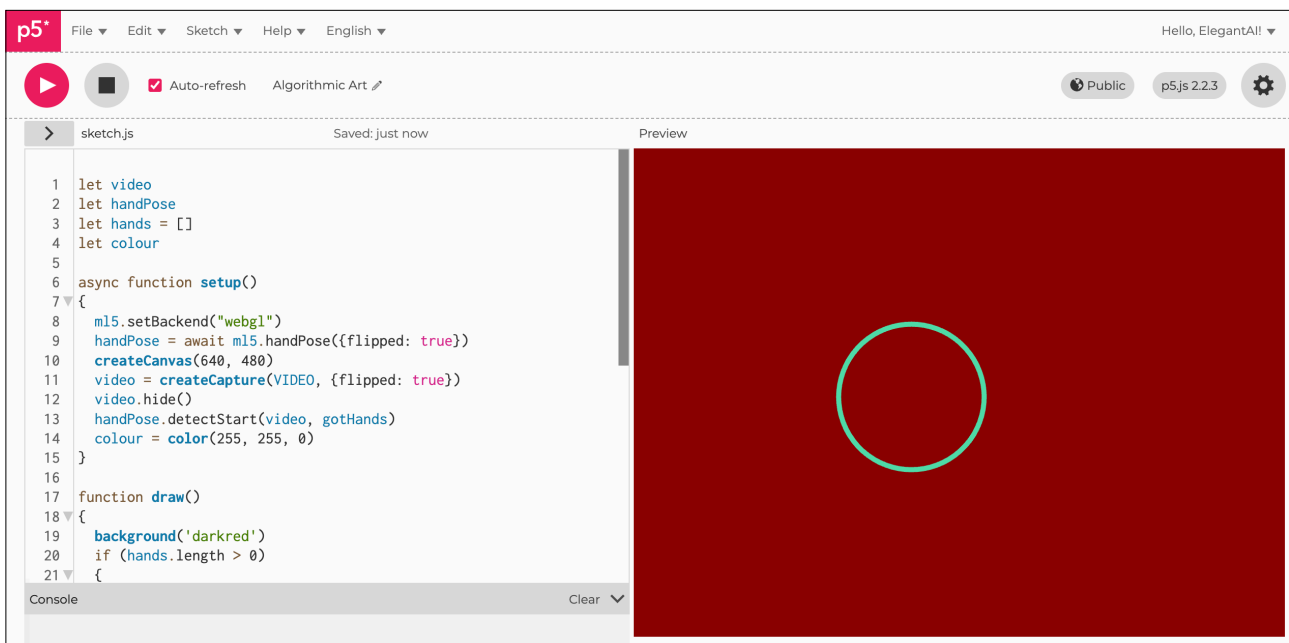
Notes

The reason why we have an `else{ }` condition is so that it settles on one random colour; otherwise, it would cycle through all the random colours, flickering continuously.

Challenge

Change the circle fill colour.

Figure H3.15





Sketch H3.16 slimmed down sketch

! Get rid of everything relating to changing the colour except what is left below. Change the highlighted code.

```
let video
let handPose
let hands = []

async function setup()
{
  ml5.setBackend("webgl")
  handPose = await ml5.handPose({flipped: true})
  createCanvas(640, 480)
  video = createCapture(VIDEO, {flipped: true})
  video.hide()
  handPose.detectStart(video, gotHands)
}

function draw()
{
  background('darkred')
  if (hands.length > 0)
  {
    noFill()
    strokeWeight(2)
    stroke('yellow')
    let index = hands[0].keypoints[8]
    let thumb = hands[0].keypoints[4]
  }
}

function gotHands(results)
{
  hands = results
}
```

Notes

We will see nothing yet. Just check for errors.



Sketch H3.17 many hands make light work

We need to make sure we detect both hands for this. One hand is `hands[0]` and the other hand is `hands[1]`.

```
let video
let handPose
let hands = []

async function setup()
{
  ml5.setBackend("webgl")
  handPose = await ml5.handPose({flipped: true})
  createCanvas(640, 480)
  video = createCapture(VIDEO, {flipped: true})
  video.hide()
  handPose.detectStart(video, gotHands)
}

function draw()
{
  background('darkred')
  if (hands[0] && hands[1])
  {
    noFill()
    strokeWeight(2)
    stroke('yellow')
    let indexL = hands[0].keypoints[8]
    let thumbL = hands[0].keypoints[4]
    let indexR = hands[1].keypoints[8]
    let thumbR = hands[1].keypoints[4]
  }
}

function gotHands(results)
{
  hands = results
}
```

Notes

We are getting the thumb and index finger of each hand. Still nothing to see here.

Code Explanation

```
if (hands[0] && hands[1])
```

Checks to see if it can see both hands



Sketch H3.18 the vertex

Now we can draw the vertex points for our irregular shape. When it detects both hands, it joins up all the points. Move your fingers around to see the effect. Just keep both hands in view.

```
let video
let handPose
let hands = []

async function setup()
{
  ml5.setBackend("webgl")
  handPose = await ml5.handPose({flipped: true})
  createCanvas(640, 480)
  video = createCapture(VIDEO, {flipped: true})
  video.hide()
  handPose.detectStart(video, gotHands)
}

function draw()
{
  background('darkred')
  if (hands[0] && hands[1])
  {
    noFill()
    strokeWeight(2)
    stroke('yellow')
    let indexL = hands[0].keypoints[8]
    let thumbL = hands[0].keypoints[4]
    let indexR = hands[1].keypoints[8]
    let thumbR = hands[1].keypoints[4]

    beginShape()
    vertex(indexL.x, indexL.y)
    vertex(indexR.x, indexR.y)
    vertex(thumbR.x, thumbR.y)
    vertex(thumbL.x, thumbL.y)
    endShape(CLOSE)
  }
}
```

```

}

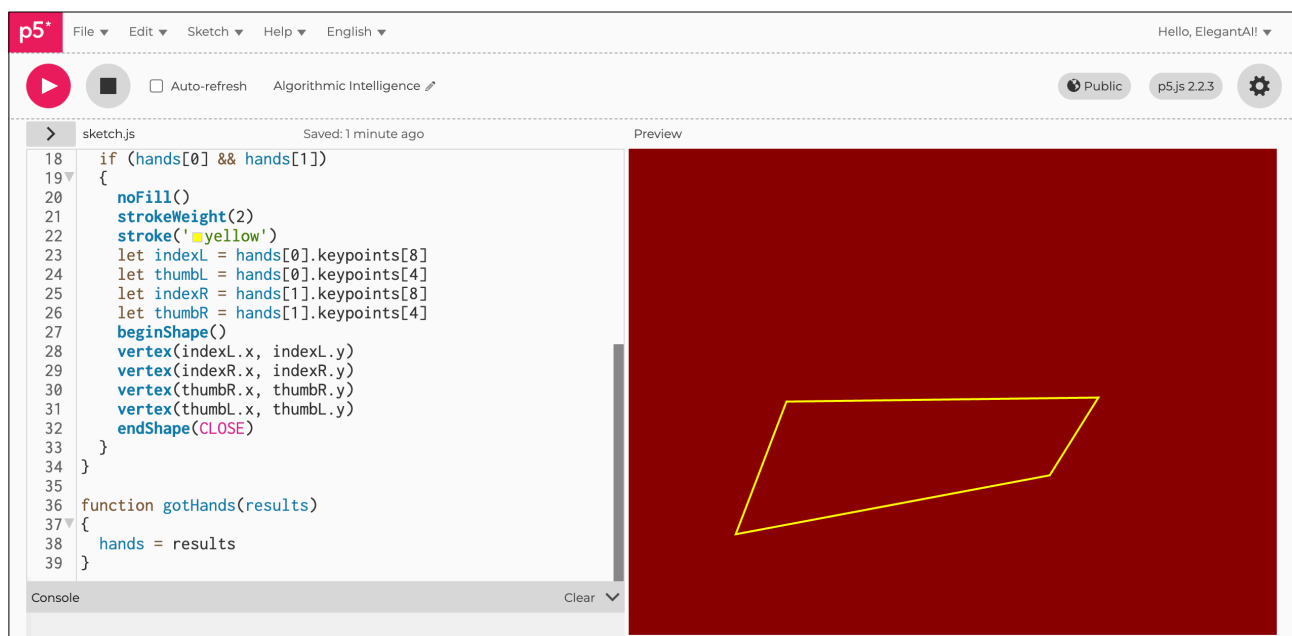
function gotHands(results)
{
  hands = results
}

```

Challenges

1. What shapes could you create with all the fingers?
2. Change the fill colour when you bring your fingers together.

Figure H3.18



Algorithmic Art

Module H

Unit #4

pretrained
handPose
emoji



Module H Unit #4 handPose emoji

This is using the same pre-trained model, but instead of points, we want to emit an emoji as a particle. This makes it a little bit more complicated, but not much. We are going to create particles, which we have covered briefly before, but I will be covering them in more depth in the Physics Simulation series of workbooks.



Sketch H4.1 index.html

As before, make sure you have the ml5.js library line of code included. You will also note that we (at the time of writing) are on version **2.2.3** of p5.js.

```
<!DOCTYPE html>
<html lang="en"><head>
  <script src="https://cdn.jsdelivr.net/npm/p5@2.2.3/lib/p5.js"></script>
  <script src="https://unpkg.com/ml5@1/dist/ml5.min.js"></script>
  <link rel="stylesheet" type="text/css" href="style.css">
  <meta charset="utf-8">
</head>
<body>
  <main>
  </main>
  <script src="sketch.js"></script>
</body></html>
```

Notes

Just a quick reminder.



Sketch H4.2 starting sketch

Pulling the key code from the previous unit, we have the starting point. If you want to see (check it is working), then comment out the background and replace it with the `image()` function, I put it there for checking purposes otherwise just leave it out, also leave it in if you would rather see your hands.

```
let video
let handPose
let hands = []

async function setup()
{
  createCanvas(640, 480)
  ml5.setBackend("webgl")
  handPose = await ml5.handPose({flipped: true})
  video = await createCapture(VIDEO, {flipped: true})
  video.hide()
  handPose.detectStart(video, gotHands)
}

function draw()
{
  background('darkred')
  // image(video, 0, 0)
}

function gotHands(results)
{
  hands = results
}
```

Notes

Nothing to see but the foundation is in place.



Sketch H4.3 hand of hands

We want to get all the data points of either (or both) hands. To do that, we create a separate variable called `hand` and copy the array called `hands`, which is full of all the data points (**keypoints**). We start with our conditional expression to see if there are any **keypoints** detected. Then, we loop through all the **keypoints** in the `hand` array.

```
let video
let handPose
let hands = []

async function setup()
{
  createCanvas(640, 480)
  ml5.setBackend("webgl")
  handPose = await ml5.handPose({flipped: true})
  video = await createCapture(VIDEO, {flipped: true})
  video.hide()
  handPose.detectStart(video, gotHands)
}

function draw()
{
  background('darkred')
  if (hands.length > 0)
  {
    for (let hand of hands)
    {
      for (let i = 0; i < hand.keypoints.length; i++)
      {
        let keypoint = hand.keypoints[i]
        console.log(hand.keypoints.length)
      }
    }
  }
}

function gotHands(results)
{

```

```
    hands = results  
}
```

Notes

This does not do anything yet; it is just adding another piece to the jigsaw. The console log will check to see how many datapoints (**keypoints**) it has detected. There should be **21**.



Sketch H4.4 emitters

We have our hands array, we need to make those datapoints become **emitters** rather than just a point or circle. We introduce an emitter class along with an array. In other words, we want the origins of the **emitters** where the **keypoints** of your hand are.

```
let video
let handPose
let hands = []
let emitters = []

async function setup()
{
  createCanvas(640, 480)
  ml5.setBackend("webgl")
  handPose = await ml5.handPose({flipped: true})
  video = await createCapture(VIDEO, {flipped: true})
  video.hide()
  handPose.detectStart(video, gotHands)

  for (let i = 0; i < 21; i++)
  {
    emitters.push(new Emitter(width/2, height/2))
  }
}

function draw()
{
  background('darkred')
  if (hands.length > 0)
  {
    for (let hand of hands)
    {
      for (let i = 0; i < hand.keypoints.length; i++)
      {
        let keypoint = hand.keypoints[i]
        let emitter = emitters[0]
        emitter.origin.x = keypoint.x
        emitter.origin.y = keypoint.y
        console.log(emitter.origin.x)
```

```
    }  
  }  
}  
  
function gotHands(results)  
{  
  hands = results  
}  
  
class Emitter  
{  
  constructor(x, y)  
  {  
    this.origin = createVector(x, y)  
  }  
}
```

Notes

We initially fill the emitter array with all the **x, y** values set to **width/2, height/2**; it doesn't really matter, we could just set them all to zero. There is nothing to see except the **x** values in the console. After this, you can get rid of the console log.



Sketch H4.5 particles

We need some particles to play with. So we need a particle class and a particle. The particle will be an emoji which you can easily get from most modern keyboards; otherwise, just use `point()` or `circle()`. I have used a flower emoji from Apple.

```
let video
let handPose
let hands = []
let emitters = []

async function setup()
{
  createCanvas(640, 480)
  ml5.setBackend("webgl")
  handPose = await ml5.handPose({flipped: true})
  video = await createCapture(VIDEO, {flipped: true})
  video.hide()
  handPose.detectStart(video, gotHands)
  for (let i = 0; i < 21; i++)
  {
    emitters.push(new Emitter(width/2, height/2))
  }
}

function draw()
{
  background('darkred')
  // image(video, 0, 0)
  if (hands.length > 0)
  {
    for (let hand of hands)
    {
      for (let i = 0; i < hand.keypoints.length; i++)
      {
        let keypoint = hand.keypoints[i]
        let emitter = emitters[0]
        emitter.origin.x = keypoint.x
        emitter.origin.y = keypoint.y
```

```

        // console.log(emitter.origin.x)
    }
}
}
}

function gotHands(results)
{
    hands = results
}

class Emitter
{
    constructor(x, y)
    {
        this.origin = createVector(x, y)
    }
}

class Particle
{
    constructor(x, y)
    {
        this.position = createVector(x, y)
        this.icon = '🌻'
    }

    show()
    {
        textSize(24)
        textAlign(CENTER)
        noStroke()
        text(this.icon, this.position.x, this.position.y)
    }
}

```

Notes

Nothing to see yet except a nice red canvas.



Sketch H4.6 movement

As with all emitters, there is movement; you want the particles to go somewhere. They could float off, fall to the ground, or fly in all directions. We are going to do the latter, so we need to give those particles a velocity and acceleration; we have already given them a location (**position**).

```
let video
let handPose
let hands = []
let emitters = []

async function setup()
{
  createCanvas(640, 480)
  ml5.setBackend("webgl")
  handPose = await ml5.handPose({flipped: true})
  video = await createCapture(VIDEO, {flipped: true})
  video.hide()
  handPose.detectStart(video, gotHands)
  for (let i = 0; i < 21; i++)
  {
    emitters.push(new Emitter(width/2, height/2))
  }
}

function draw()
{
  background('darkred')
  if (hands.length > 0)
  {
    for (let hand of hands)
    {
      for (let i = 0; i < hand.keypoints.length; i++)
      {
        let keypoint = hand.keypoints[i]
        let emitter = emitters[0]
        emitter.origin.x = keypoint.x
        emitter.origin.y = keypoint.y
```

```

    emitter.addParticle()
  }
}

function gotHands(results)
{
  hands = results
}

class Emitter
{
  constructor(x, y)
  {
    this.origin = createVector(x, y)
    this.particles = []
  }

  addParticle()
  {
    this.particles.push(new Particle(this.origin.x, this.origin.y))
  }
}

class Particle
{
  constructor(x, y)
  {
    this.position = createVector(x, y)
    this.acceleration = createVector(0, 0)
    this.velocity = p5.Vector.random2D()

    this.icon = '🌻'
  }

  show()
  {
    textSize(24)
  }
}

```

```
    textAlign(CENTER)
    noStroke()
    text(this.icon, this.position.x, this.position.y)
  }

  move()
  {
    this.velocity.add(this.acceleration)
    this.position.add(this.velocity)
    this.acceleration.mult(0)
  }
}
```

Notes

There is so much to unpack here, although nothing is seen yet. For a deeper understanding of vectors, movement, etc., it will be worth digging into the physics workbooks.

Challenge

If you comment out a line of code such as (later on): `this.acceleration.mult(0)`. You will see what difference it makes and why we multiply the acceleration by zero on each iteration.



Sketch H4.7 let's go running

Most of the pieces are in place. We now want to run everything, so we will be creating functions inside both classes called `run()`. This will produce a flurry of particles when it sees your hand.

```
let video
let handPose
let hands = []
let emitters = []

async function setup()
{
  createCanvas(640, 480)
  ml5.setBackend("webgl")
  handPose = await ml5.handPose({flipped: true})
  video = await createCapture(VIDEO, {flipped: true})
  video.hide()
  handPose.detectStart(video, gotHands)
  for (let i = 0; i < 21; i++)
  {
    emitters.push(new Emitter(width/2, height/2))
  }
}

function draw()
{
  background('darkred')
  if (hands.length > 0)
  {
    for (let hand of hands)
    {
      for (let i = 0; i < hand.keypoints.length; i++)
      {
        let keypoint = hand.keypoints[i]
        let emitter = emitters[0]
        emitter.origin.x = keypoint.x
        emitter.origin.y = keypoint.y
        emitter.addParticle()
```

```

    }
  }
}

for (let emitter of emitters)
{
  emitter.run()
}

}

function gotHands(results)
{
  hands = results
}

class Emitter
{
  constructor(x, y)
  {
    this.origin = createVector(x, y)
    this.particles = []
  }

  addParticle()
  {
    this.particles.push(new Particle(this.origin.x, this.origin.y))
  }

  run()
  {
    for (let i = this.particles.length - 1; i > 0; i--)
    {
      this.particles[i].run()
    }
  }
}

class Particle
{

```

```
constructor(x, y)
{
  this.position = createVector(x, y)
  this.acceleration = createVector(0, 0)
  this.velocity = p5.Vector.random2D()
  this.icon = '🌻'
}
```

```
run()
{
  this.show()
  this.move()
}
```

```
show()
{
  textSize(24)
  textAlign(CENTER)
  noStroke()
  text(this.icon, this.position.x, this.position.y)
}

move()
{
  this.velocity.add(this.acceleration)
  this.position.add(this.velocity)
  this.acceleration.mult(0)
}
}
```

Notes

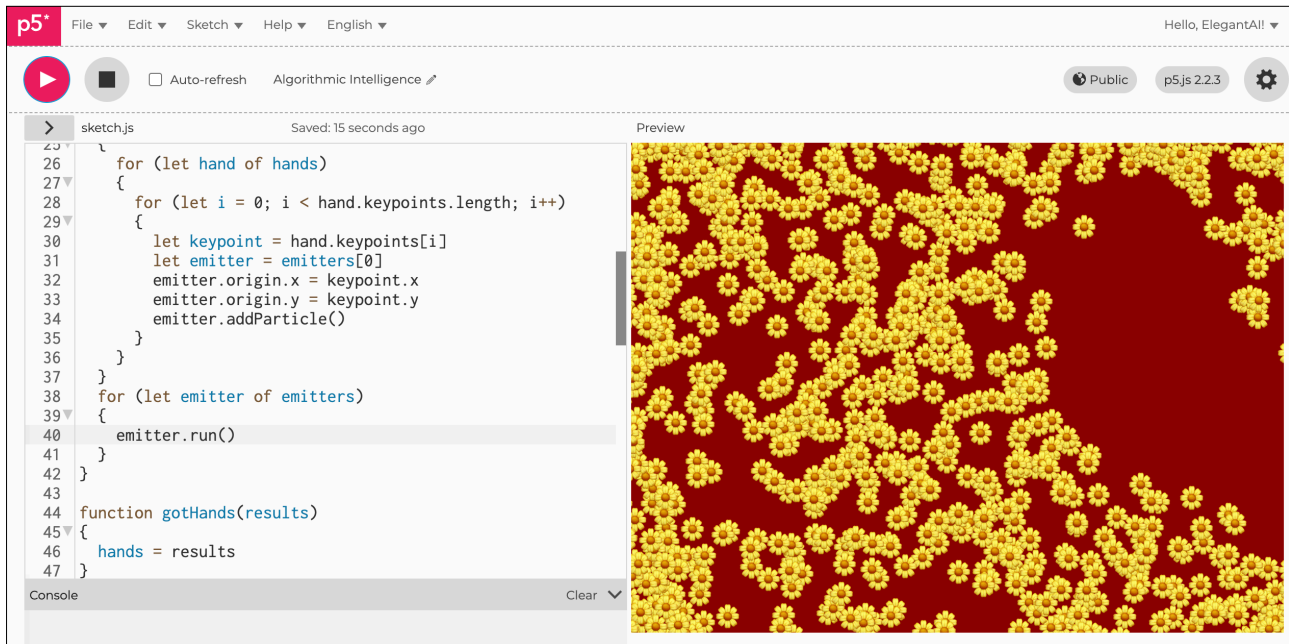
In the particle class, we do something which might seem a bit mystifying. We have a **for()** loop that works backwards through the particles array. There is a reason for this, which will become apparent shortly.

Code Explanation

```
for (let i = this.particles.length - 1; i > 0; i--)
```

This starts with the end of the array and works backwards through it

Figure H4.7





Sketch H4.8 is it alive?

As you can see, it produces many particles, and if they aren't culled, they will slow down the sketch very quickly. So we need a lifespan for the particle so that it doesn't hang around forever. Hence, the reverse `for()` loop in the particle class. We need a function and a variable to handle this.

```
let video
let handPose
let hands = []
let emitters = []

async function setup()
{
  createCanvas(640, 480)
  ml5.setBackend("webgl")
  handPose = await ml5.handPose({flipped: true})
  video = await createCapture(VIDEO, {flipped: true})
  video.hide()
  handPose.detectStart(video, gotHands)
  for (let i = 0; i < 21; i++)
  {
    emitters.push(new Emitter(width/2, height/2))
  }
}

function draw()
{
  background('darkred')
  if (hands.length > 0)
  {
    for (let hand of hands)
    {
      for (let i = 0; i < hand.keypoints.length; i++)
      {
        let keypoint = hand.keypoints[i]
        let emitter = emitters[0]
        emitter.origin.x = keypoint.x
        emitter.origin.y = keypoint.y
```

```

        emitter.addParticle()
    }
}
for (let emitter of emitters)
{
    emitter.run()
}
}

function gotHands(results)
{
    hands = results
}

class Emitter
{
    constructor(x, y)
    {
        this.origin = createVector(x, y)
        this.particles = []
    }

    addParticle()
    {
        this.particles.push(new Particle(this.origin.x, this.origin.y))
    }

    run()
    {
        for (let i = this.particles.length - 1; i > 0; i--)
        {
            this.particles[i].run()
            if (this.particles[i].expired())
            {
                this.particles.splice(i, 1)
            }
        }
    }
}

```

```

    }
}

class Particle
{
    constructor(x, y)
    {
        this.position = createVector(x, y)
        this.acceleration = createVector(0, 0)
        this.velocity = p5.Vector.random2D()
        this.icon = '🌻'
        this.lifespan = 255
    }

    run()
    {
        this.show()
        this.move()
    }

    show()
    {
        textSize(24)
        textAlign(CENTER)
        noStroke()
        text(this.icon, this.position.x, this.position.y)
    }

    move()
    {
        this.velocity.add(this.acceleration)
        this.position.add(this.velocity)
        this.acceleration.mult(0)
        this.lifespan -= 12
    }

    expired()
    {

```

```
    return this.lifespan < 0
  }
}
```

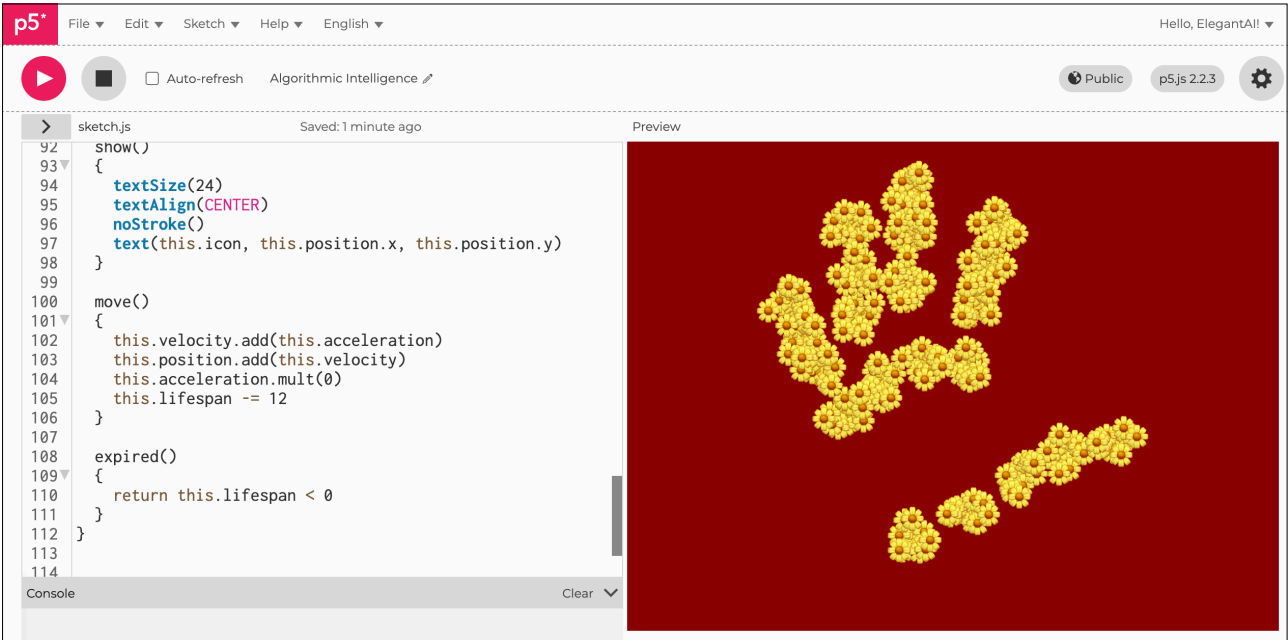
Notes

This improves the performance of the sketch no end.

Code Explanation

<code>if (this.particles[i].expired())</code>	Checks each particle to see if it has expired
<code>this.particles.splice(i, 1)</code>	When that particle has expired we remove it from the array
<code>this.lifespan = 255</code>	Our starting lifespan, the reason we use 255 is so that we can gently remove it rather than just suddenly eliminating it.
<code>this.lifespan -= 12</code>	Every iteration we reduce the lifespan by 12
<code>return this.lifespan < 0</code>	Think of this as an if statement, if the lifespan is less than zero, then return the expired() function as true for each particle.

Figure H4.8





Sketch H4.9 tweak

One final tweak to make it look a lot less sudden: we can reduce the alpha by how much lifespan it has before it is eliminated.

```
let video
let handPose
let hands = []
let emitters = []

async function setup()
{
  createCanvas(640, 480)
  ml5.setBackend("webgl")
  handPose = await ml5.handPose({flipped: true})
  video = await createCapture(VIDEO, {flipped: true})
  video.hide()
  handPose.detectStart(video, gotHands)
  for (let i = 0; i < 21; i++)
  {
    emitters.push(new Emitter(width/2, height/2))
  }
}

function draw()
{
  background('darkred')
  if (hands.length > 0)
  {
    for (let hand of hands)
    {
      for (let i = 0; i < hand.keypoints.length; i++)
      {
        let keypoint = hand.keypoints[i]
        let emitter = emitters[0]
        emitter.origin.x = keypoint.x
        emitter.origin.y = keypoint.y
        emitter.addParticle()
      }
    }
  }
}
```

```

    }
  }
  for (let emitter of emitters)
  {
    emitter.run()
  }
}

function gotHands(results)
{
  hands = results
}

class Emitter
{
  constructor(x, y)
  {
    this.origin = createVector(x, y)
    this.particles = []
  }

  addParticle()
  {
    this.particles.push(new Particle(this.origin.x, this.origin.y))
  }

  run()
  {
    for (let i = this.particles.length - 1; i > 0; i--)
    {
      this.particles[i].run()
      if (this.particles[i].expired())
      {
        this.particles.splice(i, 1)
      }
    }
  }
}

```

```

class Particle
{
  constructor(x, y)
  {
    this.position = createVector(x, y)
    this.acceleration = createVector(0, 0)
    this.velocity = p5.Vector.random2D()
    this.icon = '🌻'
    this.lifespan = 255
  }

  run()
  {
    this.show()
    this.move()
  }

  show()
  {
    textSize(24)
    textAlign(CENTER)
    noStroke()
    fill(255, this.lifespan)
    text(this.icon, this.position.x, this.position.y)
  }

  move()
  {
    this.velocity.add(this.acceleration)
    this.position.add(this.velocity)
    this.acceleration.mult(0)
    this.lifespan -= 12
  }

  expired()
  {
    return this.lifespan < 0
  }
}

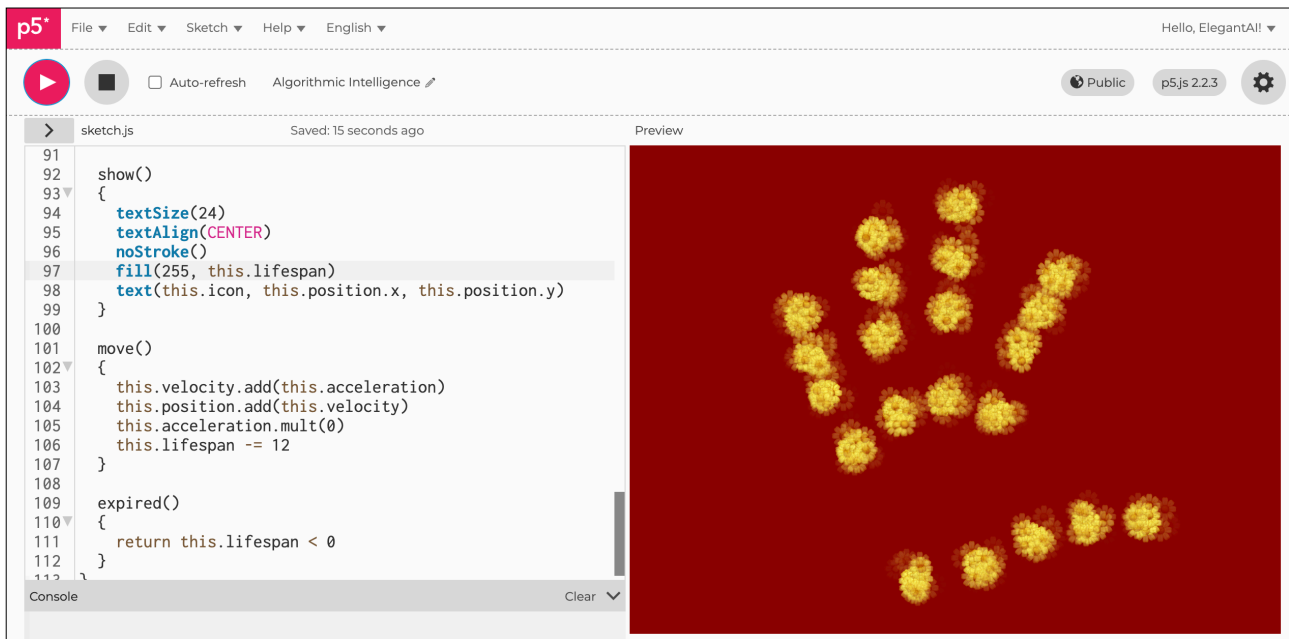
```

```
}  
}
```

Notes

A far more pleasing result, I think you will agree.

Figure H4.9





Algorithmic
Art

Module H

Unit #5

pretrained
faceMesh



Module H Unit #5 faceMesh

Using the faceMesh pre-trained model, we can create some fun stuff with your face. We will be adding an image to your face, so upload one to your sketch. In my case, I have uploaded an image of some clouds from Unsplash and superimposed it onto my face.

As you can see the code is very similar to the handPose model which makes life so much easier and why ml5.js is a great place to start.



Sketch H5.1 index.html

Just a quick reminder to have the ml5.js library listed in the index.html file.

```
<!DOCTYPE html>
<html lang="en"><head>
  <script src="https://cdn.jsdelivr.net/npm/p5@2.2.3/lib/p5.js"></script>
  <script src="https://unpkg.com/ml5@1/dist/ml5.min.js"></script>
  <link rel="stylesheet" type="text/css" href="style.css">
  <meta charset="utf-8">
</head>
<body>
  <main>
  </main>
  <script src="sketch.js"></script>
</body></html>
```

Notes

Notice I am using p5.js version **2.2.3**. I recommend that you use that version too, if not already the default.



Sketch H5.2 our starting sketch

! Please notice that we are using **WEBGL** as our rendering for this unit. If you use it with text you will get an error, hence why we didn't use it with the handPose model.

This is very similar to our hand sketch. Eliminate what needs eliminating and change what needs changing, or just start again, whichever is easiest. Basically get rid of all the classes (and references) and the **draw()** function.

```
let video
let faceMesh
let faces = []

async function setup()
{
  createCanvas(640, 480, WEBGL)
  ml5.setBackend("webgl")
  faceMesh = await ml5.faceMesh({ flipped: true })
  video = createCapture(VIDEO, { flipped: true })
  video.hide()
  faceMesh.detectStart(video, gotFaces)
}

function gotFaces(results)
{
  faces = results
}
```

Notes

Nothing to see yet.



Sketch H5.3 points of view

We now add the `draw()` function back in, here we want to draw a pixel `point()` at each data or `keypoint`. The video will be commented out. If you want to see your face and the points on your face, then just uncomment it.

```
let video
let faceMesh
let faces = []

async function setup()
{
  createCanvas(640, 480, WEBGL)
  ml5.setBackend("webgl")
  faceMesh = await ml5.faceMesh({ flipped: true })
  video = createCapture(VIDEO, { flipped: true })
  video.hide()
  faceMesh.detectStart(video, gotFaces)
}

function gotFaces(results)
{
  faces = results
}

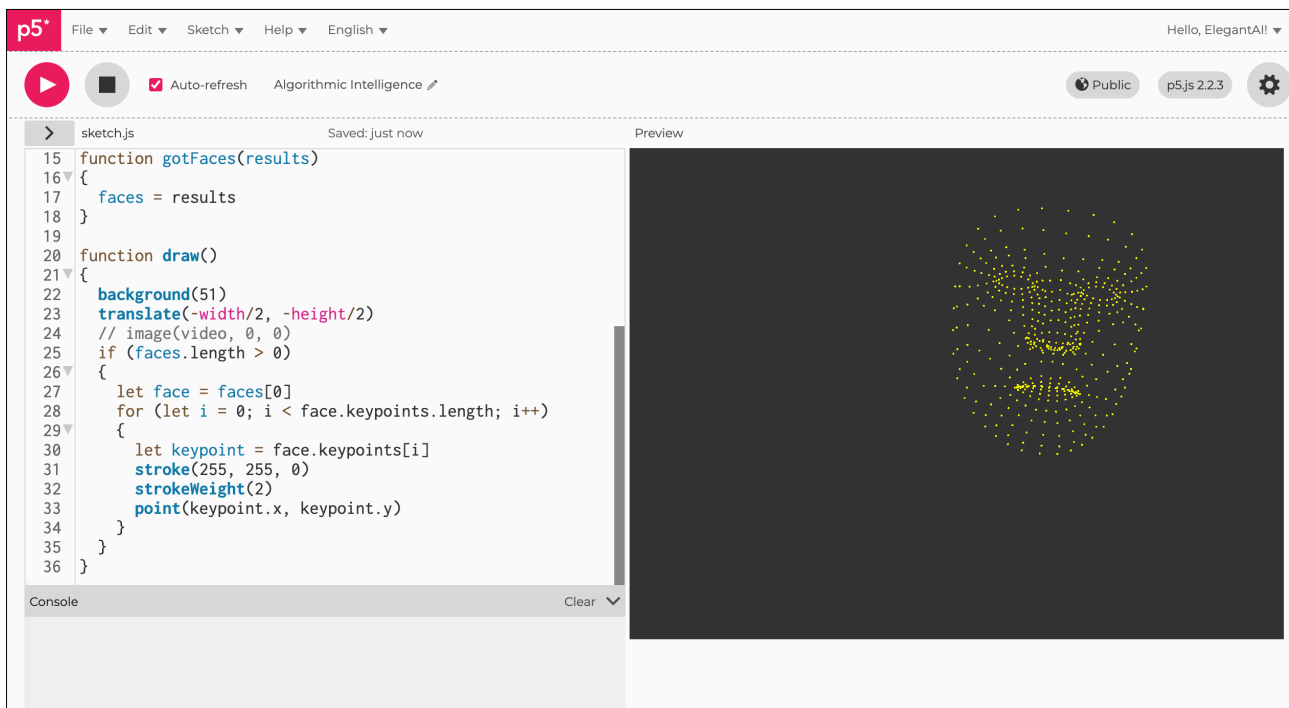
function draw()
{
  background(51)
  translate(-width/2, -height/2)
  if (faces.length > 0)
  {
    let face = faces[0]
    for (let i = 0; i < face.keypoints.length; i++)
    {
      let keypoint = face.keypoints[i]
      stroke(255, 255, 0)
      strokeWeight(2)
      point(keypoint.x, keypoint.y)
    }
  }
}
```

```
}  
}
```

Notes

I have run through this very quickly as it is so similar to the handPose models. There is plenty of interesting and fun stuff we can do with this model.

Figure H5.3





Sketch H5.4 step back

! If you want to see everything superimposed onto your face then replace the background with the `image(video, 0, 0)` line of code otherwise ignore that line.

Remove the bulk of the code in draw. We want to draw the lips. So we need all the points that refer to the inner and outer lips in order. We add this as an array for each set.

```
let video
let faceMesh
let faces = []

async function setup()
{
  createCanvas(640, 480, WEBGL)
  ml5.setBackend("webgl")
  faceMesh = await ml5.faceMesh({ flipped: true })
  video = createCapture(VIDEO, { flipped: true })
  video.hide()
  faceMesh.detectStart(video, gotFaces)
}

function gotFaces(results)
{
  faces = results
}

function draw()
{
  background(51)
  translate(-width/2, -height/2)
  // image(video, 0, 0)
  if (faces.length > 0)
  {
    let face = faces[0]
  }
}
```

```
let lipsExterior = [267, 269, 270, 409, 291, 375, 321, 405, 314, 17, 84, 181,  
91, 146, 61, 185, 40, 39, 37, 0]  
let lipsInterior = [13, 312, 311, 310, 415, 308, 324, 318, 402, 317, 14, 87,  
178, 88, 95, 78, 191, 80, 81, 82]
```

Notes

Somebody has very kindly created the lips arrays for us. I didn't do it. Whoever did that, thank you.



Sketch H5.5 drawing outer lips

Using the outer lips array, we cycle through them and draw the vertex at that data point.

```
let video
let faceMesh
let faces = []

async function setup()
{
  createCanvas(640, 480, WEBGL)
  ml5.setBackend("webgl")
  faceMesh = await ml5.faceMesh({ flipped: true })
  video = createCapture(VIDEO, { flipped: true })
  video.hide()
  faceMesh.detectStart(video, gotFaces)
}

function gotFaces(results)
{
  faces = results
}

function draw()
{
  background(51)
  translate(-width/2, -height/2)
  if (faces.length > 0)
  {
    let face = faces[0]
    beginShape()
    for (let i = 0; i < lipsExterior.length; i++)
    {
      let index = lipsExterior[i]
      let keypoint = face.keypoints[index]
      stroke('red')
      strokeWeight(2)
      noFill()
```

```

    vertex(keypoint.x, keypoint.y)
  }
endShape(CLOSE)
}
}

let lipsExterior = [267, 269, 270, 409, 291, 375, 321, 405, 314, 17, 84, 181,
91, 146, 61, 185, 40, 39, 37, 0]

let lipsInterior = [13, 312, 311, 310, 415, 308, 324, 318, 402, 317, 14, 87,
178, 88, 95, 78, 191, 80, 81, 82]

```

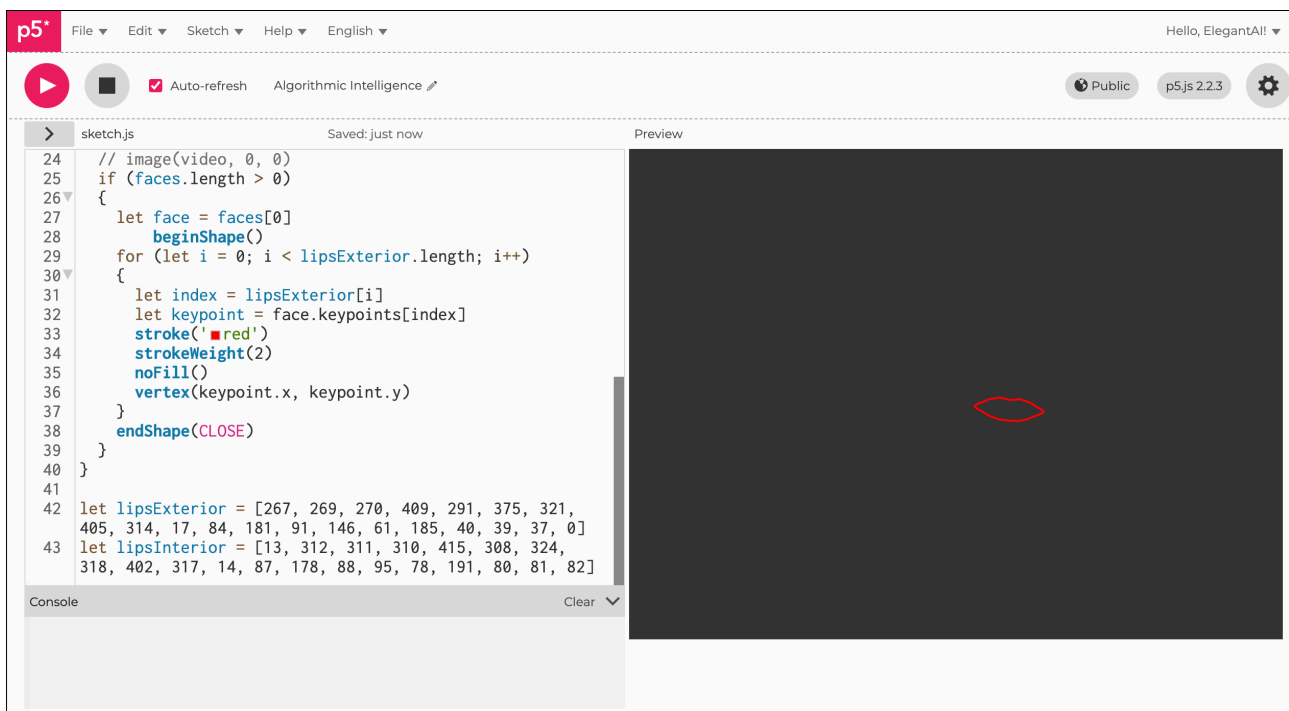
Notes

You should have your lips displayed and moving in sync.

Challenges

1. Change the colour
2. Add your face

Figure H5.5





Sketch H5.6 and the inner lips

The same again. This time cycle through the inner lips array, drawing the vertex.

```
let video
let faceMesh
let faces = []

async function setup()
{
  createCanvas(640, 480, WEBGL)
  ml5.setBackend("webgl")
  faceMesh = await ml5.faceMesh({ flipped: true })
  video = createCapture(VIDEO, { flipped: true })
  video.hide()
  faceMesh.detectStart(video, gotFaces)
}

function gotFaces(results)
{
  faces = results
}

function draw()
{
  background(51)
  translate(-width/2, -height/2)
  // image(video, 0, 0)
  if (faces.length > 0)
  {
    let face = faces[0]
    beginShape()
    for (let i = 0; i < lipsExterior.length; i++)
    {
      let index = lipsExterior[i]
      let keypoint = face.keypoints[index]
      stroke('red')
      strokeWeight(2)
      noFill()
```

```

    vertex(keypoint.x, keypoint.y)
  }
  endShape(CLOSE)
  beginShape()
  for (let i = 0; i < lipsInterior.length; i++)
  {
    let index = lipsInterior[i]
    let keypoint = face.keypoints[index]
    stroke('white')
    strokeWeight(2)
    noFill()
    vertex(keypoint.x, keypoint.y)
  }
  endShape(CLOSE)
}
}

let lipsExterior = [267, 269, 270, 409, 291, 375, 321, 405, 314, 17, 84, 181,
91, 146, 61, 185, 40, 39, 37, 0]
let lipsInterior = [13, 312, 311, 310, 415, 308, 324, 318, 402, 317, 14, 87,
178, 88, 95, 78, 191, 80, 81, 82]

```

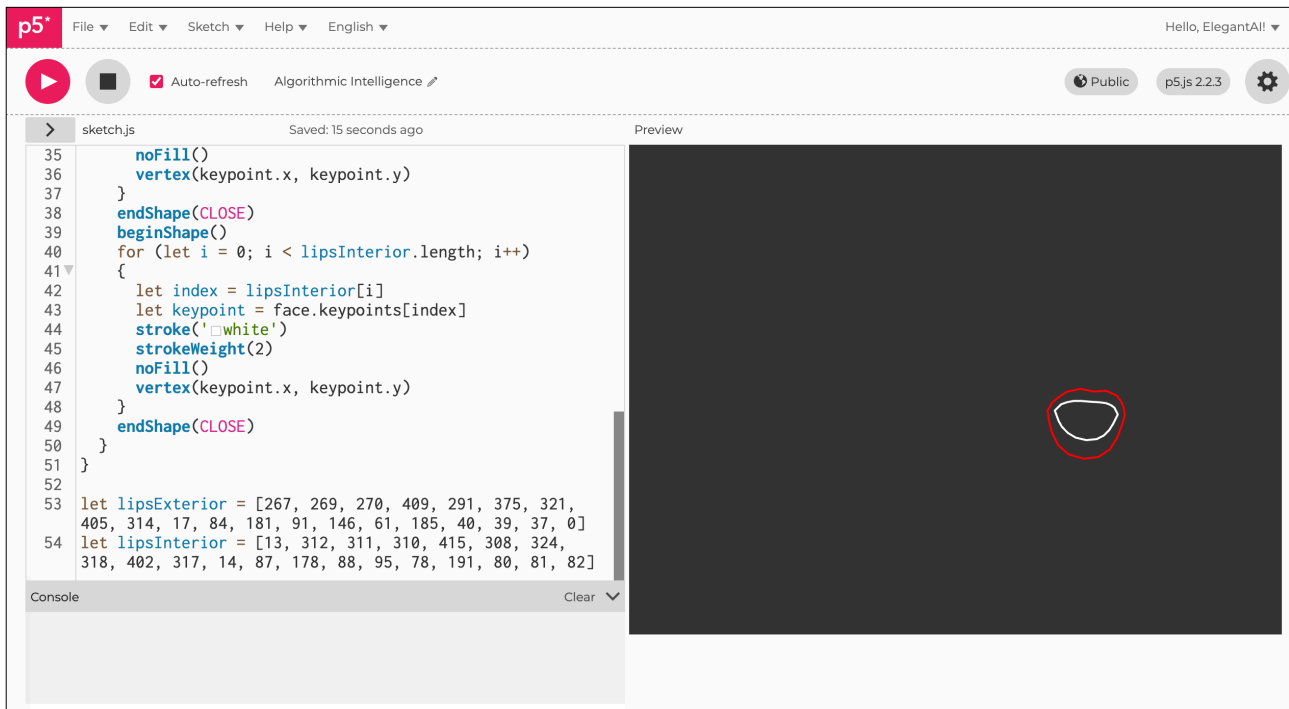
Notes

Should now have two sets of lips moving in sync.

Challenge

Can you think of a way to fill in between the inner and outer lips?

Figure H5.6





Sketch H5.7 lip sync

We are going to add a nose (simple circle) that expands when you open and close your mouth. Why not? We need to work out when you are opening and closing your mouth.

```
let video
let faceMesh
let faces = []

async function setup()
{
  createCanvas(640, 480, WEBGL)
  ml5.setBackend("webgl")
  faceMesh = await ml5.faceMesh({ flipped: true })
  video = createCapture(VIDEO, { flipped: true })
  video.hide()
  faceMesh.detectStart(video, gotFaces)
}

function gotFaces(results)
{
  faces = results
}

function draw()
{
  background(51)
  translate(-width/2, -height/2)
  // image(video, 0, 0)
  if (faces.length > 0)
  {
    let face = faces[0]
    beginShape()
    for (let i = 0; i < lipsExterior.length; i++)
    {
      let index = lipsExterior[i]
      let keypoint = face.keypoints[index]
      stroke('red')
      strokeWeight(2)
```

```

    noFill()
    vertex(keypoint.x, keypoint.y)
  }
  endShape(CLOSE)
  beginShape()
  for (let i = 0; i < lipsInterior.length; i++)
  {
    let index = lipsInterior[i]
    let keypoint = face.keypoints[index]
    stroke('white')
    strokeWeight(2)
    noFill()
    vertex(keypoint.x, keypoint.y)
  }
  endShape(CLOSE)

  let a = face.keypoints[13]
  let b = face.keypoints[14]
  let d = dist(a.x, a.y, b.x, b.y)
}
}

let lipsExterior = [267, 269, 270, 409, 291, 375, 321, 405, 314, 17, 84, 181,
91, 146, 61, 185, 40, 39, 37, 0]
let lipsInterior = [13, 312, 311, 310, 415, 308, 324, 318, 402, 317, 14, 87,
178, 88, 95, 78, 191, 80, 81, 82]

```

Notes

Working out the distance between point 13 and 14 for the interior lips. Nothing new happening yet.



Sketch H5.8 now the nose

We are going to add a new nose, nothing glamorous.

```
let video
let faceMesh
let faces = []

async function setup()
{
  createCanvas(640, 480, WEBGL)
  ml5.setBackend("webgl")
  faceMesh = await ml5.faceMesh({ flipped: true })
  video = createCapture(VIDEO, { flipped: true })
  video.hide()
  faceMesh.detectStart(video, gotFaces)
}

function gotFaces(results)
{
  faces = results
}

function draw()
{
  background(51)
  translate(-width/2, -height/2)
  if (faces.length > 0)
  {
    let face = faces[0]
    beginShape()
    for (let i = 0; i < lipsExterior.length; i++)
    {
      let index = lipsExterior[i]
      let keypoint = face.keypoints[index]
      stroke('red')
      strokeWeight(2)
      noFill()
      vertex(keypoint.x, keypoint.y)
```

```

    }
    endShape(CLOSE)
    beginShape()
    for (let i = 0; i < lipsInterior.length; i++)
    {
        let index = lipsInterior[i]
        let keypoint = face.keypoints[index]
        stroke('white')
        strokeWeight(2)
        noFill()
        vertex(keypoint.x, keypoint.y)
    }
    endShape(CLOSE)
    let a = face.keypoints[13]
    let b = face.keypoints[14]
    let d = dist(a.x, a.y, b.x, b.y)
    noStroke()
    let nose = face.keypoints[19]
    fill('yellow')
    circle(nose.x, nose.y, d)
}
}

let lipsExterior = [267, 269, 270, 409, 291, 375, 321, 405, 314, 17, 84, 181,
91, 146, 61, 185, 40, 39, 37, 0]

let lipsInterior = [13, 312, 311, 310, 415, 308, 324, 318, 402, 317, 14, 87,
178, 88, 95, 78, 191, 80, 81, 82]

```

Notes

Looks even better with the image of you being displayed.

Challenge

Could be a clown nose.



Blowing Bubbles

We are now going to blow bubbles through our mouths. Rather, we are going to emit bubbles from our mouths when we open our mouths. This has many similarities to the emoji emitter in the previous unit. We will need an emitter class and a particle class.



Sketch H5.9 adding the emitter

! Here we remove the nose, you can keep it if you want to.

We want to have bubbles emit from your mouth as you open and close your mouth. Similar to the emoji emitter with handPose. So the emitter class is almost identical, except that the particle now has a **d** value for the diameter.

```
let video
let faceMesh
let faces = []

async function setup()
{
  createCanvas(640, 480, WEBGL)
  ml5.setBackend("webgl")
  faceMesh = await ml5.faceMesh({ flipped: true })
  video = createCapture(VIDEO, { flipped: true })
  video.hide()
  faceMesh.detectStart(video, gotFaces)
}

function gotFaces(results)
{
  faces = results
}

function draw()
{
  background(51)
  translate(-width/2, -height/2)
  if (faces.length > 0)
  {
    let face = faces[0]
    beginShape()
    for (let i = 0; i < lipsExterior.length; i++)
    {
      let index = lipsExterior[i]
      let keypoint = face.keypoints[index]
      stroke('red')
```

```

    strokeWeight(2)
    noFill()
    vertex(keypoint.x, keypoint.y)
  }
  endShape(CLOSE)
  beginShape()
  for (let i = 0; i < lipsInterior.length; i++)
  {
    let index = lipsInterior[i]
    let keypoint = face.keypoints[index]
    stroke('white')
    strokeWeight(2)
    noFill()
    vertex(keypoint.x, keypoint.y)
  }
  endShape(CLOSE)
  let a = face.keypoints[13]
  let b = face.keypoints[14]
  let d = dist(a.x, a.y, b.x, b.y)
  // noStroke()
  // let nose = face.keypoints[19]
  // fill('yellow')
  // circle(nose.x, nose.y, d)
}
}

let lipsExterior = [267, 269, 270, 409, 291, 375, 321, 405, 314, 17, 84, 181,
91, 146, 61, 185, 40, 39, 37, 0]
let lipsInterior = [13, 312, 311, 310, 415, 308, 324, 318, 402, 317, 14, 87,
178, 88, 95, 78, 191, 80, 81, 82]

class Emitter
{
  constructor(x, y)
  {
    this.origin = createVector(x, y)
    this.particles = []
  }
}

```

```
addParticle(x, y, d)
{
  this.particles.push(new Particle(x, y, d))
}

run()
{
  for (let i = this.particles.length - 1; i >= 0; i--)
  {
    this.particles[i].run()
    if (this.particles[i].expired())
    {
      this.particles.splice(i, 1)
    }
  }
}
}
```

Notes

The emitter references the particle.



Sketch H5.10 adding the particle

Again, this is very similar to the previous unit, so I'm not going to go through all of it in detail. We will be adding gravity as a force shortly.

```
let video
let faceMesh
let faces = []

async function setup()
{
  createCanvas(640, 480, WEBGL)
  ml5.setBackend("webgl")
  faceMesh = await ml5.faceMesh({ flipped: true })
  video = createCapture(VIDEO, { flipped: true })
  video.hide()
  faceMesh.detectStart(video, gotFaces)
}

function gotFaces(results)
{
  faces = results
}

function draw()
{
  background(51)
  translate(-width/2, -height/2)
  if (faces.length > 0)
  {
    let face = faces[0]
    beginShape()
    for (let i = 0; i < lipsExterior.length; i++)
    {
      let index = lipsExterior[i]
      let keypoint = face.keypoints[index]
      stroke('red')
      strokeWeight(2)
      noFill()
```

```

    vertex(keypoint.x, keypoint.y)
  }
  endShape(CLOSE)
  beginShape()
  for (let i = 0; i < lipsInterior.length; i++)
  {
    let index = lipsInterior[i]
    let keypoint = face.keypoints[index]
    stroke('white')
    strokeWeight(2)
    noFill()
    vertex(keypoint.x, keypoint.y)
  }
  endShape(CLOSE)
  let a = face.keypoints[13]
  let b = face.keypoints[14]
  let d = dist(a.x, a.y, b.x, b.y)
}
}

let lipsExterior = [267, 269, 270, 409, 291, 375, 321, 405, 314, 17, 84, 181,
91, 146, 61, 185, 40, 39, 37, 0]
let lipsInterior = [13, 312, 311, 310, 415, 308, 324, 318, 402, 317, 14, 87,
178, 88, 95, 78, 191, 80, 81, 82]

class Emitter
{
  constructor(x, y)
  {
    this.origin = createVector(x, y)
    this.particles = []
  }

  addParticle(x, y, d)
  {
    this.particles.push(new Particle(x, y, d))
  }

  run()

```

```

{
  for (let i = this.particles.length - 1; i >= 0; i--)
  {
    this.particles[i].run()
    if (this.particles[i].expired())
    {
      this.particles.splice(i, 1)
    }
  }
}
}

```

```

class Particle
{
  constructor(x, y, d)
  {
    this.position = createVector(x, y)
    this.acceleration = createVector(0, 0)
    this.velocity = createVector(random(-2, 2), random(-2, 2))
    this.lifespan = 255
    this.r = d * 0.5
  }

  run()
  {
    this.move()
    this.show()
  }

  move()
  {
    this.velocity.add(this.acceleration)
    this.position.add(this.velocity)
    this.lifespan -= 2
    this.acceleration.mult(0)
  }

  show()

```

```
{
  stroke(255, this.lifespan)
  fill(255, this.lifespan * 0.5)
  strokeWeight(4)
  circle(this.position.x, this.position.y, this.r * 2)
}

expired()
{
  return this.lifespan < 0
}
}
```

Notes

The particle in this case is a simple circle. The fill is relative to half its lifespan.

Challenge

Use an emoji instead of the particle.



Sketch H5.11 now the bubbles

We need an emitter variable and create an emitter at some place initially. Then we calculate the halfway point for the x and y values for each bubble as your mouth opens, closes, and moves. The bubble only emits if the mouth is open more than 20 pixels. To have it slow down a bit, we use `random(1)` so that it emits just 75% of the time.

```
let video
let faceMesh
let faces = []
let emitter

async function setup()
{
  createCanvas(640, 480, WEBGL)
  ml5.setBackend("webgl")
  faceMesh = await ml5.faceMesh({ flipped: true })
  video = createCapture(VIDEO, { flipped: true })
  video.hide()
  faceMesh.detectStart(video, gotFaces)
  emitter = new Emitter(300, 300)
}

function gotFaces(results)
{
  faces = results
}

function draw()
{
  background(51)
  translate(-width/2, -height/2)
  if (faces.length > 0)
  {
    let face = faces[0]
    beginShape()
    for (let i = 0; i < lipsExterior.length; i++)
    {
      let index = lipsExterior[i]
```

```

    let keypoint = face.keypoints[index]
    stroke('red')
    strokeWeight(2)
    noFill()
    vertex(keypoint.x, keypoint.y)
  }
  endShape(CLOSE)
  beginShape()
  for (let i = 0; i < lipsInterior.length; i++)
  {
    let index = lipsInterior[i]
    let keypoint = face.keypoints[index]
    stroke('white')
    strokeWeight(2)
    noFill()
    vertex(keypoint.x, keypoint.y)
  }
  endShape(CLOSE)
  let a = face.keypoints[13]
  let b = face.keypoints[14]
  let d = dist(a.x, a.y, b.x, b.y)

  let x = (a.x + b.x) * 0.5
  let y = (a.y + b.y) * 0.5
  if (d > 20 && random(1) < 0.25)
  {
    emitter.addParticle(x, y, d)
  }
}
emitter.run()
}

let lipsExterior = [267, 269, 270, 409, 291, 375, 321, 405, 314, 17, 84, 181,
91, 146, 61, 185, 40, 39, 37, 0]
let lipsInterior = [13, 312, 311, 310, 415, 308, 324, 318, 402, 317, 14, 87,
178, 88, 95, 78, 191, 80, 81, 82]

class Emitter
{
  constructor(x, y)

```

```

{
  this.origin = createVector(x, y)
  this.particles = []
}

addParticle(x, y, d)
{
  this.particles.push(new Particle(x, y, d))
}

run()
{
  for (let i = this.particles.length - 1; i >= 0; i--)
  {
    this.particles[i].run()
    if (this.particles[i].expired())
    {
      this.particles.splice(i, 1)
    }
  }
}
}

class Particle
{
  constructor(x, y, d)
  {
    this.position = createVector(x, y)
    this.acceleration = createVector(0, 0)
    this.velocity = createVector(random(-2, 2), random(-2, 2))
    this.lifespan = 255
    this.r = d * 0.5
  }

  run()
  {
    this.move()
    this.show()
  }
}

```

```

}

move()
{
  this.velocity.add(this.acceleration)
  this.position.add(this.velocity)
  this.lifespan -= 2
  this.acceleration.mult(0)
}

show()
{
  stroke(255, this.lifespan)
  fill(255, this.lifespan * 0.5)
  strokeWeight(4)
  circle(this.position.x, this.position.y, this.r * 2)
}

expired()
{
  return this.lifespan < 0
}
}

```

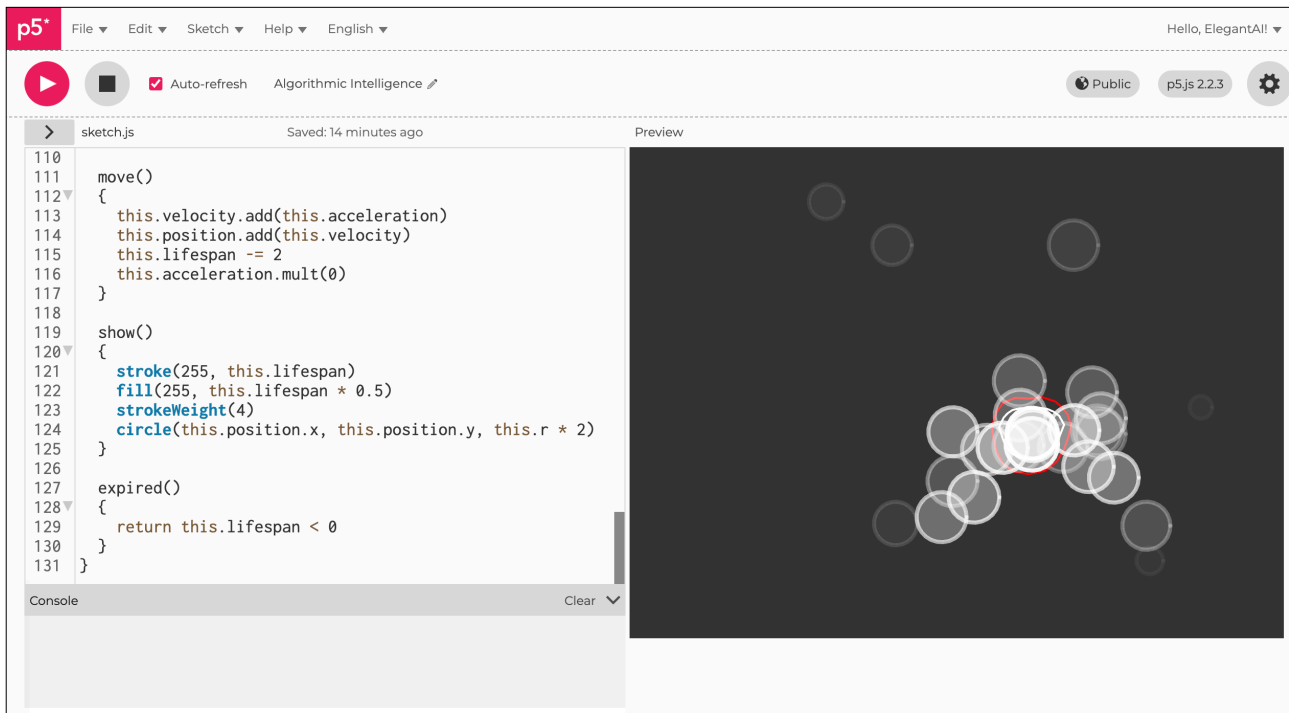
Notes

When the condition is met, we add a particle to the emitter and run the emitter. You should get bubbles coming out of your mouth. Notice that they go in all directions. You may notice a point on the circle. This is something to do with using WebGL as the render, but I have no idea why, it may be just a bug in this version of p5.js.

Challenge

You could do this without drawing the lips for a better effect.

Figure H5.11





Sketch H5.12 negative gravity

It would be better if the bubbles rose upwards. So let us add some negative gravity!

```
let video
let faceMesh
let faces = []
let emitter

async function setup()
{
  createCanvas(640, 480, WEBGL)
  ml5.setBackend("webgl")
  faceMesh = await ml5.faceMesh({ flipped: true })
  video = createCapture(VIDEO, { flipped: true })
  video.hide()
  faceMesh.detectStart(video, gotFaces)
  emitter = new Emitter(300, 300)
}

function gotFaces(results)
{
  faces = results
}

function draw()
{
  background(51)
  translate(-width/2, -height/2)
  if (faces.length > 0)
  {
    let face = faces[0]
    beginShape()
    for (let i = 0; i < lipsExterior.length; i++)
    {
      let index = lipsExterior[i]
      let keypoint = face.keypoints[index]
      stroke('red')
      strokeWeight(2)
```

```

    noFill()
    vertex(keypoint.x, keypoint.y)
  }
  endShape(CLOSE)
  beginShape()
  for (let i = 0; i < lipsInterior.length; i++)
  {
    let index = lipsInterior[i]
    let keypoint = face.keypoints[index]
    stroke('white')
    strokeWeight(2)
    noFill()
    vertex(keypoint.x, keypoint.y)
  }
  endShape(CLOSE)
  let a = face.keypoints[13]
  let b = face.keypoints[14]
  let d = dist(a.x, a.y, b.x, b.y)
  let x = (a.x + b.x) * 0.5
  let y = (a.y + b.y) * 0.5
  if (d > 20 && random(1) < 0.25)
  {
    emitter.addParticle(x, y, d)
  }
}
emitter.run()
}

let lipsExterior = [267, 269, 270, 409, 291, 375, 321, 405, 314, 17, 84, 181,
91, 146, 61, 185, 40, 39, 37, 0]
let lipsInterior = [13, 312, 311, 310, 415, 308, 324, 318, 402, 317, 14, 87,
178, 88, 95, 78, 191, 80, 81, 82]

class Emitter
{
  constructor(x, y)
  {
    this.origin = createVector(x, y)
    this.particles = []
  }
}

```

```

}

addParticle(x, y, d)
{
  this.particles.push(new Particle(x, y, d))
}

run()
{
  for (let i = this.particles.length - 1; i >= 0; i--)
  {
    this.particles[i].run()
    if (this.particles[i].expired())
    {
      this.particles.splice(i, 1)
    }
  }
}
}

class Particle
{
  constructor(x, y, d)
  {
    this.position = createVector(x, y)
    this.acceleration = createVector(0, 0)
    this.velocity = createVector(random(-2, 2), random(-2, 2))
    this.lifespan = 255
    this.r = d * 0.5
  }

  run()
  {
    let gravity = createVector(0, -0.05)
    this.applyForce(gravity)
    this.move()
    this.show()
  }
}

```

```
    applyForce(force)
  {
    this.acceleration.add(force)
  }

  move()
  {
    this.velocity.add(this.acceleration)
    this.position.add(this.velocity)
    this.lifespan -= 2
    this.acceleration.mult(0)
  }

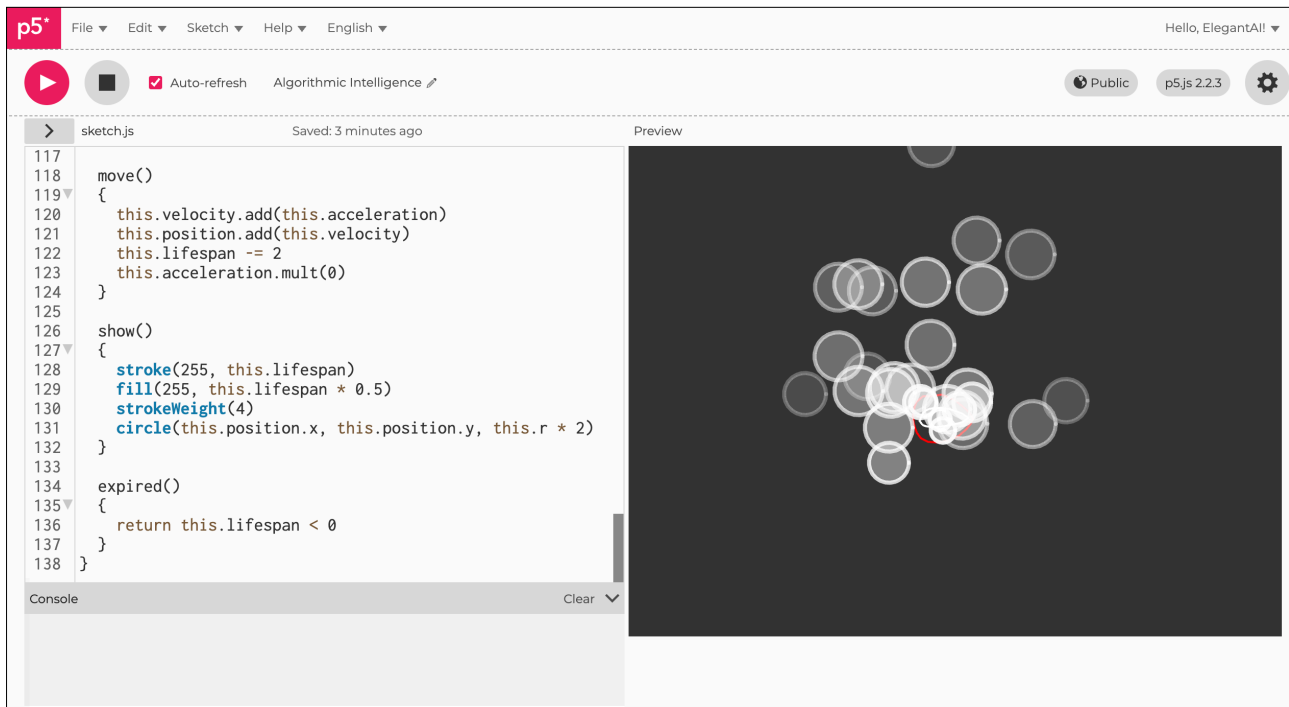
  show()
  {
    stroke(255, this.lifespan)
    fill(255, this.lifespan * 0.5)
    strokeWeight(4)
    circle(this.position.x, this.position.y, this.r * 2)
  }

  expired()
  {
    return this.lifespan < 0
  }
}
```

Notes

Looks really nice now.

Figure H5.12



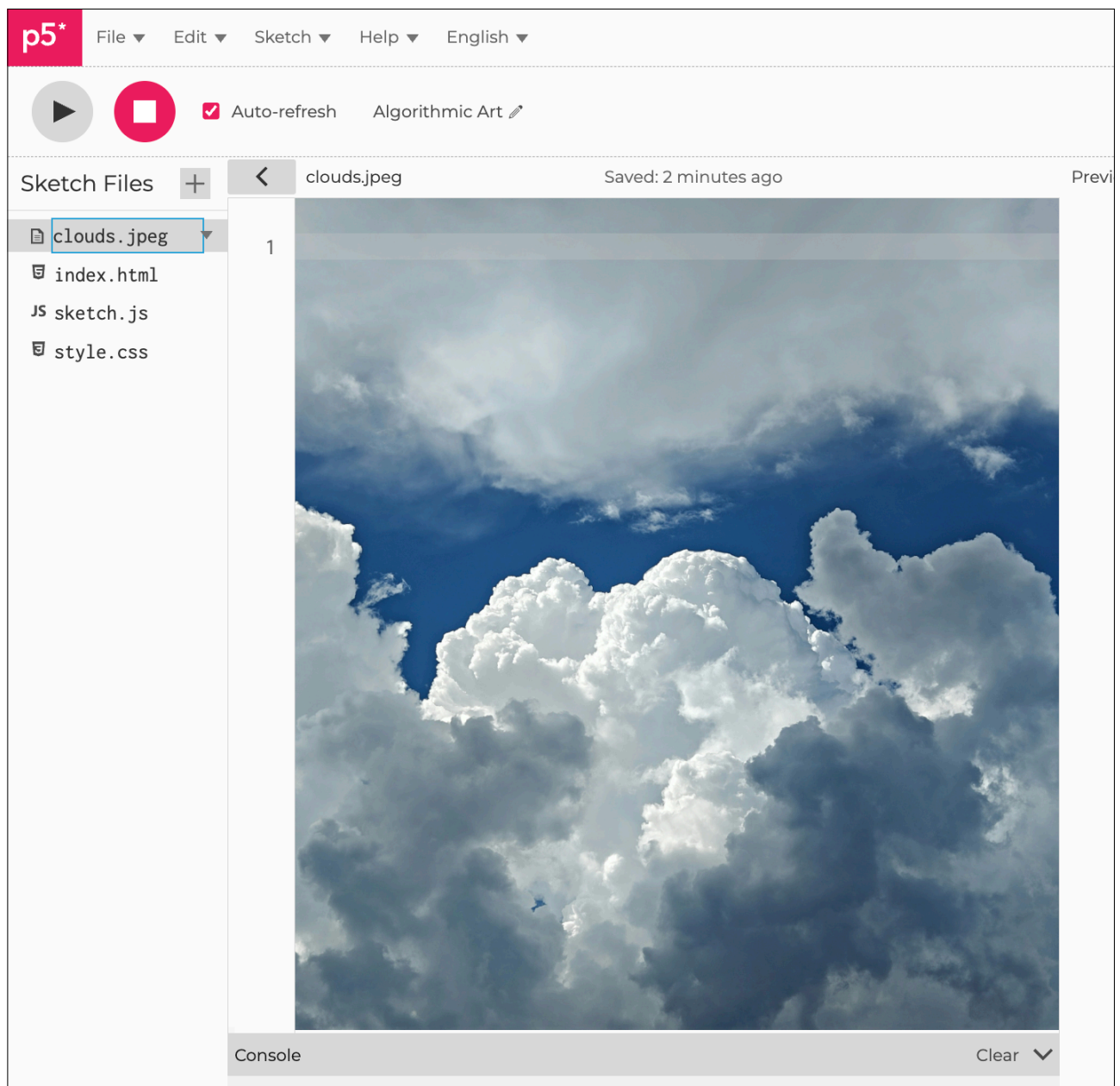


A face mask

We are going to create a mask to put on the faceMesh. For this we will need an image and do quite a bit of refactoring. This is all about triangles. I have a jpeg ready for you to download on my website, it is of some clouds.

You could upload your own photos, images and I would recommend that you try. In my example the image is clouds.jpeg:

Face mask clouds





Sketch H5.13 back to basics

Remove everything except leave what is shown below. We are removing all references to emitters and particles.

```
let video
let faceMesh
let faces = []

async function setup()
{
  createCanvas(640, 480, WEBGL)
  ml5.setBackend("webgl")
  faceMesh = await ml5.faceMesh({ flipped: true })
  video = createCapture(VIDEO, { flipped: true })
  video.hide()
  faceMesh.detectStart(video, gotFaces)
}

function gotFaces(results)
{
  faces = results
}

function draw()
{
  background(51)
  translate(-width/2, -height/2)
  if (faces.length > 0)
  {

  }
}
```

Notes

Just a blank canvas.



Sketch H5.14 triangles

One of the features of **faceMesh** is that it has a built-in function to get the triangles that join all the data points on the face. We create a variable called **triangles** to hold that data (console log to see it). There are three data points for each triangle. There are **852** triangles with a total of **2,556** data points.

```
let video
let faceMesh
let faces = []
let triangles

async function setup()
{
  createCanvas(640, 480, WEBGL)
  ml5.setBackend("webgl")
  faceMesh = await ml5.faceMesh({ flipped: true })
  video = createCapture(VIDEO, { flipped: true })
  video.hide()
  faceMesh.detectStart(video, gotFaces)
  triangles = faceMesh.getTriangles()
}

function gotFaces(results)
{
  faces = results
}

function draw()
{
  background(51)
  translate(-width/2, -height/2)
  if (faces.length > 0)
  {
    let face = faces[0]
  }
}
```

Notes

You might see it running slower in certain circumstances.



Sketch H5.15 showing the triangles

There is a lot happening here. The triangles hold the array of arrays (2556 data points) for each triangle. We cycle through each triangle, pull out the data points (a, b, c). We use `beginShape(TRIANGLES)` to draw these triangles as `vertex()`.

```
let video
let faceMesh
let faces = []
let triangles

async function setup()
{
  createCanvas(640, 480, WEBGL)
  ml5.setBackend("webgl")
  faceMesh = await ml5.faceMesh({ flipped: true })
  video = createCapture(VIDEO, { flipped: true })
  video.hide()
  faceMesh.detectStart(video, gotFaces)
  triangles = faceMesh.getTriangles()
}

function gotFaces(results)
{
  faces = results
}

function draw()
{
  background(51)
  translate(-width/2, -height/2)
  if (faces.length > 0)
  {
    let face = faces[0]
    beginShape(TRIANGLES)
    for (let i = 0; i < triangles.length; i++)
    {
      let tri = triangles[i]
      let [a, b, c] = tri
    }
  }
}
```

```

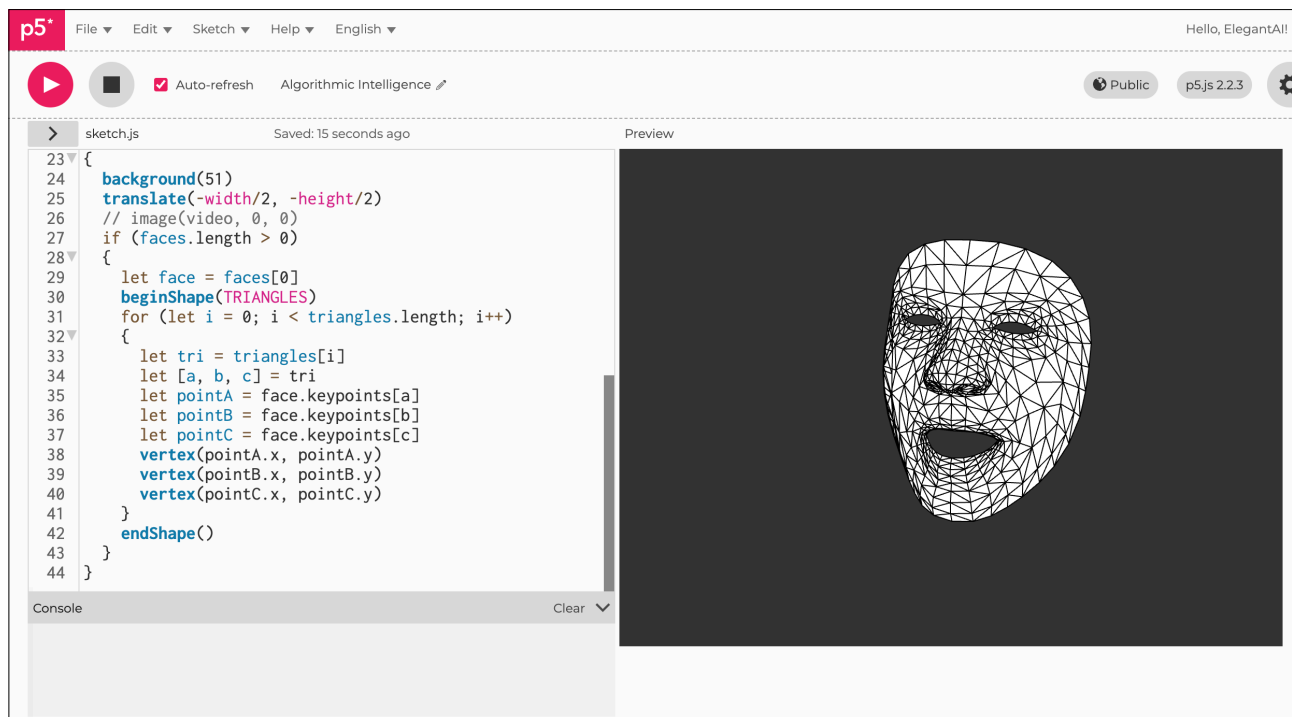
    let pointA = face.keypoints[a]
    let pointB = face.keypoints[b]
    let pointC = face.keypoints[c]
    vertex(pointA.x, pointA.y)
    vertex(pointB.x, pointB.y)
    vertex(pointC.x, pointC.y)
  }
  endShape()
}
}

```

Challenge

If you remove the triangle lines using `noStroke()`.

Figure H5.15





Sketch H5.16 adding a mask

! Upload an image and rename it in the file and sketch accordingly. I have uploaded a cloud image as shown in the fig below.

We are going to use an image; any image will do. In my case, I have downloaded an image of some clouds from [Unsplash](#), but you can use any image or your own. The mask has co-ordinates x, y, z , but these need to be mapped onto the image. The image is 2D, and the co-ordinates are called u and v , hence (u, v) . We create a variable to hold the image (`img`) and the u, v co-ordinates (`uvcoords`). There is a function in `faceMesh` to get those co-ordinates, called `getUVCoords()`.

```
let video
let faceMesh
let faces = []
let triangles

let uvcoords
let img

async function setup()
{
  createCanvas(640, 480, WEBGL)
  ml5.setBackend("webgl")
  faceMesh = await ml5.faceMesh({ flipped: true })
  img = await loadImage("clouds.jpeg")
  video = createCapture(VIDEO, { flipped: true })
  video.hide()
  faceMesh.detectStart(video, gotFaces)
  triangles = faceMesh.getTriangles()
  uvcoords = faceMesh.getUVCoords()
}

function gotFaces(results)
{
  faces = results
}

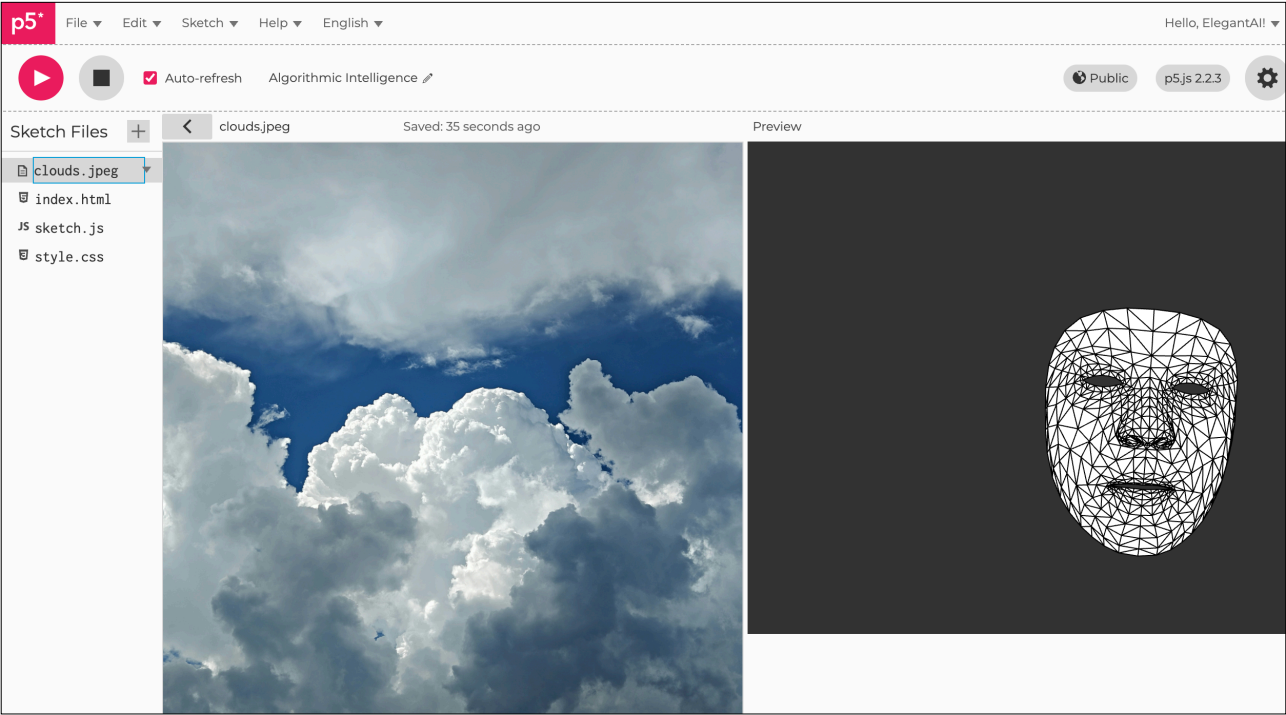
function draw()
{
  background(51)
```

```
translate(-width/2, -height/2)
if (faces.length > 0)
{
  let face = faces[0]
  beginShape(TRIANGLES)
  for (let i = 0; i < triangles.length; i++)
  {
    let tri = triangles[i]
    let [a, b, c] = tri
    let pointA = face.keypoints[a]
    let pointB = face.keypoints[b]
    let pointC = face.keypoints[c]
    vertex(pointA.x, pointA.y)
    vertex(pointB.x, pointB.y)
    vertex(pointC.x, pointC.y)
  }
  endShape()
}
}
```

Notes

This is just the beginning and is straightforward. You can use any image of anything; I have just used clouds for effect.

Figure H5.16





Sketch H5.17 texture mapping

We now apply texture mapping to the detected face mesh.

```
let video
let faceMesh
let faces = []
let triangles
let uvcoords
let img

async function setup()
{
  createCanvas(640, 480, WEBGL)
  ml5.setBackend("webgl")
  faceMesh = await ml5.faceMesh({ flipped: true })
  img = await loadImage("clouds.jpeg")
  video = createCapture(VIDEO, { flipped: true })
  video.hide()
  faceMesh.detectStart(video, gotFaces)
  triangles = faceMesh.getTriangles()
  uvcoords = faceMesh.getUVCoords()
}

function gotFaces(results)
{
  faces = results
}

function draw()
{
  background(51)
  translate(-width/2, -height/2)
  if (faces.length > 0)
  {
    let face = faces[0]
    texture(img)
    textureMode(NORMAL)
    noStroke()
```

```

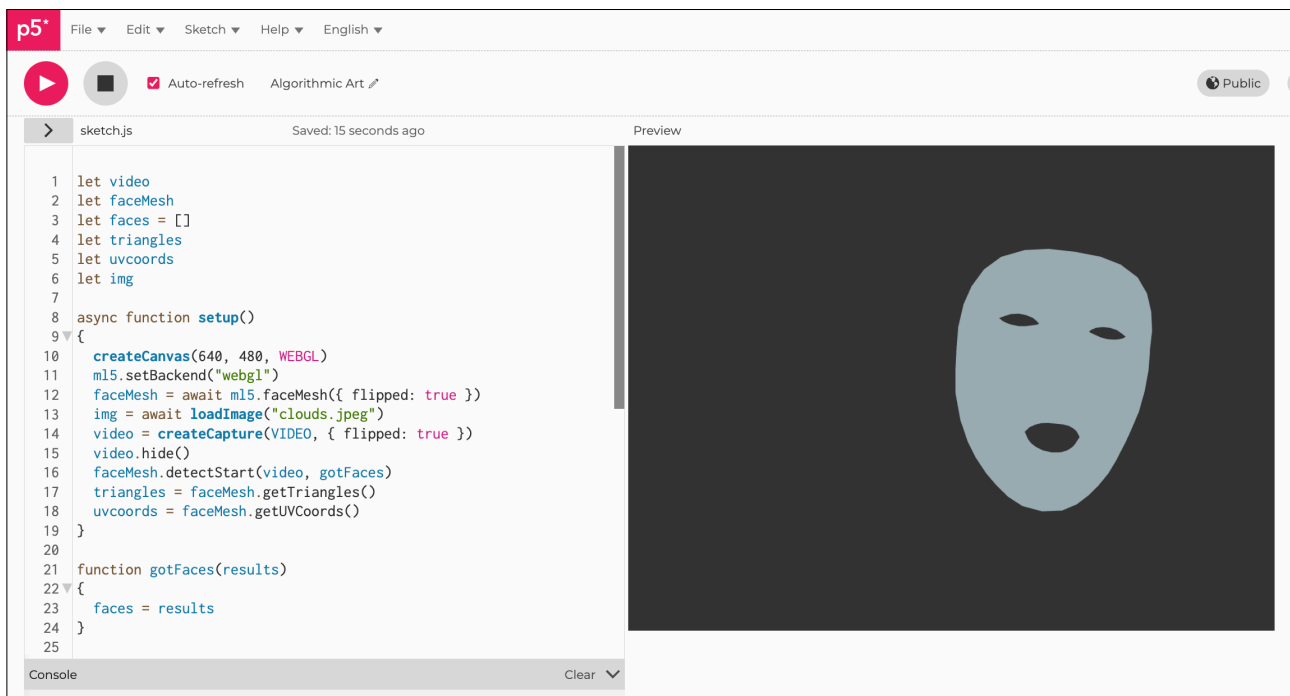
beginShape(TRIANGLES)
for (let i = 0; i < triangles.length; i++)
{
  let tri = triangles[i]
  let [a, b, c] = tri
  let pointA = face.keypoints[a]
  let pointB = face.keypoints[b]
  let pointC = face.keypoints[c]
  vertex(pointA.x, pointA.y)
  vertex(pointB.x, pointB.y)
  vertex(pointC.x, pointC.y)
}
endShape()
}
}

```

Notes

All we get is a blue/green mask.

Figure H5.17





Sketch H5.18 mapping uv coordinates

Retrieve the corresponding **UV** coordinates for texture mapping and then define the triangle with both position (**x**, **y**) and **UV** texture coordinates.

```
let video
let faceMesh
let faces = []
let triangles
let uvcoords
let img

async function setup()
{
  createCanvas(640, 480, WEBGL)
  ml5.setBackend("webgl")
  faceMesh = await ml5.faceMesh({ flipped: true })
  img = await loadImage("clouds.jpeg")
  video = createCapture(VIDEO, { flipped: true })
  video.hide()
  faceMesh.detectStart(video, gotFaces)
  triangles = faceMesh.getTriangles()
  uvcoords = faceMesh.getUVCoords()
}

function gotFaces(results)
{
  faces = results
}

function draw()
{
  background(51)
  translate(-width/2, -height/2)
  if (faces.length > 0)
  {
    let face = faces[0]
    texture(img)
    textureMode(NORMAL)
```

```
noStroke()
beginShape(TRIANGLES)
for (let i = 0; i < triangles.length; i++)
{
  let tri = triangles[i]
  let [a, b, c] = tri
  let pointA = face.keypoints[a]
  let pointB = face.keypoints[b]
  let pointC = face.keypoints[c]

  let uvA = uvcoords[a]
  let uvB = uvcoords[b]
  let uvC = uvcoords[c]
  vertex(pointA.x, pointA.y, uvA[0], uvA[1])
  vertex(pointB.x, pointB.y, uvB[0], uvB[1])
  vertex(pointC.x, pointC.y, uvC[0], uvC[1])
}
endShape()
}
```

Notes

We have a cloud mask, how cool is that!

Figure H5.18

