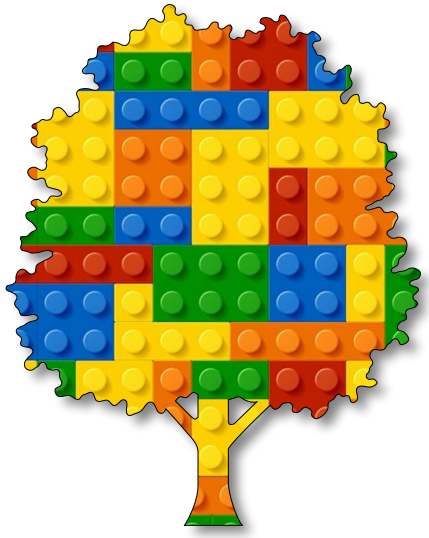
Algorithmic Art

Module E
Unit #6

perlin flowfield



Content

Module E Unit #6 perlin flowfield

Introduction to perlin flowfield

We are going to replace pixels with arrows with Perlin noise to show a flowfield. We start where we left off with 2D Perlin noise and instead of a grayscale value we will have a vector which will point in a direction according to Perlin noise.

Keep the fireworks display sketch from the previous unit, suggest duplicating it. We can use the `particle.js` file for this. We don't need the firework.js file. Delete all the code in those files for now.

Sketch E6.1 starting point

❗ our sketch in sketch.js

We will start with familiar code from module A unit #2, Perlin 2D. Saves a bit of time, assuming you have completed that unit.

sketch.js

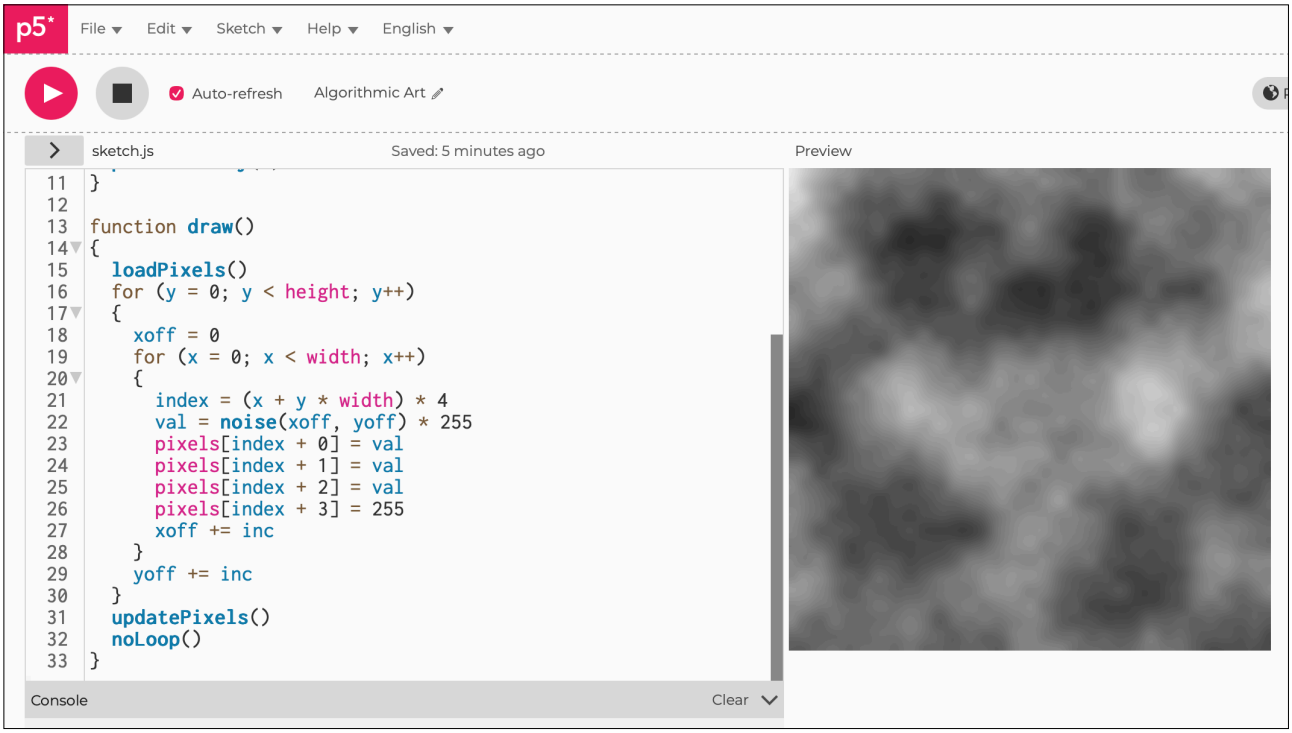
```
let xoff = 0
let yoff = 0
let inc = 0.01
let index
let val

function setup()
{
  createCanvas(400, 400)
  pixelDensity(1)
}

function draw()
{
  loadPixels()
  for (y = 0; y < height; y++)
  {
    xoff = 0
    for (x = 0; x < width; x++)
    {
      index = (x + y * width) * 4
      val = noise(xoff, yoff) * 255
      pixels[index + 0] = val
      pixels[index + 1] = val
      pixels[index + 2] = val
      pixels[index + 3] = 255
      xoff += inc
    }
    yoff += inc
  }
  updatePixels()
```

```
  noLoop()
}
```

Notes

This is what we had before.

Figure E6.1



Sketch E6.2 vectors not pixels

But we want vectors, not pixels. So we also don't want a vector for every pixel. So we will space them out with a variable, which we will call `scl` (short for scale). We will need columns (`cols`) and rows (`rows`). To get the right number of rows and columns, we divide the width and height by `scl` and also use `floor` so we have an integer number.

We can also get rid of all the references to `pixels` (// commented out and highlighted in blue).

sketch.js

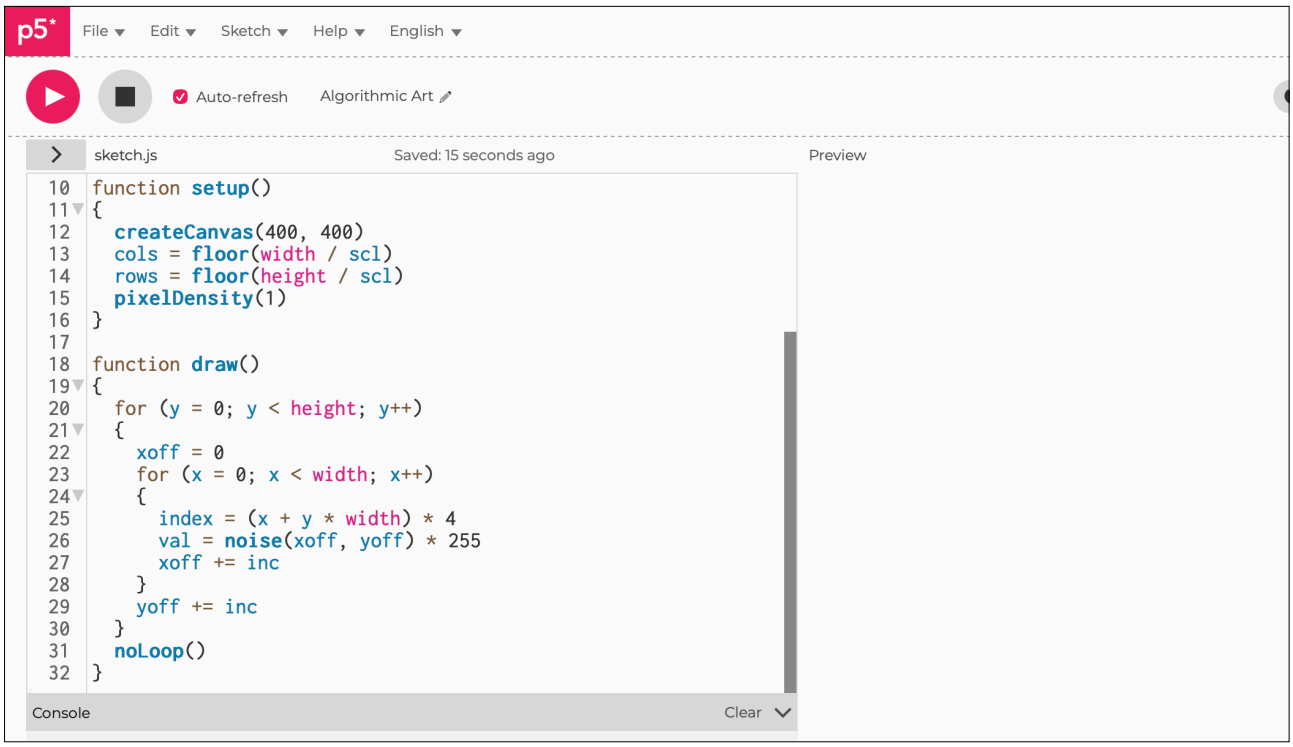
```
let xoff = 0
let yoff = 0
let inc = 0.01
let index
let val
let scl = 20
let cols
let rows

function setup()
{
  createCanvas(400, 400)
  cols = floor(width / scl)
  rows = floor(height / scl)
  pixelDensity(1)
}

function draw()
{
  // loadPixels()
  for (y = 0; y < height; y++)
  {
    xoff = 0
    for (x = 0; x < width; x++)
    {
      index = (x + y * width) * 4
      val = noise(xoff, yoff) * 255
      // pixels[index + 0] = val
```

```
      // pixels[index + 1] = val
      // pixels[index + 2] = val
      // pixels[index + 3] = 255
      xoff += inc
    }
    yoff += inc
  }
  // updatePixels()
  noLoop()
}
```

🗒️ Notes

Nothing to show for it.

Figure E6.2



Sketch E6.3 random squares

To make sure everything is working ok we will use those variables we have just created to draw a square and fill it randomly.

sketch.js

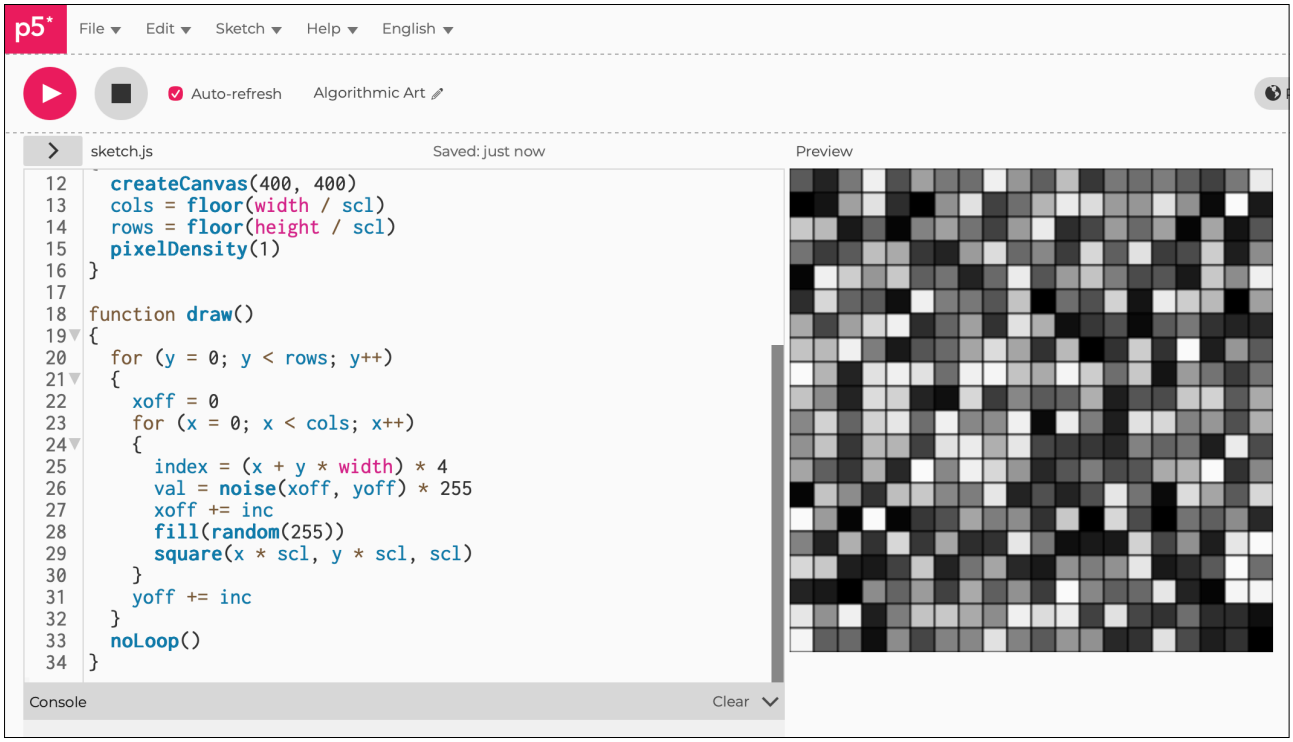
```
let xoff = 0
let yoff = 0
let inc = 0.01
let index
let val
let scl = 20
let cols
let rows

function setup()
{
  createCanvas(400, 400)
  cols = floor(width / scl)
  rows = floor(height / scl)
  pixelDensity(1)
}

function draw()
{
  for (y = 0; y < rows; y++)
  {
    xoff = 0
    for (x = 0; x < cols; x++)
    {
      index = (x + y * width) * 4
      val = noise(xoff, yoff) * 255
      xoff += inc
      fill(random(255))
      square(x * scl, y * scl, scl)
    }
    yoff += inc
  }
```

```
  noLoop()
}
```

Notes

Just randomly filled squares.

Figure E6.3



Sketch E6.4 perlin fill

Instead of randomly filling the squares with greyscale, let's change it to Perlin. We already have the Perlin value as val and make the inc (increment) a little bigger.

sketch.js

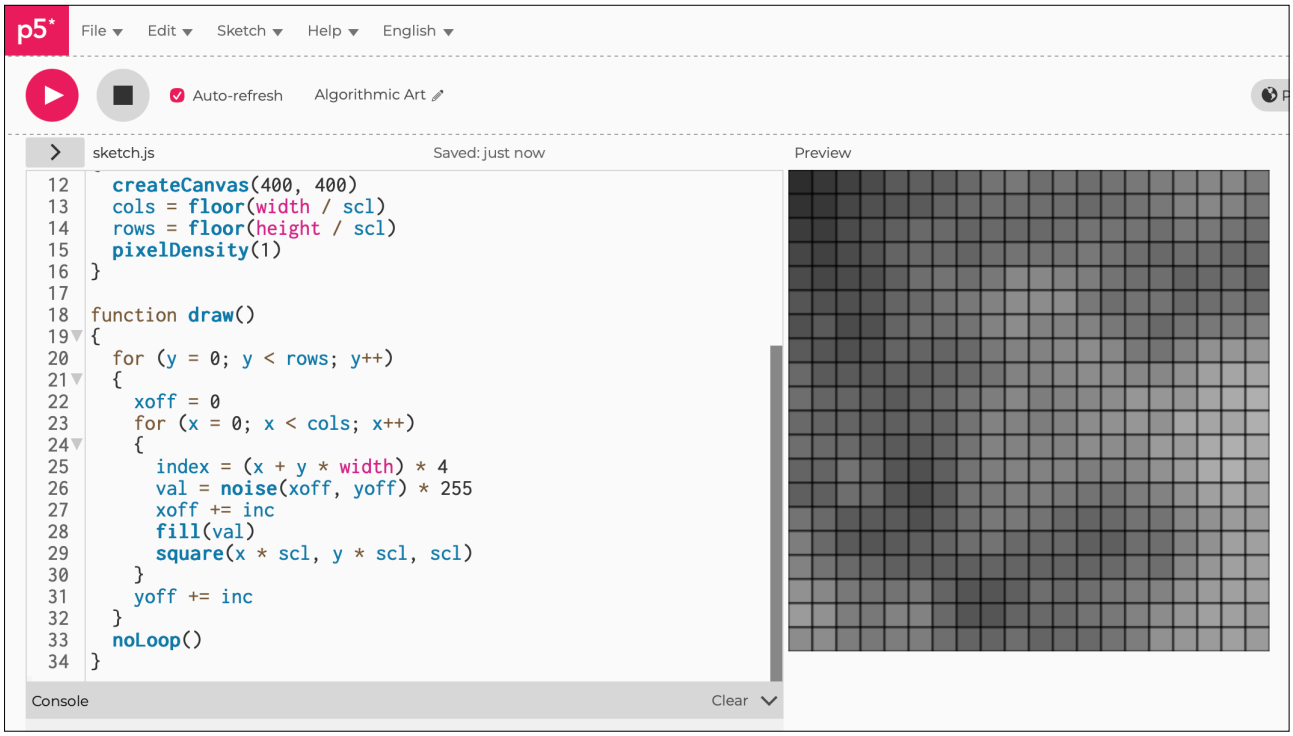
```
let xoff = 0
let yoff = 0
let inc = 0.1
let index
let val
let scl = 20
let cols
let rows

function setup()
{
  createCanvas(400, 400)
  cols = floor(width / scl)
  rows = floor(height / scl)
  pixelDensity(1)
}

function draw()
{
  for (y = 0; y < rows; y++)
  {
    xoff = 0
    for (x = 0; x < cols; x++)
    {
      index = (x + y * width) * 4
      val = noise(xoff, yoff) * 255
      xoff += inc
      fill(val)
      square(x * scl, y * scl, scl)
    }
    yoff += inc
  }
```

```
  noLoop()
}
```

Notes

The grey scaling is linked to the Perlin noise() value.

Figure E6.4



Sketch E6.5 vector lines

Instead of colouring in a square, we are going to draw a vector as a line, removing the `fill()` and `square()`.

sketch.js

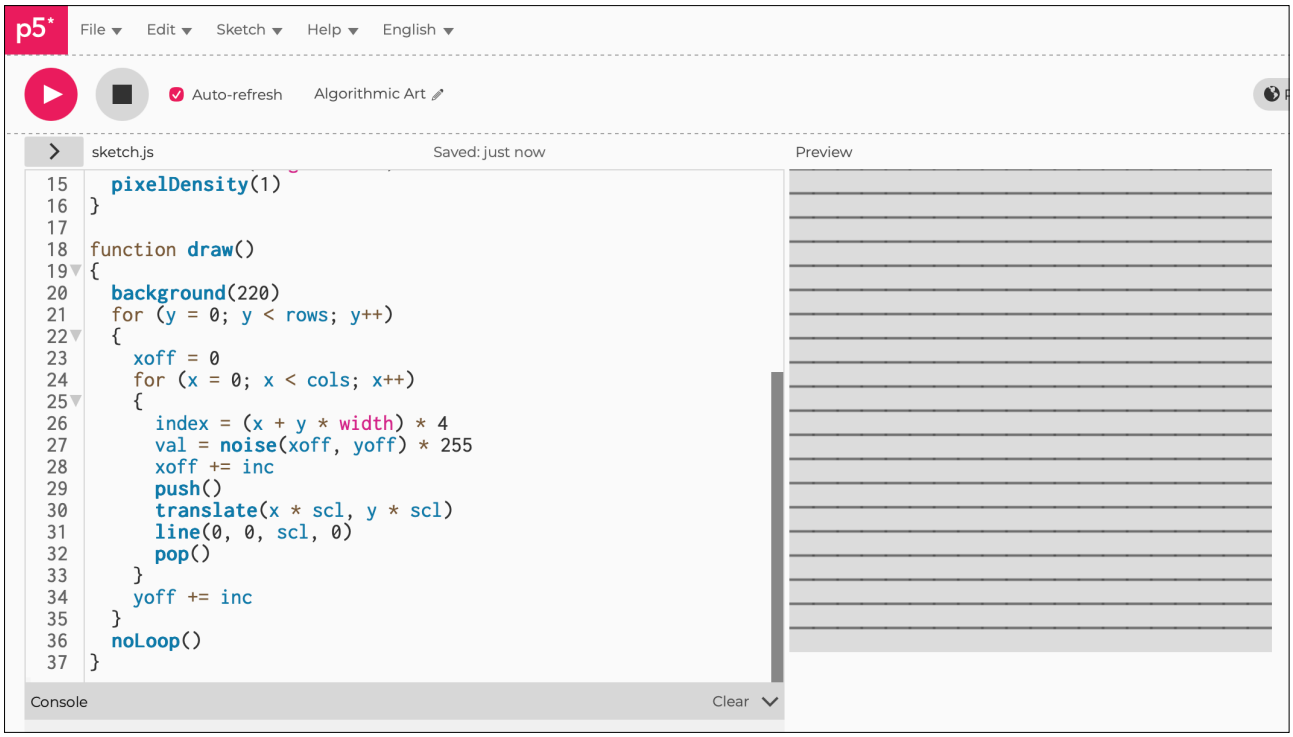
```
let xoff = 0
let yoff = 0
let inc = 0.1
let index
let val
let scl = 20
let cols
let rows

function setup()
{
  createCanvas(400, 400)
  cols = floor(width / scl)
  rows = floor(height / scl)
  pixelDensity(1)
}

function draw()
{
  background(220)
  for (y = 0; y < rows; y++)
  {
    xoff = 0
    for (x = 0; x < cols; x++)
    {
      index = (x + y * width) * 4
      val = noise(xoff, yoff) * 255
      xoff += inc
      push()
      translate(x * scl, y * scl)
      line(0, 0, scl, 0)
      pop()
```

```
      // fill(val)
      // square(x * scl, y * scl, scl)
    }
    yoff += inc
  }
  noLoop()
}
```

Notes

We aren't making use of the noise() yet.

Figure E6.5



Sketch E6.6 rotate

We are going to rotate a vector from an angle using a p5 function called: `p5.Vector.fromAngle()` and using the `heading()` function. We will rotate by `0.5 radians`, which is `30° degrees`.

sketch.js

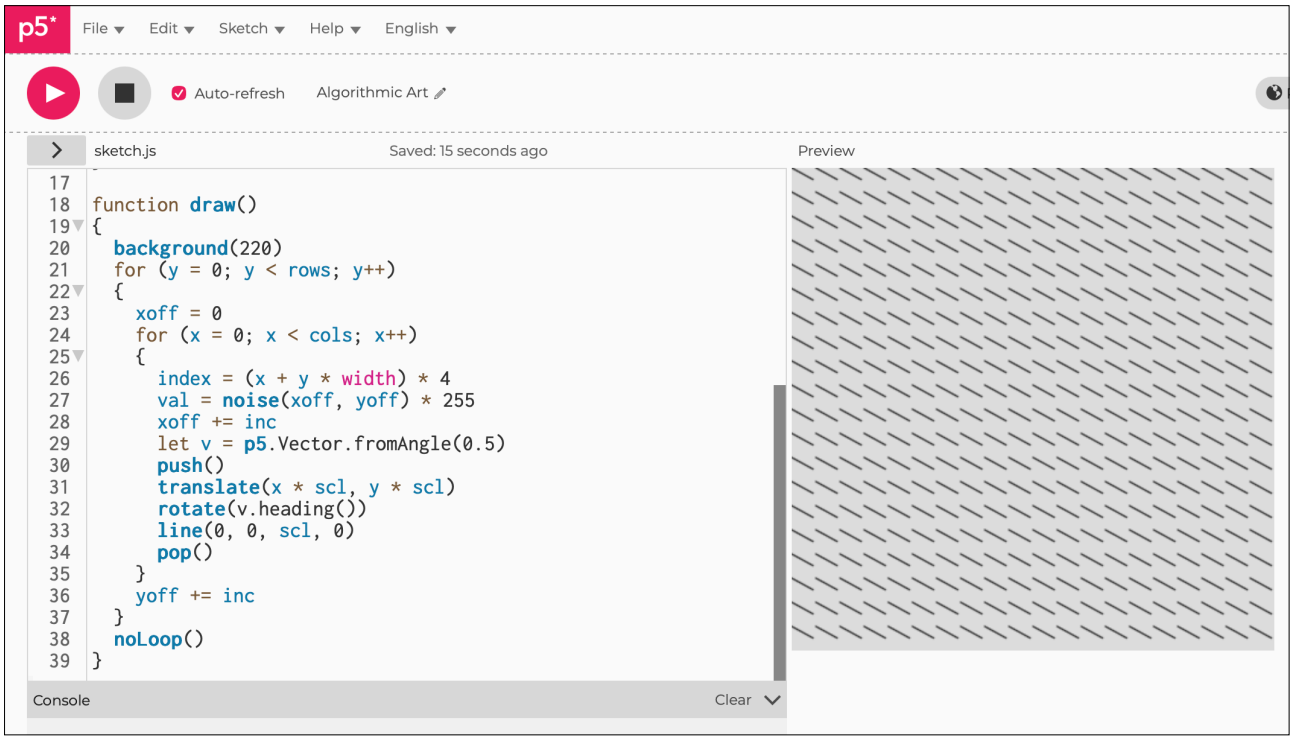
```
let xoff = 0
let yoff = 0
let inc = 0.1
let index
let val
let scl = 20
let cols
let rows

function setup()
{
  createCanvas(400, 400)
  cols = floor(width / scl)
  rows = floor(height / scl)
  pixelDensity(1)
}

function draw()
{
  background(220)
  for (y = 0; y < rows; y++)
  {
    xoff = 0
    for (x = 0; x < cols; x++)
    {
      index = (x + y * width) * 4
      val = noise(xoff, yoff) * 255
      xoff += inc
      let v = p5.Vector.fromAngle(0.5)
      push()
      translate(x * scl, y * scl)
```

```
      rotate(v.heading())
      line(0, 0, scl, 0)
      pop()
    }
    yoff += inc
  }
  noLoop()
}
```

Notes

All angled nicely.

Figure E6.6



Sketch E6.7 random angle

Instead of a fixed angle, let us use `random()` from `0` to `2π`.

sketch.js

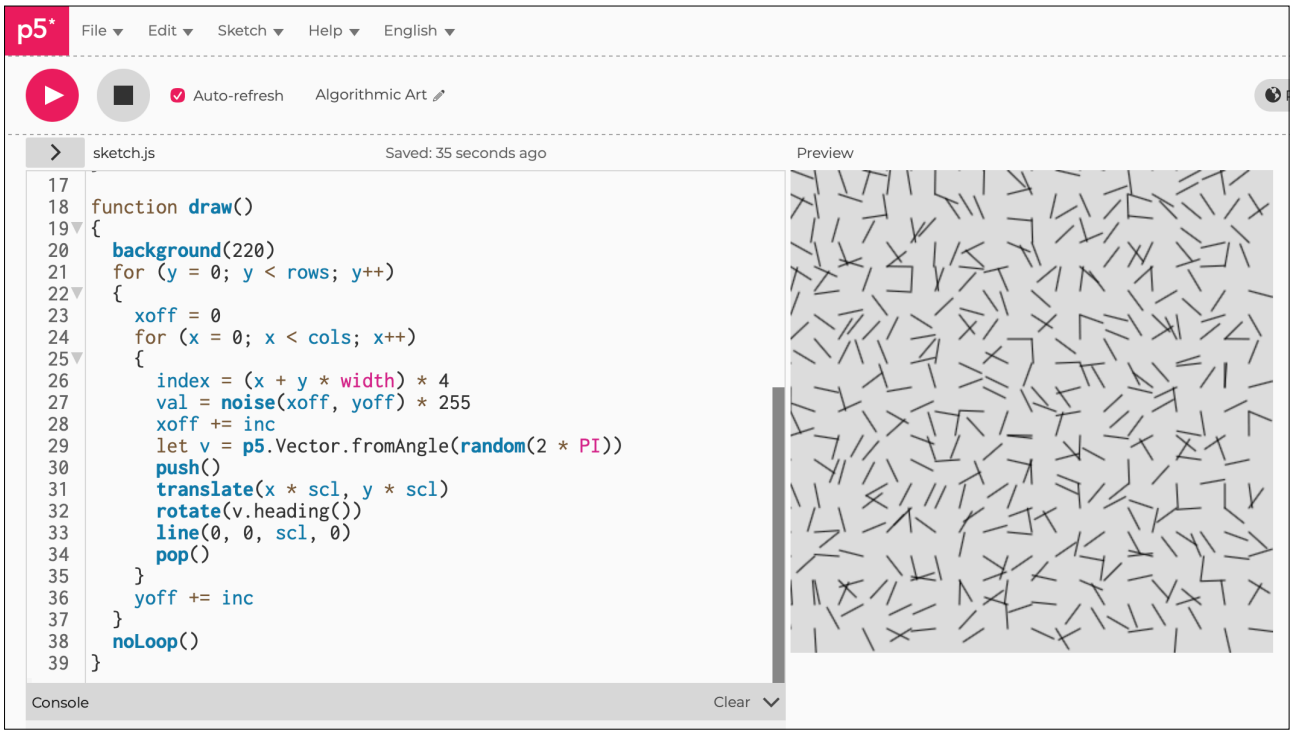
```
let xoff = 0
let yoff = 0
let inc = 0.1
let index
let val
let scl = 20
let cols
let rows

function setup()
{
  createCanvas(400, 400)
  cols = floor(width / scl)
  rows = floor(height / scl)
  pixelDensity(1)
}

function draw()
{
  background(220)
  for (y = 0; y < rows; y++)
  {
    xoff = 0
    for (x = 0; x < cols; x++)
    {
      index = (x + y * width) * 4
      val = noise(xoff, yoff) * 255
      xoff += inc
      let v = p5.Vector.fromAngle(random(2 * PI))
      push()
      translate(x * scl, y * scl)
      rotate(v.heading())
      line(0, 0, scl, 0)
```

```
      pop()
    }
    yoff += inc
  }
  noLoop()
}
```

🗒️ Notes

They are randomly arranged.

Figure E6.7



Sketch E6.8 perlin angle

Rather than pure random, we incorporate Perlin noise. We change the variable name from `val` to something more relevant (`angle`). We multiply by 2π (360°) and then draw the vector to that angle. Every time you refresh it, it changes, but you can see the Perlin noise effect.

sketch.js

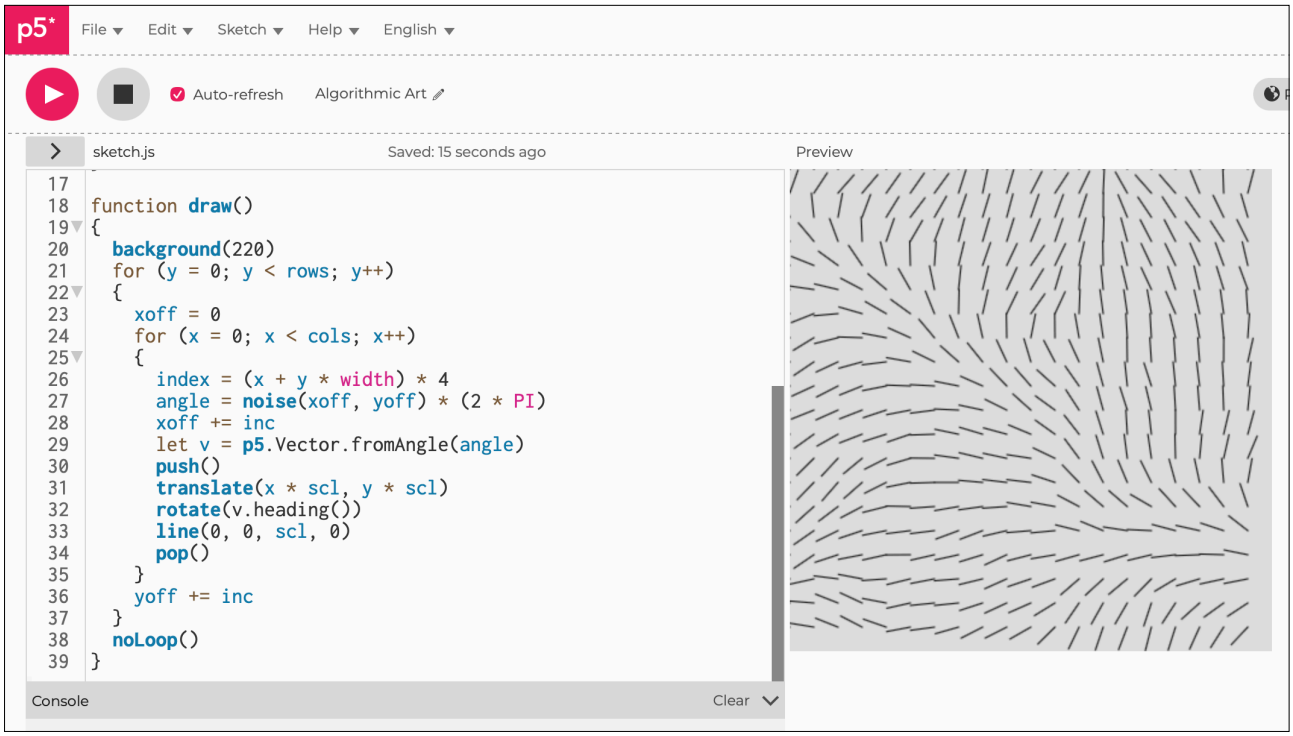
```
let xoff = 0
let yoff = 0
let inc = 0.1
let index
let angle
let scl = 20
let cols
let rows

function setup()
{
  createCanvas(400, 400)
  cols = floor(width / scl)
  rows = floor(height / scl)
  pixelDensity(1)
}

function draw()
{
  background(220)
  for (y = 0; y < rows; y++)
  {
    xoff = 0
    for (x = 0; x < cols; x++)
    {
      index = (x + y * width) * 4
      angle = noise(xoff, yoff) * (2 * PI)
      xoff += inc
      let v = p5.Vector.fromAngle(angle)
      push()
      translate(x * scl, y * scl)
```

```
      rotate(v.heading())
      line(0, 0, scl, 0)
      pop()
    }
    yoff += inc
  }
  noLoop()
}
```

Notes

The angles now have something in common with each other.

Figure E6.8



Sketch E6.9 third dimension

Let's create a third dimension for time and call it zoff. We will change the time in small increments. This will animate as if we have slices of time. Each slice is connected to the previous one through Perlin noise. We need to make a few alterations to have a slow animated Perlin noise effect. You can play with the parameters.

sketch.js

```
let xoff = 0
let yoff = 0
let zoff = 0
let inc = 0.1
let index
let angle
let scl = 20
let cols
let rows

function setup()
{
  createCanvas(400, 400)
  cols = floor(width / scl)
  rows = floor(height / scl)
  pixelDensity(1)
}

function draw()
{
  background(220)
  yoff = 0
  for (y = 0; y < rows; y++)
  {
    xoff = 0
    for (x = 0; x < cols; x++)
    {
      index = (x + y * width) * 4
      angle = noise(xoff, yoff, zoff) * (2 * PI)
      xoff += inc
      let v = p5.Vector.fromAngle(angle)
```

```
      push()
      translate(x * scl, y * scl)
      rotate(v.heading())
      line(0, 0, scl, 0)
      pop()
    }
    yoff += inc
    zoff += 0.0005
  }
  // noLoop()
}
```

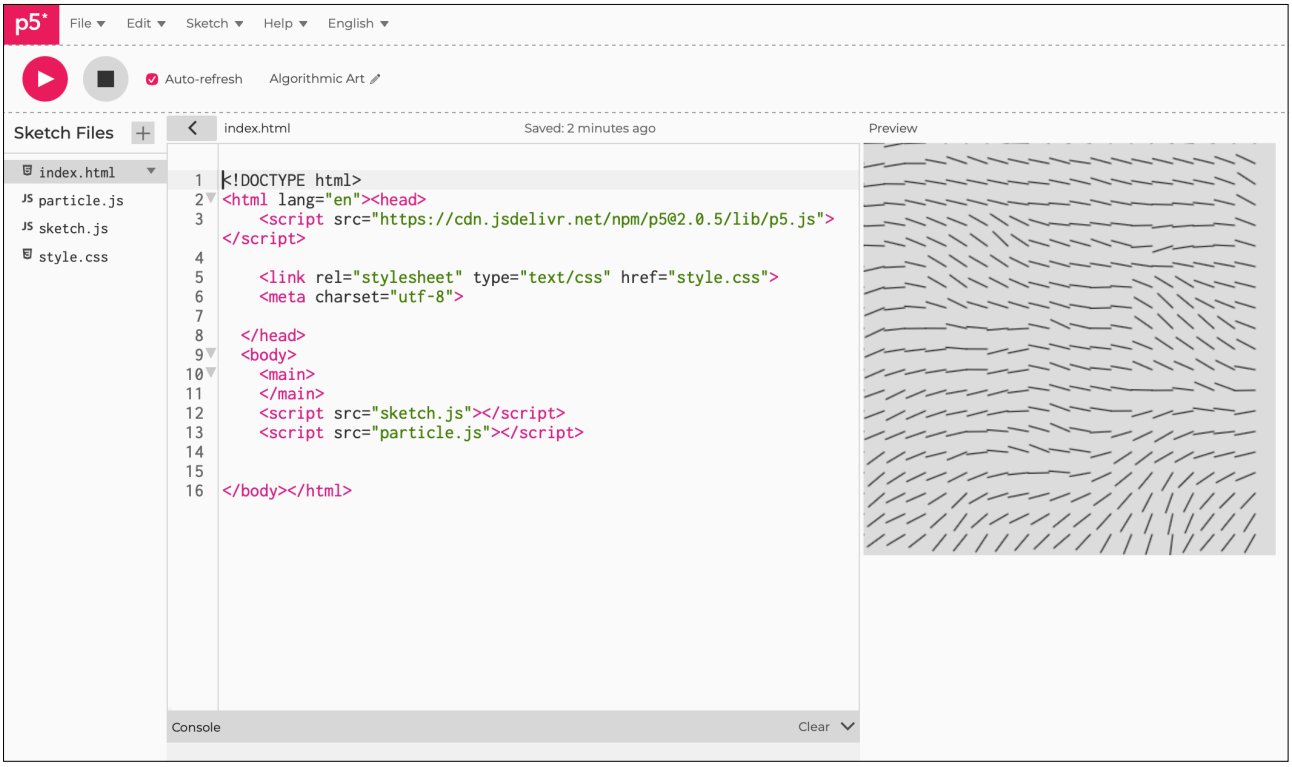
🗒️ Notes

You now get a very pleasing, slowly moving, choreographed set of vectors.

A particle file

If you don't already have the file `particle.js`, then please add it now as you did with the previous unit on fireworks. Remember to add it to the `index.html` file.

Adding the particle.js file



Sketch E6.10 particle class

❗ our first particle.js sketch
We have added a `constructor()` function to the `Particle class`.

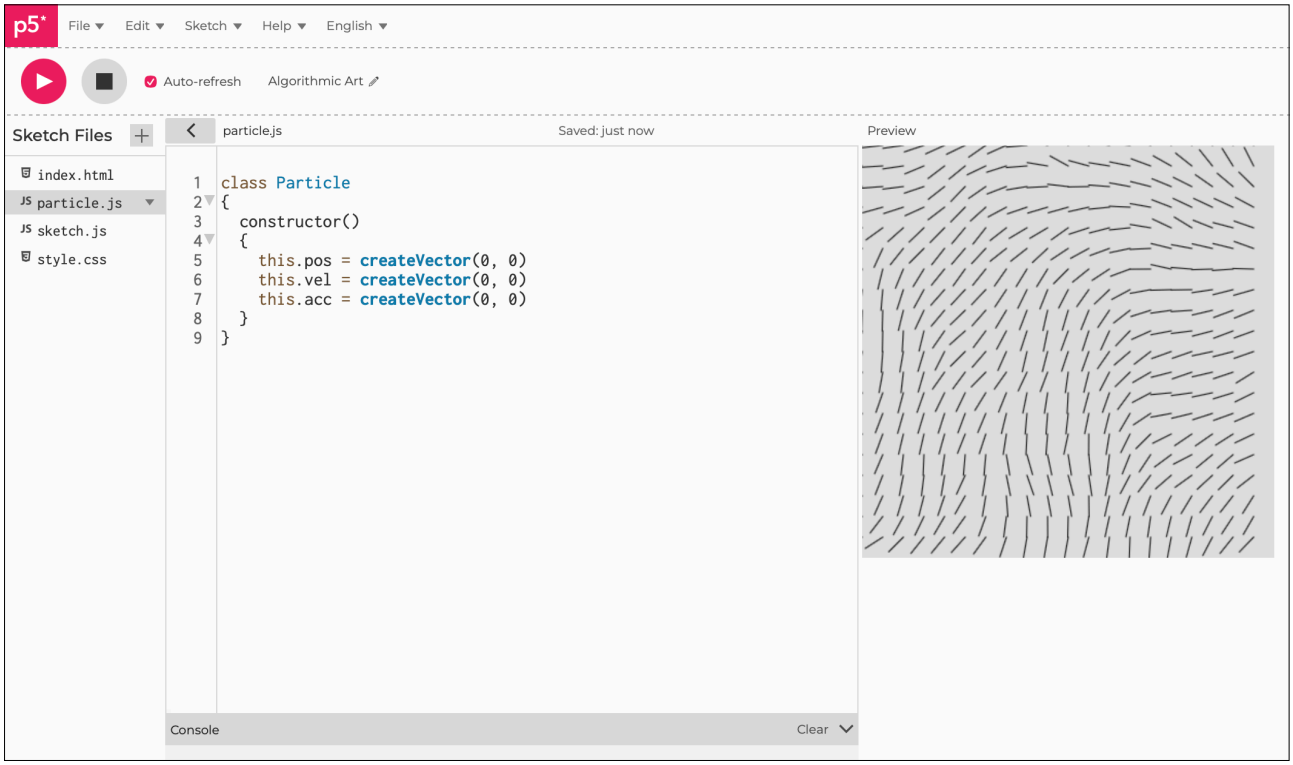
particle.js

```
class Particle
{
  constructor()
  {
    this.pos = createVector(0, 0)
    this.vel = createVector(0, 0)
    this.acc = createVector(0, 0)
  }
}
```

Notes

We have created three vectors for the `position`, the `velocity`, and the `acceleration`.

Figure E6.10



Sketch E6.11 position, velocity and acceleration

We will add in the position, velocity, and acceleration to the move() function and reset the acceleration to zero.

particle.js

```
class Particle
{
  constructor()
  {
    this.pos = createVector(0, 0)
    this.vel = createVector(0, 0)
    this.acc = createVector(0, 0)
  }

  move()
  {
    this.vel.add(this.acc)
    this.pos.add(this.vel)
    this.acc.mult(0)
  }
}
```

Notes

Nothing is changing just yet.

Sketch E6.12 the force

A force is applied to an acceleration

particle.js

```
class Particle
{
  constructor()
  {
    this.pos = createVector(0, 0)
    this.vel = createVector(0, 0)
    this.acc = createVector(0, 0)
  }

  move()
  {
    this.vel.add(this.acc)
    this.pos.add(this.vel)
    this.acc.mult(0)
  }

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

Sketch E6.13 particle

Now, to draw the particle.

particle.js

```
class Particle
{
  constructor()
  {
    this.pos = createVector(0, 0)
    this.vel = createVector(0, 0)
    this.acc = createVector(0, 0)
  }

  move()
  {
    this.vel.add(this.acc)
    this.pos.add(this.vel)
    this.acc.mult(0)
  }

  applyForce(force)
  {
    this.acc.add(force)
  }

  show()
  {
    point(this.pos.x, this.pos.y)
  }
}
```

Notes

Still nothing to see here. We need to go to the main sketch and reference it there.

Sketch E6.14 it is there honestly!

❗ sketch.js

Let us draw the particle in the main sketch. You may not be able to see it very well, but it is there, honest. We create a particle array and put one particle in it at `index [0]`.

sketch.js

```
let xoff = 0
let yoff = 0
let zoff = 0
let inc = 0.1
let index
let angle
let scl = 20
let cols
let rows
let particles = []

function setup()
{
  createCanvas(400, 400)
  cols = floor(width / scl)
  rows = floor(height / scl)
  pixelDensity(1)
  particles[0] = new Particle()
}

function draw()
{
  background(220)
  yoff = 0
  for (y = 0; y < rows + 1; y++)
  {
    xoff = 0
    for (x = 0; x < cols + 1; x++)
    {
      index = (x + y * width) * 4
```

```
      angle = noise(xoff, yoff, zoff) * (2 * PI)
      xoff += inc
      let v = p5.Vector.fromAngle(angle)
      push()
      translate(x * scl, y * scl)
      rotate(v.heading())
      line(0, 0, scl, 0)
      pop()
    }
    yoff += inc
    zoff += 0.0005
  }
  particles[0].show()
  particles[0].move()
}
```

Sketch E6.15 many particles

Let's make 100 particles with a for() loop.

sketch.js

```
let xoff = 0
let yoff = 0
let zoff = 0
let inc = 0.1
let index
let angle
let scl = 20
let cols
let rows
let particles = []

function setup()
{
  createCanvas(400, 400)
  cols = floor(width / scl)
  rows = floor(height / scl)
  pixelDensity(1)
  for (let i = 0; i < 100; i++)
  {
    particles[i] = new Particle()
  }
}

function draw()
{
  background(220)
  yoff = 0
  for (y = 0; y < rows + 1; y++)
  {
    xoff = 0
    for (x = 0; x < cols + 1; x++)
    {
      index = (x + y * width) * 4
```

```
      angle = noise(xoff, yoff, zoff) * (2 * PI)
      xoff += inc
      let v = p5.Vector.fromAngle(angle)
      push()
      translate(x * scl, y * scl)
      rotate(v.heading())
      line(0, 0, scl, 0)
      pop()
    }
    yoff += inc
    zoff += 0.0005
  }
  for (let i = 0; i < particles.length; i++)
  {
    particles[i].show()
    particles[i].move()
  }
}
```

Sketch E6.16 random particles

❗ particle.js

We can check if this is working in particle.js and give them a random velocity.

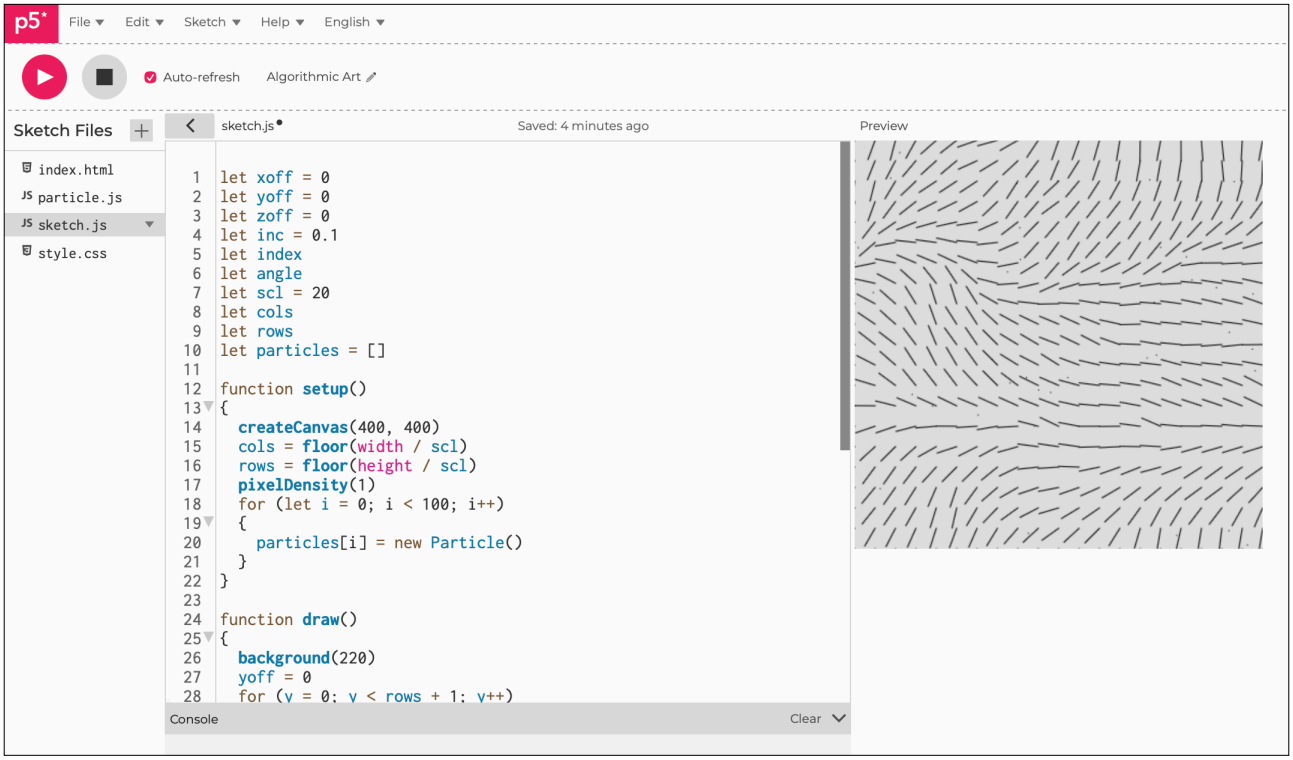
particle.js

```
class Particle
{
  constructor()
  {
    this.pos = createVector(random(width), random(height))
    this.vel = p5.Vector.random2D()
    this.acc = createVector(0, 0)
  }

  move()
  {
    this.vel.add(this.acc)
    this.pos.add(this.vel)
    this.acc.mult(0)
  }

  applyForce(force)
  {
    this.acc.add(force)
  }

  show()
  {
    point(this.pos.x, this.pos.y)
  }
}
```

🗒️ Notes

You should see them if you look closely.

Figure E6.16


p5*
File
Edit
Sketch
Help
English
Auto-refresh
Algorithmic Art
Sketch Files
index.html
particle.js
sketch.js
style.css
sketch.js
Saved: 4 minutes ago
Preview
let xoff = 0
let yoff = 0
let zoff = 0
let inc = 0.1
let index
let angle
let scl = 20
let cols
let rows
let particles = []
function setup()
{
createCanvas(400, 400)
cols = floor(width / scl)
rows = floor(height / scl)
pixelDensity(1)
for (let i = 0; i < 100; i++)
{
particles[i] = new Particle()
}
}
function draw()
{
background(220)
yoff = 0
for (y = 0; y < rows + 1; y++)
Console
Clear

Sketch E6.17 falling off the edge

To stop them disappearing off the edge of the canvas, we will create an `edge()` function boundary in particle.js. In the `setup()` part of the sketch.js, I have commented out the `pixelDensity()` and introduced a `strokeWeight(3)` for clarity (both of these are temporary and optional).

particle.js

```
class Particle
{
  constructor()
  {
    this.pos = createVector(random(width), random(height))
    this.vel = p5.Vector.random2D()
    this.acc = createVector(0, 0)
  }

  move()
  {
    this.vel.add(this.acc)
    this.pos.add(this.vel)
    this.acc.mult(0)
  }

  applyForce(force)
  {
    this.acc.add(force)
  }

  show()
  {

    point(this.pos.x, this.pos.y)
  }

  edges()
  {
    if (this.pos.x > width)
    {
```

```
      this.pos.x = 0
    }
    if (this.pos.x < 0)
    {
      this.pos.x = width
    }
    if (this.pos.y > height)
    {
      this.pos.y = 0
    }
    if (this.pos.y < 0)
    {
      this.pos.y = height
    }
  }
}
```

Sketch E6.18 hide the lines

❗ sketch.js

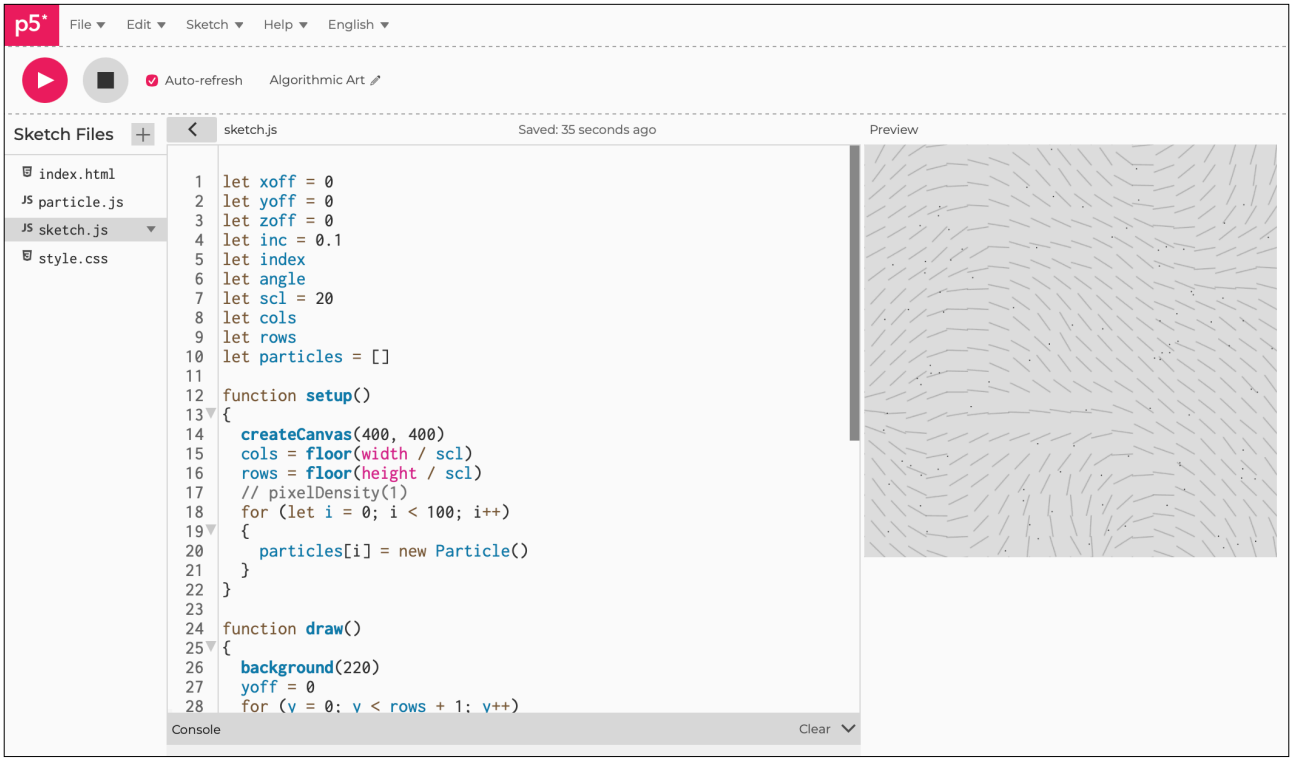
To hide the lines a bit, make the following adjustments (as well as adding the `edges()` function).

sketch.js

```
let xoff = 0
let yoff = 0
let zoff = 0
let inc = 0.1
let index
let angle
let scl = 20
let cols
let rows
let particles = []

function setup()
{
  createCanvas(400, 400)
  cols = floor(width / scl)
  rows = floor(height / scl)
  // pixelDensity(1)
  for (let i = 0; i < 100; i++)
  {
    particles[i] = new Particle()
  }
}

function draw()
{
  background(220)
  yoff = 0
  for (y = 0; y < rows + 1; y++)
  {
    xoff = 0
    for (x = 0; x < cols + 1; x++)
```

```
    {
      index = (x + y * width) * 4
      angle = noise(xoff, yoff, zoff) * (2 * PI)
      xoff += inc
      let v = p5.Vector.fromAngle(angle)
      push()
      translate(x * scl, y * scl)
      rotate(v.heading())
      strokeWeight(1)
      stroke(0, 50)
      line(0, 0, scl, 0)
      pop()
    }
    yoff += inc
    zoff += 0.0005
  }
  for (let i = 0; i < particles.length; i++)
  {
    particles[i].show()
    particles[i].move()
    particles[i].edges()
  }
}
```

Figure E6.18


p5*
File
Edit
Sketch
Help
English
Auto-refresh
Algorithmic Art
Sketch Files
index.html
particle.js
sketch.js
style.css
sketch.js
Saved: 35 seconds ago
Preview
let xoff = 0
let yoff = 0
let zoff = 0
let inc = 0.1
let index
let angle
let scl = 20
let cols
let rows
let particles = []
function setup()
{
createCanvas(400, 400)
cols = floor(width / scl)
rows = floor(height / scl)
// pixelDensity(1)
for (let i = 0; i < 100; i++)
{
particles[i] = new Particle()
}
}
function draw()
{
background(220)
yoff = 0
for (y = 0; y < rows + 1; y++)
Console
Clear

Sketch E6.19 better particles

❗ particle.js
In particle.js, I want to see the particles clearly.

particle.js

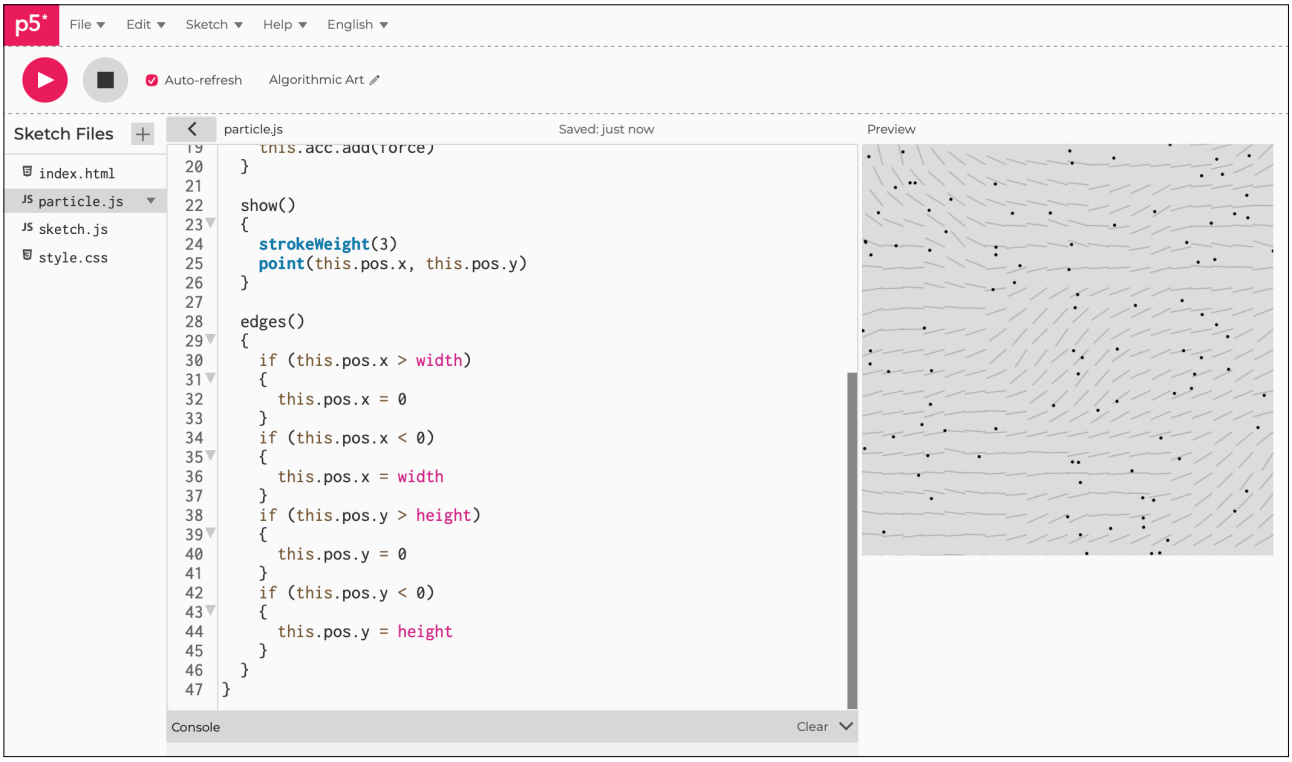
```
class Particle
{
  constructor()
  {
    this.pos = createVector(random(width), random(height))
    this.vel = p5.Vector.random2D()
    this.acc = createVector(0, 0)
  }

  move()
  {
    this.vel.add(this.acc)
    this.pos.add(this.vel)
    this.acc.mult(0)
  }

  applyForce(force)
  {
    this.acc.add(force)
  }

  show()
  {
    strokeWeight(3)
    point(this.pos.x, this.pos.y)
  }

  edges()
  {
    if (this.pos.x > width)
    {
      this.pos.x = 0
```

```
    }
    if (this.pos.x < 0)
    {
      this.pos.x = width
    }
    if (this.pos.y > height)
    {
      this.pos.y = 0
    }
    if (this.pos.y < 0)
    {
      this.pos.y = height
    }
  }
}
```

Notes

The points aren't moving according to perlin noise().

Figure E6.19



Sketch E6.20 array of vectors

❗ sketch.js

We want to store the values of the vectors in an array and then attribute them to the particles nearest to them. We create an empty array called `flowfield[]`. We use the formula for calculating the index and change the width to `cols` (and remove the times 4). Now we can fill the array with those vectors `v`.

sketch.js

```
let xoff = 0
let yoff = 0
let zoff = 0
let inc = 0.1
let index
let angle
let scl = 20
let cols
let rows
let particles = []
let flowfield = []

function setup()
{
  createCanvas(400, 400)
  cols = floor(width / scl)
  rows = floor(height / scl)
  // pixelDensity(1)
  flowfield = new Array(cols * rows)
  for (let i = 0; i < 100; i++)
  {
    particles[i] = new Particle()
  }
}

function draw()
{
  background(220)
  yoff = 0
  for (y = 0; y < rows + 1; y++)
```

```
  {
    xoff = 0
    for (x = 0; x < cols + 1; x++)
    {
      index = (x + y * cols)
      angle = noise(xoff, yoff, zoff) * (2 * PI)
      xoff += inc
      let v = p5.Vector.fromAngle(angle)
      flowfield[index] = v
      push()
      translate(x * scl, y * scl)
      rotate(v.heading())
      strokeWeight(1)
      stroke(0, 50)
      line(0, 0, scl, 0)
      pop()
    }
    yoff += inc
    zoff += 0.0005
  }
  for (let i = 0; i < particles.length; i++)
  {
    particles[i].show()
    particles[i].move()
    particles[i].edges()
  }
}
```

🗒️ Notes

Nothing new happening yet.

Sketch E6.21 following the vectors

! particle.js

We need to write a function called `follow()` in the particle class that helps us identify which particle is near which vector. It will have an argument called vectors. We have to find the co-ordinates of the particle according to how the vector lines are laid out.

particle.js

```
class Particle
{
  constructor()
  {
    this.pos = createVector(random(width), random(height))
    this.vel = p5.Vector.random2D()
    this.acc = createVector(0, 0)
  }

  move()
  {
    this.vel.add(this.acc)
    this.pos.add(this.vel)
    this.acc.mult(0)
  }

  applyForce(force)
  {
    this.acc.add(force)
  }

  show()
  {
    strokeWeight(3)
    point(this.pos.x, this.pos.y)
  }

  edges()
  {
    if (this.pos.x > width)
```

```
    {
      this.pos.x = 0
    }
    if (this.pos.x < 0)
    {
      this.pos.x = width
    }
    if (this.pos.y > height)
    {
      this.pos.y = 0
    }
    if (this.pos.y < 0)
    {
      this.pos.y = height
    }
  }

  follow(vectors)
  {
    let x = floor(this.pos.x/scl)
    let y = floor(this.pos.y/scl)
  }
}
```

Sketch E6.22 the index value

From that, we calculate the index value from x and y using the formula.

particle.js

```
class Particle
{
  constructor()
  {
    this.pos = createVector(random(width), random(height))
    this.vel = p5.Vector.random2D()
    this.acc = createVector(0, 0)
  }

  move()
  {
    this.vel.add(this.acc)
    this.pos.add(this.vel)
    this.acc.mult(0)
  }

  applyForce(force)
  {
    this.acc.add(force)
  }

  show()
  {
    strokeWeight(3)
    point(this.pos.x, this.pos.y)
  }

  edges()
  {
    if (this.pos.x > width)
    {
      this.pos.x = 0
    }
```

```
    if (this.pos.x < 0)
    {
      this.pos.x = width
    }
    if (this.pos.y > height)
    {
      this.pos.y = 0
    }
    if (this.pos.y < 0)
    {
      this.pos.y = height
    }
  }

  follow(vectors)
  {
    let x = floor(this.pos.x/scl)
    let y = floor(this.pos.y/scl)
    let index = x + (y * cols)
  }
}
```

Sketch E6.23 index force

We calculate the force of the vector at that index.

particle.js

```
class Particle
{
  constructor()
  {
    this.pos = createVector(random(width), random(height))
    this.vel = p5.Vector.random2D()
    this.acc = createVector(0, 0)
  }

  move()
  {
    this.vel.add(this.acc)
    this.pos.add(this.vel)
    this.acc.mult(0)
  }

  applyForce(force)
  {
    this.acc.add(force)
  }

  show()
  {
    strokeWeight(3)
    point(this.pos.x, this.pos.y)
  }

  edges()
  {
    if (this.pos.x > width)
    {
      this.pos.x = 0
    }
```

```
    if (this.pos.x < 0)
    {
      this.pos.x = width
    }
    if (this.pos.y > height)
    {
      this.pos.y = 0
    }
    if (this.pos.y < 0)
    {
      this.pos.y = height
    }
  }

  follow(vectors)
  {
    let x = floor(this.pos.x/scl)
    let y = floor(this.pos.y/scl)
    let index = x + (y * cols)
    let force = vectors[index]
  }
}
```

Sketch E6.24 may the force be with you

We then apply the force.

particle.js

```
class Particle
{
  constructor()
  {
    this.pos = createVector(random(width), random(height))
    this.vel = p5.Vector.random2D()
    this.acc = createVector(0, 0)
  }

  move()
  {
    this.vel.add(this.acc)
    this.pos.add(this.vel)
    this.acc.mult(0)
  }

  applyForce(force)
  {
    this.acc.add(force)
  }

  show()
  {
    strokeWeight(3)
    point(this.pos.x, this.pos.y)
  }

  edges()
  {
    if (this.pos.x > width)
    {
      this.pos.x = 0
    }
```

```
    if (this.pos.x < 0)
    {
      this.pos.x = width
    }
    if (this.pos.y > height)
    {
      this.pos.y = 0
    }
    if (this.pos.y < 0)
    {
      this.pos.y = height
    }
  }

  follow(vectors)
  {
    let x = floor(this.pos.x/scl)
    let y = floor(this.pos.y/scl)
    let index = x + (y * cols)
    let force = vectors[index]
    this.applyForce(force)
  }
}
```

Sketch E6.25 mad dash

Let's make the `velocity` zero in particle.js, then set a maximum speed `this.maxSpeed = 4`, and then limit the speed in the `move()` function.

particle.js

```
class Particle
{
  constructor()
  {
    this.pos = createVector(random(width), random(height))
    this.vel = createVector(0, 0)
    this.acc = createVector(0, 0)
    this.maxSpeed = 4
  }

  move()
  {
    this.vel.add(this.acc)
    this.vel.limit(this.maxSpeed)
    this.pos.add(this.vel)
    this.acc.mult(0)
  }

  applyForce(force)
  {
    this.acc.add(force)
  }

  show()
  {
    strokeWeight(3)
    point(this.pos.x, this.pos.y)
  }

  edges()
  {
    if (this.pos.x > width)
```

```
    {
      this.pos.x = 0
    }
    if (this.pos.x < 0)
    {
      this.pos.x = width
    }
    if (this.pos.y > height)
    {
      this.pos.y = 0
    }
    if (this.pos.y < 0)
    {
      this.pos.y = height
    }
  }

  follow(vectors)
  {
    let x = floor(this.pos.x/scl)
    let y = floor(this.pos.y/scl)
    let index = x + (y * cols)
    let force = vectors[index]
    this.applyForce(force)
  }
}
```

Notes

We have come to a complete standstill. Be patient, just wait.

Sketch E6.26 the magnitude of it all

❗ sketch.js

In the sketch.js, we can set the magnitude of the force. You will notice that it goes a bit mad if you run it, but it is the beginning of something.

sketch.js

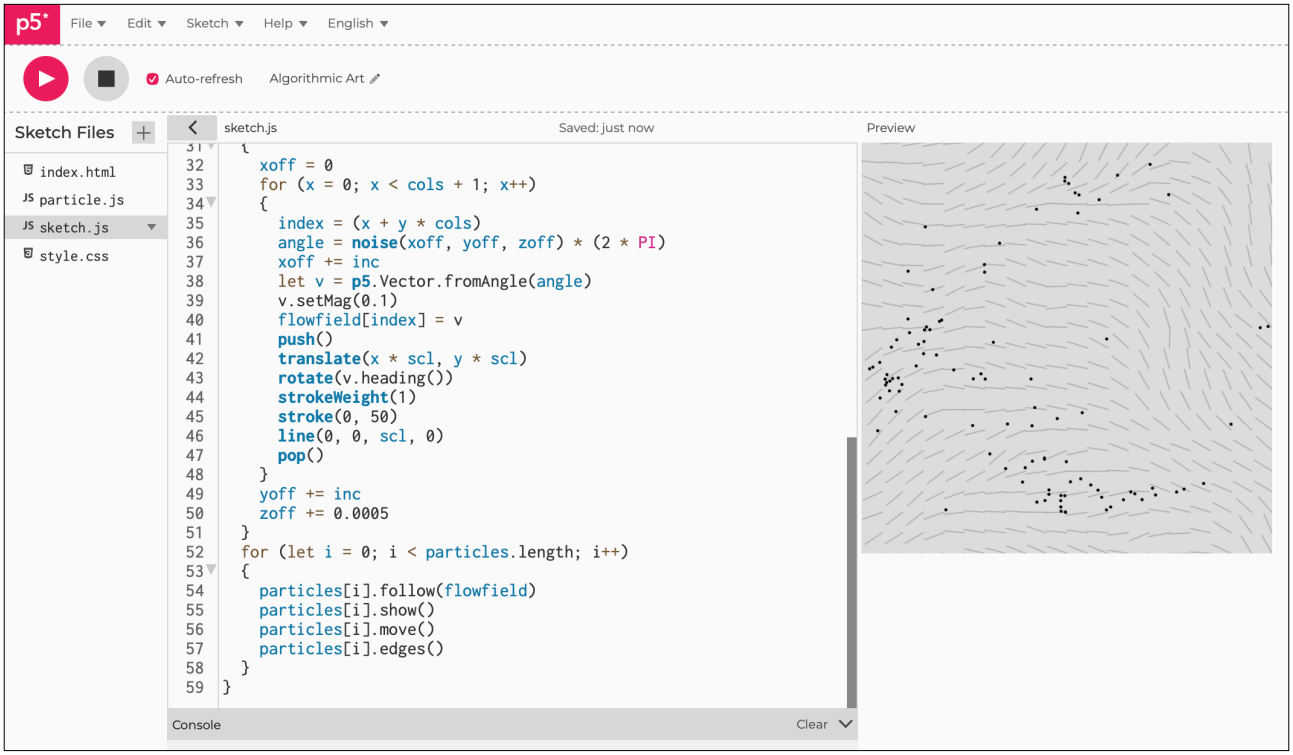
```
let xoff = 0
let yoff = 0
let zoff = 0
let inc = 0.1
let index
let angle
let scl = 20
let cols
let rows
let particles = []
let flowfield = []

function setup()
{
  createCanvas(400, 400)
  cols = floor(width / scl)
  rows = floor(height / scl)
  // pixelDensity(1)
  // flowfield = new Array(cols * rows)
  for (let i = 0; i < 100; i++)
  {
    particles[i] = new Particle()
  }
}

function draw()
{
  background(220)
  yoff = 0
  for (y = 0; y < rows + 1; y++)
  {
```

```
    xoff = 0
    for (x = 0; x < cols + 1; x++)
    {
      index = (x + y * cols)
      angle = noise(xoff, yoff, zoff) * (2 * PI)
      xoff += inc
      let v = p5.Vector.fromAngle(angle)
      v.setMag(0.1)
      flowfield[index] = v
      push()
      translate(x * scl, y * scl)
      rotate(v.heading())
      strokeWeight(1)
      stroke(0, 50)
      line(0, 0, scl, 0)
      pop()
    }
    yoff += inc
    zoff += 0.0005
  }
  for (let i = 0; i < particles.length; i++)
  {
    particles[i].follow(flowfield)
    particles[i].show()
    particles[i].move()
    particles[i].edges()
  }
}
```

🗒️ Notes

You should see them start to follow the flowfield.

Figure E6.26


p5*
File
Edit
Sketch
Help
English
Auto-refresh
Algorithmic Art
Sketch Files
index.html
particle.js
sketch.js
style.css
sketch.js
Saved: just now
Preview
{
xoff = 0
for (x = 0; x < cols + 1; x++)
{
index = (x + y * cols)
angle = noise(xoff, yoff, zoff) * (2 * PI)
xoff += inc
let v = p5.Vector.fromAngle(angle)
v.setMag(0.1)
flowfield[index] = v
push()
translate(x * scl, y * scl)
rotate(v.heading())
strokeWeight(1)
stroke(0, 50)
line(0, 0, scl, 0)
pop()
}
yoff += inc
zoff += 0.0005
}
for (let i = 0; i < particles.length; i++)
{
particles[i].follow(flowfield)
particles[i].show()
particles[i].move()
particles[i].edges()
}
}
Console
Clear

Sketch E6.27 zoff off

Now set the magnitude to 5 and turn off the zoff and you should see it very clearly.

sketch.js

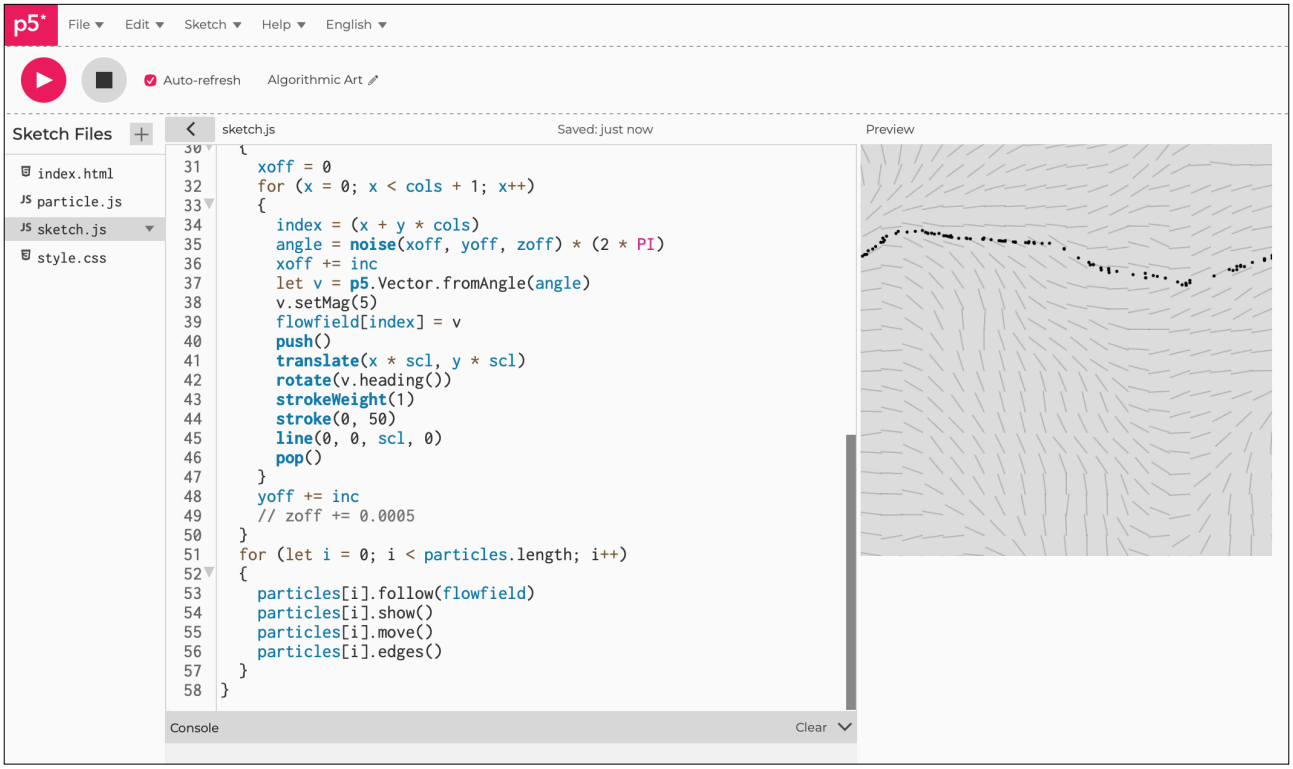
```
let xoff = 0
let yoff = 0
let zoff = 0
let inc = 0.1
let index
let angle
let scl = 20
let cols
let rows
let particles = []
let flowfield = []

function setup()
{
  createCanvas(400, 400)
  cols = floor(width / scl)
  rows = floor(height / scl)
  // pixelDensity(1)
  for (let i = 0; i < 100; i++)
  {
    particles[i] = new Particle()
  }
}

function draw()
{
  background(220)
  yoff = 0
  for (y = 0; y < rows + 1; y++)
  {
    xoff = 0
    for (x = 0; x < cols + 1; x++)
    {
```

```
      index = (x + y * cols)
      angle = noise(xoff, yoff, zoff) * (2 * PI)
      xoff += inc
      let v = p5.Vector.fromAngle(angle)
      v.setMag(5)
      flowfield[index] = v
      push()
      translate(x * scl, y * scl)
      rotate(v.heading())
      strokeWeight(1)
      stroke(0, 50)
      line(0, 0, scl, 0)
      pop()
    }
    yoff += inc
    // zoff += 0.0005
  }
  for (let i = 0; i < particles.length; i++)
  {
    particles[i].follow(flowfield)
    particles[i].show()
    particles[i].move()
    particles[i].edges()
  }
}
```

🗒️ Notes

You can see it following an element of the flowfield.

Figure E6.27


p5*
File
Edit
Sketch
Help
English
Auto-refresh
Algorithmic Art
Sketch Files
index.html
particle.js
sketch.js
style.css
sketch.js
Saved: just now
Preview
{
xoff = 0
for (x = 0; x < cols + 1; x++)
{
index = (x + y * cols)
angle = noise(xoff, yoff, zoff) * (2 * PI)
xoff += inc
let v = p5.Vector.fromAngle(angle)
v.setMag(5)
flowfield[index] = v
push()
translate(x * scl, y * scl)
rotate(v.heading())
strokeWeight(1)
stroke(0, 50)
line(0, 0, scl, 0)
pop()
}
yoff += inc
// zoff += 0.0005
}
for (let i = 0; i < particles.length; i++)
{
particles[i].follow(flowfield)
particles[i].show()
particles[i].move()
particles[i].edges()
}
}
Console
Clear

Sketch E6.28 no lines

Removing the line vectors and adding more particles gives us a better feel, as well as reducing the `force` to `1` and also making the angle of rotation larger.

sketch.js

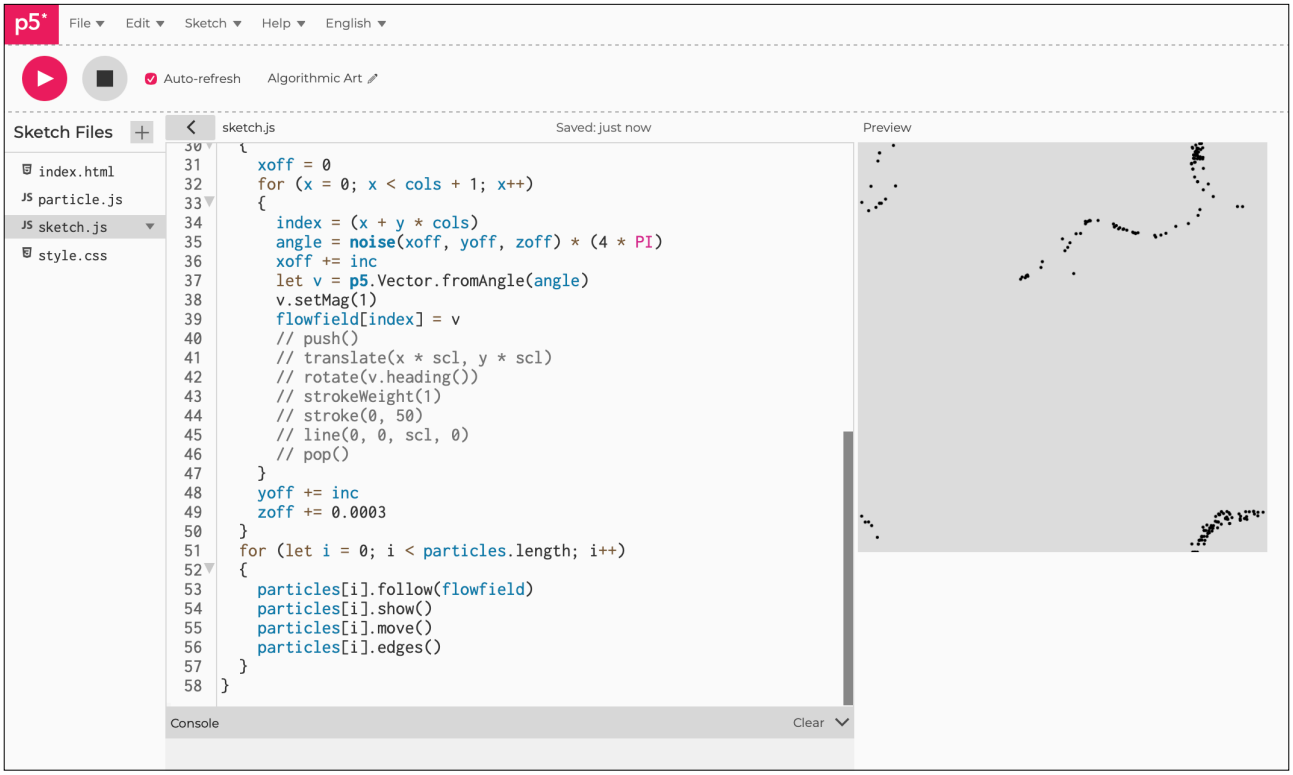
```
let xoff = 0
let yoff = 0
let zoff = 0
let inc = 0.1
let index
let angle
let scl = 20
let cols
let rows
let particles = []
let flowfield = []

function setup()
{
  createCanvas(400, 400)
  cols = floor(width / scl)
  rows = floor(height / scl)
  // pixelDensity(1)
  for (let i = 0; i < 200; i++)
  {
    particles[i] = new Particle()
  }
}

function draw()
{
  background(220)
  yoff = 0
  for (y = 0; y < rows + 1; y++)
  {
    xoff = 0
    for (x = 0; x < cols + 1; x++)
```

```
    {
      index = (x + y * cols)
      angle = noise(xoff, yoff, zoff) * (4 * PI)
      xoff += inc
      let v = p5.Vector.fromAngle(angle)
      v.setMag(1)
      flowfield[index] = v
      // push()
      // translate(x * scl, y * scl)
      // rotate(v.heading())
      // strokeWeight(1)
      // stroke(0, 50)
      // line(0, 0, scl, 0)
      // pop()
    }
    yoff += inc
    zoff += 0.0003
  }
  for (let i = 0; i < particles.length; i++)
  {
    particles[i].follow(flowfield)
    particles[i].show()
    particles[i].move()
    particles[i].edges()
  }
}
```

🗒️ Notes

You do see some strange movement.

Figure E6.28


p5*
File
Edit
Sketch
Help
English
Auto-refresh
Algorithmic Art
Sketch Files
index.html
particle.js
sketch.js
style.css
sketch.js
Saved: just now
Preview
{
xoff = 0
for (x = 0; x < cols + 1; x++)
{
index = (x + y * cols)
angle = noise(xoff, yoff, zoff) * (4 * PI)
xoff += inc
let v = p5.Vector.fromAngle(angle)
v.setMag(1)
flowfield[index] = v
// push()
// translate(x * scl, y * scl)
// rotate(v.heading())
// strokeWeight(1)
// stroke(0, 50)
// line(0, 0, scl, 0)
// pop()
}
yoff += inc
zoff += 0.0003
}
for (let i = 0; i < particles.length; i++)
{
particles[i].follow(flowfield)
particles[i].show()
particles[i].move()
particles[i].edges()
}
}
Console
Clear

Sketch E6.29 nice visual

Making more amendments to get a nice visual. Move the `background()` into the `setup()` function so that it draws it only once, and increase the number of particles to `500`.

sketch.js

```
let xoff = 0
let yoff = 0
let zoff = 0
let inc = 0.1
let index
let angle
let scl = 20
let cols
let rows
let particles = []
let flowfield = []

function setup()
{
  createCanvas(400, 400)
  cols = floor(width / scl)
  rows = floor(height / scl)
  // pixelDensity(1)
  for (let i = 0; i < 500; i++)
  {
    particles[i] = new Particle()
  }
  background(220)
}

function draw()
{
  yoff = 0
  for (y = 0; y < rows + 1; y++)
  {
    xoff = 0
```

```
    for (x = 0; x < cols + 1; x++)
    {
      index = (x + y * cols)
      angle = noise(xoff, yoff, zoff) * (4 * PI)
      xoff += inc
      let v = p5.Vector.fromAngle(angle)
      v.setMag(1)
      flowfield[index] = v
    }
    yoff += inc
    zoff += 0.0003
  }
  for (let i = 0; i < particles.length; i++)
  {
    particles[i].follow(flowfield)
    particles[i].show()
    particles[i].move()
    particles[i].edges()
  }
}
```

🗒️ Notes

Interesting, but we can do even better.

Figure E6.29


p5*
File
Edit
Sketch
Help
English
Auto-refresh
Algorithmic Art
Sketch Files
index.html
particle.js
sketch.js
style.css
sketch.js
Saved: just now
Preview
background(220)
}
function draw()
{
yoff = 0
for (y = 0; y < rows + 1; y++)
{
xoff = 0
for (x = 0; x < cols + 1; x++)
{
index = (x + y * cols)
angle = noise(xoff, yoff, zoff) * (4 * PI)
xoff += inc
let v = p5.Vector.fromAngle(angle)
v.setMag(1)
flowfield[index] = v
}
yoff += inc
zoff += 0.0003
}
for (let i = 0; i < particles.length; i++)
{
particles[i].follow(flowfield)
particles[i].show()
particles[i].move()
particles[i].edges()
}
}
Console
Clear

Sketch E6.30 alpha stroke

! particle.js

In particle.js, we change the drawing of the point. Making it a `strokeWeight()` of `1` and a `stroke()` with some `alpha`.

particle.js

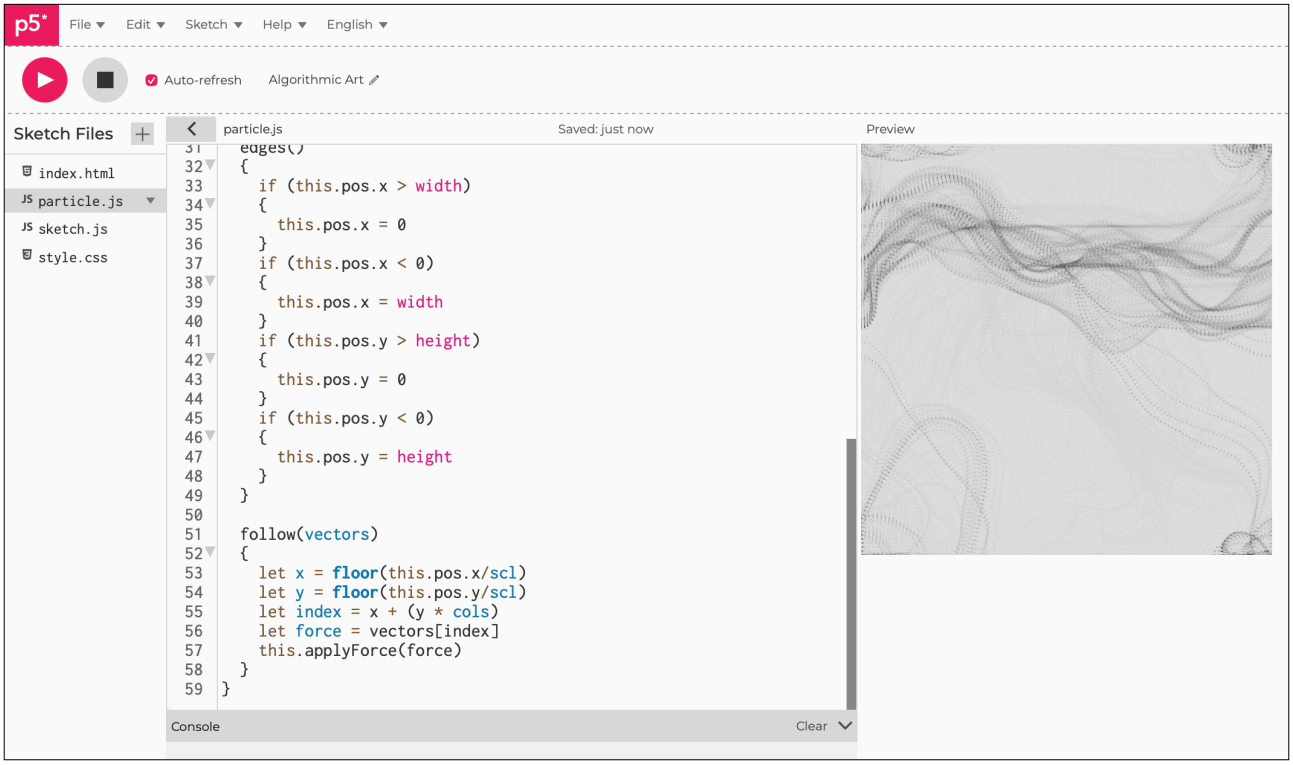
```
class Particle
{
  constructor()
  {
    this.pos = createVector(random(width), random(height))
    this.vel = createVector(0, 0)
    this.acc = createVector(0, 0)
    this.maxSpeed = 4
  }

  move()
  {
    this.vel.add(this.acc)
    this.vel.limit(this.maxSpeed)
    this.pos.add(this.vel)
    this.acc.mult(0)
  }

  applyForce(force)
  {
    this.acc.add(force)
  }

  show()
  {
    strokeWeight(1)
    stroke(0, 5)
    point(this.pos.x, this.pos.y)
  }

  edges()
```

```
  {
    if (this.pos.x > width)
    {
      this.pos.x = 0
    }
    if (this.pos.x < 0)
    {
      this.pos.x = width
    }
    if (this.pos.y > height)
    {
      this.pos.y = 0
    }
    if (this.pos.y < 0)
    {
      this.pos.y = height
    }
  }

  follow(vectors)
  {
    let x = floor(this.pos.x/scl)
    let y = floor(this.pos.y/scl)
    let index = x + (y * cols)
    let force = vectors[index]
    this.applyForce(force)
  }
}
```

🗒️ Notes

You will notice that it is a bit pixelated because the point is moving faster than the frame rate. We will address this in the next part.

Figure E6.30


p5*
File
Edit
Sketch
Help
English
Auto-refresh
Algorithmic Art
Sketch Files
index.html
particle.js
sketch.js
style.css
particle.js
Saved: just now
Preview
edges()
{
if (this.pos.x > width)
{
this.pos.x = 0
}
if (this.pos.x < 0)
{
this.pos.x = width
}
if (this.pos.y > height)
{
this.pos.y = 0
}
if (this.pos.y < 0)
{
this.pos.y = height
}
}
follow(vectors)
{
let x = floor(this.pos.x/scl)
let y = floor(this.pos.y/scl)
let index = x + (y * cols)
let force = vectors[index]
this.applyForce(force)
}
}
Console
Clear

Sketch E6.31 the previous position

We copy the position of the pixel before it moves and call `this.prevPos` using the `copy()` function. Then, when the pixel moves, it draws a line between its position and its previous position.

particle.js

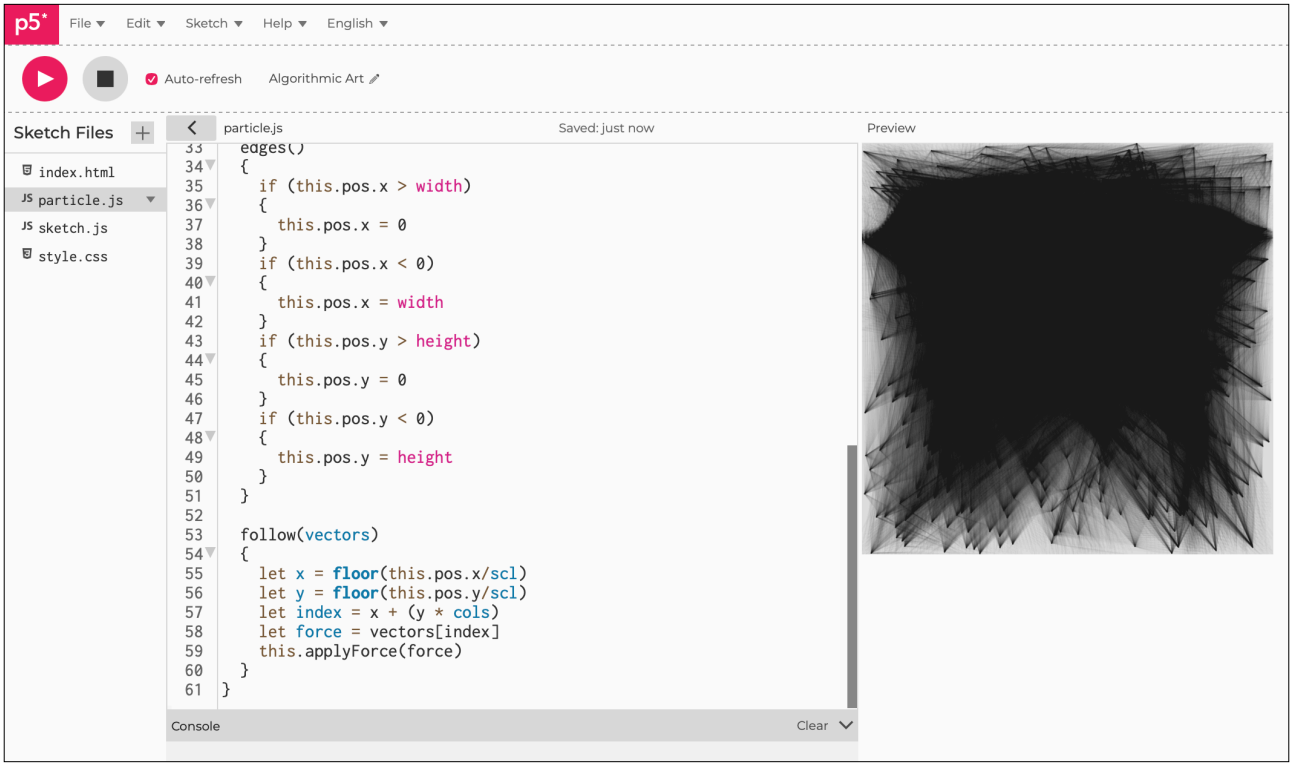
```
class Particle
{
  constructor()
  {
    this.pos = createVector(random(width), random(height))
    this.vel = createVector(0, 0)
    this.acc = createVector(0, 0)
    this.maxSpeed = 4
    this.prevPos = this.pos.copy()
  }

  move()
  {
    this.vel.add(this.acc)
    this.vel.limit(this.maxSpeed)
    this.pos.add(this.vel)
    this.acc.mult(0)
  }

  applyForce(force)
  {
    this.acc.add(force)
  }

  show()
  {
    strokeWeight(1)
    stroke(0, 5)
    line(this.pos.x, this.pos.y, this.prevPos.x, this.prevPos.y)
    // point(this.pos.x, this.pos.y)
  }
```

```
  edges()
  {
    if (this.pos.x > width)
    {
      this.pos.x = 0
    }
    if (this.pos.x < 0)
    {
      this.pos.x = width
    }
    if (this.pos.y > height)
    {
      this.pos.y = 0
    }
    if (this.pos.y < 0)
    {
      this.pos.y = height
    }
  }

  follow(vectors)
  {
    let x = floor(this.pos.x/scl)
    let y = floor(this.pos.y/scl)
    let index = x + (y * cols)
    let force = vectors[index]
    this.applyForce(force)
  }
}
```

🗒️ Notes

Hmmm, very quickly we have a problem because of the edges. However, we can rectify this.

Figure E6.31


p5*
File
Edit
Sketch
Help
English
Auto-refresh
Algorithmic Art
Sketch Files
index.html
particle.js
sketch.js
style.css
particle.js
Saved: just now
Preview
edges()
{
if (this.pos.x > width)
{
this.pos.x = 0
}
if (this.pos.x < 0)
{
this.pos.x = width
}
if (this.pos.y > height)
{
this.pos.y = 0
}
if (this.pos.y < 0)
{
this.pos.y = height
}
}
follow(vectors)
{
let x = floor(this.pos.x/scl)
let y = floor(this.pos.y/scl)
let index = x + (y * cols)
let force = vectors[index]
this.applyForce(force)
}
}
Console
Clear

Sketch E6.32 updating everything

We create another function to update the previous position.

particle.js

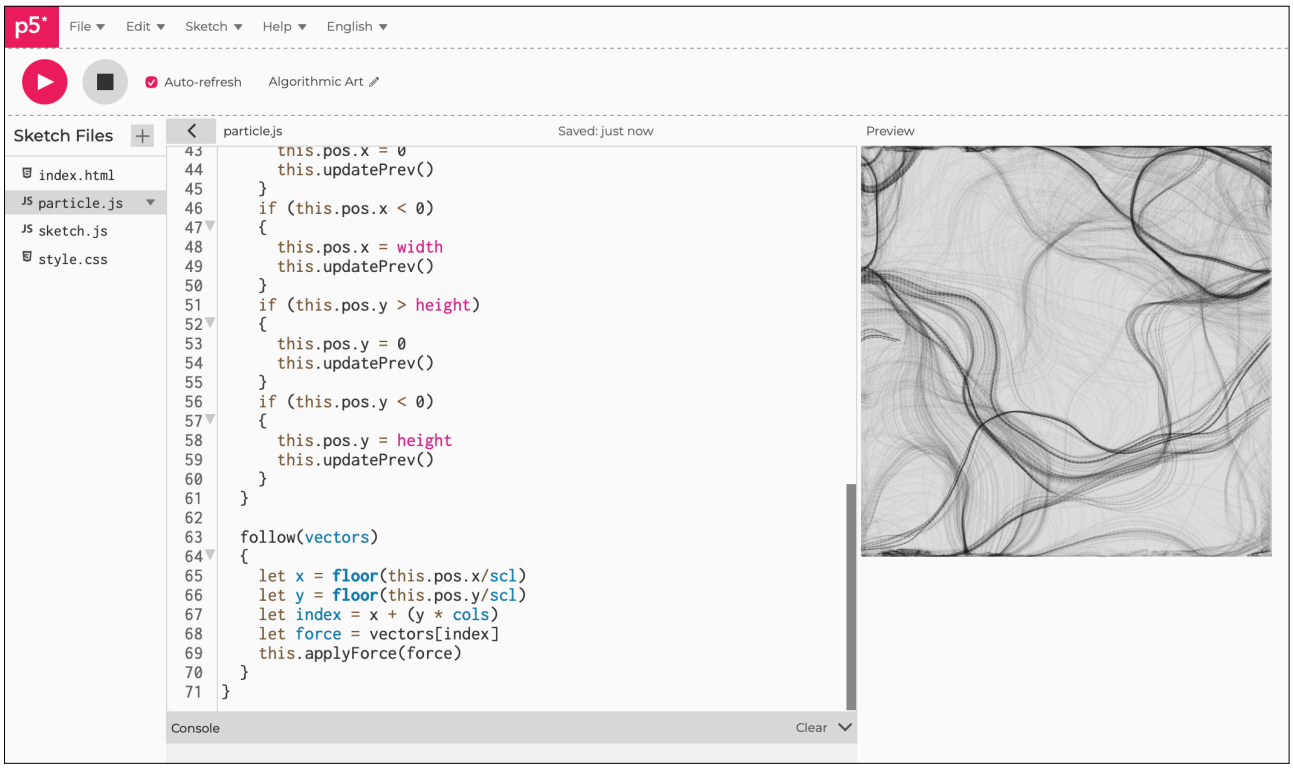
```
class Particle
{
  constructor()
  {
    this.pos = createVector(random(width), random(height))
    this.vel = createVector(0, 0)
    this.acc = createVector(0, 0)
    this.maxSpeed = 4
    this.prevPos = this.pos.copy()
  }

  move()
  {
    this.vel.add(this.acc)
    this.vel.limit(this.maxSpeed)
    this.pos.add(this.vel)
    this.acc.mult(0)
  }

  applyForce(force)
  {
    this.acc.add(force)
  }

  show()
  {
    strokeWeight(1)
    stroke(0, 5)
    line(this.pos.x, this.pos.y, this.prevPos.x, this.prevPos.y)
    this.updatePrev()
  }

  updatePrev()
```

```
  {
    this.prevPos.x = this.pos.x
    this.prevPos.y = this.pos.y
  }

  edges()
  {
    if (this.pos.x > width)
    {
      this.pos.x = 0
      this.updatePrev()
    }
    if (this.pos.x < 0)
    {
      this.pos.x = width
      this.updatePrev()
    }
    if (this.pos.y > height)
    {
      this.pos.y = 0
      this.updatePrev()
    }
    if (this.pos.y < 0)
    {
      this.pos.y = height
      this.updatePrev()
    }
  }

  follow(vectors)
  {
    let x = floor(this.pos.x/scl)
    let y = floor(this.pos.y/scl)
    let index = x + (y * cols)
    let force = vectors[index]
    this.applyForce(force)
  }
}
```

Notes

A much improved response.

Figure E6.32



Sketch E6.33 edges before show

❗ sketch.js

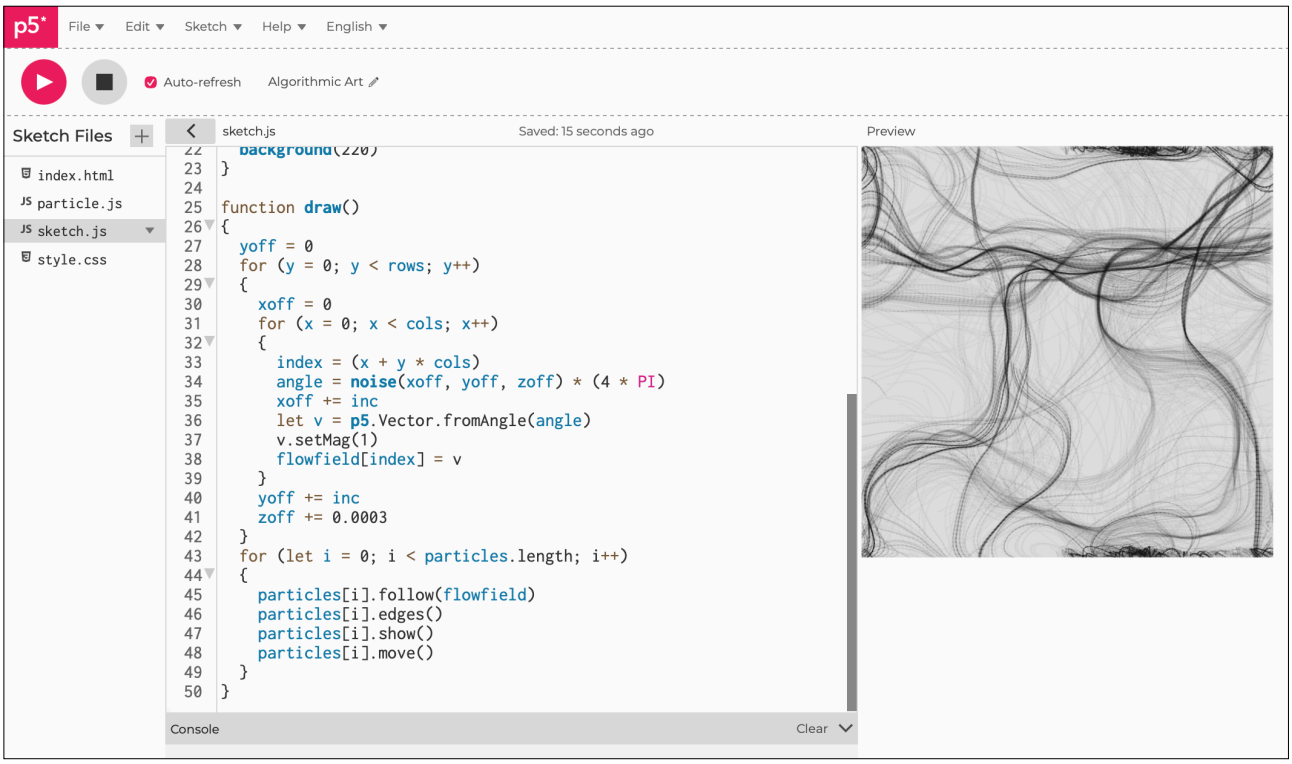
Move the `edges()` before `show()` so they are calculated first before drawing the particles/lines. Also removed the `pixelDensity()` and the drawing of lines (all commented out anyway //).

sketch.js

```
let xoff = 0
let yoff = 0
let zoff = 0
let inc = 0.1
let index
let angle
let scl = 20
let cols
let rows
let particles = []
let flowfield = []

function setup()
{
  createCanvas(400, 400)
  cols = floor(width / scl)
  rows = floor(height / scl)
  for (let i = 0; i < 300; i++)
  {
    particles[i] = new Particle()
  }
  background(220)
}

function draw()
{
  yoff = 0
  for (y = 0; y < rows; y++)
  {
    xoff = 0
```

```
    for (x = 0; x < cols; x++)
    {
      index = (x + y * cols)
      angle = noise(xoff, yoff, zoff) * (4 * PI)
      xoff += inc
      let v = p5.Vector.fromAngle(angle)
      v.setMag(1)
      flowfield[index] = v
    }
    yoff += inc
    zoff += 0.0003
  }
  for (let i = 0; i < particles.length; i++)
  {
    particles[i].follow(flowfield)
    particles[i].edges()
    particles[i].show()
    particles[i].move()
  }
}
```

Figure E6.33



Challenges

To improve the effect, I suggest altering the following:

1. Change the max force in `setMag()` value.
2. Alter `xoff`.
3. Change the amount of `alpha`.
4. Change the `strokeWeight()` to `0.5`.
5. Reduce the `maxSpeed`.
6. Change the number of particles.
7. Background to black and the lines to white/grey.
8. Introduce coloured lines.

All the above are tweaks to the parameters; an example below of what I mean.