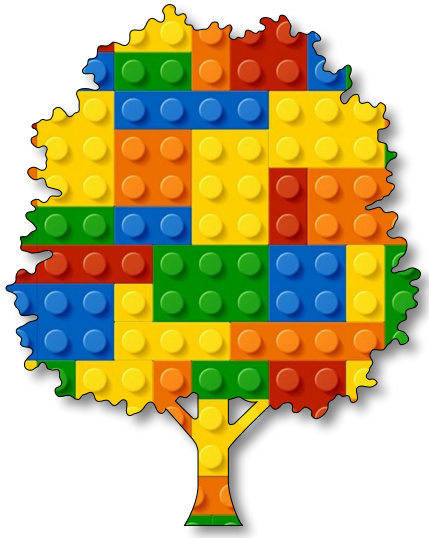
Algorithmic Art

Module E

Unit #3

2D perlin noise



Content

Section E Unit #3 perlin noise 2D

Introduction to 2D Perlin Noise

Although we touched on Perlin noise in the first unit, we will continue by revisiting it from another angle, which may help you in understanding Perlin noise intuitively before jumping into 2D Perlin noise.

Sketch E3.1 starting sketch

❗ Our starting sketch

This is a little bit of a recap of 1D Perlin noise; we will build on this very quickly.

```
function setup()
{
  createCanvas(400, 400)
}

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

Sketch E3.2 row of pixels

We create a `for()` loop and a point every pixel across the width of the canvas.

```
function setup()
{
  createCanvas(400, 400)
}

function draw()
{
  background(220)
  for (x = 0; x < width; x++)
  {
    point(x, 200)
  }
}
```

Figure E3.2

Sketch E3.3 vertex replaces point

Replace the point with a `vertex` and join them all up. You should get a line as before.

```
function setup()
{
  createCanvas(400, 400)
}

function draw()
{
  background(220)
  beginShape()
  for (x = 0; x < width; x++)
  {
    vertex(x, 200)
  }
  endShape()
}
```

Notes

You should have what looks like the same as before. Instead, we have line joined the points (pixels) but too small to see.

Sketch E3.4 random y

Instead of a `y` value of `200`, let's give it a random value between the top and the bottom of the canvas. We also include a `noLoop()` function so that we have a static image; if you comment it out, you will see the random effect animated.

```
function setup()
{
  createCanvas(400, 400)
}

function draw()
{
  background(220)
  noFill()
  beginShape()
  for (x = 0; x < width; x++)
  {
    vertex(x, random(height))
  }
  endShape()
  noLoop()
}
```

Figure E3.4

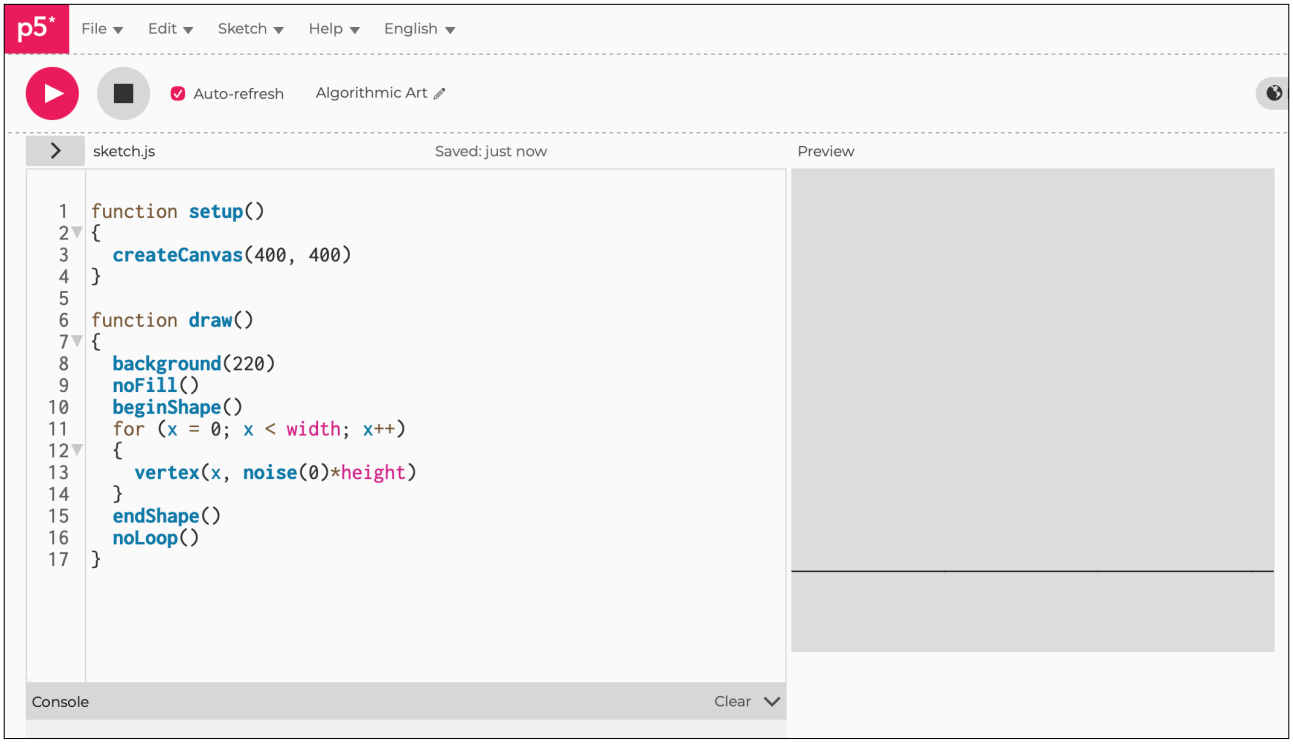

p5*
File
Edit
Sketch
Help
English
Auto-refresh
Algorithmic Art
sketch.js
Saved: just now
Preview
function setup()
{
createCanvas(400, 400)
}
function draw()
{
background(220)
noFill()
beginShape()
for (x = 0; x < width; x++)
{
vertex(x, random(height))
}
endShape()
noLoop()
}
Console
Clear

Sketch E3.5 introducing the perlin noise

We can introduce Perlin `noise()` rather than `random()`. We will get the `y` value at `0` and also multiply it by the `height` because noise returns a value between `0` and `1`. We will get a straight line somewhere between the top and the bottom of the canvas. Refresh and you will get a different value returned.

```
function setup()
{
  createCanvas(400, 400)
}

function draw()
{
  background(220)
  noFill()
  beginShape()
  for (x = 0; x < width; x++)
  {
    vertex(x, noise(0)*height)
  }
  endShape()
  noLoop()
}
```

Figure E3.5


p5*
File
Edit
Sketch
Help
English
Auto-refresh
Algorithmic Art
sketch.js
Saved: just now
Preview
1 function setup()
2 {
3 createCanvas(400, 400)
4 }
5
6 function draw()
7 {
8 background(220)
9 noFill()
10 beginShape()
11 for (x = 0; x < width; x++)
12 {
13 vertex(x, noise(0)*height)
14 }
15 endShape()
16 noLoop()
17 }
Console
Clear

Sketch E3.6 xoff

But what we have is a static value of Perlin noise. We want to move through the values, so we need another variable called `xoff` (short for x offset). Previously, we just used the variable time. This is what Perlin `noise()` random looks like compared to pure `random()` as we had before. Nice and smooth.

```
let xoff = 0

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

function draw()
{
  background(220)
  noFill()
  beginShape()
  for (x = 0; x < width; x++)
  {
    vertex(x, noise(xoff)*height)
    xoff += 0.01
  }
  endShape()
  noLoop()
}
```

Figure E3.6


p5*
File
Edit
Sketch
Help
English
Auto-refresh
Algorithmic Art
sketch.js
Saved: 15 seconds ago
Preview
let xoff = 0
function setup()
{
createCanvas(400, 400)
}
function draw()
{
background(220)
noFill()
beginShape()
for (x = 0; x < width; x++)
{
vertex(x, noise(xoff)*height)
xoff += 0.01
}
endShape()
noLoop()
}
Console
Clear

Sketch E3.7 a bit static

This is still static, and we want to animate it so that you can see it flowing. We will replace the initial value with a variable called `inc` (short for increment). Nothing changes yet.

```
let xoff = 0
let inc = 0.01

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

function draw()
{
  background(220)
  noFill()
  beginShape()
  for (x = 0; x < width; x++)
  {
    vertex(x, noise(xoff)*height)
    xoff += inc
  }
  endShape()
  noLoop()
}
```

Sketch E3.8 scrolling graph

Let's add another variable so that it doesn't begin at zero. We also comment out the noLoop() so we can animate. What you should get is a scrolling graph as Perlin `noise()` changes. We are effectively moving the `start` forward.

```
let xoff = 0
let inc = 0.01
let start = 0

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

function draw()
{
  background(220)
  noFill()
  xoff = start
  beginShape()
  for (x = 0; x < width; x++)
  {
    vertex(x, noise(xoff)*height)
    xoff += inc
  }
  endShape()
  start += inc
  // noLoop()
}
```

🗒️ Notes

You should have the Perlin `noise()` line scrolling slowly across the canvas.

🌻 Challenge

Try different `inc` values.

Sketch E3.9 noise detail

We can control the Perlin `noise()` with `noiseDetail()`. It takes two arguments, the default is `4` and `0.5`. The first one is the `octave` and the second one is the `fall-off`. Think detail rather than worry about what they mean.

```
let xoff = 0
let inc = 0.01
let start = 0

function setup()
{
  createCanvas(400, 400)
  noiseDetail(4, 0.5)
}

function draw()
{
  background(220)
  noFill()
  xoff = start
  beginShape()
  for (x = 0; x < width; x++)
  {
    vertex(x, noise(xoff)*height)
    xoff += inc
  }
  endShape()
  start += inc
  // noLoop()
}
```

🗒️ Notes

There is no appreciable difference immediately.

🌻 Challenge

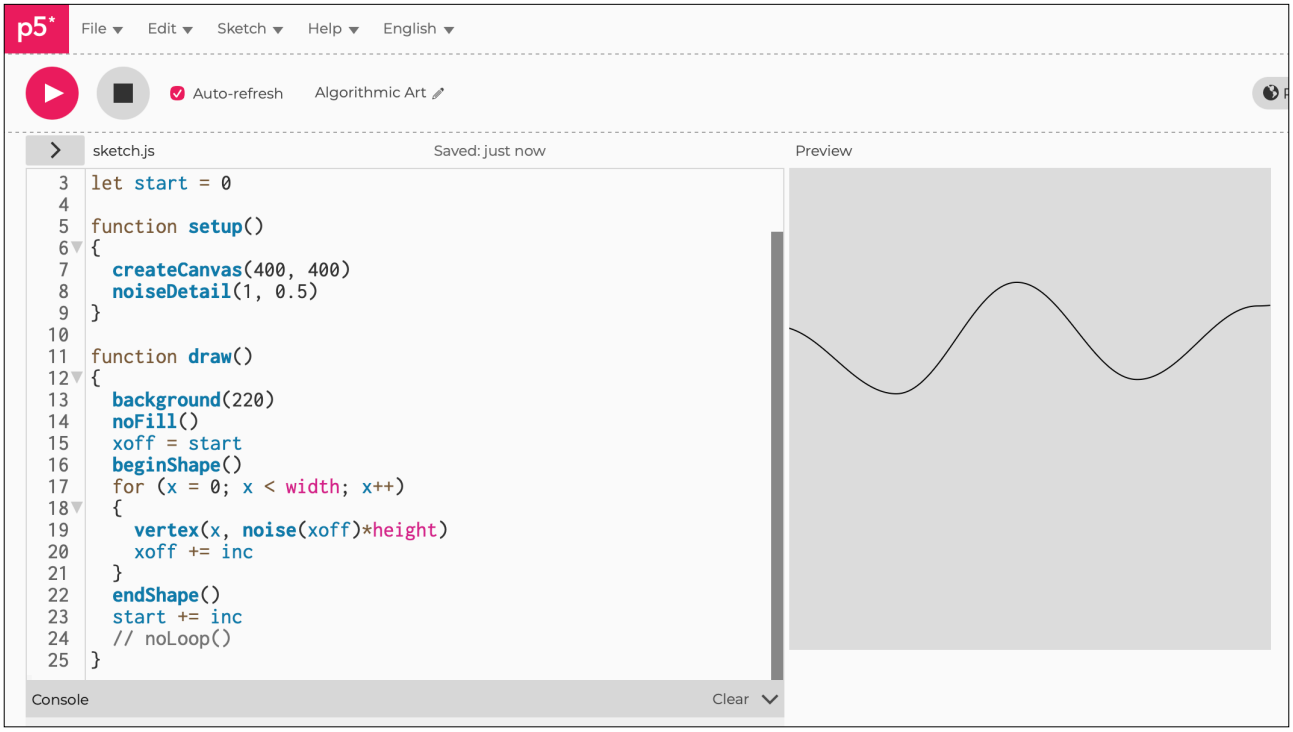
Play with the values. I recommend between `1` and `24` for the first argument and `0` to `1` for the second.

Sketch E3.10 smoothing

Creating a much smoother graph.

```
let xoff = 0
let inc = 0.01
let start = 0

function setup()
{
  createCanvas(400, 400)
  noiseDetail(1, 0.5)
}

function draw()
{
  background(220)
  noFill()
  xoff = start
  beginShape()
  for (x = 0; x < width; x++)
  {
    vertex(x, noise(xoff)*height)
    xoff += inc
  }
  endShape()
  start += inc
  // noLoop()
}
```

Figure E3.10

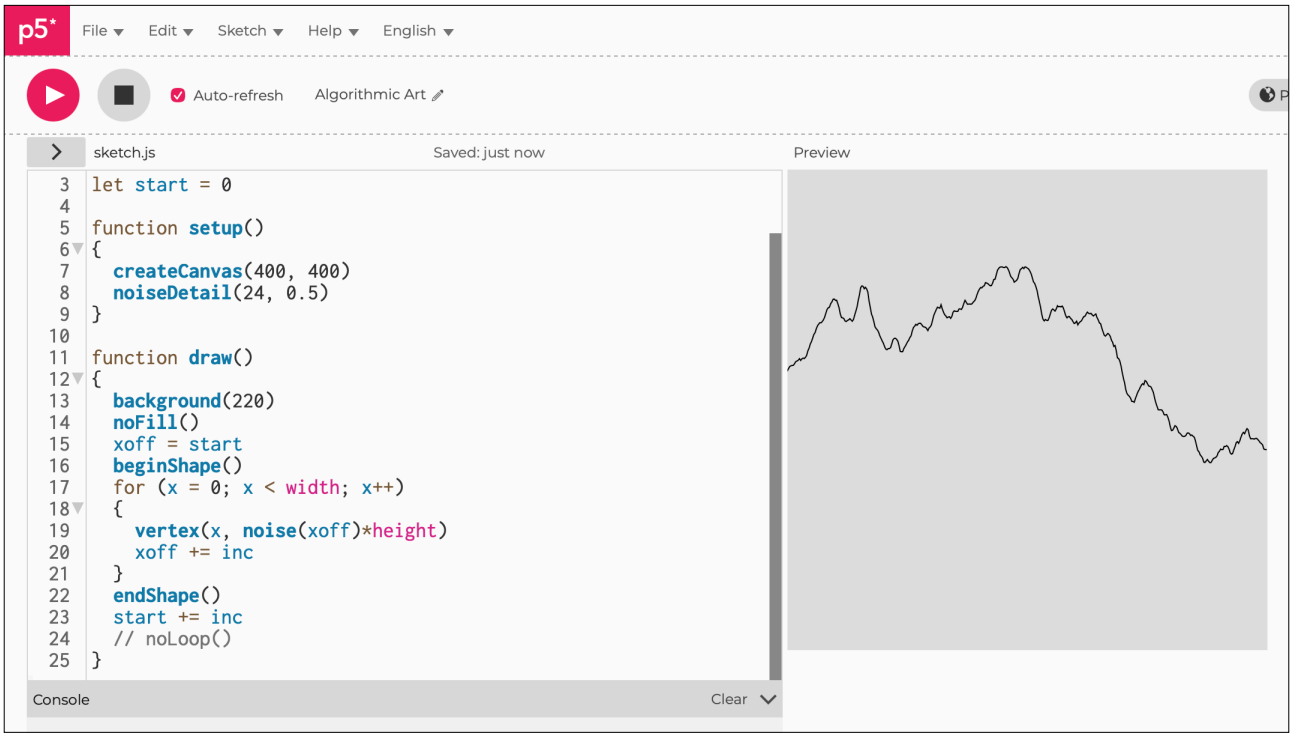

p5*
File
Edit
Sketch
Help
English
Auto-refresh
Algorithmic Art
sketch.js
Saved: just now
Preview
3 let start = 0
4
5 function setup()
6 {
7 createCanvas(400, 400)
8 noiseDetail(1, 0.5)
9 }
10
11 function draw()
12 {
13 background(220)
14 noFill()
15 xoff = start
16 beginShape()
17 for (x = 0; x < width; x++)
18 {
19 vertex(x, noise(xoff)*height)
20 xoff += inc
21 }
22 endShape()
23 start += inc
24 // noLoop()
25 }
Console
Clear

Sketch E3.11 not so smooth

Less smooth.

```
let xoff = 0
let inc = 0.01
let start = 0

function setup()
{
  createCanvas(400, 400)
  noiseDetail(24, 0.5)
}

function draw()
{
  background(220)
  noFill()
  xoff = start
  beginShape()
  for (x = 0; x < width; x++)
  {
    vertex(x, noise(xoff)*height)
    xoff += inc
  }
  endShape()
  start += inc
  // noLoop()
}
```

Figure E3.11


p5*
File
Edit
Sketch
Help
English
Auto-refresh
Algorithmic Art
sketch.js
Saved: just now
Preview
let start = 0
function setup()
{
createCanvas(400, 400)
noiseDetail(24, 0.5)
}
function draw()
{
background(220)
noFill()
xoff = start
beginShape()
for (x = 0; x < width; x++)
{
vertex(x, noise(xoff)*height)
xoff += inc
}
endShape()
start += inc
// noLoop()
}
Console
Clear

2D perlin noise

So far, we have only explored 1-dimensional (1D) noise(), which is the value of y as x moves along the x-axis (time). For 2D Perlin noise(), we need to consider all the surrounding values for any pixel on the canvas.

We are going to use the basis of the above sketch for the next section, which will generate a more interesting pattern for you to explore. This will form the basis for FlowFields later.

Sketch E3.12 time for 2D

Remove all the highlighted in blue (and commented out).

```
let xoff = 0
let inc = 0.01
// let start = 0

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

function draw()
{
  // background(220)
  // noFill()
  // xoff = start
  // beginShape()
  for (x = 0; x < width; x++)
  {
    // vertex(x, noise(xoff)*height)
    // xoff += inc
  }
  // endShape()
  // start += inc
  // noLoop()
}
```

Sketch E3.13 are we ready?

You should have something like this.

```
let xoff = 0
let inc = 0.01

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

function draw()
{
  for (x = 0; x < width; x++)
  {

  }
}
```

Notes

We have no canvas.

Sketch E3.14 loading the pixels

We are going to use a function called `loadPixels()` to examine whichever pixels we want to look at (in this case, all of them) and then `updatePixels()` after we have done something to them. We cycle through every pixel, hence the `y for()` loop.

```
let xoff = 0
let inc = 0.01

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

function draw()
{
  loadPixels()
  for (x = 0; x < width; x++)
  {
    for (y = 0; y < height; y++)
    {

    }
  }
  updatePixels()
}
```

Notes

Again, no canvas; be patient.

Sketch E3.15 four channels

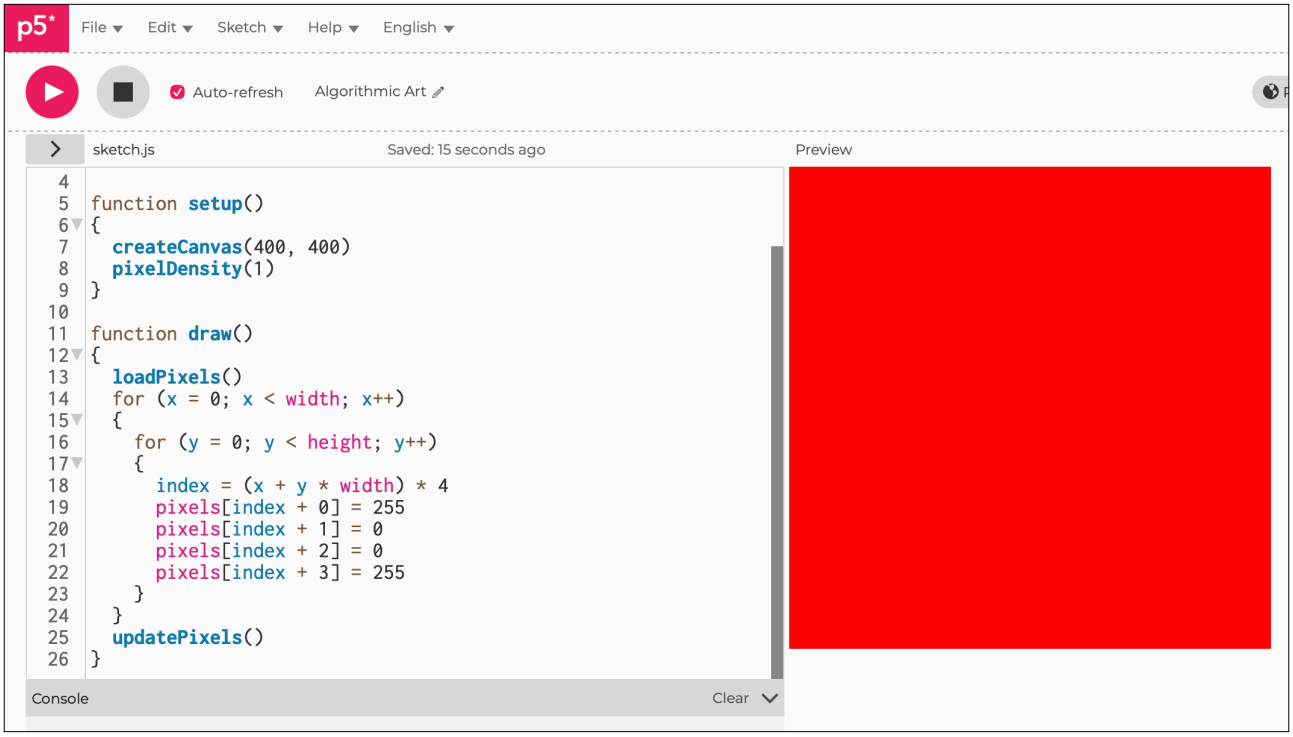
There are four channels: red, green, blue, and alpha for each pixel. So we are going to set the red to full (255) with no green or blue and full alpha (255). We have to set the pixelDensity() to 1 because of my display (Mac Retina); you may not need it depending on your monitor/screen.

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

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

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

Notes
What you will see is a red canvas. Play with the other values and see how it works.

Challenge
Create other colours.

Figure E3.15


p5*
File
Edit
Sketch
Help
English
Auto-refresh
Algorithmic Art
sketch.js
Saved: 15 seconds ago
Preview
4
5 function setup()
6 {
7 createCanvas(400, 400)
8 pixelDensity(1)
9 }
10
11 function draw()
12 {
13 loadPixels()
14 for (x = 0; x < width; x++)
15 {
16 for (y = 0; y < height; y++)
17 {
18 index = (x + y * width) * 4
19 pixels[index + 0] = 255
20 pixels[index + 1] = 0
21 pixels[index + 2] = 0
22 pixels[index + 3] = 255
23 }
24 }
25 updatePixels()
26 }
Console
Clear

Sketch E3.16 random static

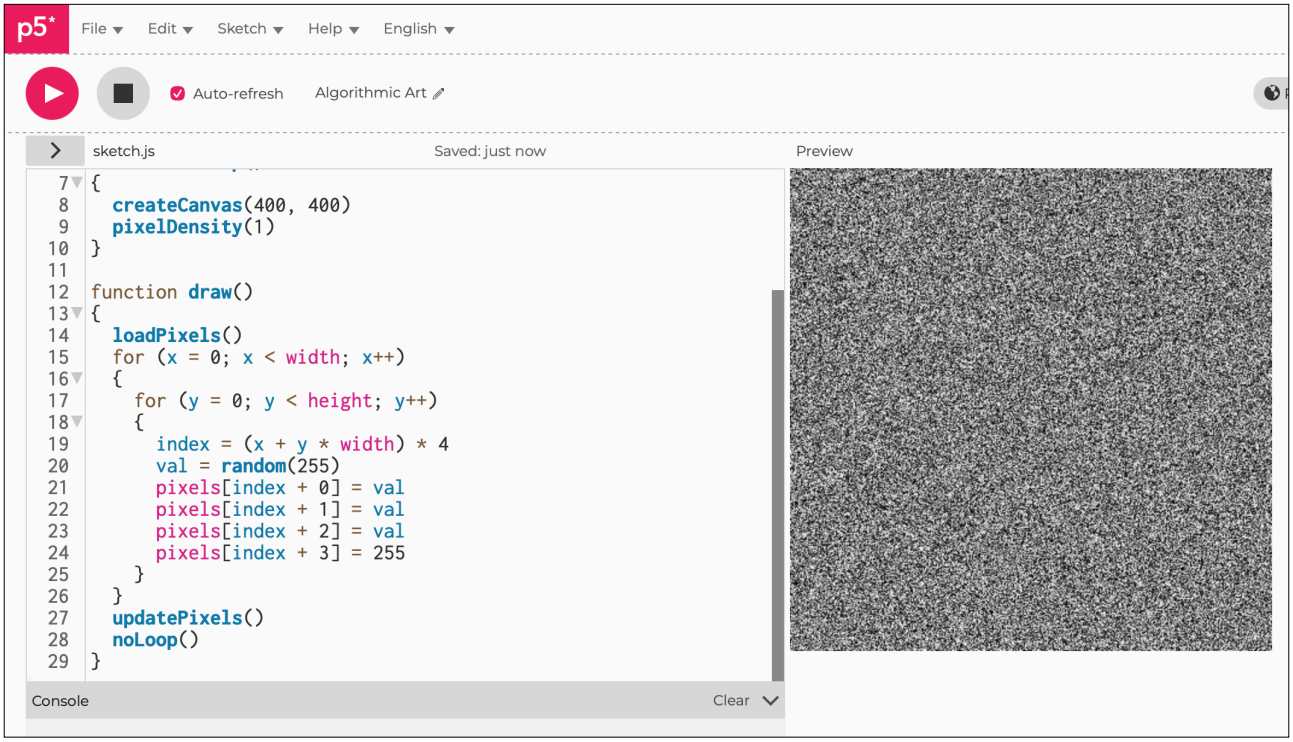
If you randomise the values for RGB, you get a random image (add `noLoop()` to stop it flickering). However, each pixel has no correlation with any of the surrounding pixels; every pixel is random in isolation.

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

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

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

Notes

What do you think would happen if we gave each pixel its separate random value?

Figure E3.16


p5*
File
Edit
Sketch
Help
English
Auto-refresh
Algorithmic Art
sketch.js
Saved: just now
Preview
{
createCanvas(400, 400)
pixelDensity(1)
}
function draw()
{
loadPixels()
for (x = 0; x < width; x++)
{
for (y = 0; y < height; y++)
{
index = (x + y * width) * 4
val = random(255)
pixels[index + 0] = val
pixels[index + 1] = val
pixels[index + 2] = val
pixels[index + 3] = 255
}
}
updatePixels()
noLoop()
}
Console
Clear

Sketch E3.17 make it noisy

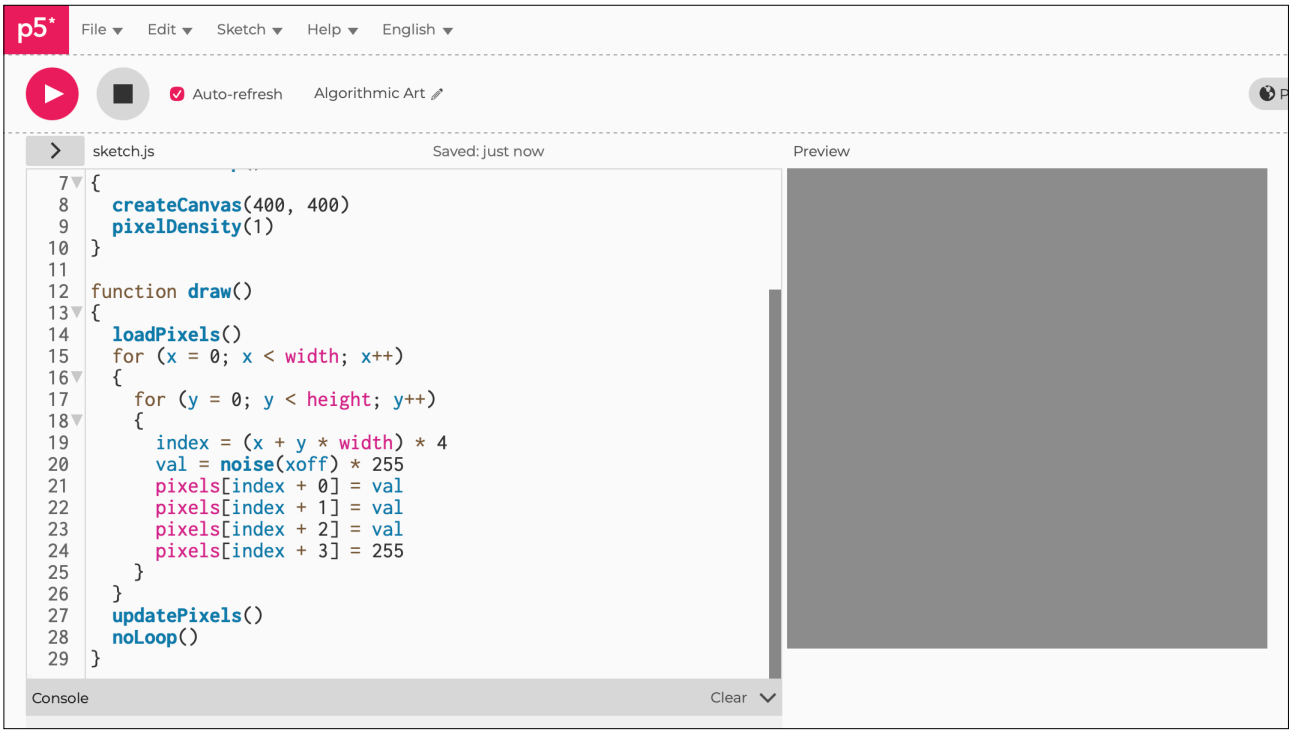
Next step is to introduce `noise()` so that we can have the situation where each pixel is related to the surrounding pixels. We first replace `random()` with `noise()` and multiply by `255` because `noise()` returns a value between `0` and `1`, what we want are values between `0` and `255`. Every time you refresh the sketch, you get a different shade of grey.

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

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

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

Notes

You get a single colour of some grey value.

Figure E3.17



Sketch E3.18 a wee amount

If we change each pixel `xoff` value by a small amount, we get the following effect (which is still not it, by the way).

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

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

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

Figure E3.18


p5*
File
Edit
Sketch
Help
English
Auto-refresh
Algorithmic Art
sketch.js
Saved: just now
Preview
createCanvas(400, 400)
pixelDensity(1)
}
function draw()
{
loadPixels()
for (x = 0; x < width; x++)
{
for (y = 0; y < height; y++)
{
index = (x + y * width) * 4
val = noise(xoff) * 255
pixels[index + 0] = val
pixels[index + 1] = val
pixels[index + 2] = val
pixels[index + 3] = 255
xoff += inc
}
}
updatePixels()
noLoop()
}
Console
Clear

Sketch E3.19 reversing the loops

Presently, we have the `y loops()` inside the `x`, but they need to be the other way round. We do a `row` first, so `y` needs to stay static as we move through from left to right.

```
let xoff = 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++)
  {
    for (x = 0; x < width; x++)
    {
      index = (x + y * width) * 4
      val = noise(xoff) * 255
      pixels[index + 0] = val
      pixels[index + 1] = val
      pixels[index + 2] = val
      pixels[index + 3] = 255
      xoff += inc
    }
  }
  updatePixels()
  noLoop()
}
```

Notes

What is happening is that the Perlin noise() value is changing as it goes across the canvas and then continues at the start of the next row of pixels. So what you are seeing is Perlin noise(), but it has no relationship with the value above or below.

Figure E3.19

Sketch E3.20 yoff

We want a `yoff` to work downwards as well, but when it moves down a row, we want the `xoff` to reset, not carry on with the Perlin noise() increments.

```
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

What you get is the image where each pixel has now some relationship with its neighbouring ones. If the lines of code above aren't obvious, this is where you take the time to work through slowly what is happening to each pixel as you move across the canvas and then onto the next row of pixels. This is where the logic of coding comes in.

🌻 Challenges

1. Change the `inc` value to see the effect.
2. Add noise values for each colour (RGB).
3. Use `noiseDetail()` to see what difference you can make to the image.

Figure E3.20