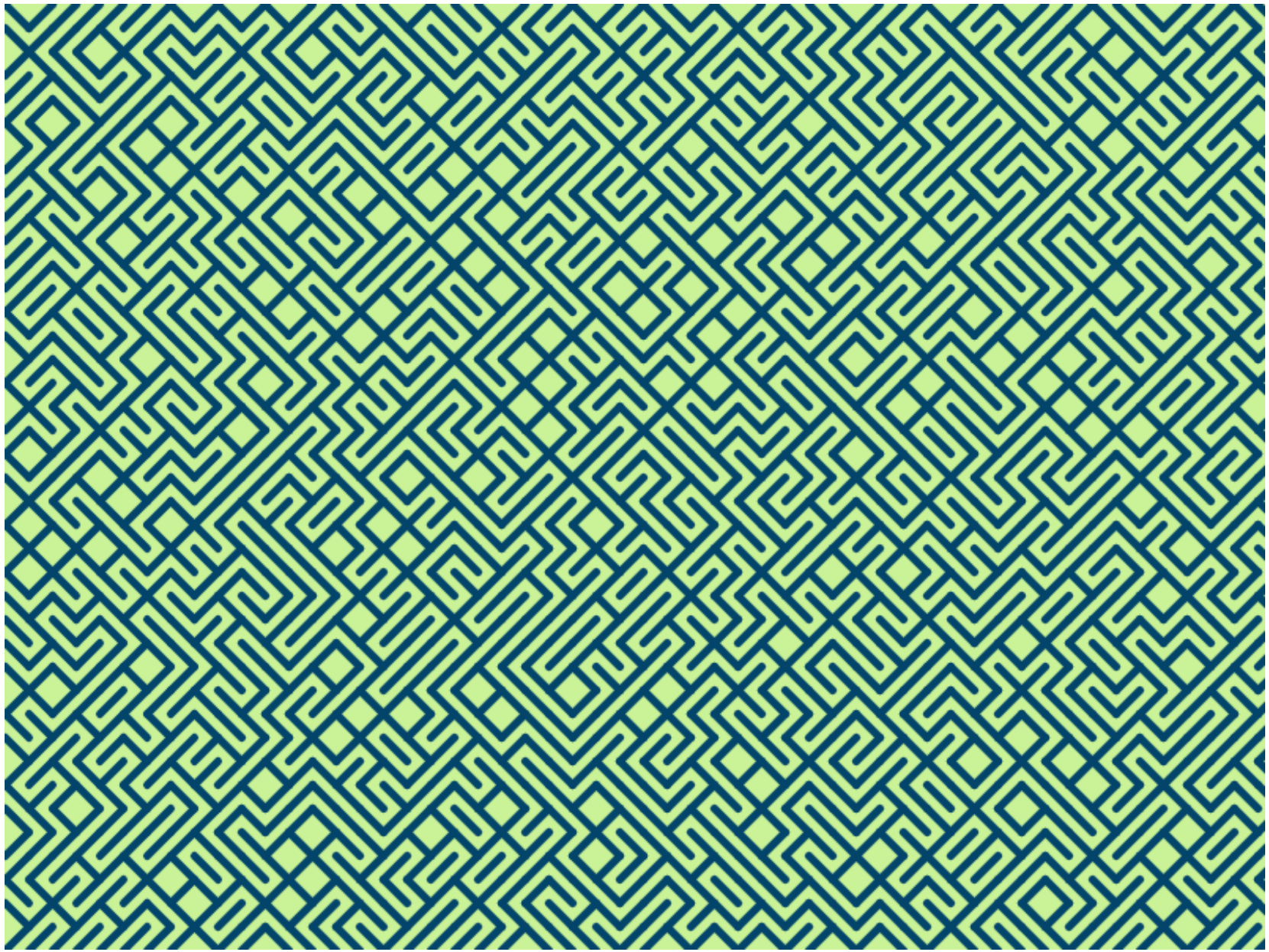

Generative Art

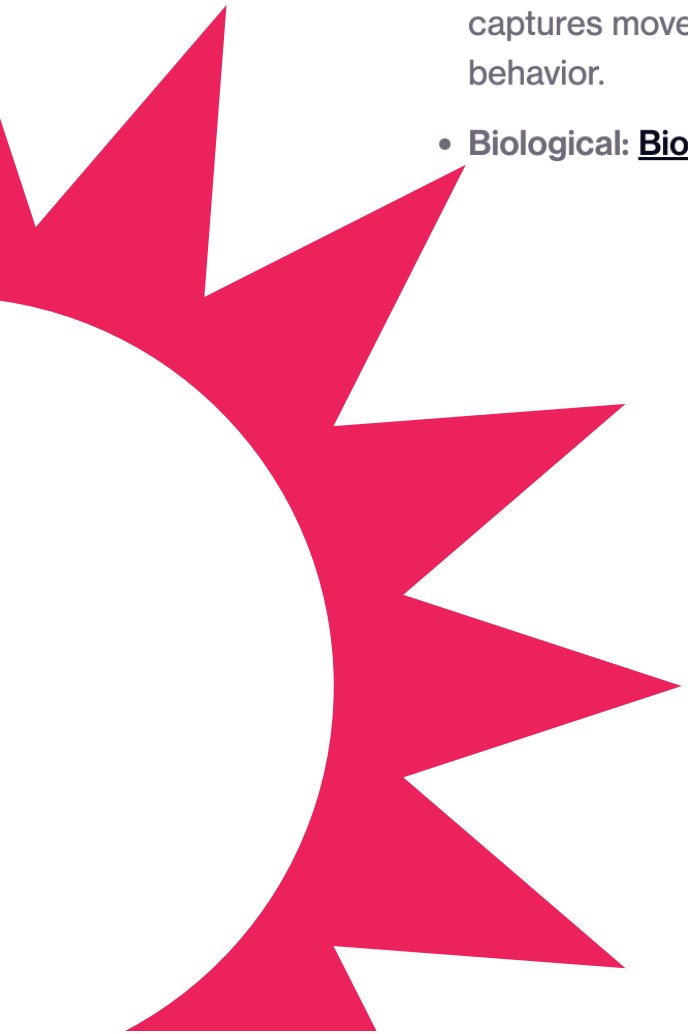


http://www.galaxykate.com/apps/Prototypes/LTrees/

**http://math.hws.edu/eck/js/edge-of-chaos/CA.html**

> **Generative art is a form of art that is either completely or partially created using an autonomous system.**

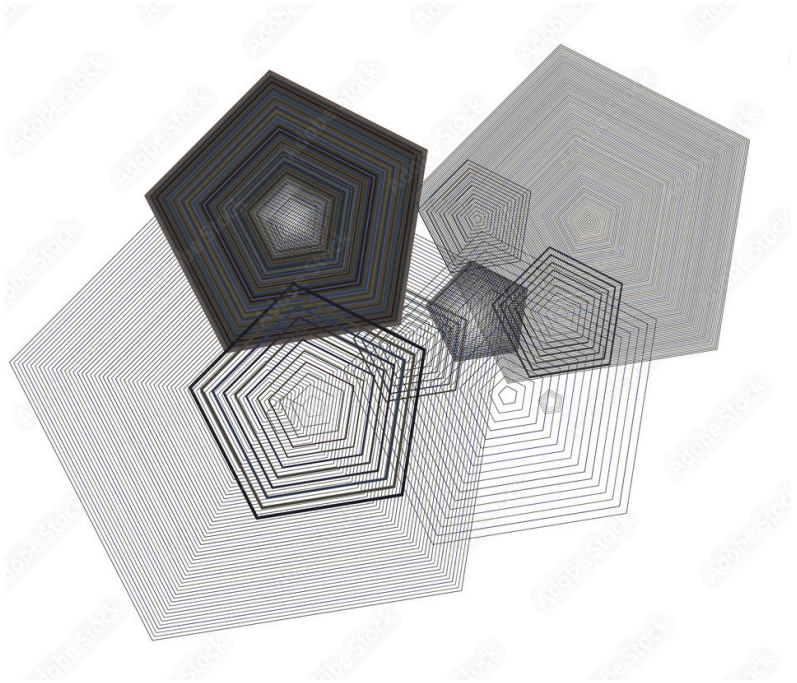
- **Mathematical:** For instance, the ***'Game of Life'*** by John Conway describes a lined field with cells inhabited by blocks that can maneuver among each other using certain rules. An artist can take this algorithm as a foundation and use it to draw.
- **Mechanical:** In kinetic art, an artist creates a **special algorithm** that wavers in the wind and captures movement. In this case, the autonomous system is an algorithm of windblast behavior.
- **Biological:** **Bio-art** enthusiasts use bacteria pattern behavior to create their unusual works.



**https://editor.p5js.org/**

# What goes into a piece of generative art?

- **Randomness** is important for creating generative art that is different each time. The art should be different each time you run the generation script.
- **Algorithms** — Implementing an algorithm visually can often generate awesome art, for example, the binary tree above.
- **Geometry** — Most generative art incorporates shapes, and the math from high school geometry class can aid in some really cool effects.

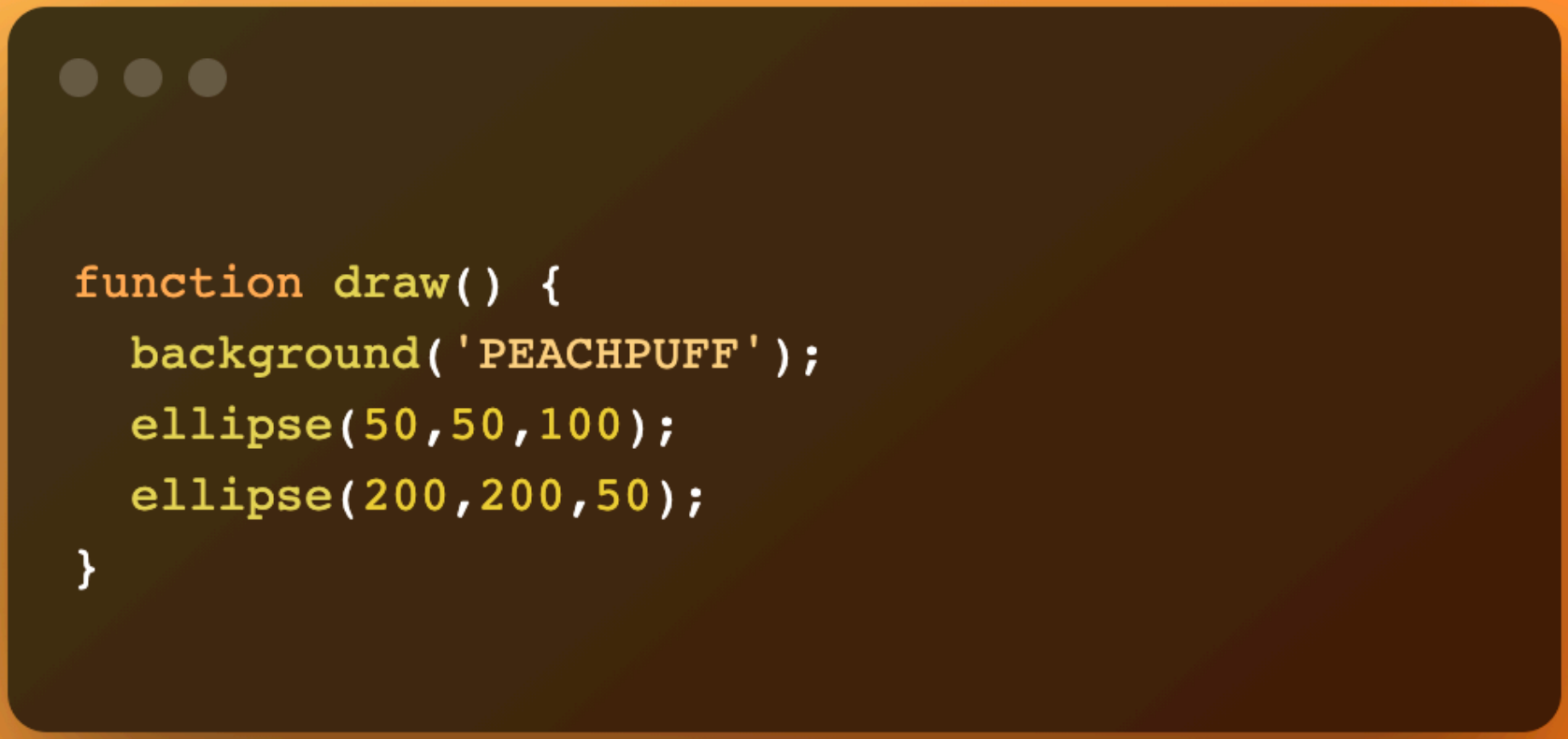


sketch.js

```
1  function setup() {
2    createCanvas(400, 400);
3  }
4
5  function draw() {
6    background(220);
7  }
```

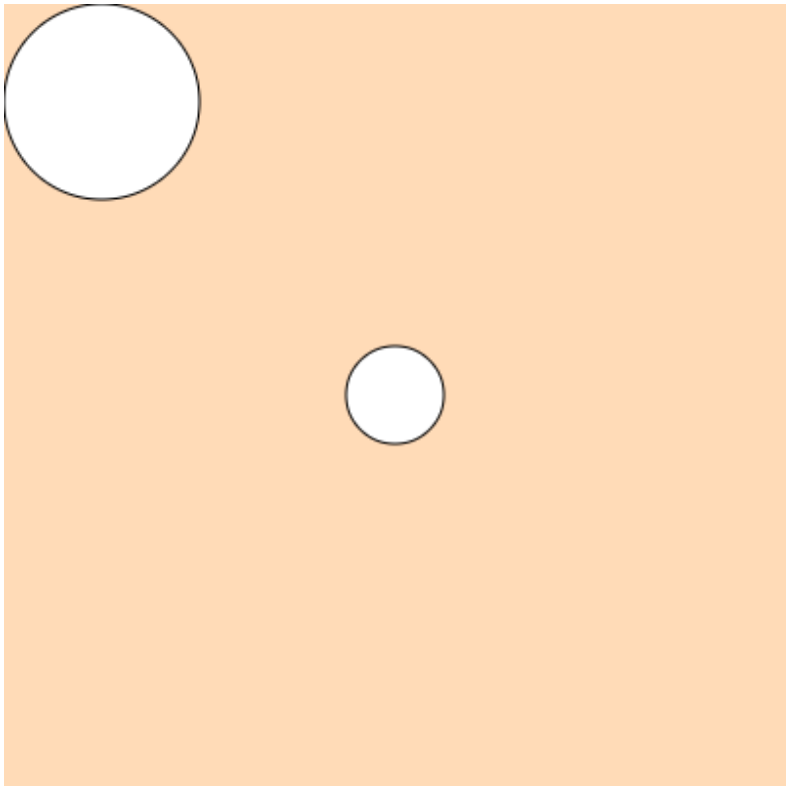
**Code is a precise way of explaining something and it is written in a programming language.**

Preview


function draw() {
background('PEACHPUFF');
ellipse(50,50,100);
ellipse(200,200,50);
}

PEACHPUFF is from CSS colours. You can check out some of these colours:

http://colours.neilorangepeel.com/



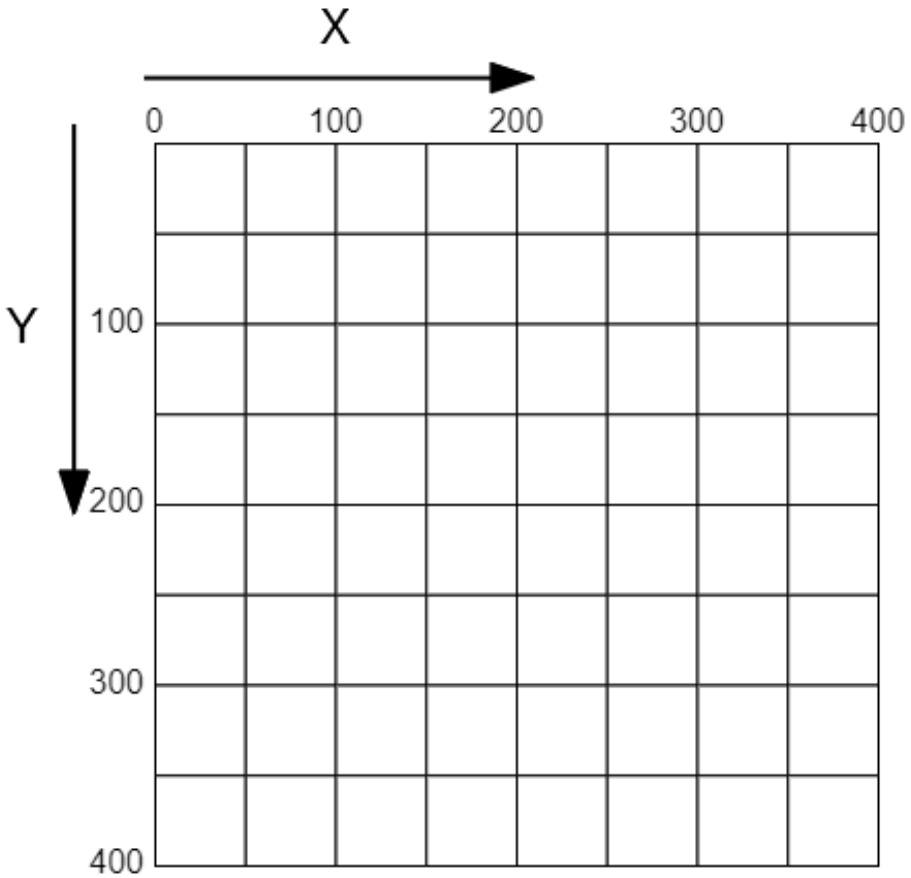
**Some other examples are**:

- mediumslateblue
- darkmagenta
- mediumturquoise

```
line(startX, startY, endX, endY)

//circle
ellipse(xCoordinate, yCoordinate, width, height)
//square
rect(xCoordinate, yCoordinate, width, height)

//triangle
triangle(xCoordinate1, yCoordinate1, x2, y2, x3, y3)
```

```
ellipse(50,50,100);
```

```
line(0,0,400,400);
```

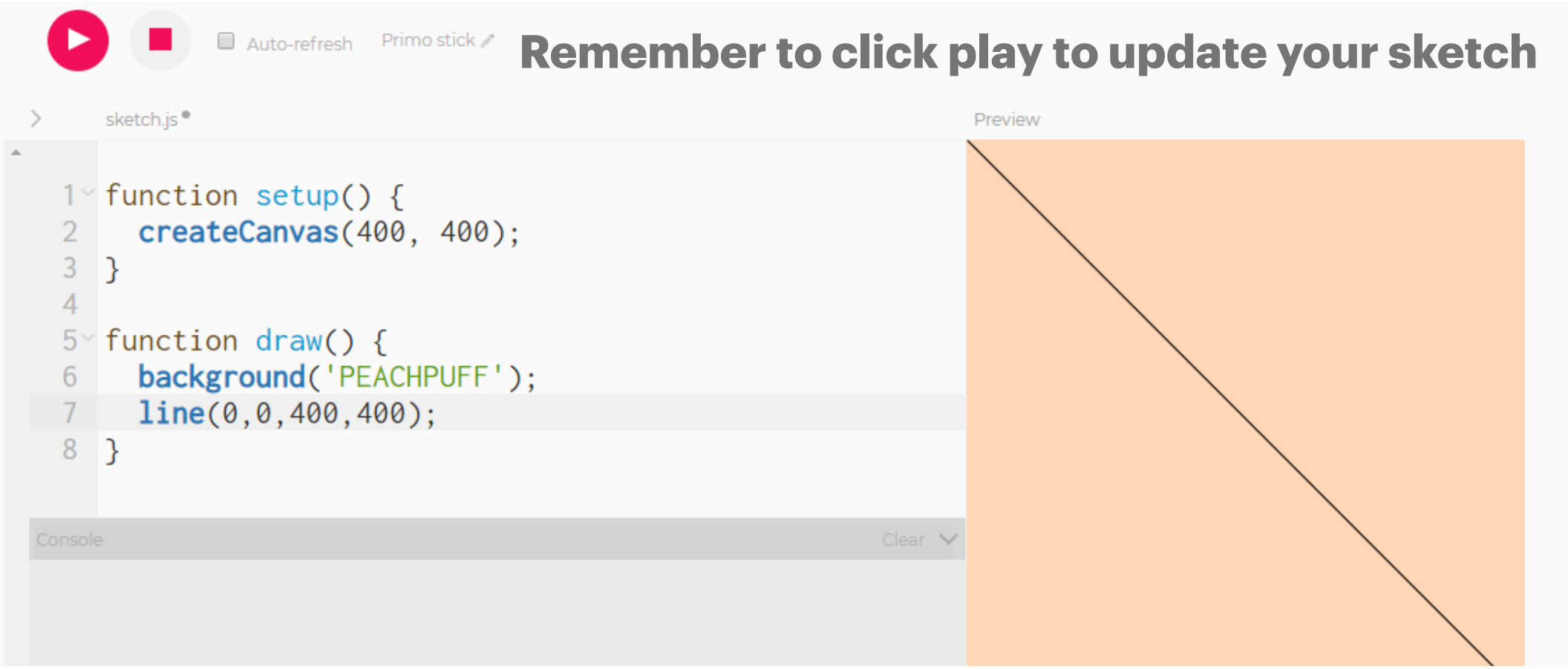



This means the line goes from (0,0) to (400, 400) ->
the top left corner to the bottom right.

in your JavaScript file, you will add two functions that will be used in pretty much ever p5 script: `setup` and `draw`. These names are necessary for p5 to call them.

`setup` is called when the program starts. You will probably create a canvas to draw on within it using the `createCanvas` function, and you may set some defaults there. It is only called once!

`draw` is called after setup, and is executed constantly until you call `noLoop`, which will stop it from running again. You can control how many times `draw` runs each second with the `frameRate` function.



# For Loops

for loops

```
for (var count = 0; count < 400; count = count + 100) {
  doAnAction
}
```

1) it declares a variable `var count = 0`

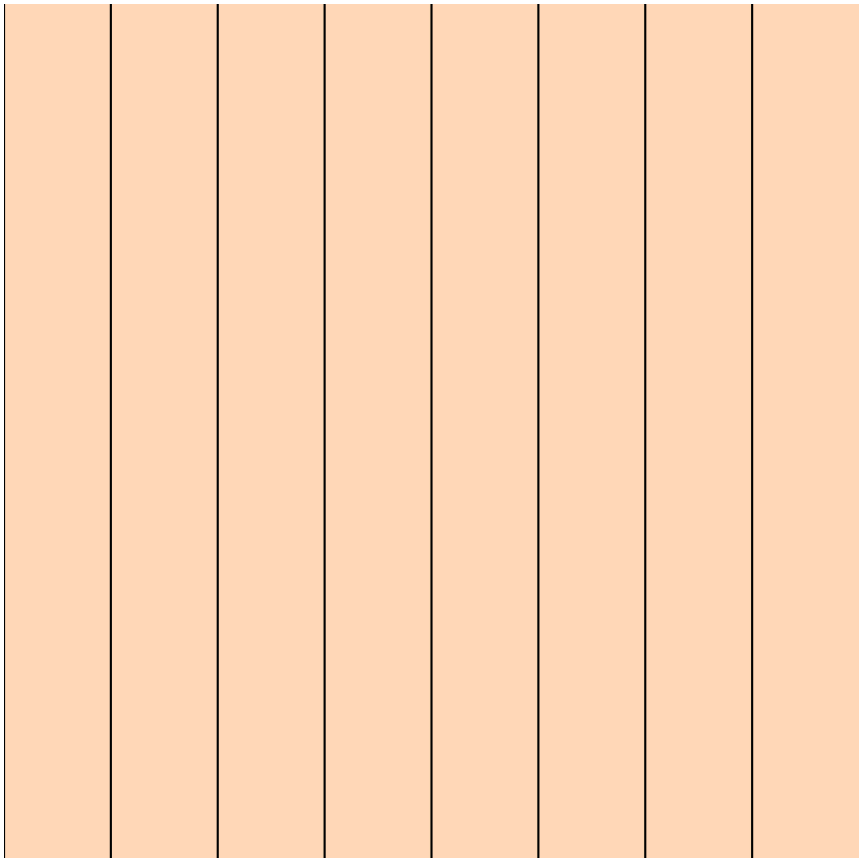
2) it then adds one hundred *after* running the loop: `count = count + 100`

3) it then checks if count is < 400 with `count < 400`

`for (var x=0; x<10; x=x+1)` goes from 0, 1, 2, 3..., 8, 9! (10 is not less than 10, so we exit without running the code.

```
function setup() {
  createCanvas(400, 400);
  noLoop();
}
function draw() {
  background('PEACHPUFF');
  for (var x=0; x<400; x=x+50) {
    line(x,0,x,400);
  }
}
```

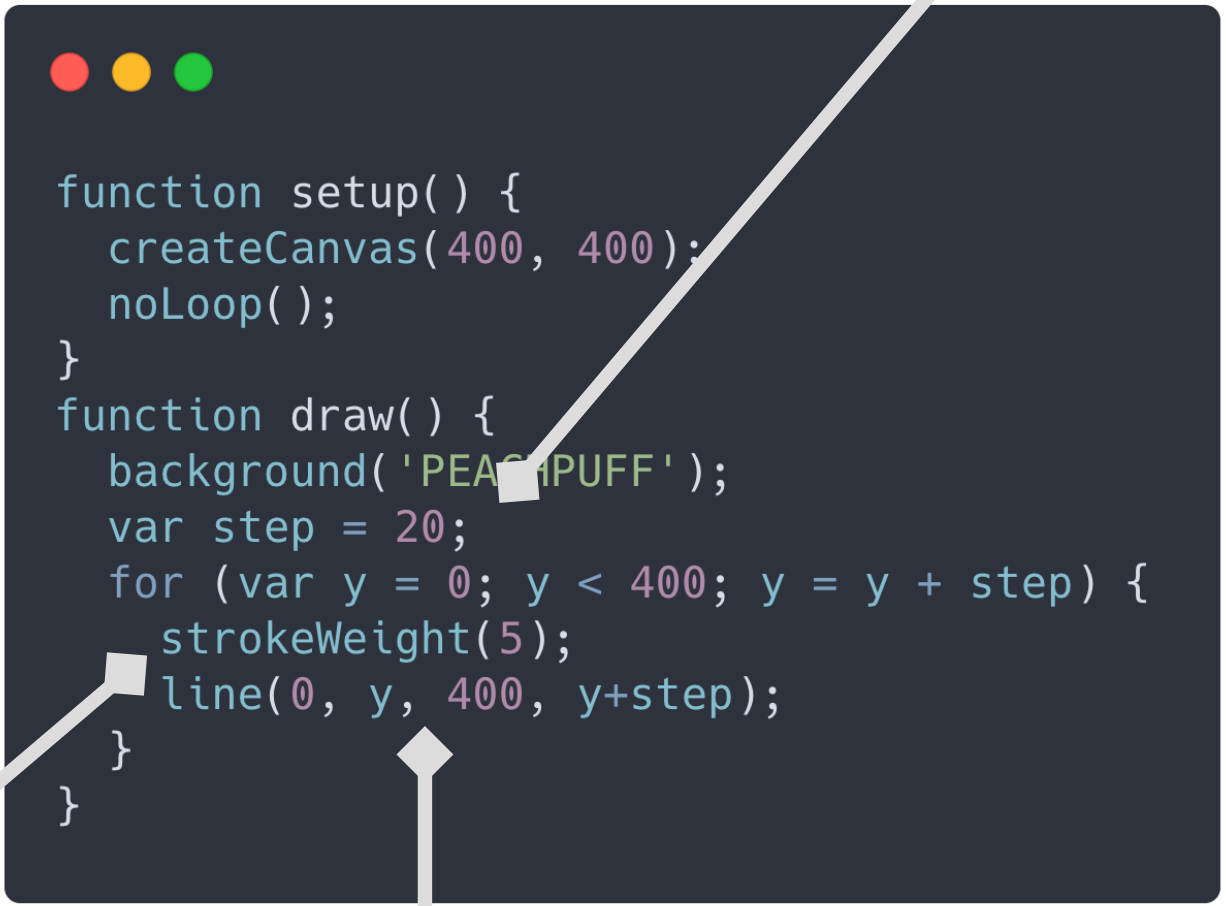
draws lines (400/50 = 8 times)



We can use the for loop in our drawing! This draws multiple lines, as expected.

# What does this draw?

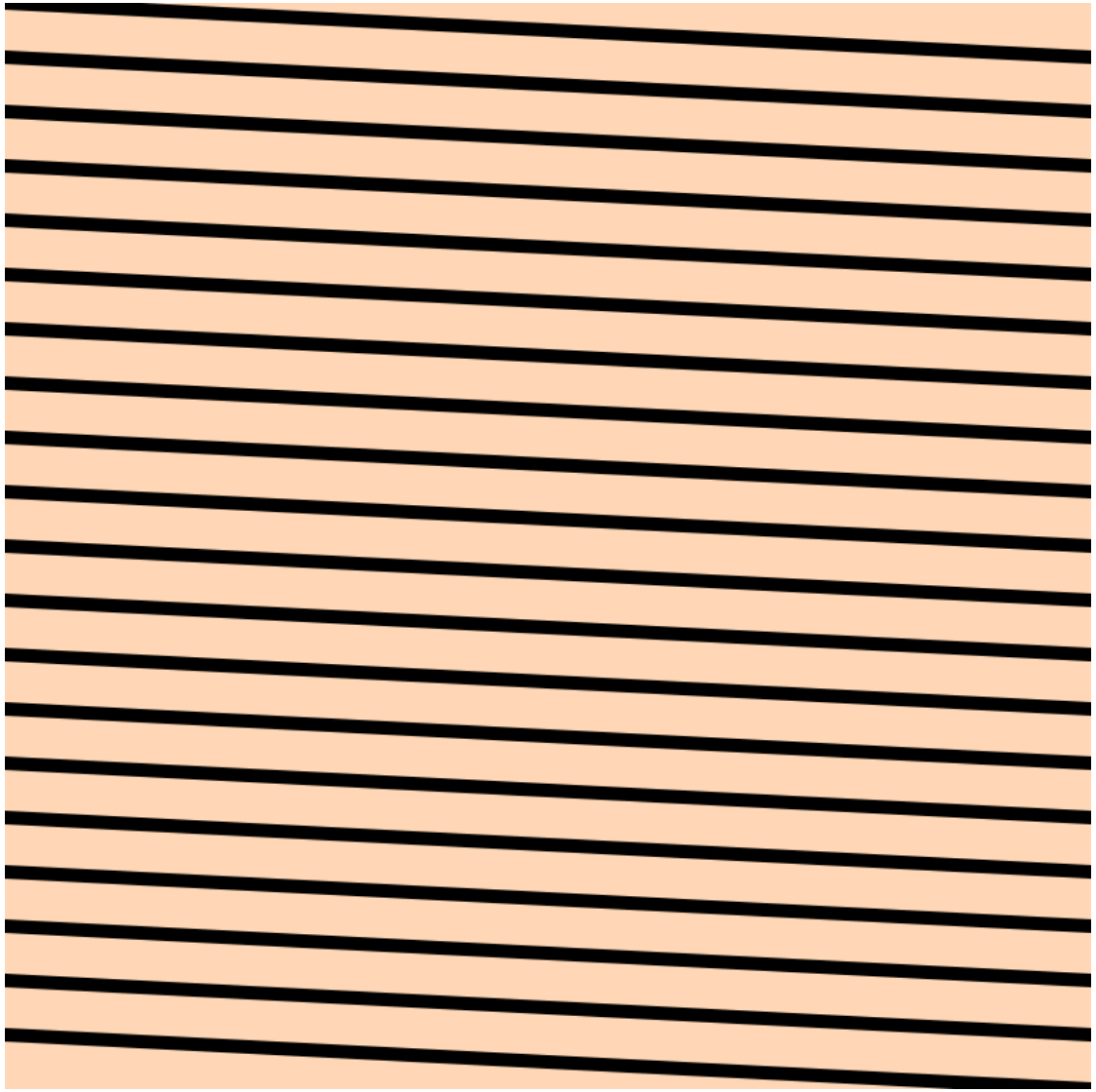
The gap between the lines



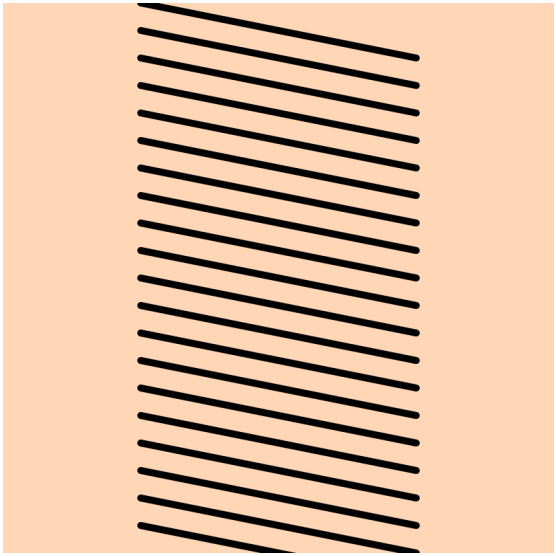
```
function setup() {
  createCanvas(400, 400);
  noLoop();
}
function draw() {
  background('PEACHPUFF');
  var step = 20;
  for (var y = 0; y < 400; y = y + step) {
    strokeWeight(5);
    line(0, y, 400, y+step);
  }
}
```

Sets the lines' thickness

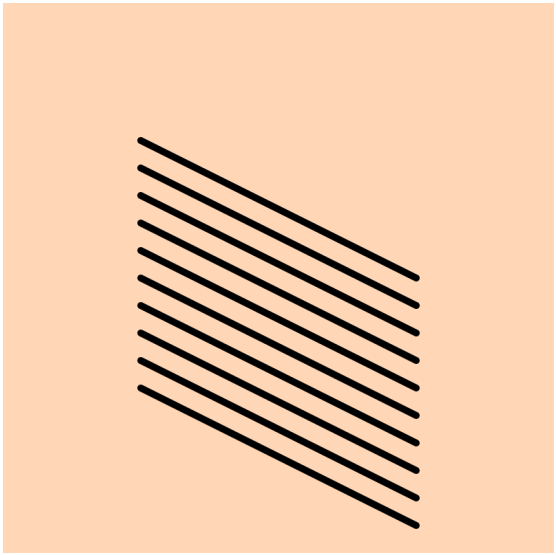
Starts are same horizontal position, and moves up
Ends at bottom and also moves up



```
function setup() {
  createCanvas(400, 400);
  noLoop();
}
function draw() {
  background('PEACHPUFF');
  var step = 20;
  for (var y = 0; y < 400; y = y + step) {
    strokeWeight(5);
    line(100, y, 300, y+step*2);
  }
}
```



```
function setup() {
  createCanvas(400, 400);
  noLoop();
}
function draw() {
  background('PEACHPUFF');
  var step = 20;
  for (var y = 100; y < 300; y = y + step) {
    strokeWeight(5);
    line(100, y, 300, y+step*5);
  }
}
```



# For loops inside for loops!

```
for (var x=0;x<5;x=x+1) {
  for (var y=0;y<5;y=y+1) {
    drawTheLines;
  }
}
```

```
x=0, y=0; x=0, y=1; x=0, y=2; x=0, y=3; x=0, y=4;
x=1, y=0; x=1, y=1; x=1, y=2; x=1, y=3; x=1, y=4;
x=2, y=0; x=2, y=1; x=2, y=2; x=2, y=3; x=2, y=4;
x=3, y=0; x=3, y=1; x=3, y=2; x=3, y=3; x=3, y=4;
x=4, y=0; x=4, y=1; x=4, y=2; x=4, y=3; x=4, y=4;
```

Here's what it looks like, in actuality...

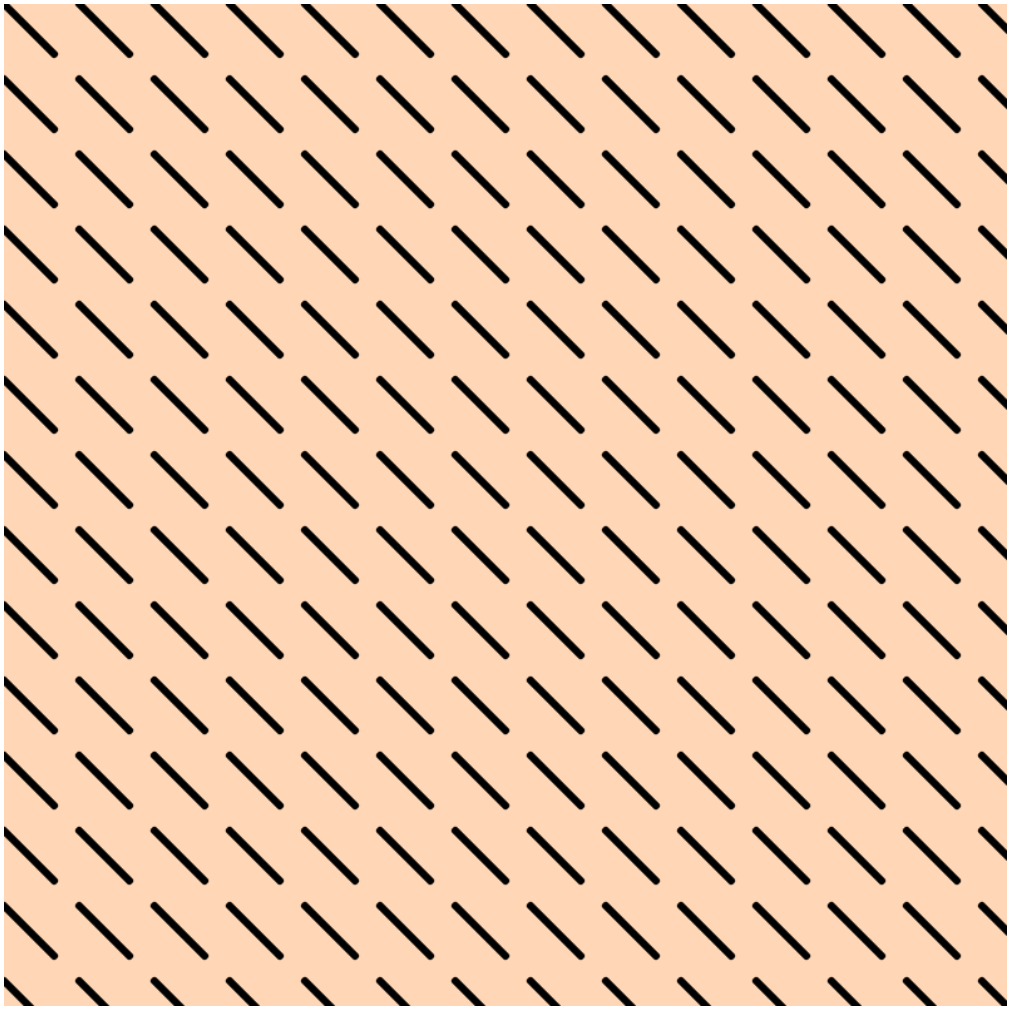
```
function setup() {
  createCanvas(400, 400);
  noLoop();
}
function draw() {
  background('PEACHPUFF');

  for (var x=0;x<400;x=x+1) {
    for (var y=0;y<400;y=y+1) {
       strokeWeight(3);
        line(x,y,x+20, y+20);
    }
  }
}
```

We go from (0,0) to (400,400)!

What do you think this outputs?

Ha! It’s all black, because the changes are just x=x+1 and y=y+1.

Too small of a change so we just fill it all.



```
function setup() {
  createCanvas(400, 400);
  noLoop();
}
function draw() {
  background('PEACHPUFF');
  var step = 30;
  for (var x=0;x<400;x=x+step) {
    for (var y=0;y<400;y=y+step) {
       strokeWeight(3);
        line(x,y,x+20, y+20);
    }
  }
}
```

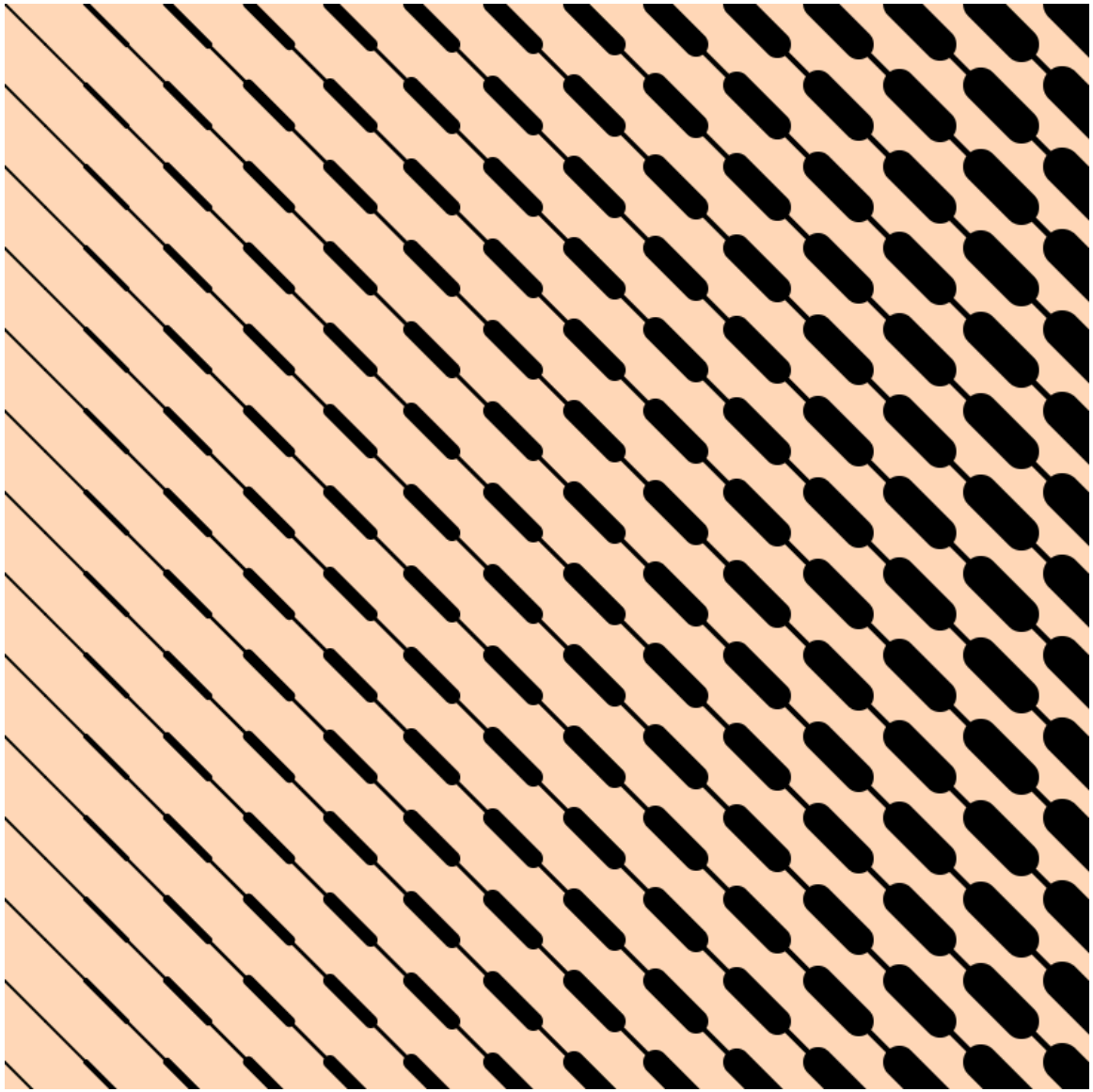
Notice how step is > 20, which is the jump we have in the lines.

This makes the gap effect!

```
function setup() {
  createCanvas(400, 400);
  noLoop();
}
function draw() {
  background('PEACHPUFF');
  var step = 30;
  for (var x=0;x<400;x=x+step) {
    for (var y=0;y<400;y=y+step) {
      strokeWeight(1+x/step);
      line(x,y,x+(step/2),y+(step/2));
      strokeWeight(1+x/step/10);
      line(x+(step/2),y+(step/2),x+step,y+step);
    }
  }
}
```

We change the `strokeWeight` to be heavier as we near (400,400)

The lines also have some that jump = step and
some < step (more precisely, step/2)



# `random()`

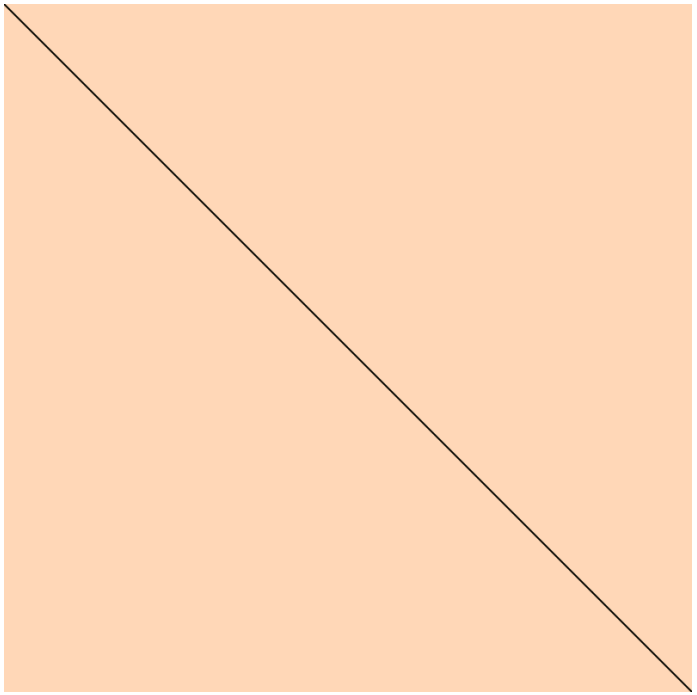
This command will result in a random value between **0 and 1**. That means around half the time the value will be bigger than 0.5 and around half the time the value will be less than 0.5.

So what we could do is use the `random()` command and draw a left line if the value is bigger than 0.5, otherwise, we draw the right-to-left line.

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

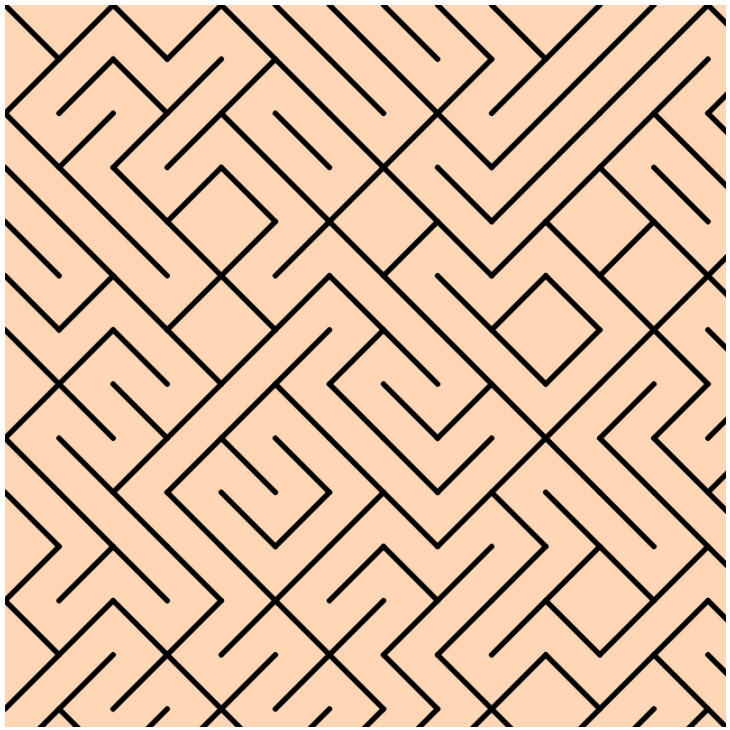
```
if (random()>0.5) {
  doAnAction
}
else {
  doSomeOtherAction
}
```

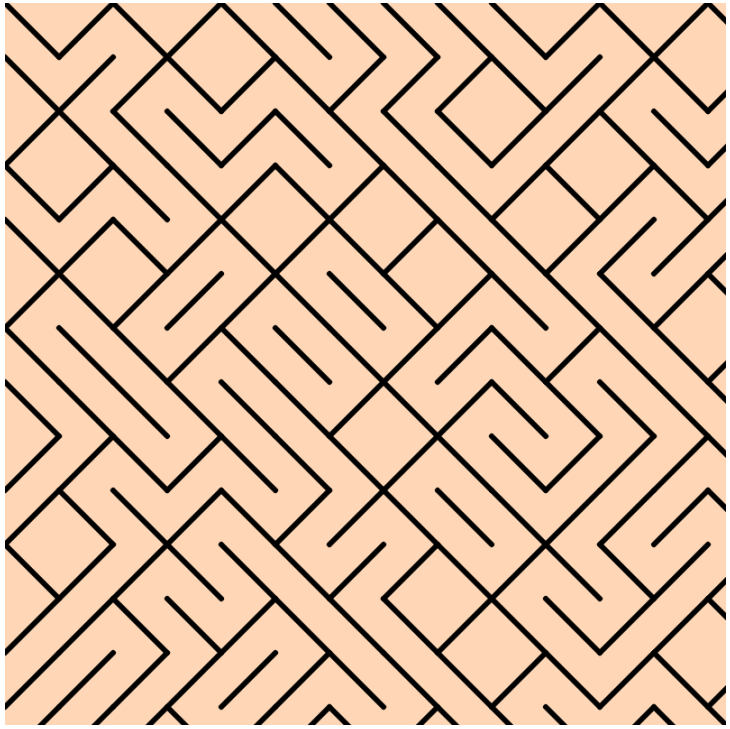
```
function setup() {
  createCanvas(400, 400);
  noLoop();
}
function draw() {
    background('PEACHPUFF');
  if (random()>0.5) {
    line(0,0,400,400);
  }
  else {
    line(400, 0, 0, 400);
  }
}
```



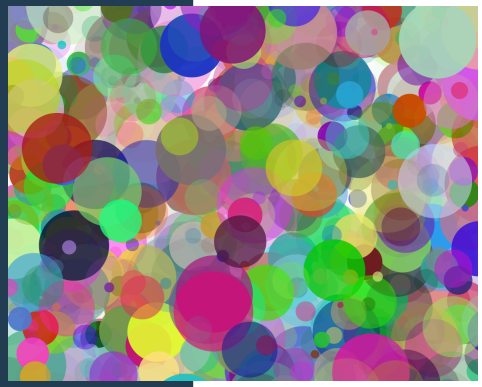
Make sure to add `noLoop();` - that makes sure the line is not flashing.

```
function setup() {
  createCanvas(400, 400);
  noLoop();
}
function draw() {
  background('PEACHPUFF');
  var step = 30;
  for (var x=0;x<400;x=x+step) {
    for (var y=0;y<400;y=y+step) {
      strokeWeight(3);
      if (random()>0.5) {
        line(x,y,x+step,y+step);
      }
      else {
        line(x+step,y,x,y+step);
      }
    }
  }
}
```





```
function setup() {
  createCanvas(windowWidth, windowHeight);
  noLoop();
}
function draw() {
  noStroke();
  for (let i = 0; i < 1000; i++) {
    fill(random(255), random(255), random(255), random(255));
    ellipse(random(windowWidth), random(windowHeight), random(100));
  }
}
```



Here, we're randomizing the color, size, and location of the circles I'm drawing on the canvas.

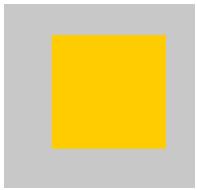
The fill takes in 4 values, which are, respectively, red, green, blue, and transparency. All the values range from between 0 and 255.

Then, for ellipse, I determine a random center point within the `windowWidth` and `windowHeight`, and a random radius.

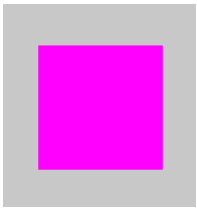
If you get rid of `noLoop();` see what happens!

Before we go on, let's talk a *bit* about color

```
let c = color(255, 204, 0);
fill(c);
noStroke();
rect(30, 20, 55, 55);
```



```
let c = color('magenta');
fill(c);
noStroke();
rect(20, 20, 60, 60);
```



`setRed()`

`setBlue()`

`setGreen()`

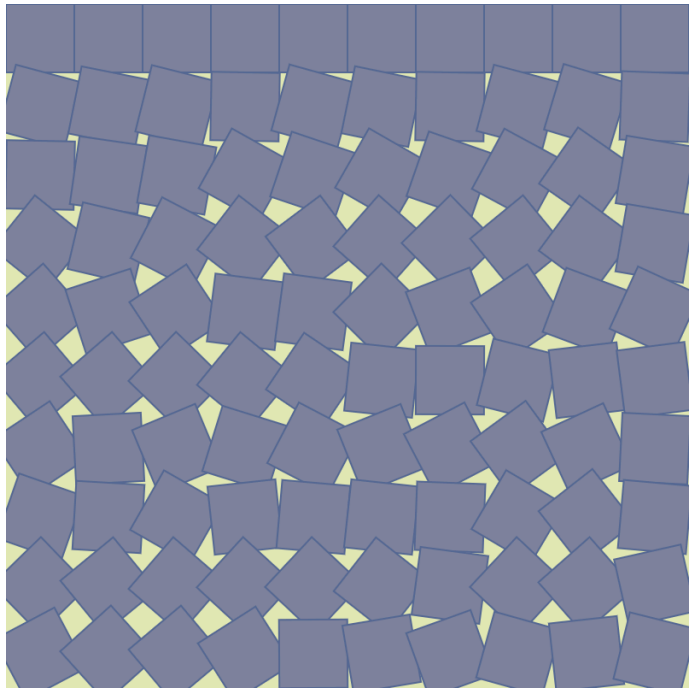
`setAlpha()`

`toString()`

https://colours.neilorangepeel.com/

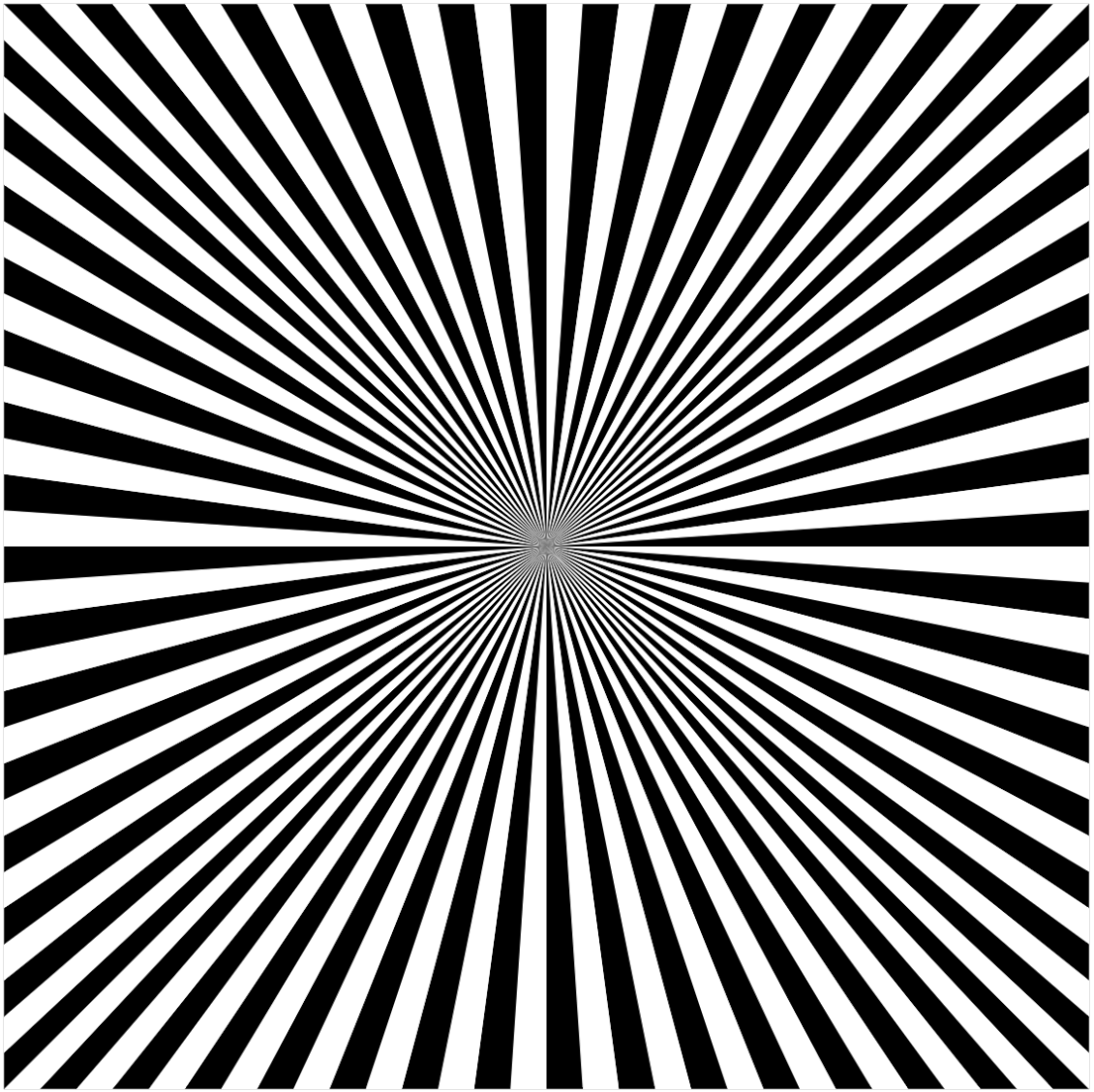
```
function draw() {
  // RGB and RGBA color strings are also supported
  // these all set to the same color (solid blue)
  let c;
  noStroke();
  c = color('rgb(0,0,255)');
  fill(c);
  rect(10, 10, 35, 35); // Draw rectangle
  c = color('rgb(0%, 0%, 100%)');
  fill(c);
  rect(55, 10, 35, 35); // Draw rectangle
  c = color('rgba(0, 0, 255, 1)');
  fill(c);
  rect(10, 55, 35, 35); // Draw rectangle
  c = color('rgba(0%, 0%, 100%, 1)');
  fill(c);
  rect(55, 55, 35, 35); // Draw rectangle
}
```

```
function setup() {
  createCanvas(400, 400);
  noLoop();
  rectMode(CENTER);
}

function draw() {
  background(229, 232, 182);
  var step = 40;
  for (var x = 0; x < 400; x = x + step) {
    for (var y = 0; y < 400; y = y + step) {
      drawRandomSquare(x, y, step);
    }
  }
}

function drawRandomSquare(x,y,step) {
    fill(125, 134, 156);
  stroke(88, 105, 148);
  push();
  translate(x+step/2,y+step/2);
  var maxRotation = map(y,0,400,0,PI);
  rotate( random(0,maxRotation) );
  rect(0,0,step,step);
  pop();
}
```



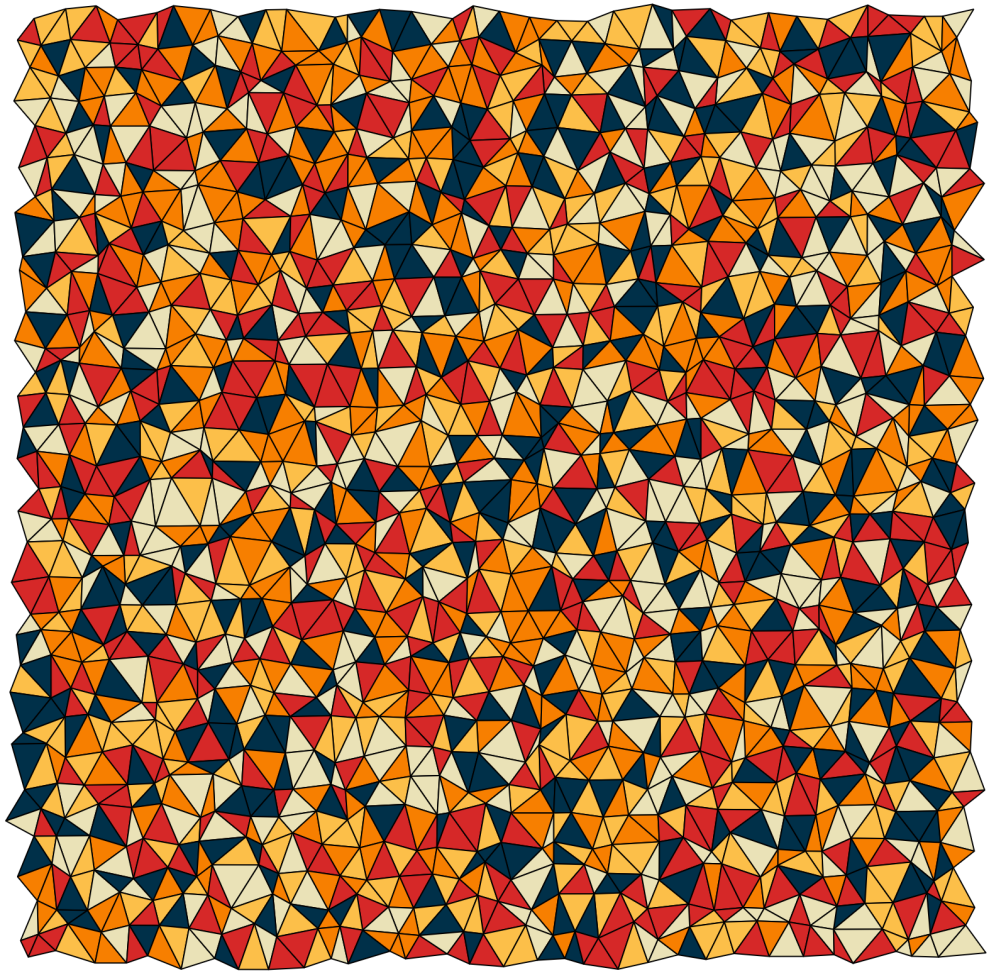
https://editor.p5js.org/ClaireBookworm/sketches/h7NYjvFof

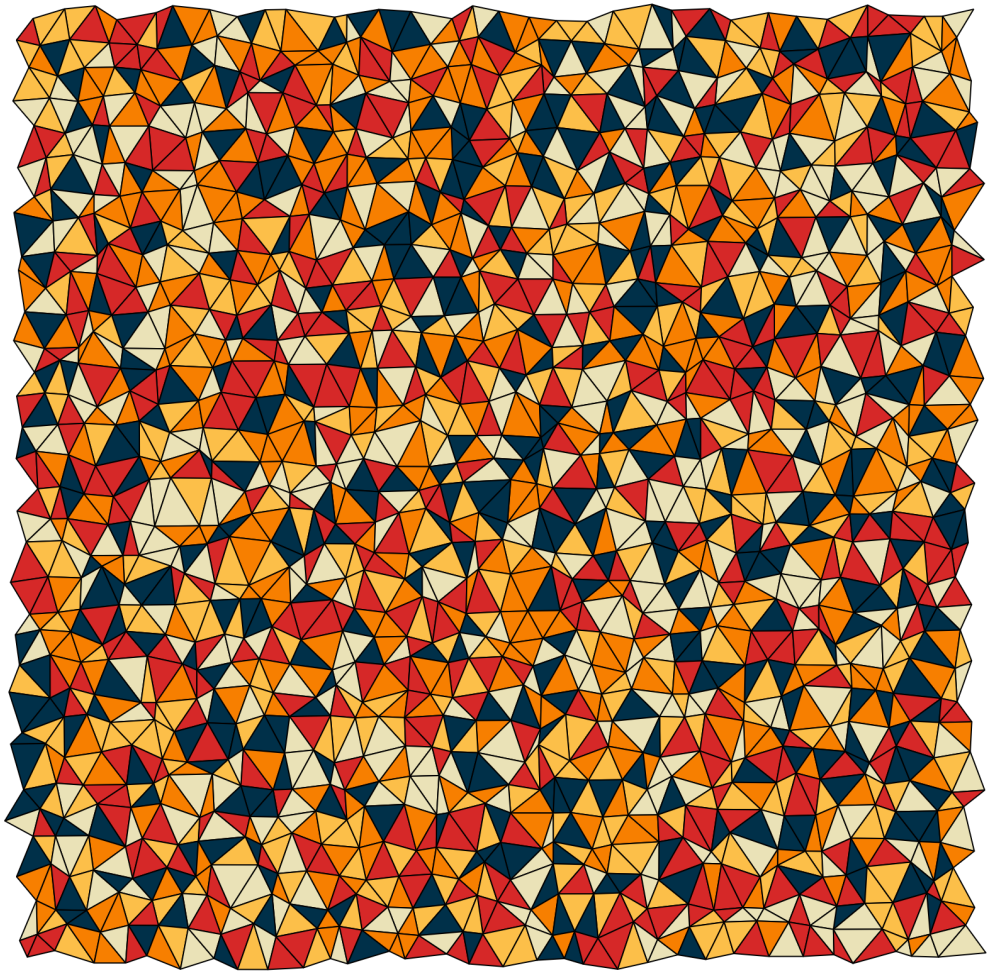
```
function setup() {
  createCanvas(800, 800);
  angleMode(DEGREES);
  rectMode(CENTER);
  const ctx = drawingContext;
  const x = width / 2;
  const y = height / 2;
  const squareSideDotsCount = 30;

  const squareVertices = [];
  let startAngle = 45;
  for (let i = 0; i < 4; i += 1) {
    squareVertices.push({
      x: 400 * cos(startAngle),
      y: 400 * sin(startAngle),
    });
    startAngle += 360 / 4;
  }

  const square = [];
  for (let i = 0; i < 4; i += 1) {
    for (let j = 0; j < squareSideDotsCount; j += 1) {
      const x = lerp(
        squareVertices[i].x,
        squareVertices[(i + 1) % squareVertices.length].x,
        j / squareSideDotsCount,
      );
      const y = lerp(
        squareVertices[i].y,
        squareVertices[(i + 1) % squareVertices.length].y,
        j / squareSideDotsCount,
      );
      square.push({ x, y });
    }
  }

  push();
  translate(x, y);
   for (let i = 0; i < square.length; i += 1) {
     push();
     noStroke();
     if (i % 2 === 0) fill(0);
     else fill(255);
     beginShape();
     vertex(square[i].x, square[i].y);
     vertex(0, 0);
     vertex(
        square[(i + 1) % square.length].x,
        square[(i + 1) % square.length].y,
      );
     endShape(CLOSE);
     pop();
   }
  pop();
}
```

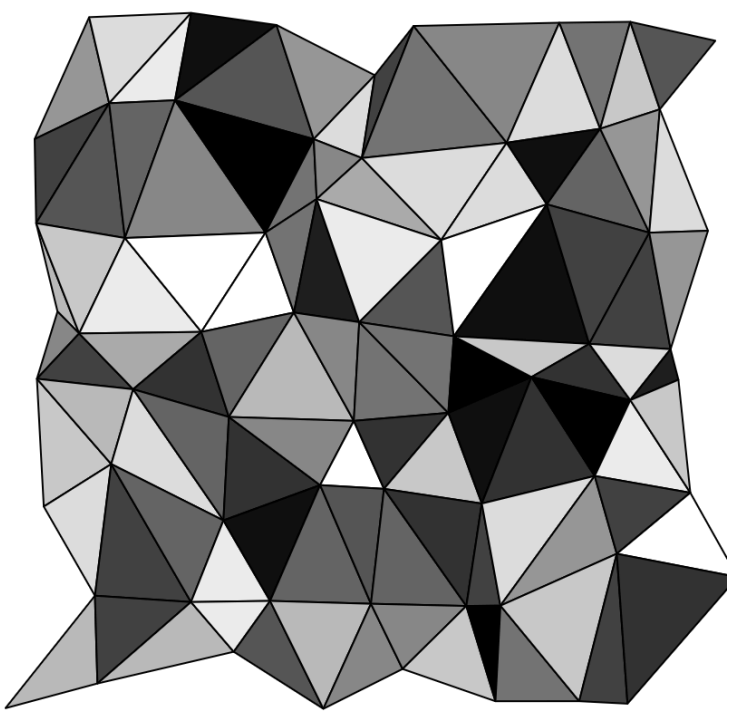


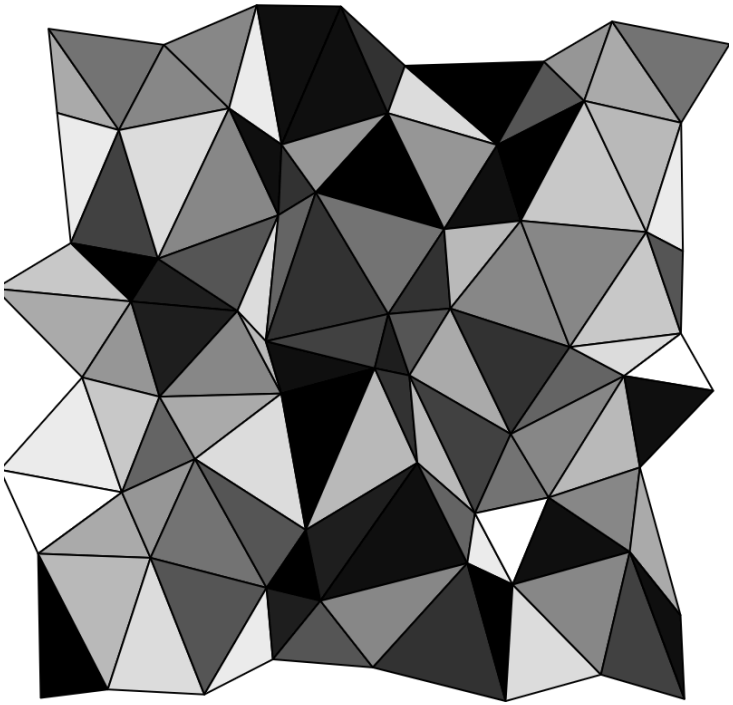
https://gist.github.com/ClaireBookworm/5df4783f15f018d6f83c5aac32c6b0a0

# openprocessing.org

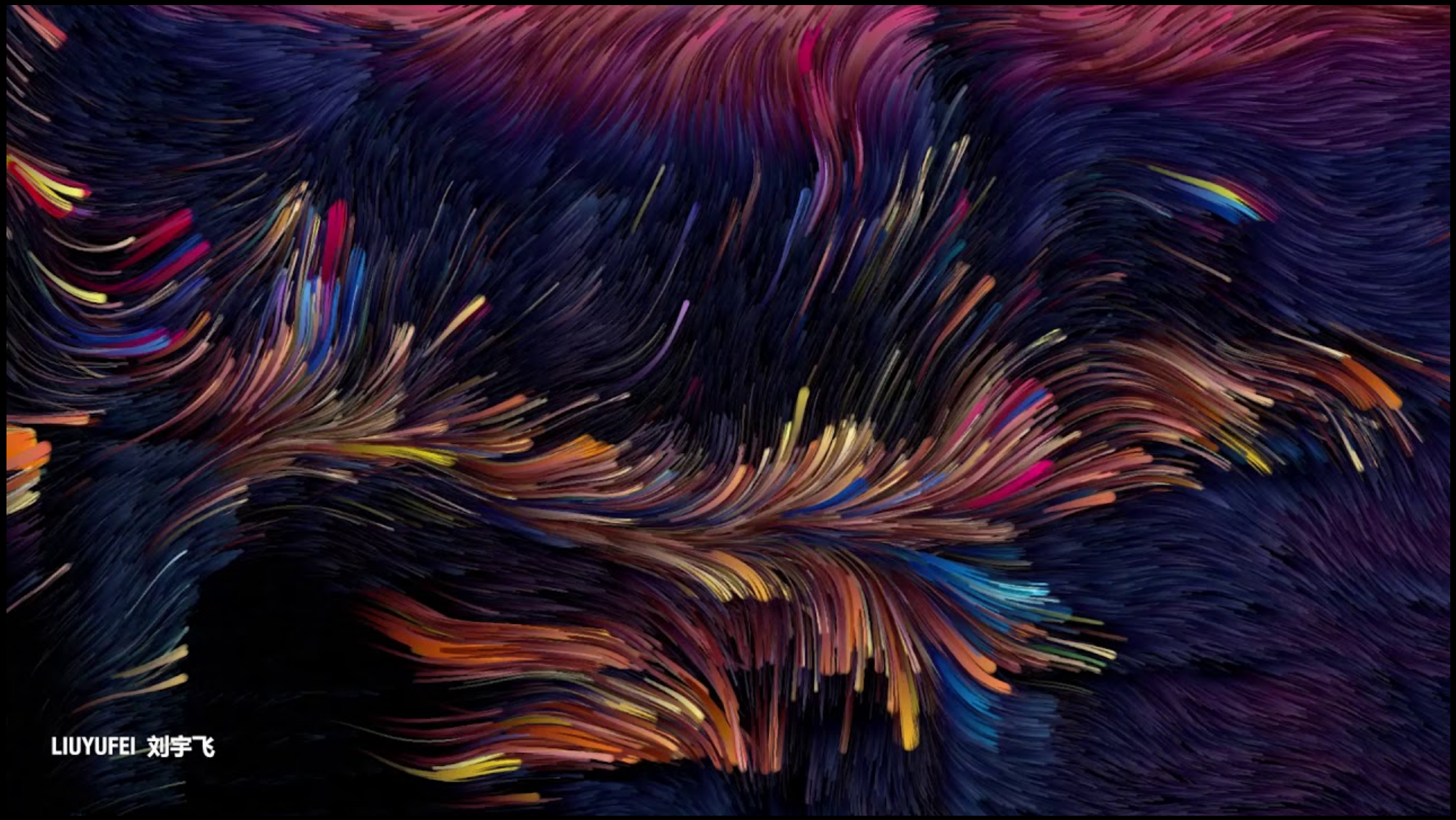




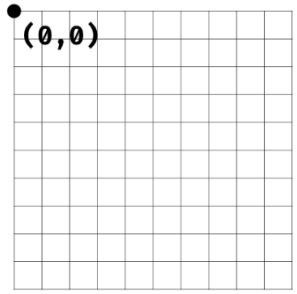




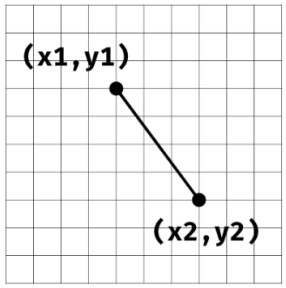
https://gist.github.com/ClaireBookworm/04b139695749f53ad896df0c67668f3c


LIUYUFEI 刘宇飞

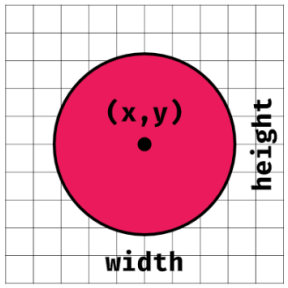
## grid system



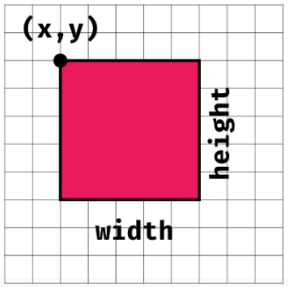

## line()



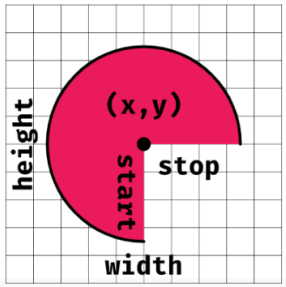

## ellipse()



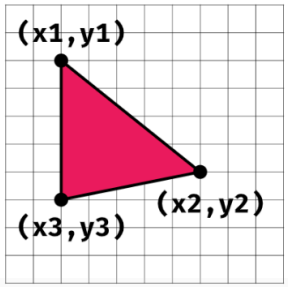

## rect()




## arc()




## vertex()




## 2d primitives

```
line(x1, y1, x2, y2);

ellipse(x, y, width, height);

rect(x, y, width, height);

arc(x, y, width, height, start, stop);

beginShape();
  vertex(x1, y1);
  vertex(x2, y2);
  vertex(x3, y3);
  //add more vertex
endShape(CLOSE);

text("string", x, y, boxwidth, boxheight);
```

## color

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

## program structure

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

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

## system variables

```
windowWidth / windowHeight
//width / height of window

width / height
//width / height of canvas

mouseX / mouseY
//current horizontal / vertical mouse position
```