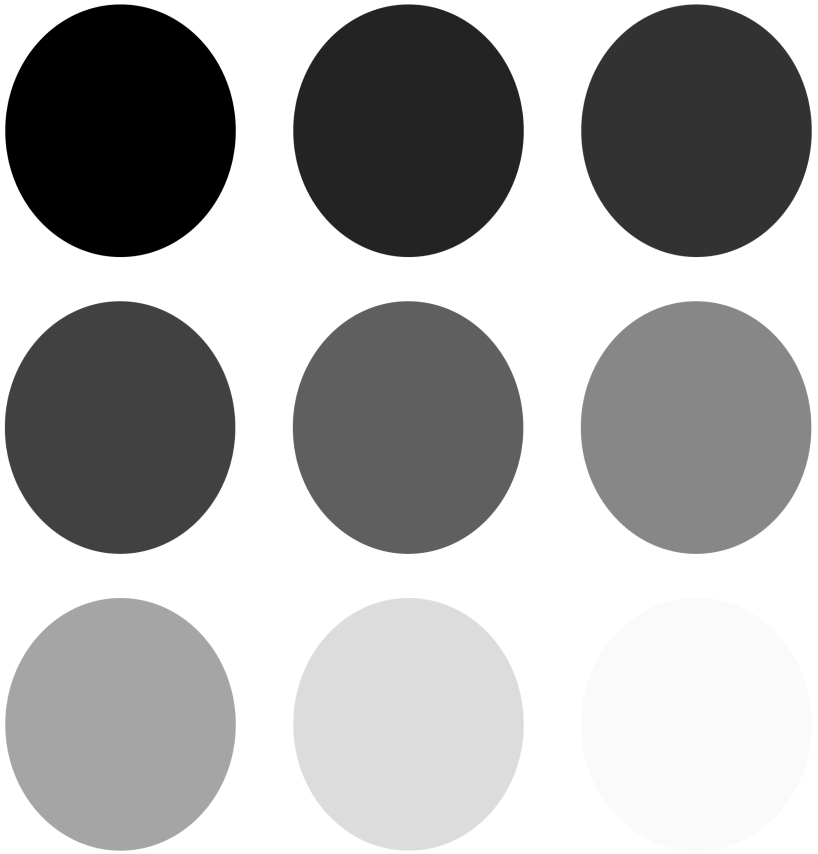


CHOO CHOO!

A CODING TRAIN ZINE

A CODING TRAIN ZINE

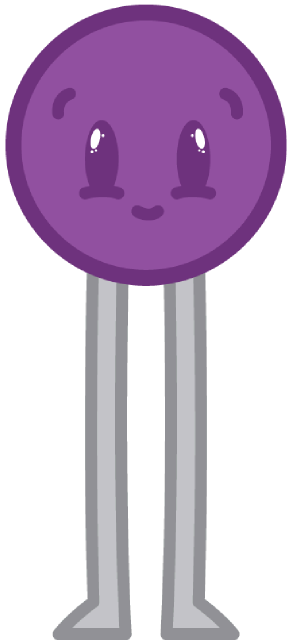


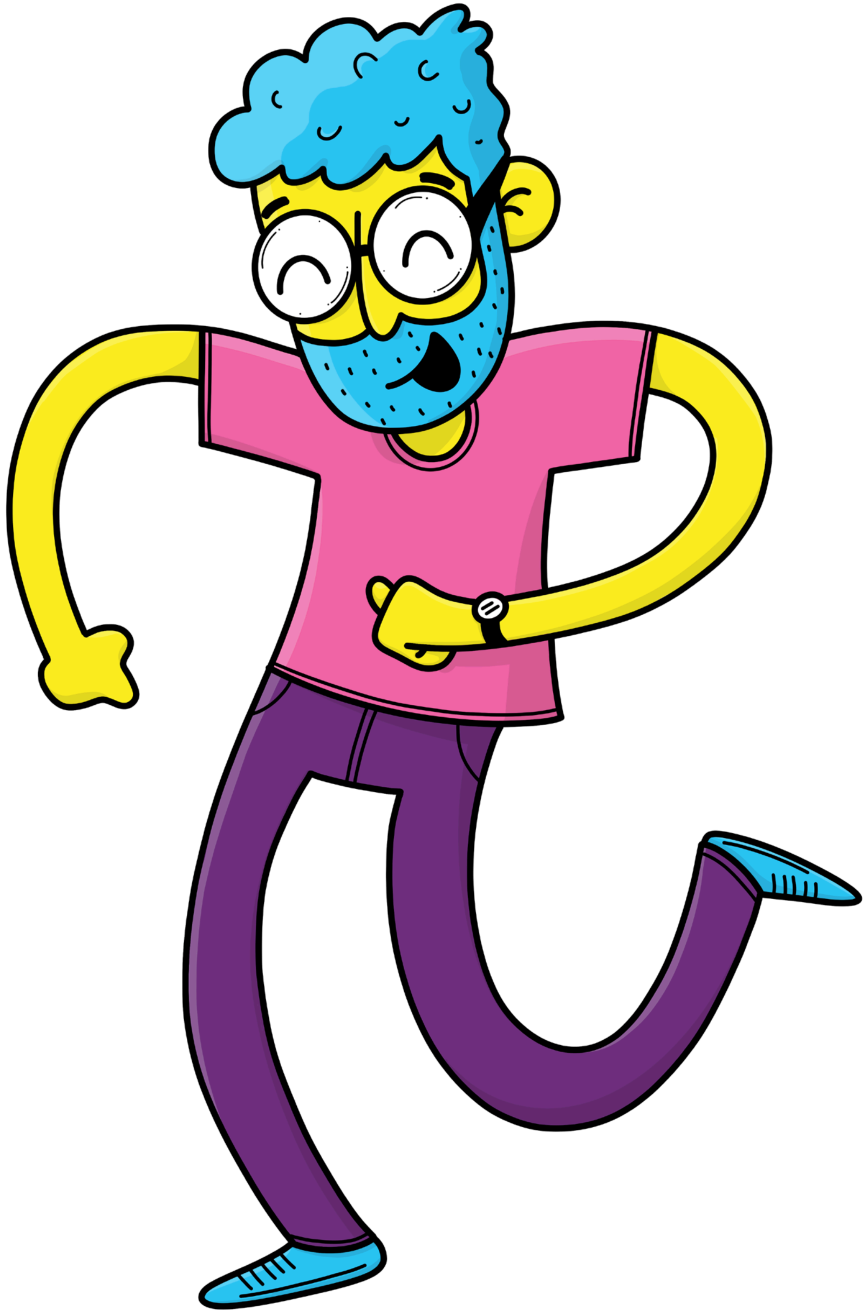


Curated and produced by Dan Shiffman, Coding Train founder, and Cy X, Coding Train Community Manager.

All images were made with code from The Coding Train and can be found on our website at thecodingtrain.com

FOR THE PROCESSING FOUNDATION





CONTENTS

I. STARFIELD 6
II. BUBBLE SORT 8
III. RAYCASTING 10
IV. MAURER ROSE 12
V. PERLIN NOISE 14
VI. COLLATZ CONJECTURE 18
VII. BLACK HOLE 20
VIII. TIMES TABLE CARDIOID 22
VIV. FRACTAL TREES 26
VV. 3D TERRAIN GENERATION 28
VVI. SELECTED CHALLENGES 32
VVII. ORIGIN STORY 34



That day, the channel gained 22 subscribers going from eight to thirty.

In 2016, mostly as a joke but also an homage to the beloved Reading Rainbow, an educational children's television series that began airing in 1983 (when I was 10 years old), I started referring to the channel informally as the Coding Rainbow. As the subscriber base grew, the name became more and more official.

On October 5, 2016 I received a very kind letter from Western New York Public Broadcasting Association, the owners of the Reading Rainbow trademark, asking that I change the name. Claire Duggan, a high school French teacher, suggested The Coding Train and once I realized a logo could include a rainbow emerging from a train's steam engine with some unicorn passengers, there was no need to search any further.

As of early 2020, there have been over 200 live streams, and around 800 videos uploaded. The total number of views across every video is a little over 60 million, which adds up to roughly 707 years, 9 months, 6 days, and 8 hours of viewing time.

Thank you for riding 🚂🌈💖

All aboard!

There was never a plan to start a YouTube channel called The Coding Train.

My journey began February 29, 2012 when I brought a camera into my classroom at New York University and broadcast a session of the ITP course The Nature of Code. The results were not great and no one watched. Later that summer, I painted a wall in my office green, put up a whiteboard, and with the help of Craig Protzel (who had just graduated from ITP) installed a couple cameras in my office. With my new studio setup, I got to work recording a set of tutorials on how to program with Processing. I created a Vimeo account and uploaded them.

On July 29, 2013, using a combination of Google Hangouts and screen sharing I attempted my very first live stream. A few people watched, and the archive can still be seen today as a 39 minute 59 second video entitled “Processing Live Test.” (https://youtu.be/x7ZKr6_QEuk)

I moved to YouTube in July 2015 after receiving an email asking if I might consider uploading my existing tutorials there so that they could be watched at double speed. Four weeks later I had uploaded 118 videos to my previously dormant YouTube account. And on September 4, 2015, I broadcast the very first “official” Coding Train Live Stream: https://youtu.be/TdUC3JgwUJY


```
class Star {

  constructor() {
    this.x = random(-width, width);
    this.y = random(-height, height);
    this.z = random(width);
    this.pz = this.z;
  }

  show() {
    fill(255);
    noStroke();
    let sx = width  * this.x / this.z;
    let sy = height * this.y / this.z;
    let r = map(this.z, 0, width, 16, 0);
    ellipse(sx, sy, r, r);

    let px = width  * this.x / this.pz;
    let py = height * this.y / this.pz;
    stroke(255);
    line(px, py, sx, sy);
  }

}
```

II. BUBBLE SORT

Coding Challenge 133 Times Table Cardioid Visualization: Visualization of the “mathematical heart” Cardioid using times tables calculations.

Coding Challenge 14 Fractal Trees: Part of algorithmic botany series. Implement fractal trees with recursion in p5.js

Coding Challenge 11 3D Terrain Generation with Perlin Noise in Processing: 3D procedural terrain using Perlin Noise and the beginShape() function in Processing.



SELECTED CHALLENGES

Coding Challenge 1 Starfield: Processing, Creation of “Starfield” or “Warpspeed” visualization

Coding Challenge 114 BubbleSort Visualization: “Bublle sort” algorithm visualization in Processing

Coding Challenge 145 2D Raycasting: Basic ray casting engine with line segment “surfaces” and vector “rays”

Coding in the Cabana 1 Maurer Rose: First challenge in new series, Coding in the Cabana. Visualization of the Maurer Rose patter in p5.js.

Coding Challenge 13.1 Introduction - Perlin Noise: Use of Perlin Noise to create a two-dimensional flow field with the p5.js library.

Coding in the Cabana 2 Collatz Conjecture: Second episde of Coding in the Cabana. Visualization of the Collatz Conjecture in Processing.

Coding Challenge 144 2D Black Hole Visualization: Inspired by image of the Powehi m87 black hole from the Event Horizon Telescope. Visualization of the behavior of light (photos) following the path of spacetime around the black hole.


```
let values = [];
let i = 0;

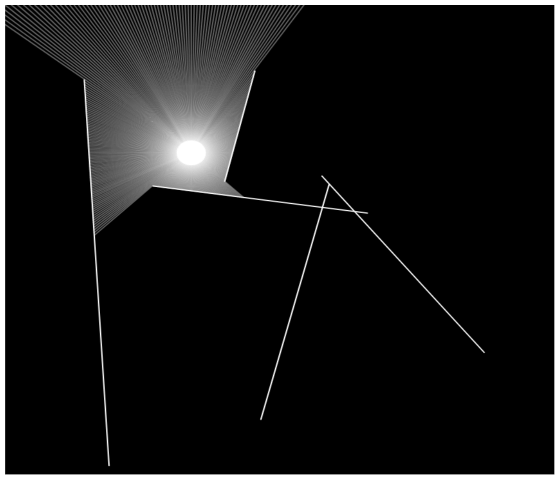
function setup() {
  for (let i = 0; i < width; i++) {
    values[i] = random(height);
  }
}

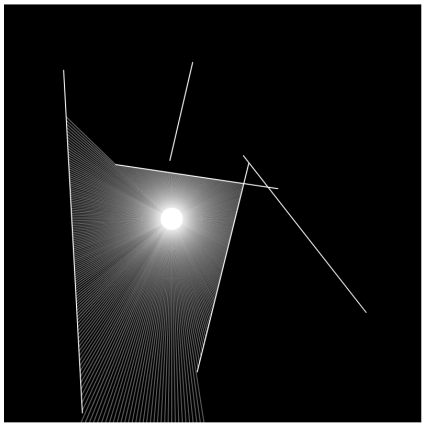
function draw() {
  background(0);
  if (i < values.length) {
    for (let j = 0; j < values.length - i - 1; j++) {
      let a = values[j];
      let b = values[j + 1];
      if (a > b) {
        swap(values, j, j + 1);
      }
    }
  }
  i++;
  for (let i = 0; i < values.length; i++) {
    stroke(255);
    line(i, height, i, height - values[i]);
  }
}

function swap(arr, a, b) {
  let temp = arr[a];
  arr[a] = arr[b];
  arr[b] = temp;
}
```

```
class Particle {
  constructor() {
    this.pos = createVector(width / 2, height / 2);
    this.rays = [];
    for (let a = 0; a < 360; a += 1) {
      this.rays.push(new Ray(this.pos, radians(a)));
    }
  }

  update(x, y) {
    this.pos.set(x, y);
  }
}
```





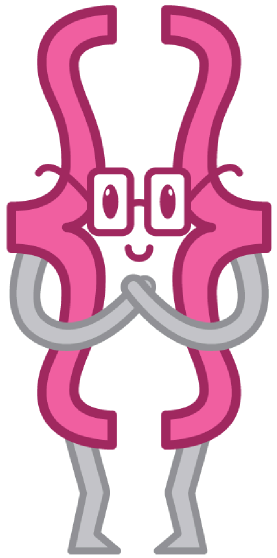
```
class Ray {
  constructor(pos, angle) {
    this.pos = pos;
    this.dir = p5.Vector.fromAngle(angle);
  }

  lookAt(x, y) {
    this.dir.x = x - this.pos.x;
    this.dir.y = y - this.pos.y;
    this.dir.normalize();
  }

  show() {
    stroke(255);
    push();
    translate(this.pos.x, this.pos.y);
    line(0, 0, this.dir.x * 10, this.dir.y * 10);
    pop();
  }
}
```



```
function draw() {
  background(0);
  translate(0, 50);
  rotateX(PI / 3);
  fill(200, 200, 200, 150);
  translate(-w / 2, -h / 2);
  for (let y = 0; y < rows - 1; y++) {
    beginShape(TRIANGLE_STRIP);
    for (var x = 0; x < cols; x++) {
      vertex(x * scl, y * scl, terrain[x][y]);
      vertex(x * scl, (y + 1) * scl, terrain[x][y + 1]);
    }
    endShape();
  }
}
```



```
let cols, rows;
let scl = 20;
let w = 1400;
let h = 1000;

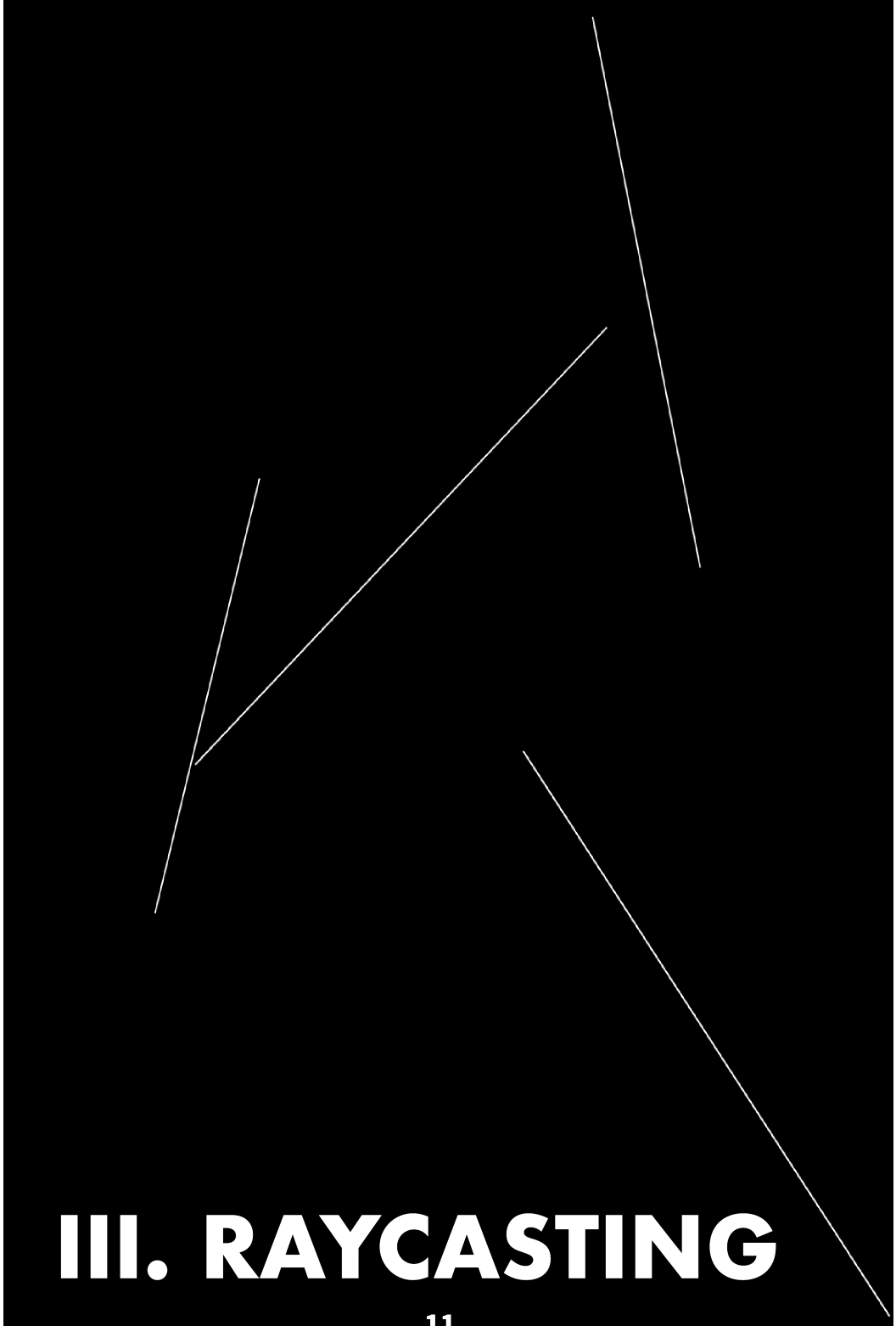
let flying = 0;

let terrain = [];

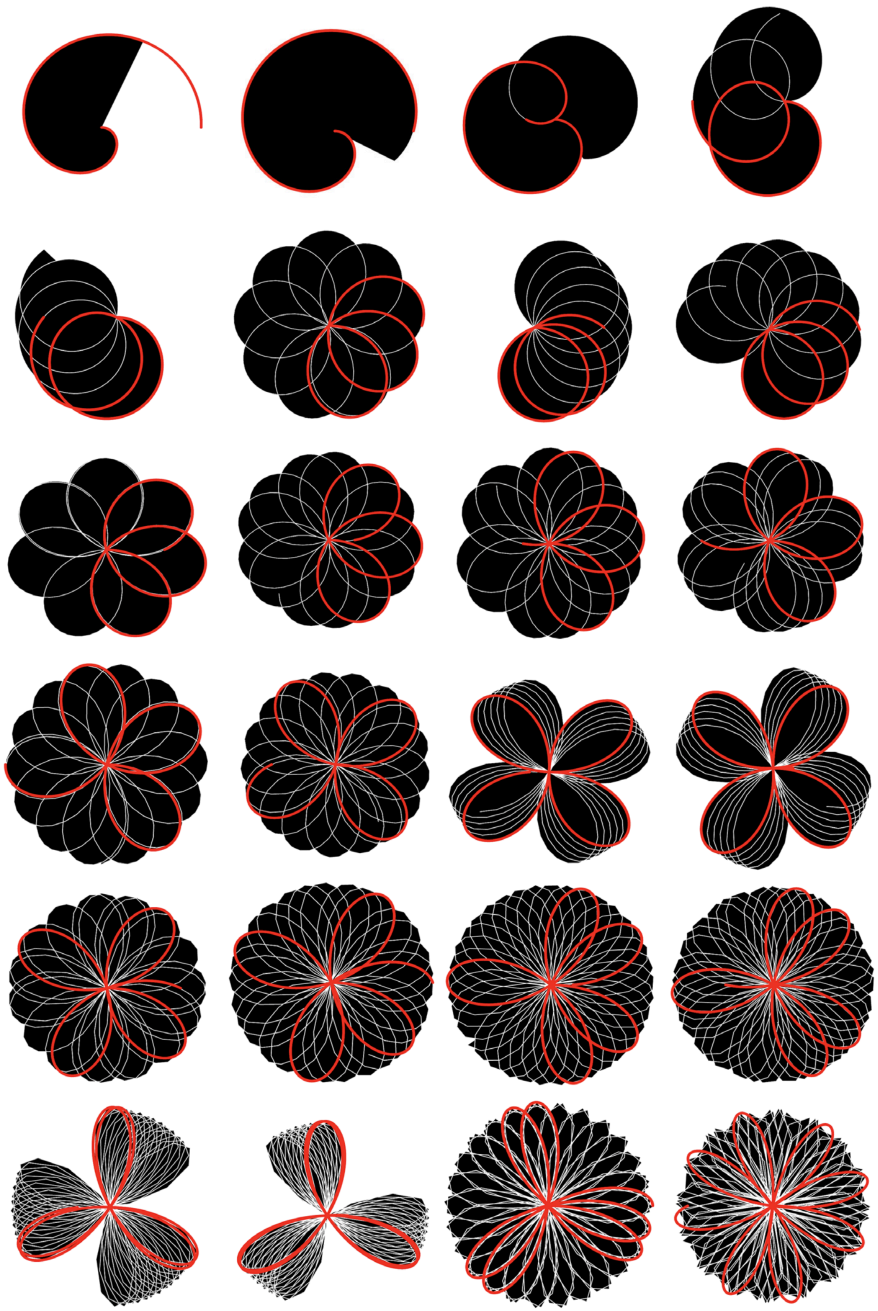
function setup() {
  createCanvas(600, 600, WEBGL);
  cols = w / scl;
  rows = h / scl;

  for (let x = 0; x < cols; x++) {
    terrain[x] = [];
    for (let y = 0; y < rows; y++) {
      terrain[x][y] = 0;
    }
  }
}

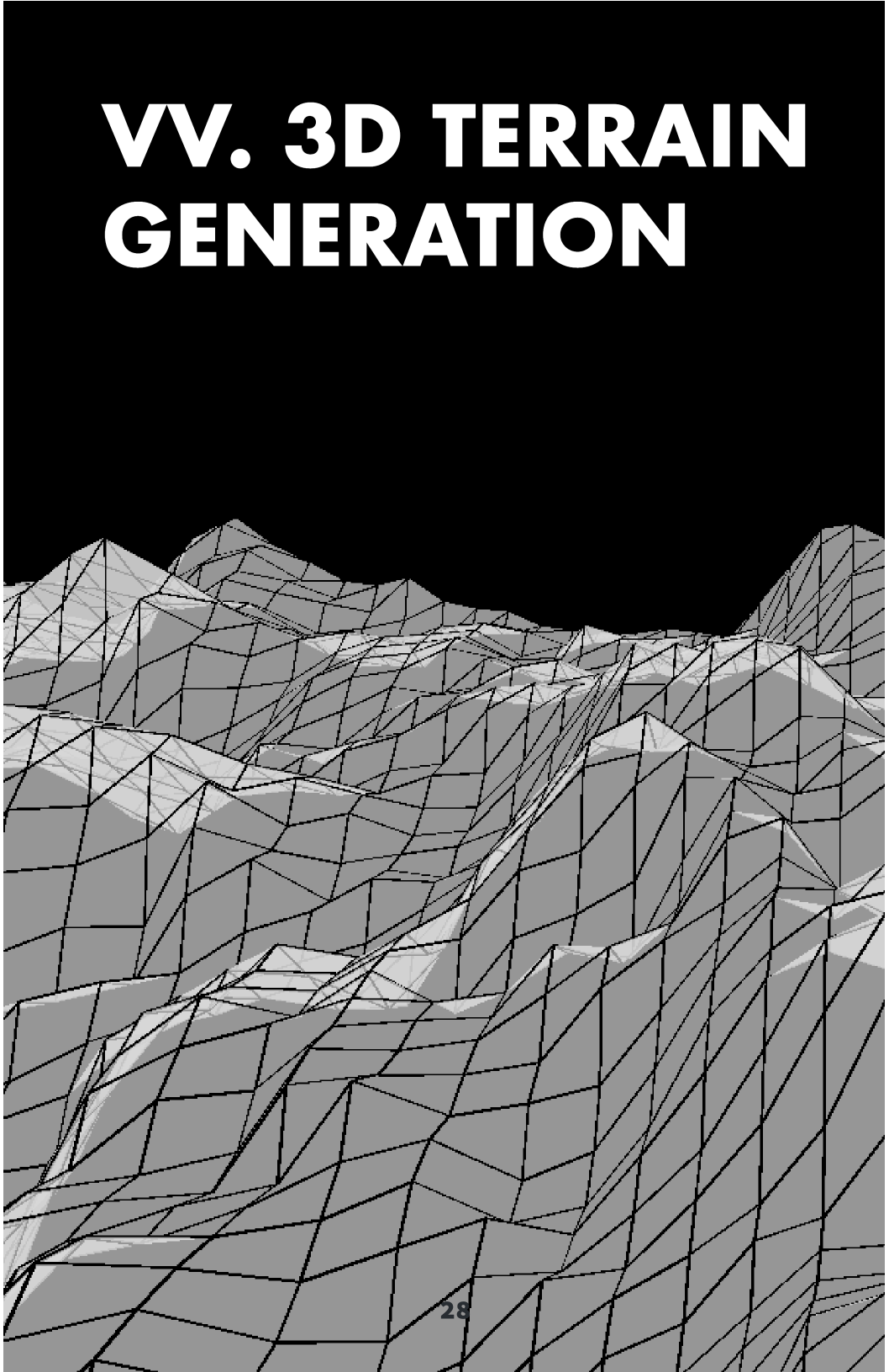
function draw() {
  flying -= 0.1;
  let yoff = flying;
  for (let y = 0; y < rows; y++) {
    let xoff = 0;
    for (let x = 0; x < cols; x++) {
      terrain[x][y] = map(noise(xoff, yoff), 0, 1, -100, 100);
      xoff += 0.2;
    }
    yoff += 0.2;
  }
}
```

III. RAYCASTING



VV. 3D TERRAIN GENERATION



IV. MAURER ROSE

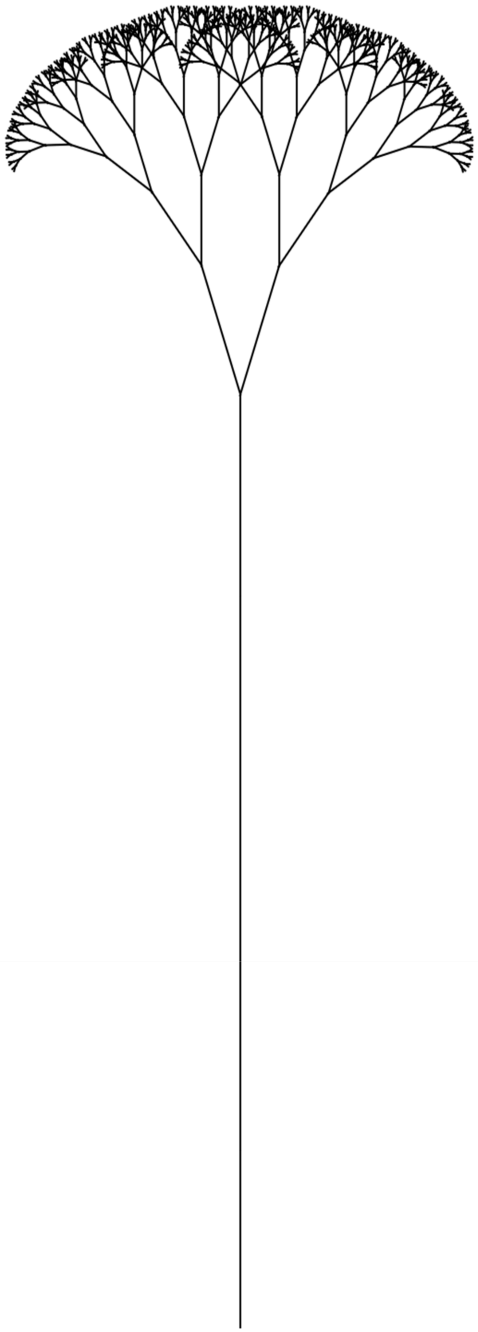
```
let n = 2;
let d = 39;

angleMode(DEGREES);
translate(width / 2, height / 2);

beginShape();
for (let i = 0; i < 361; i++) {
  let k = i * d;
  let r = 150 * sin(n * k);
  let x = r * cos(k);
  let y = r * sin(k);
  vertex(x, y);
}
endShape();

beginShape();
for (let i = 0; i < 361; i++) {
  let k = i;
  let r = 150 * sin(n * k);
  let x = r * cos(k);
  let y = r * sin(k);
  vertex(x, y);
}
endShape();
```

V. PERLIN NOISE



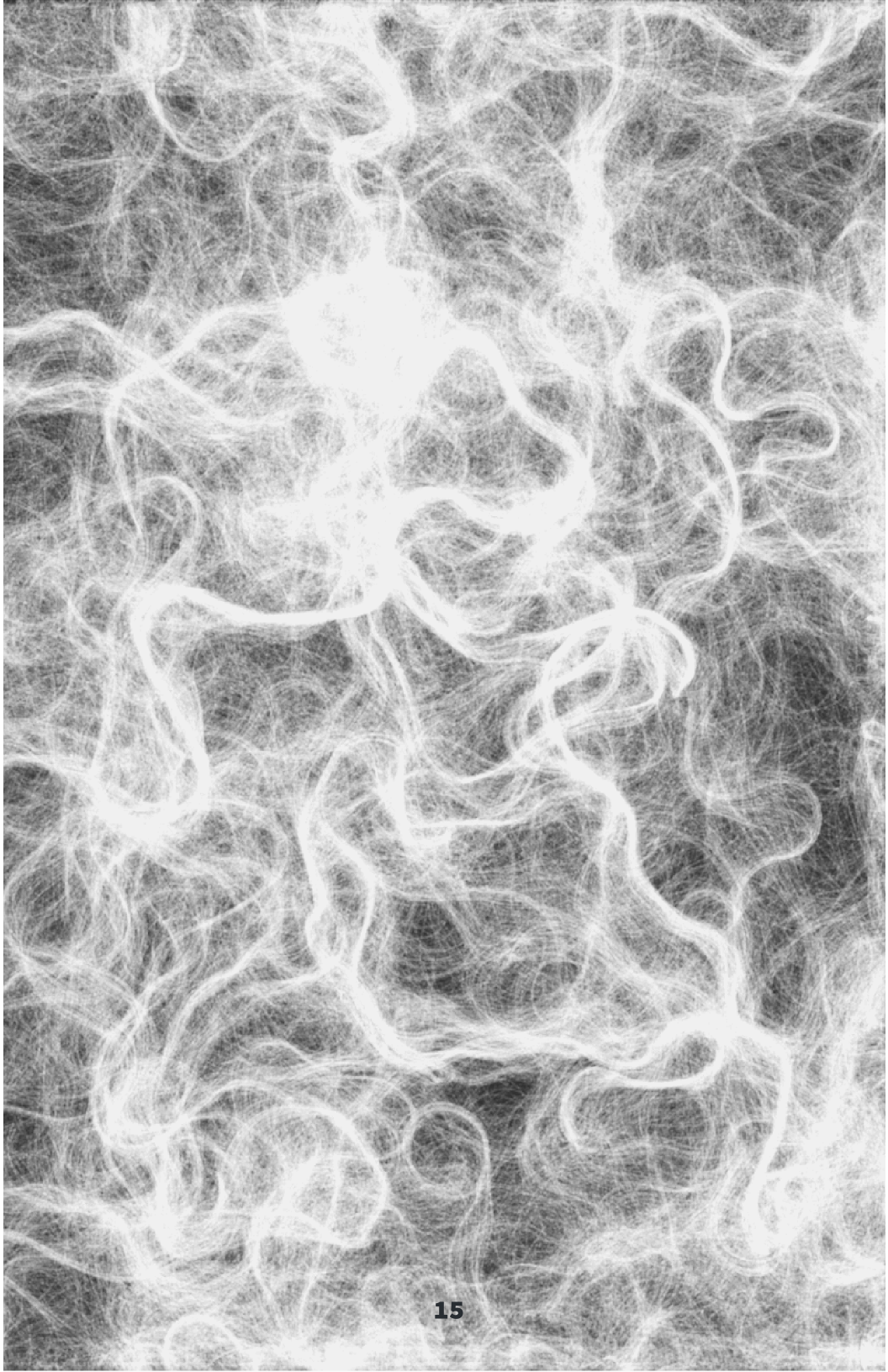
VIV. FRACTAL TREES

```
let angle = 0;
let slider;

function setup() {
  createCanvas(400, 400);
  slider = createSlider(0, TWO_PI, PI / 4, 0.01);
}

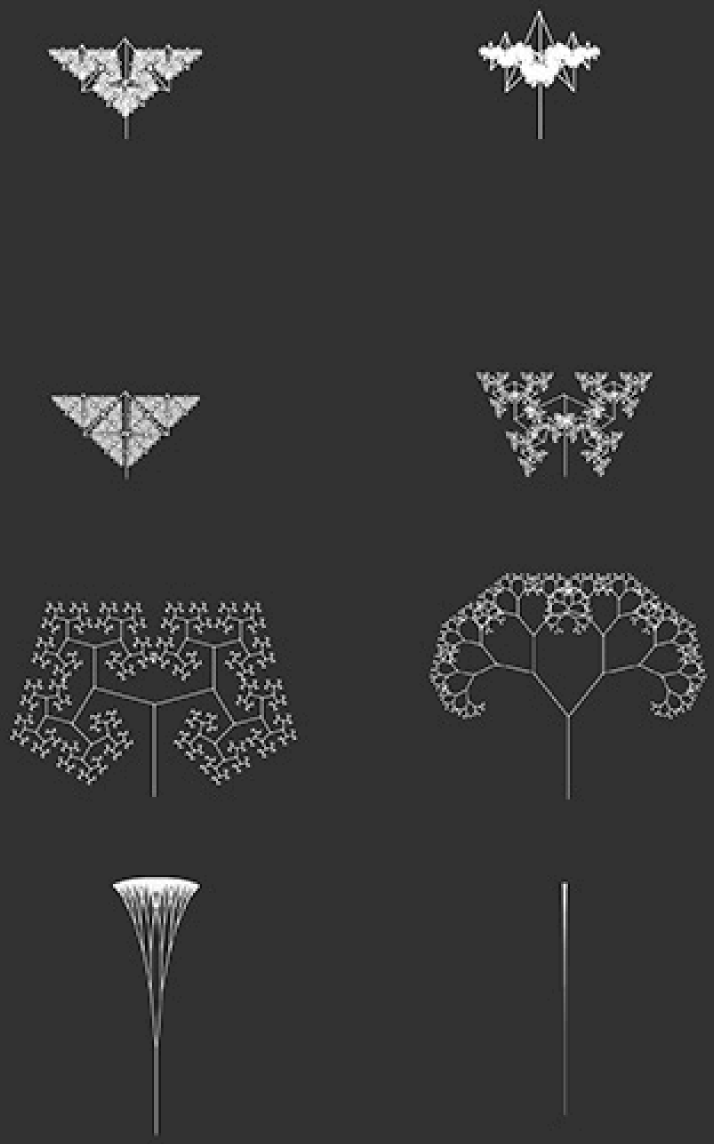
function draw() {
  background(51);
  angle = slider.value();
  stroke(255);
  translate(200, height);
  branch(100);
}

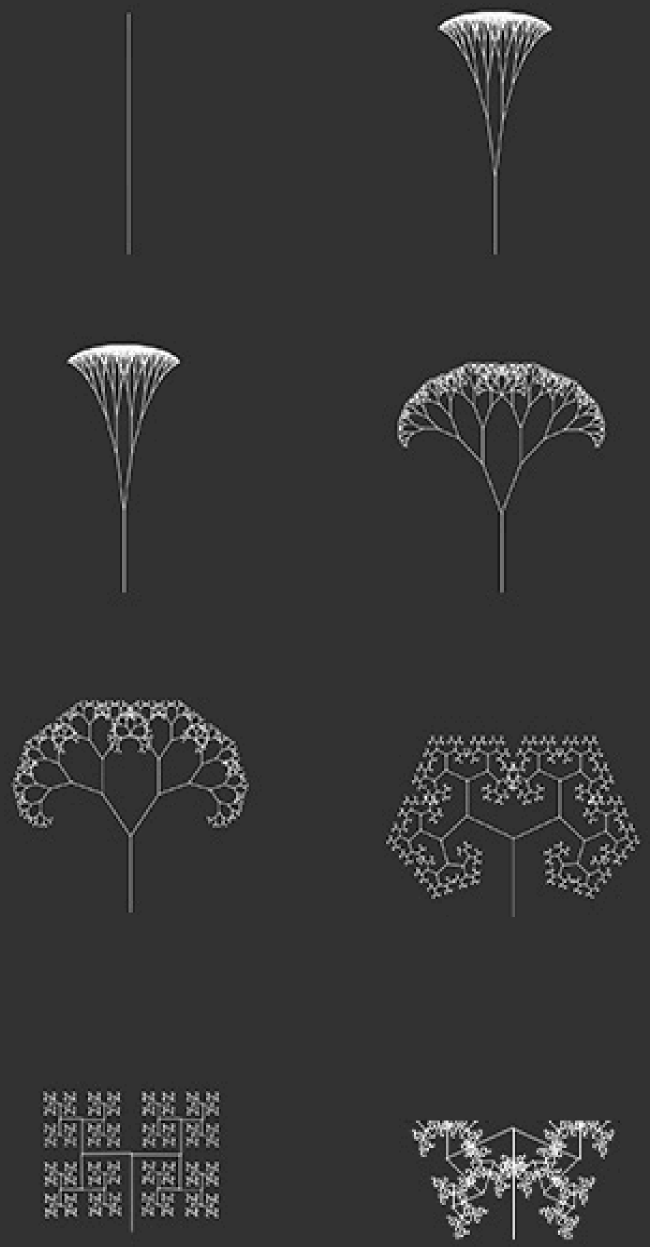
function branch(len) {
  line(0, 0, 0, -len);
  translate(0, -len);
  if (len > 4) {
    push();
    rotate(angle);
    branch(len * 0.67);
    pop();
    push();
    rotate(-angle);
    branch(len * 0.67);
    pop();
  }
}
```

```
let increment = 0.1;
let z_increment = 0.003;

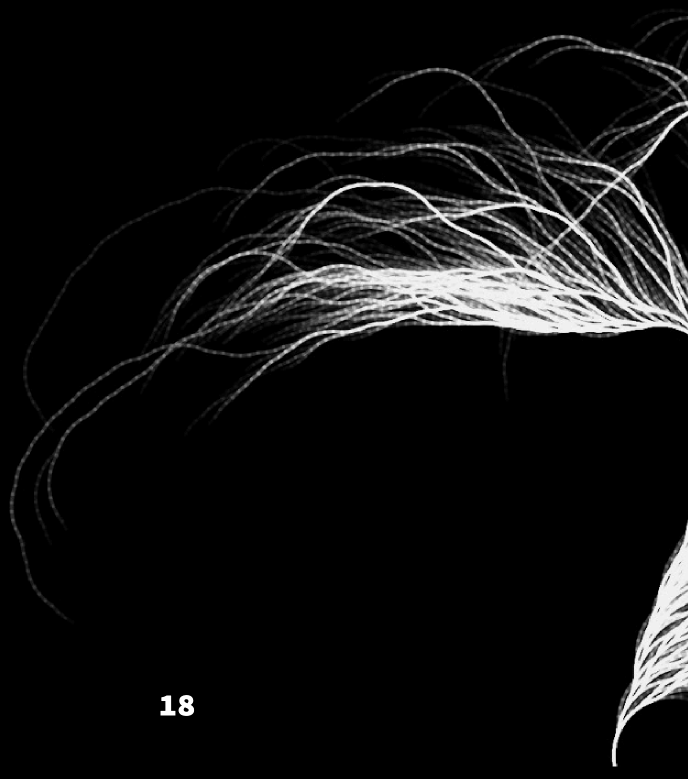
let yoff = 0;
for (let y = 0; y < rows; y++) {
  let xoff = 0;
  for (let x = 0; x < cols; x++) {
    let index = x + y * cols;
    let angle = noise(xoff, yoff, zoff) *
      TWO_PI * 4;
    let v = p5.Vector.fromAngle(angle);
    flowfield[index] = v;
    xoff += increment;
  }
  yoff += increment;
}
zoff += z_increment;
```

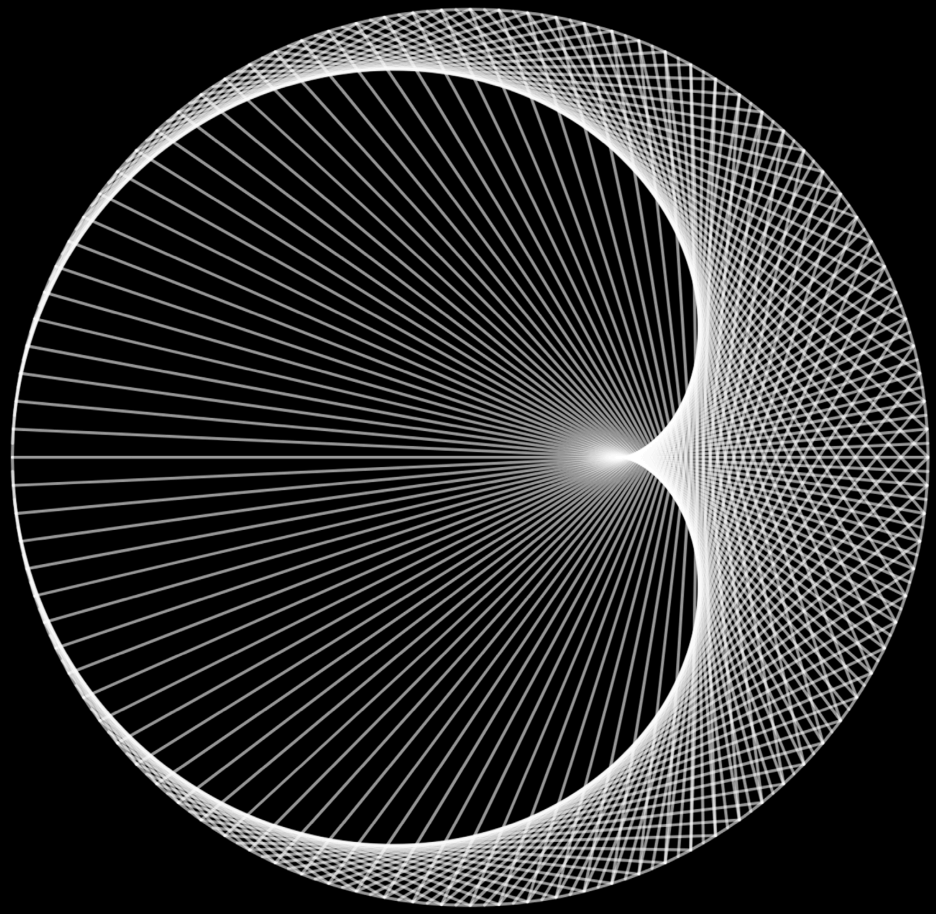





VI. COLLATZ CONJECTURE

```
for (let i = 1; i < 10000; i++) {
  let sequence = [];
  let n = i;
  do {
    sequence.push(n);
    n = collatz(n);
  } while (n != 1);
  sequence.push(1);
  sequence.reverse();
}
```





VIII. TIMES TABLE CARDIOID



```
function getVector(index, total, r) {
  const angle = map(index % total, 0, total, 0, TWO_PI);
  const v = p5.Vector.fromAngle(angle + PI);
  v.mult(r);
  return v;
}

function setup() {
  createCanvas(windowWidth, windowHeight);
  let r = height / 2 - 16;
  let factor = 2;
  let total = 200;

  background(0);
  translate(width / 2, height / 2);
  stroke(255, 150);
  strokeWeight(2);
  noFill();
  ellipse(0, 0, r * 2);
  strokeWeight(2);
  for (let i = 0; i < total; i++) {
    const a = getVector(i, total, r);
    const b = getVector(i * factor, total, r);
    line(a.x, a.y, b.x, b.y);
  }
}
```

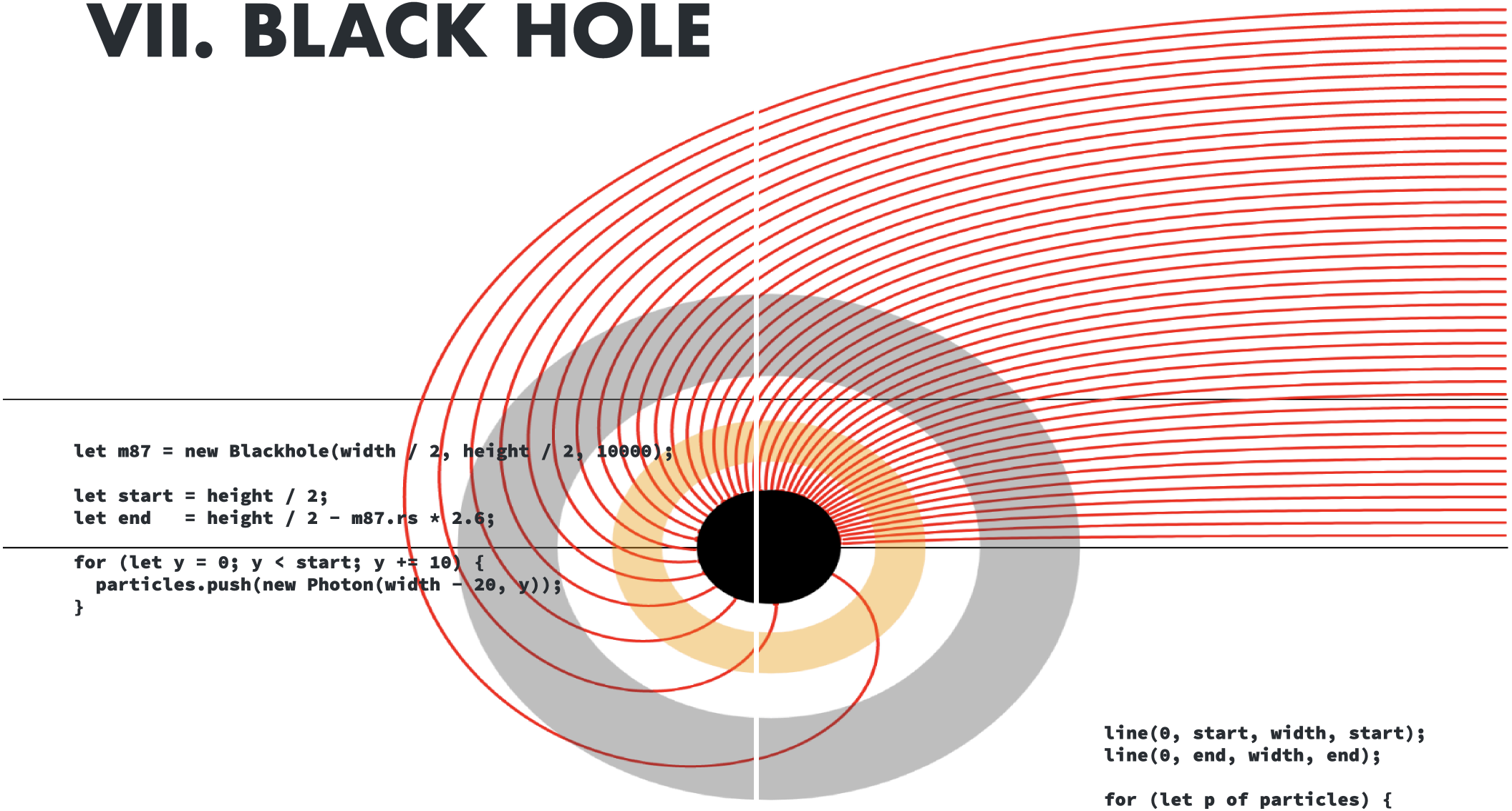


```
let len = 5;
let angle = 0.15;
resetMatrix();
translate(width / 2, height);
for (let j = 0; j < sequence.length; j++) {
  let value = sequence[j];
  if (value % 2 == 0) {
    rotate(angle);
  } else {
    rotate(-angle);
  }
  strokeWeight(2);
  stroke(255, 10);
  line(0, 0, 0, -len);
  translate(0, -len);
}
```



```
function collatz(n) {
  // even
  if (n % 2 == 0) {
    return n / 2;
    // odd
  } else {
    return (n * 3 + 1) / 2;
  }
}
```

VII. BLACK HOLE



```
let m87 = new Blackhole(width / 2, height / 2, 10000);

let start = height / 2;
let end   = height / 2 - m87.rs * 2.6;

for (let y = 0; y < start; y += 10) {
  particles.push(new Photon(width - 20, y));
}
```

```
line(0, start, width, start);
line(0, end, width, end);

for (let p of particles) {
  m87.pull(p);
  p.update();
  p.show();
}
m87.show();
```