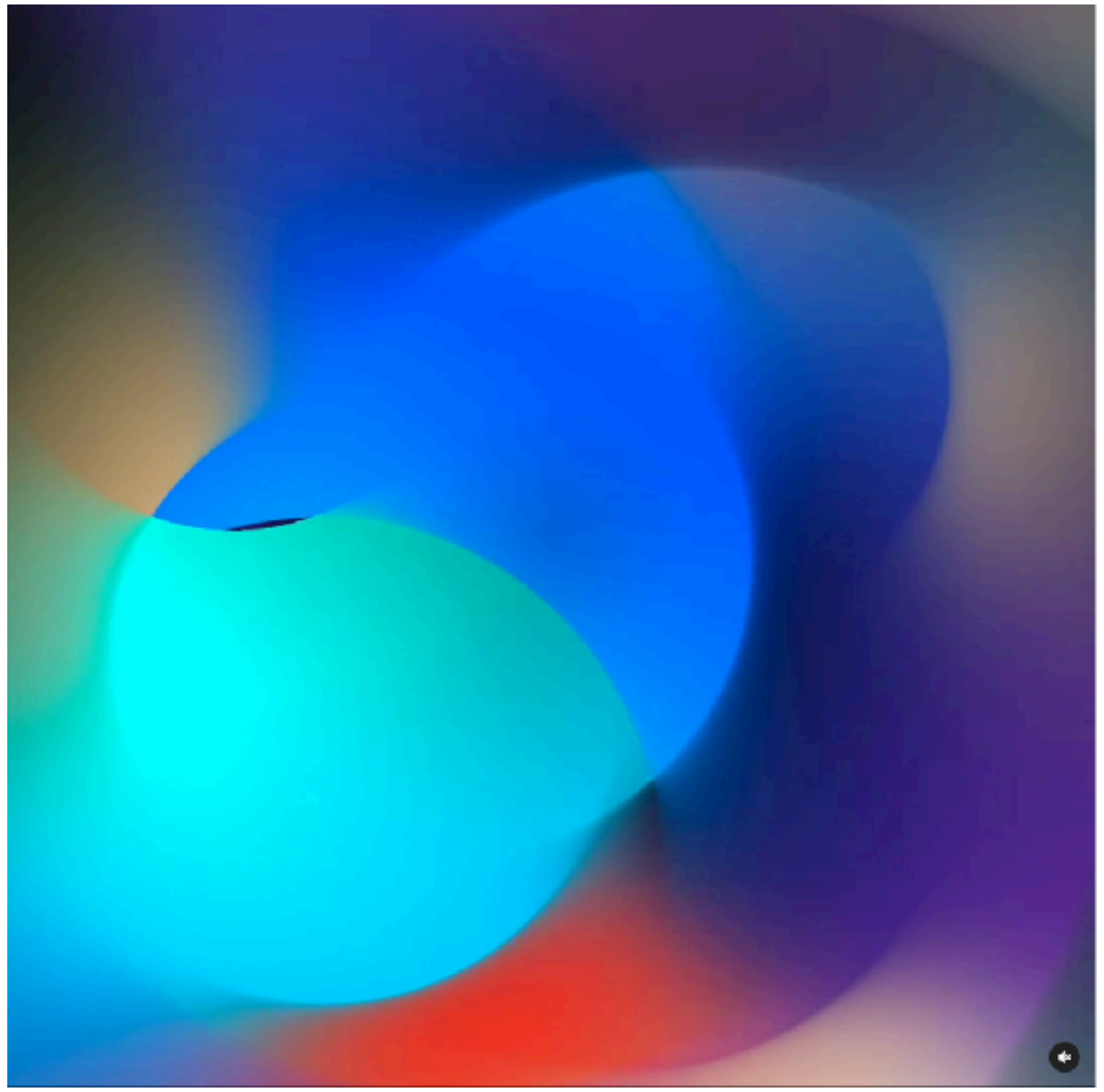
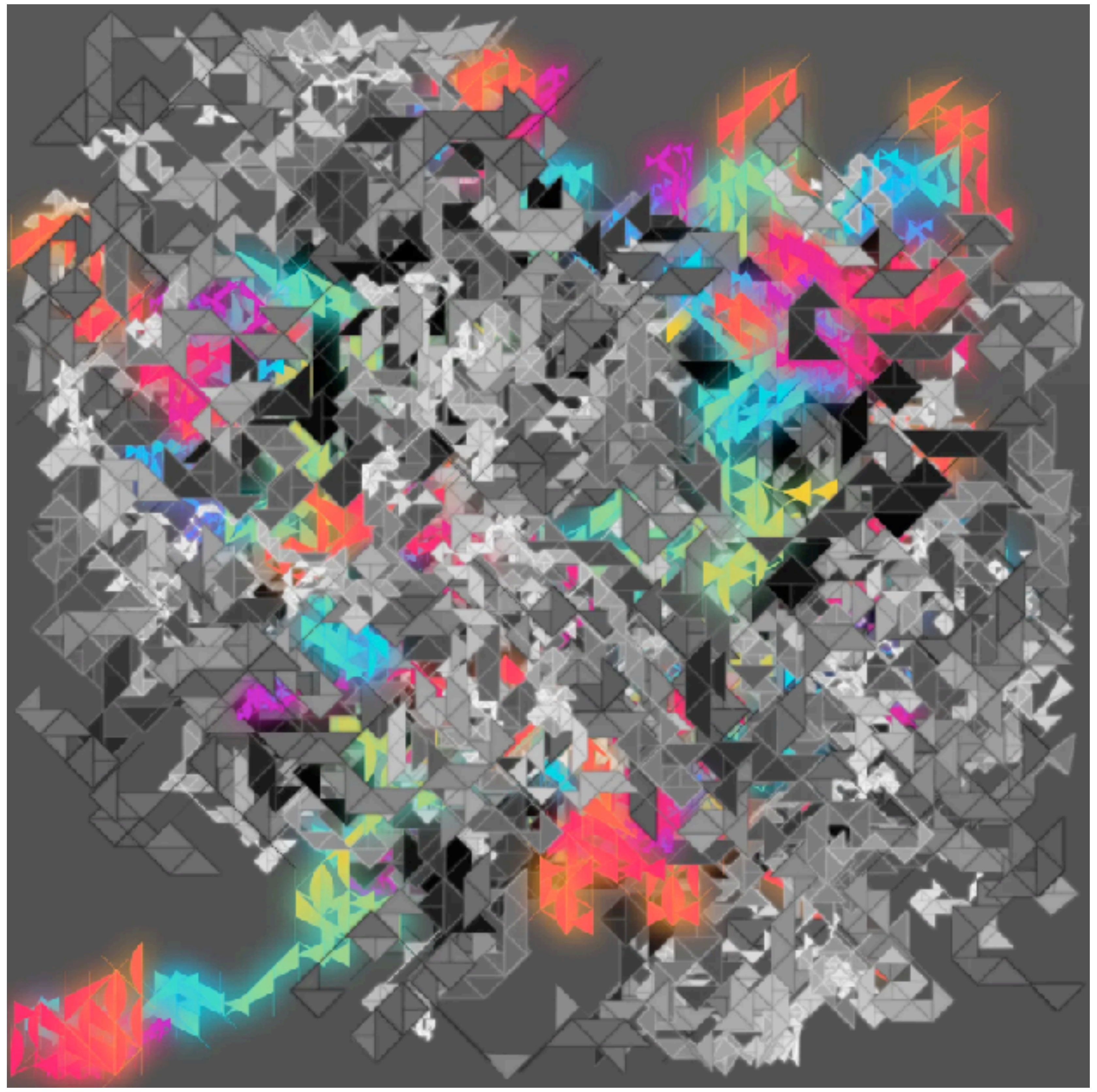
CS122 Class 9: Creative Coding


Arcs by Zach Lieberman

Sketch Aquarium by teamLab


p5.js generative piece by shvembldr

Class 9 agenda

- Zipcrit
- PM3 artwalk
- Mini lecture: Creative coding
- Break
- Creative coding studio in p5.js

Press Fit Kit Crit (10 min)

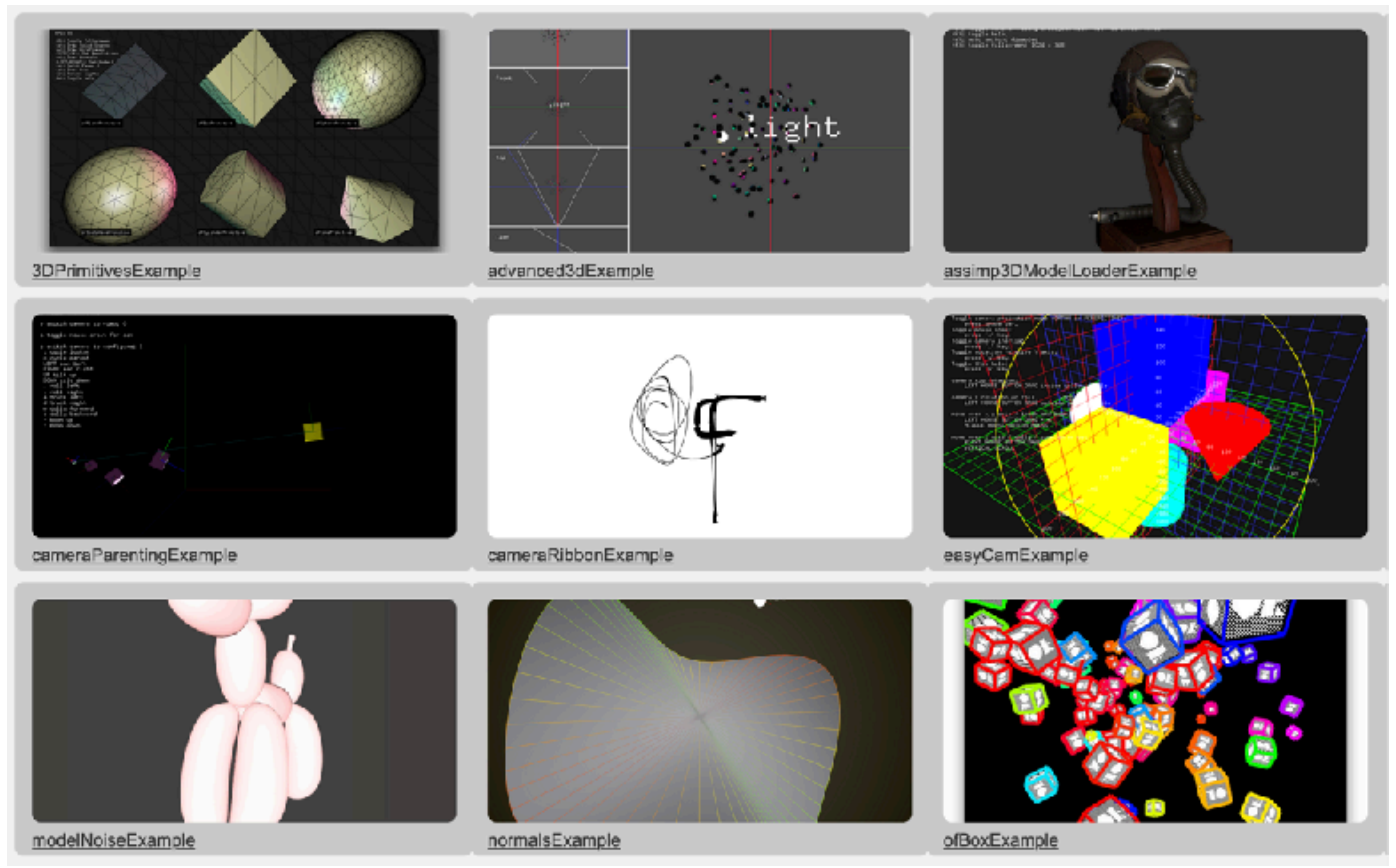
- 1 index card + 5 post-its
- On the index card, write how many sheets of new plywood you used and any messages to your audience
  - I’ll collect these at the end so the course can pay the HMC makerspace :)
- 5 post-its: initial impressions for at least 5 pieces
  - Every piece should have at least 2 comments

Intro to Creative Coding

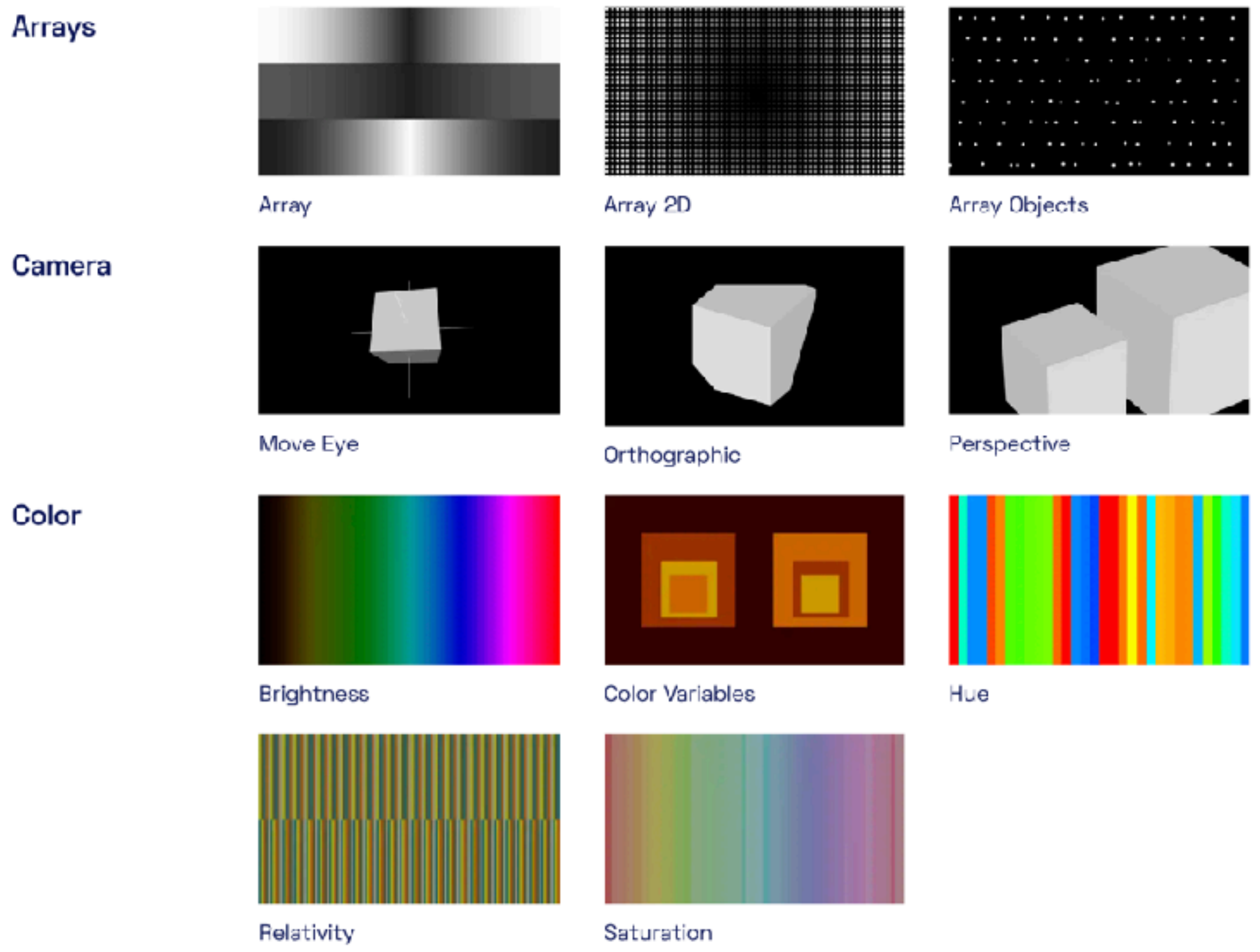
Creative coding

- Code written to be *expressive* rather than functional
- Many “domain specific languages” (DSLs), such as…

Compared to non-coding digital art tools (like Photoshop), what kinds of art can only be made with code?



openFrameworks (C++)



Processing (Java)

Processing.py

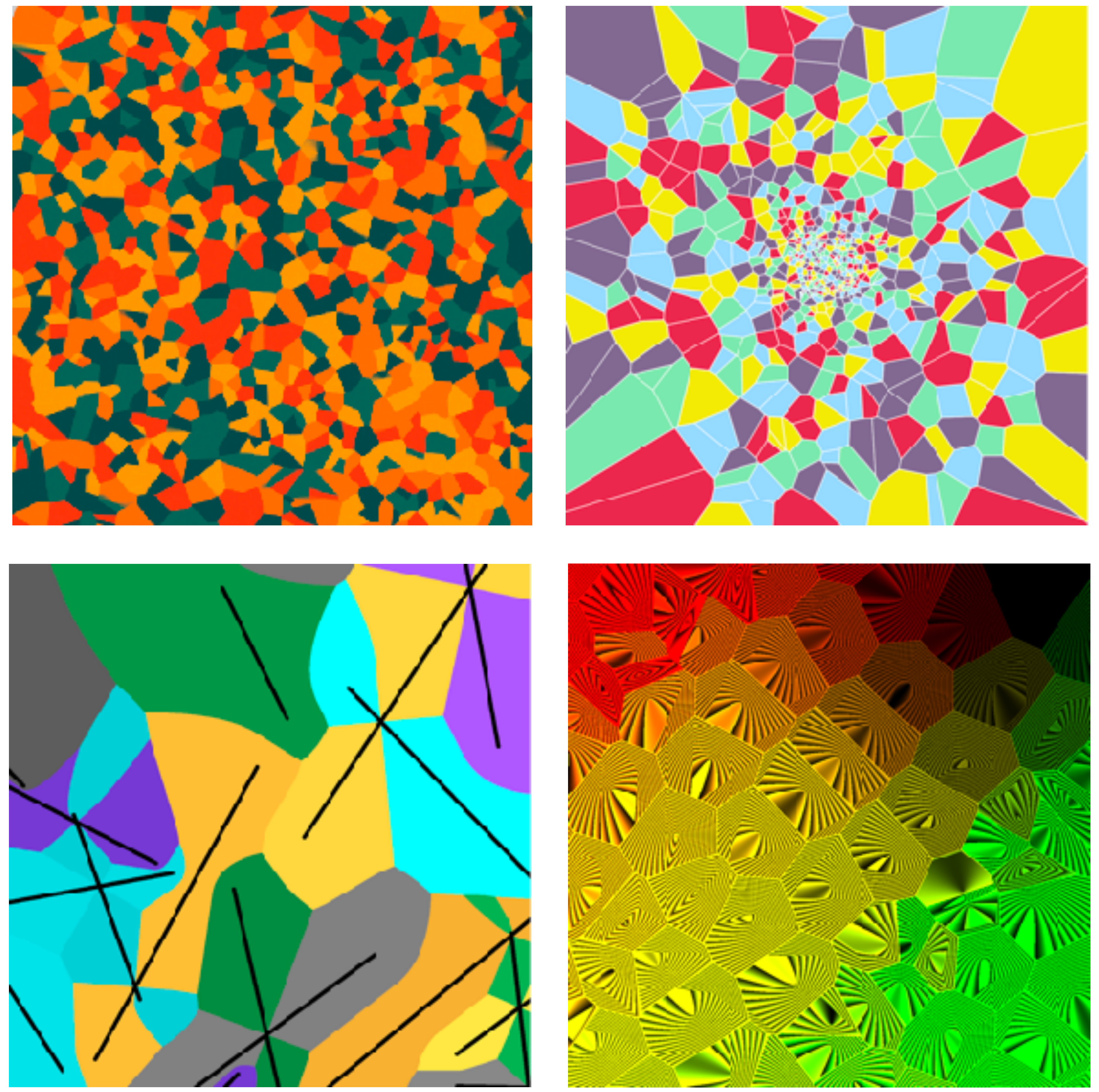
p5.js

Processing for Android

Processing for Pi

Use cases

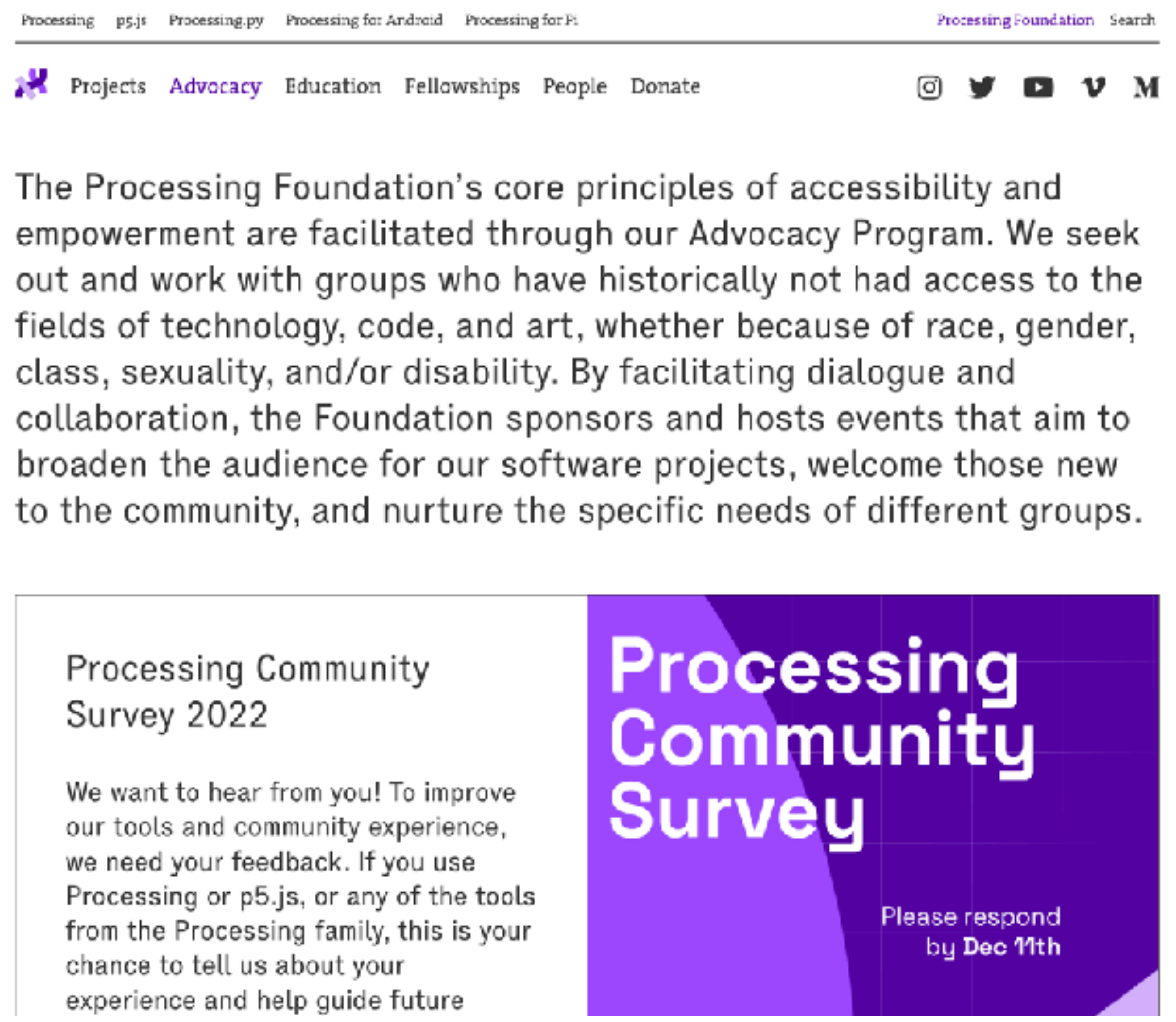
Interactive Art


Algorithmic art

Video Jockeying (VJing)

Community oriented

- “From the beginning, Processing was designed to be as simple as possible for beginners, knowing that its simplicity would also benefit more experienced users as well.”
- “[…] to empower people of all interests and backgrounds to learn how to program and make creative work with code, especially those who might not otherwise have access to these tools and resources.”



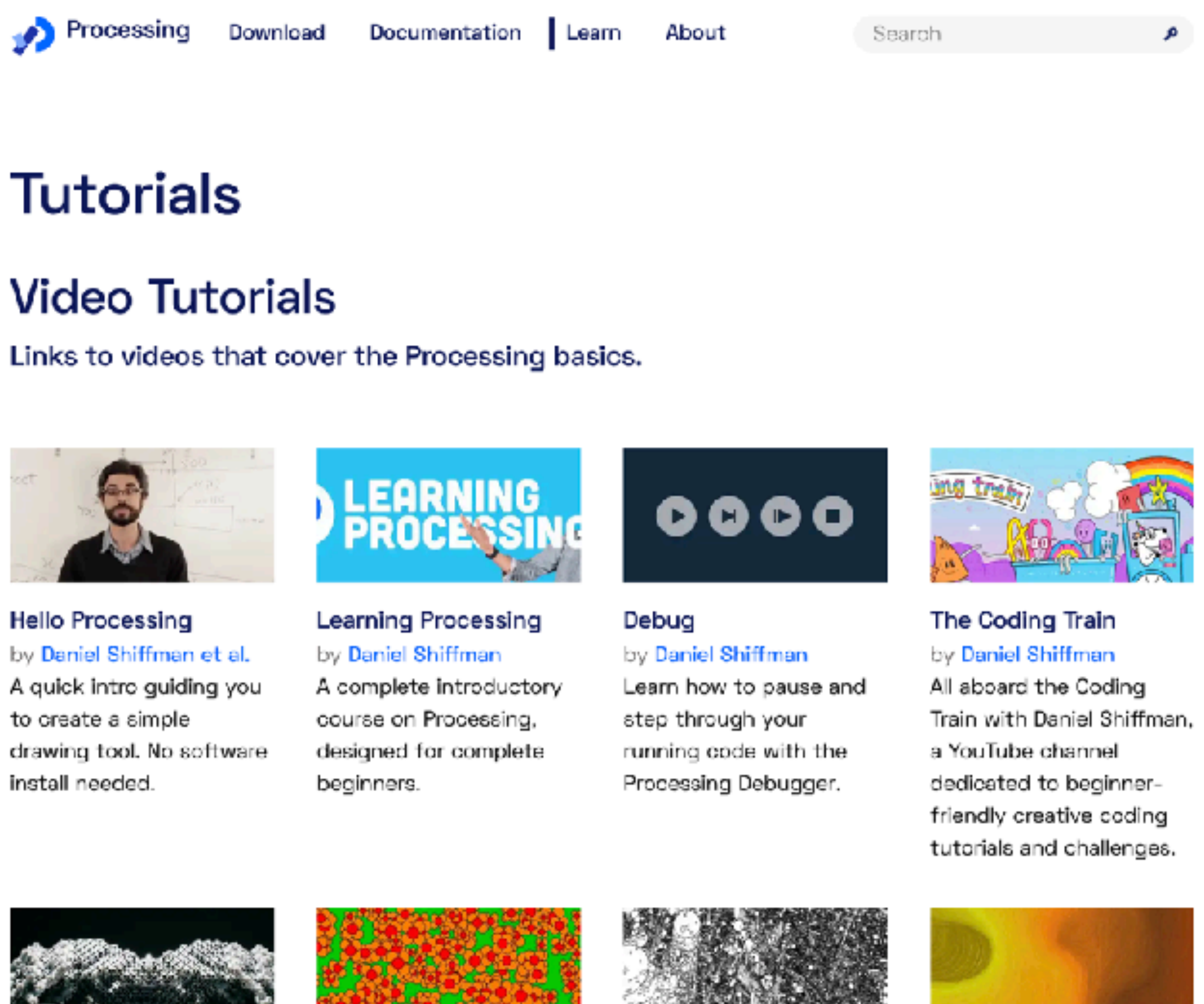
The Processing Foundation’s core principles of accessibility and empowerment are facilitated through our Advocacy Program. We seek out and work with groups who have historically not had access to the fields of technology, code, and art, whether because of race, gender, class, sexuality, and/or disability. By facilitating dialogue and collaboration, the Foundation sponsors and hosts events that aim to broaden the audience for our software projects, welcome those new to the community, and nurture the specific needs of different groups.

Processing Community Survey 2022

We want to hear from you! To improve our tools and community experience, we need your feedback. If you use Processing or p5.js, or any of the tools from the Processing family, this is your chance to tell us about your experience and help guide future

Processing Community Survey

Please respond by Dec 11th



Tutorials

Video Tutorials

Links to videos that cover the Processing basics.

Hello Processing
by Daniel Shiffman et al.
A quick intro guiding you to create a simple drawing tool. No software install needed.

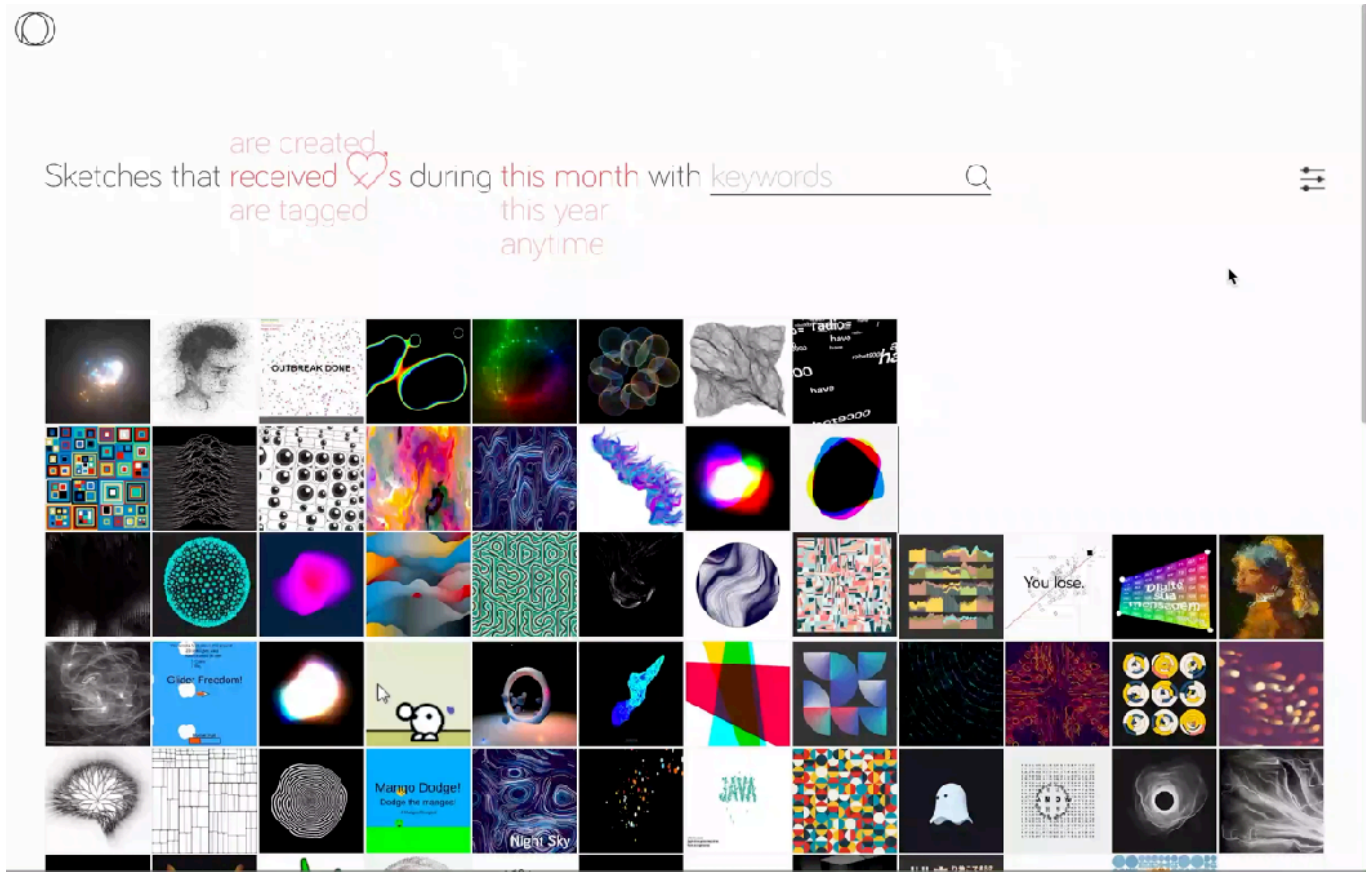
Learning Processing
by Daniel Shiffman
A complete introductory course on Processing, designed for complete beginners.

Debug
by Daniel Shiffman
Learn how to pause and step through your running code with the Processing Debugger.

The Coding Train
by Daniel Shiffman
All aboard the Coding Train with Daniel Shiffman, a YouTube channel dedicated to beginner-friendly creative coding tutorials and challenges.

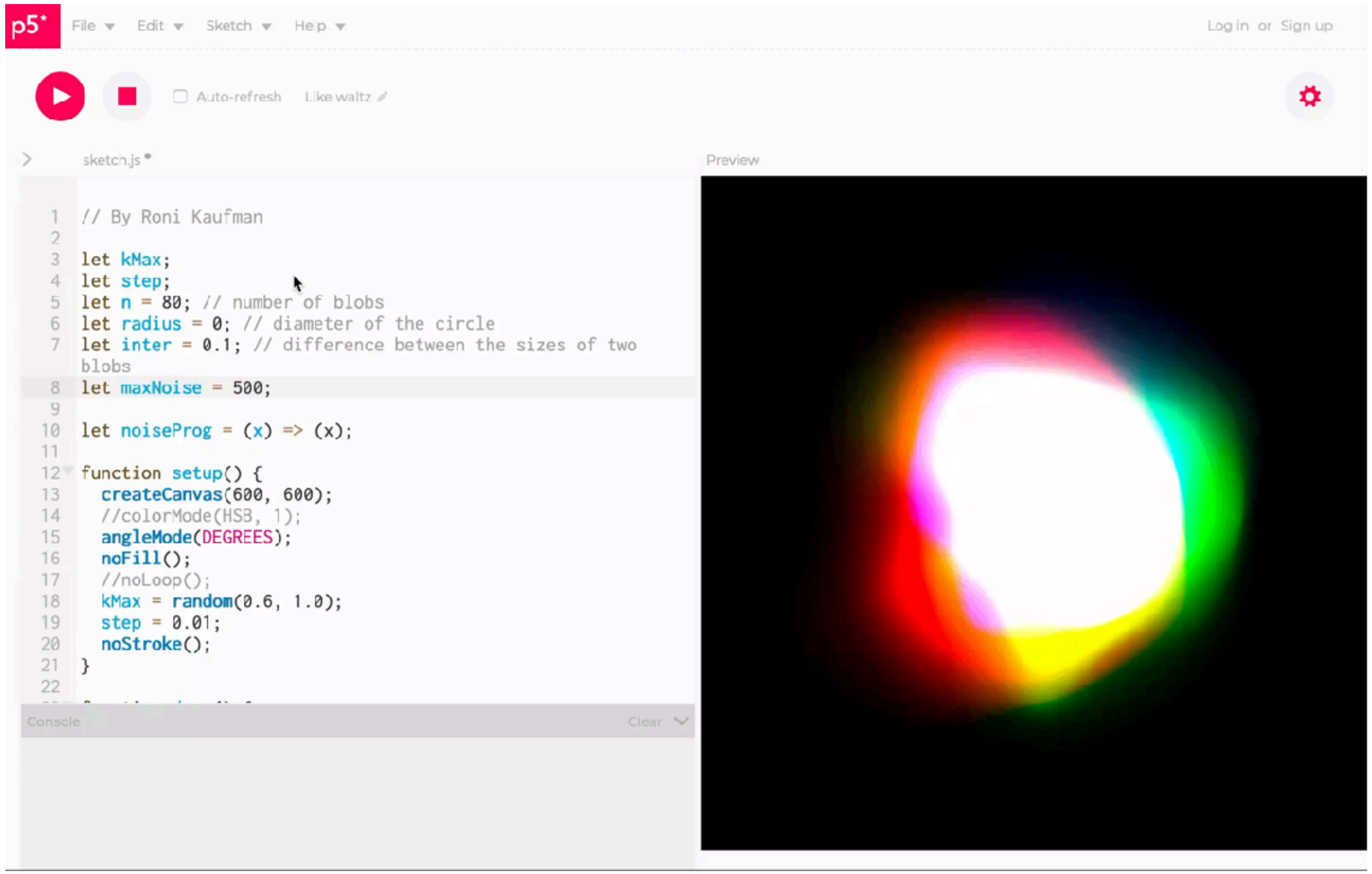
Accessible to new programmers

- Language designed in response to a frustration of the cultural elitism of programming
- Learning through editing lots of community made examples and tutorials, or in person workshops


Sketches that are created / received ♡s / are tagged during this month / this year / anytime with keywords
OUTBREAK DONE
You lose.
Glider Freedom!
Mango Dodge!
Night Sky
JAVA

Accessible to new programmers

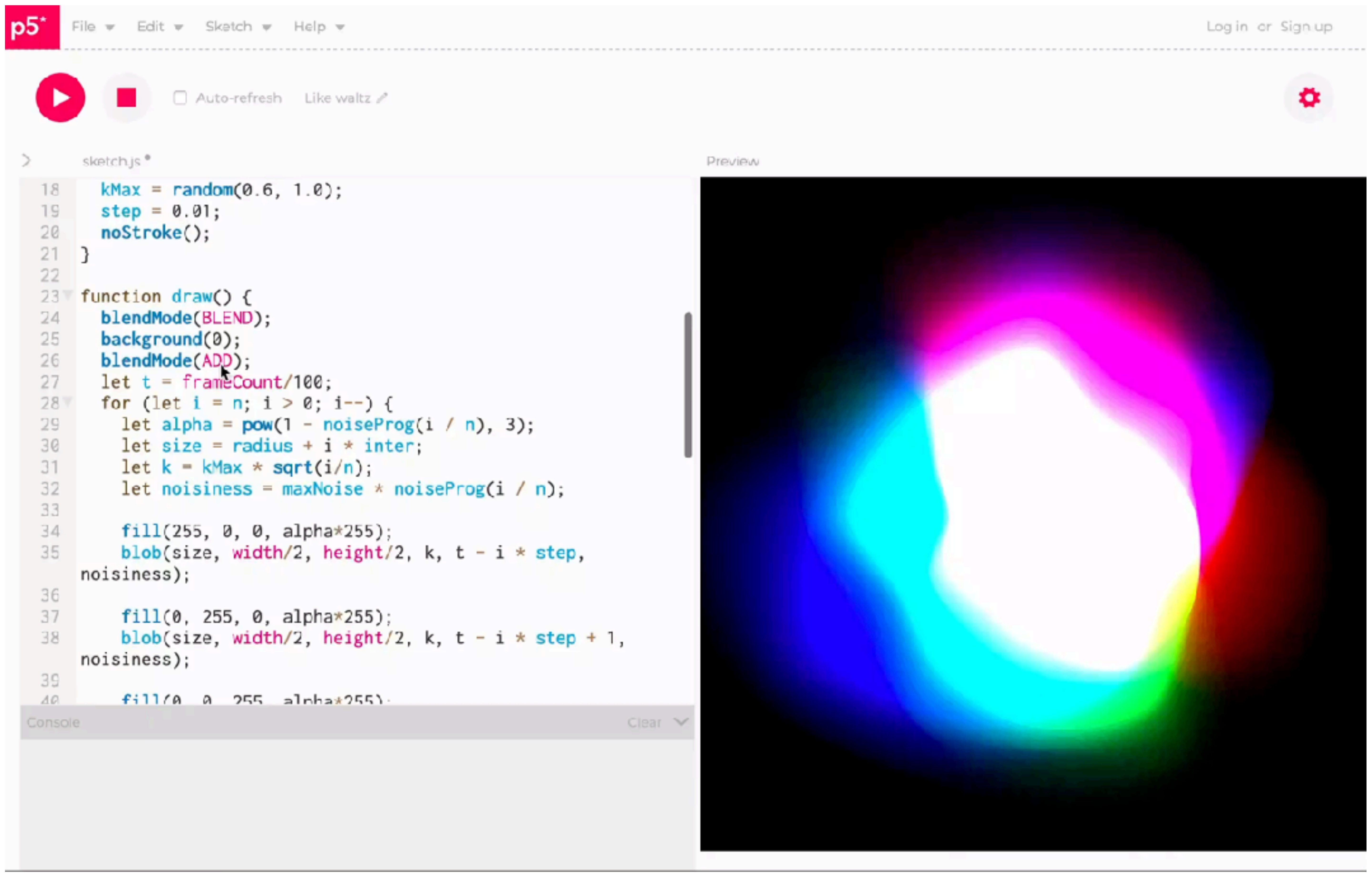
- Language designed in response to a frustration of the cultural elitism of programming
- Learning through editing lots of community made examples and tutorials, or in person workshops



```
// By Roni Kaufman

let kMax;
let step;
let n = 80; // number of blobs
let radius = 0; // diameter of the circle
let inter = 0.1; // difference between the sizes of two
blobs
let maxNoise = 500;

let noiseProg = (x) => (x);

function setup() {
  createCanvas(600, 600);
  //colorMode(HSB, 1);
  angleMode(DEGREES);
  noFill();
  //noLoop();
  kMax = random(0.6, 1.0);
  step = 0.01;
  noStroke();
}
```

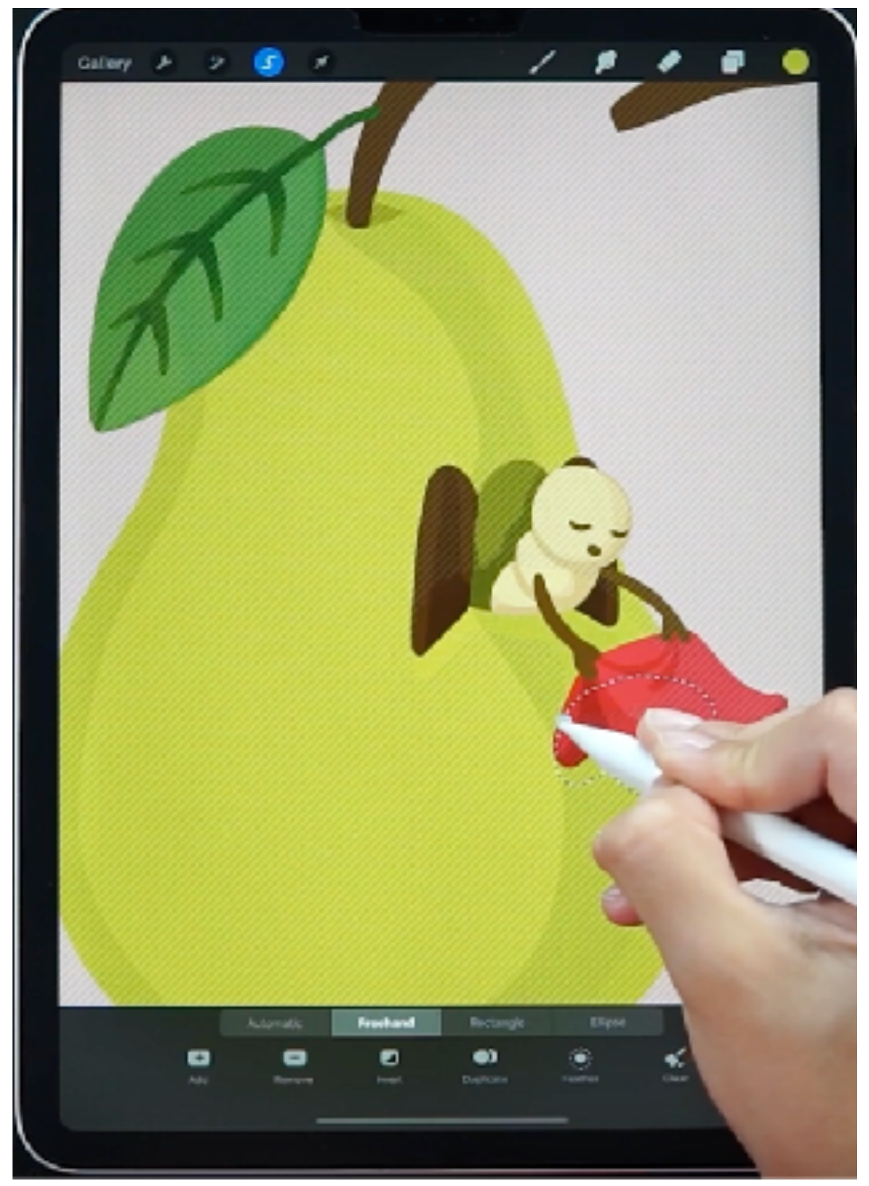
Accessible to new programmers, but different than manually making art

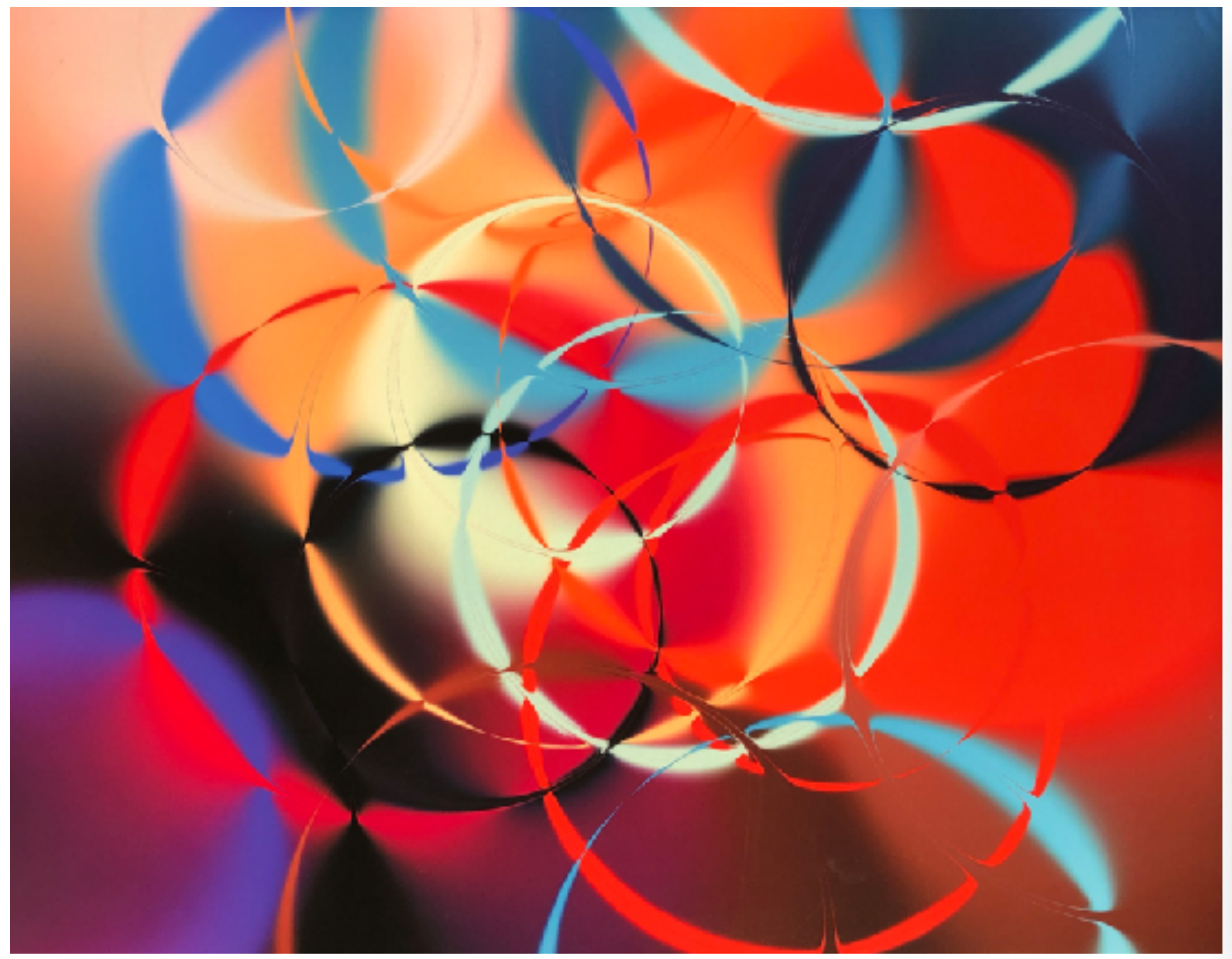
- Language designed in response to a frustration of the cultural elitism of programming
- Learning through editing lots of community made examples and tutorials, or in person workshops

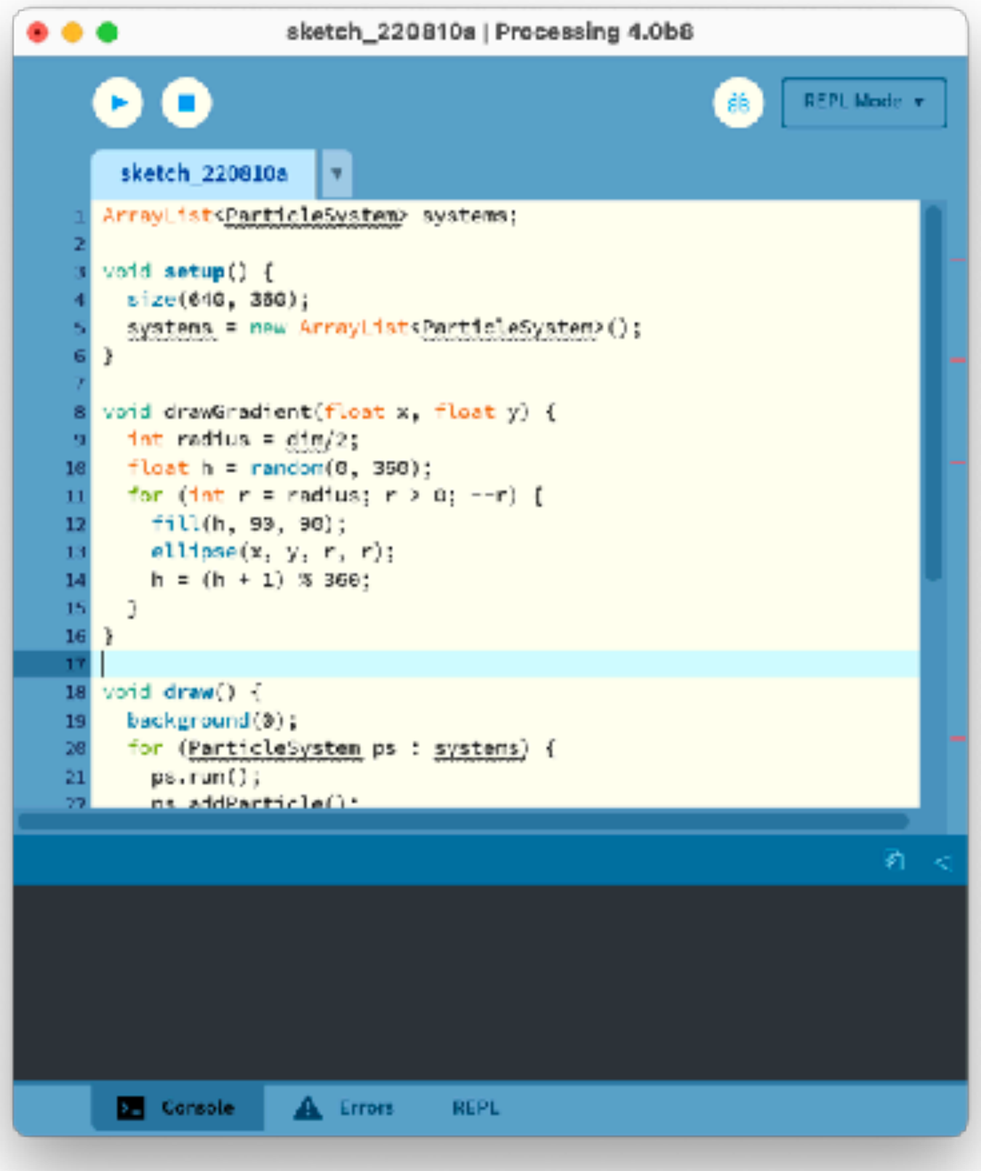


```
  kMax = random(0.6, 1.0);
  step = 0.01;
  noStroke();
}

function draw() {
  blendMode(BLEND);
  background(0);
  blendMode(ADD);
  let t = frameCount/100;
  for (let i = n; i > 0; i--) {
    let alpha = pow(1 - noiseProg(i / n), 3);
    let size = radius + i * inter;
    let k = kMax * sqrt(i/n);
    let noisiness = maxNoise * noiseProg(i / n);

    fill(255, 0, 0, alpha*255);
    blob(size, width/2, height/2, k, t - i * step,
noisiness);

    fill(0, 255, 0, alpha*255);
    blob(size, width/2, height/2, k, t - i * step + 1,
noisiness);

    fill(0, 0, 255, alpha*255);
```

Artwork


Canvas


Artwork



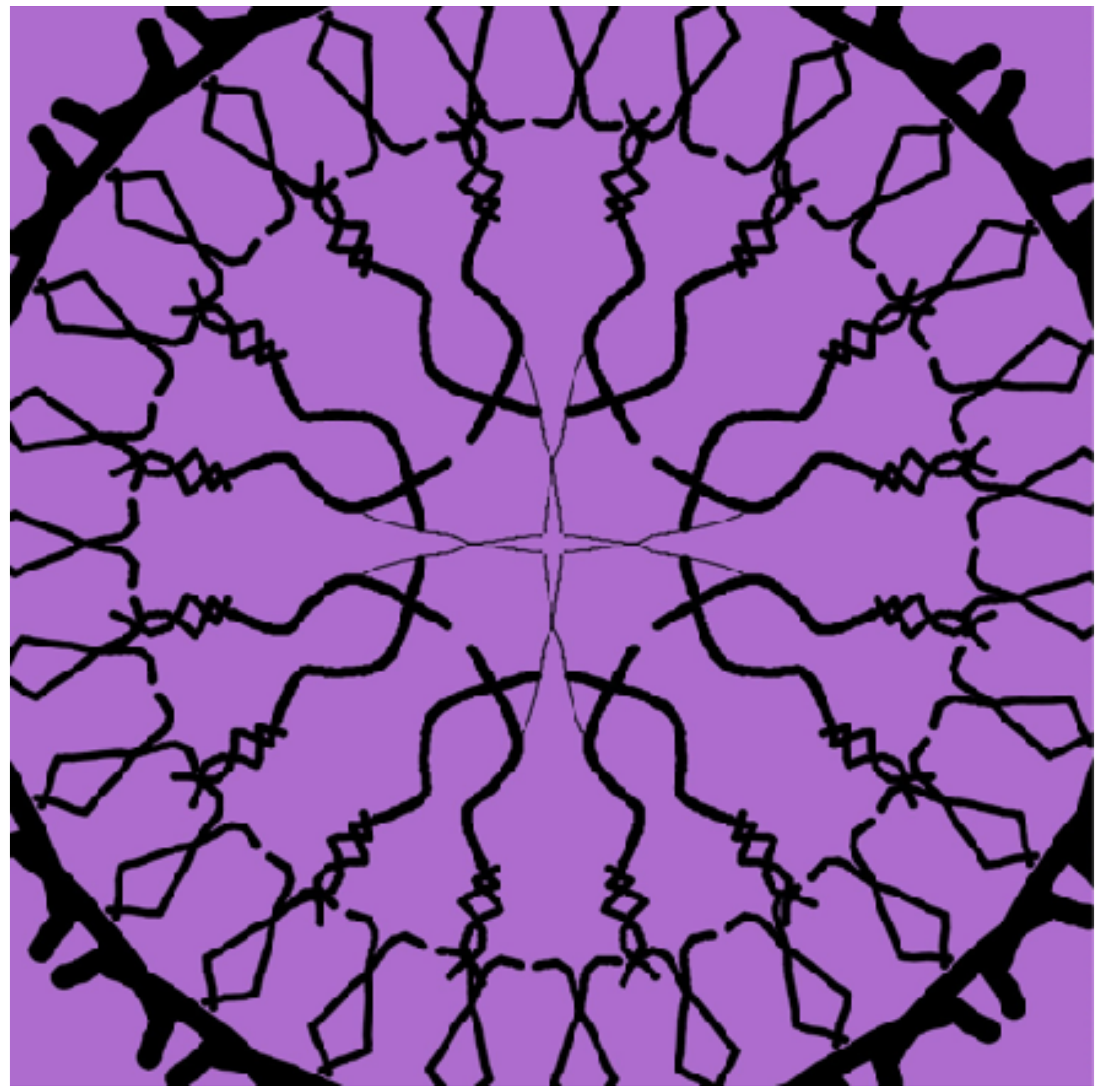
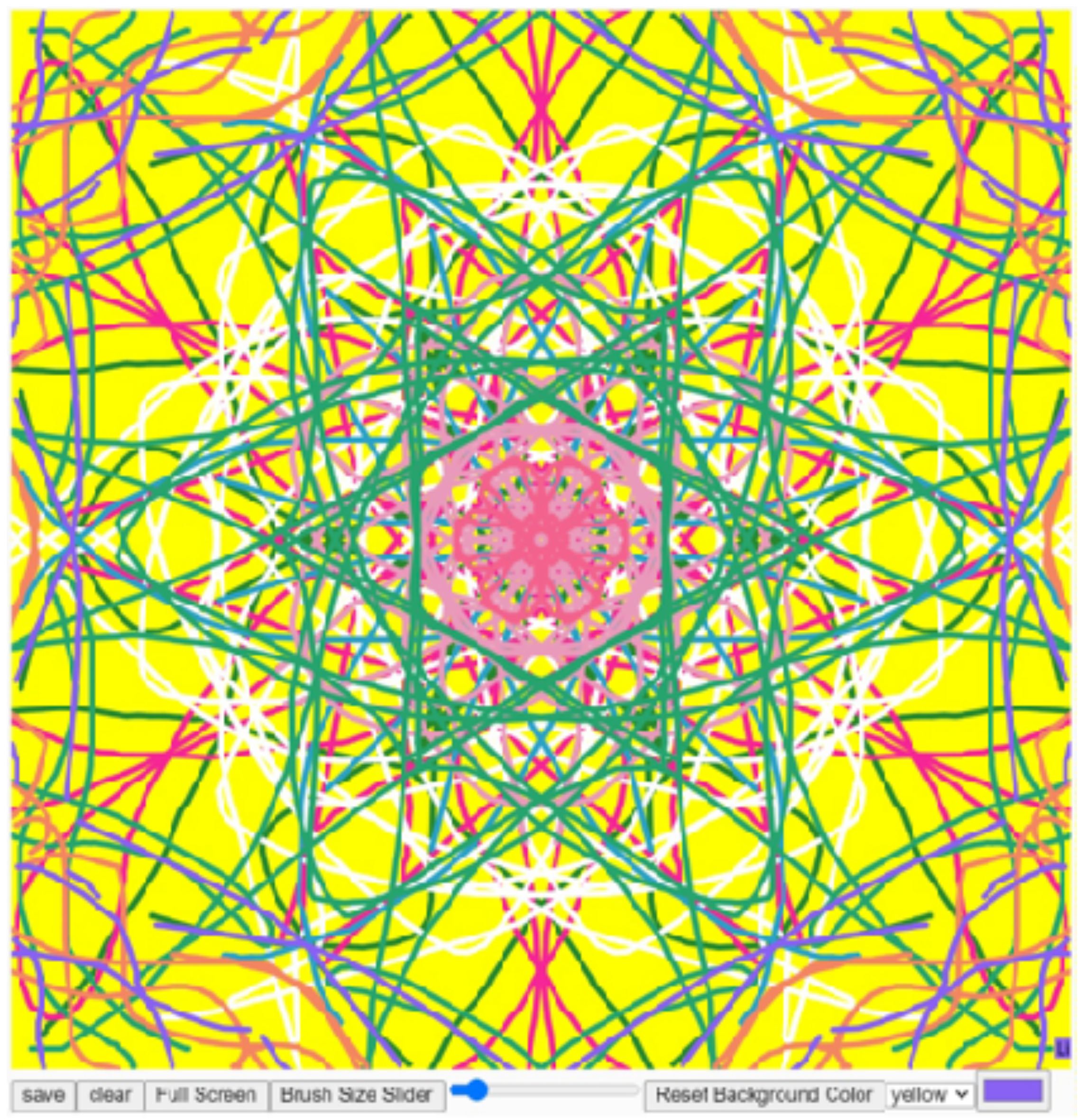
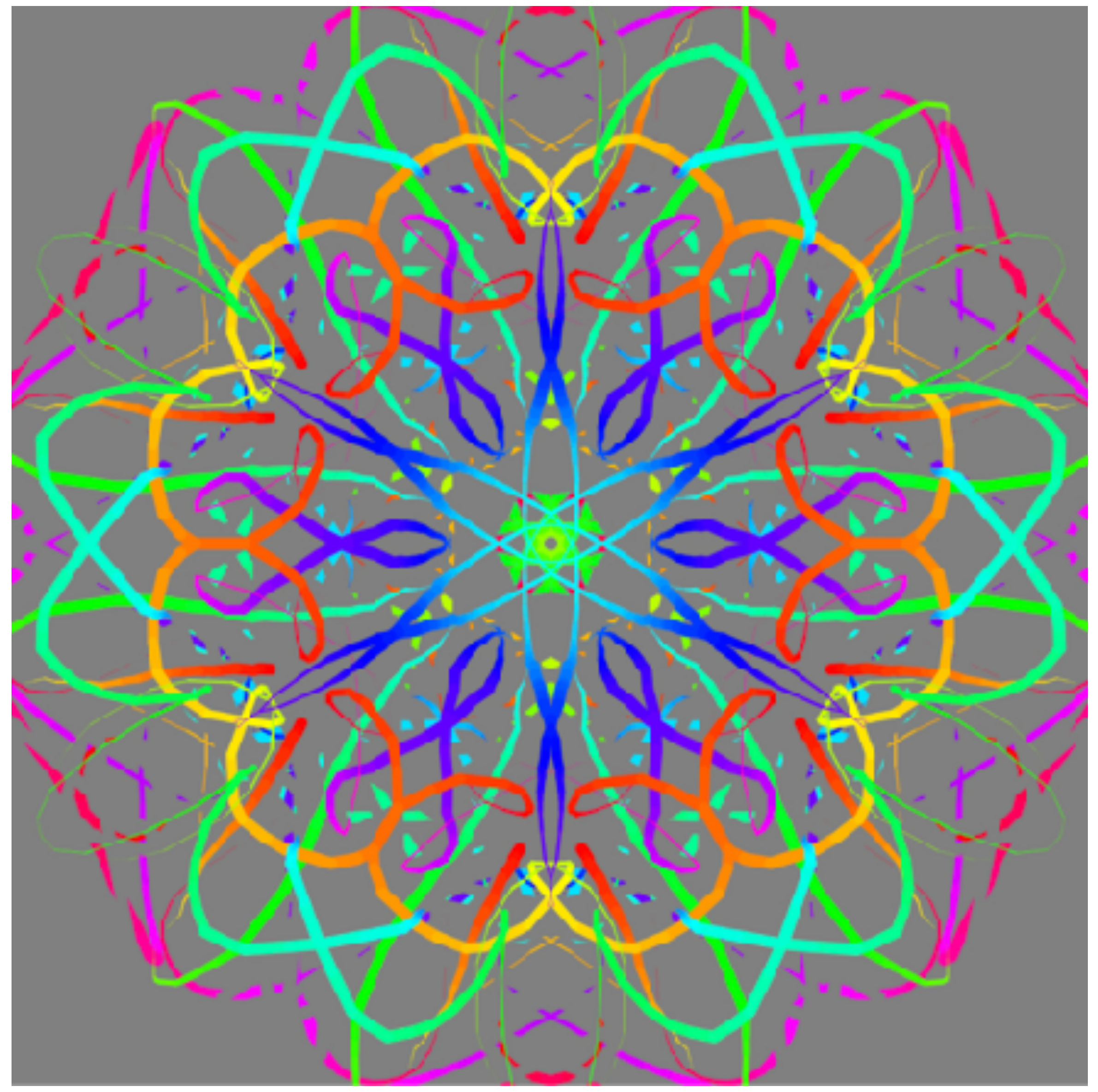
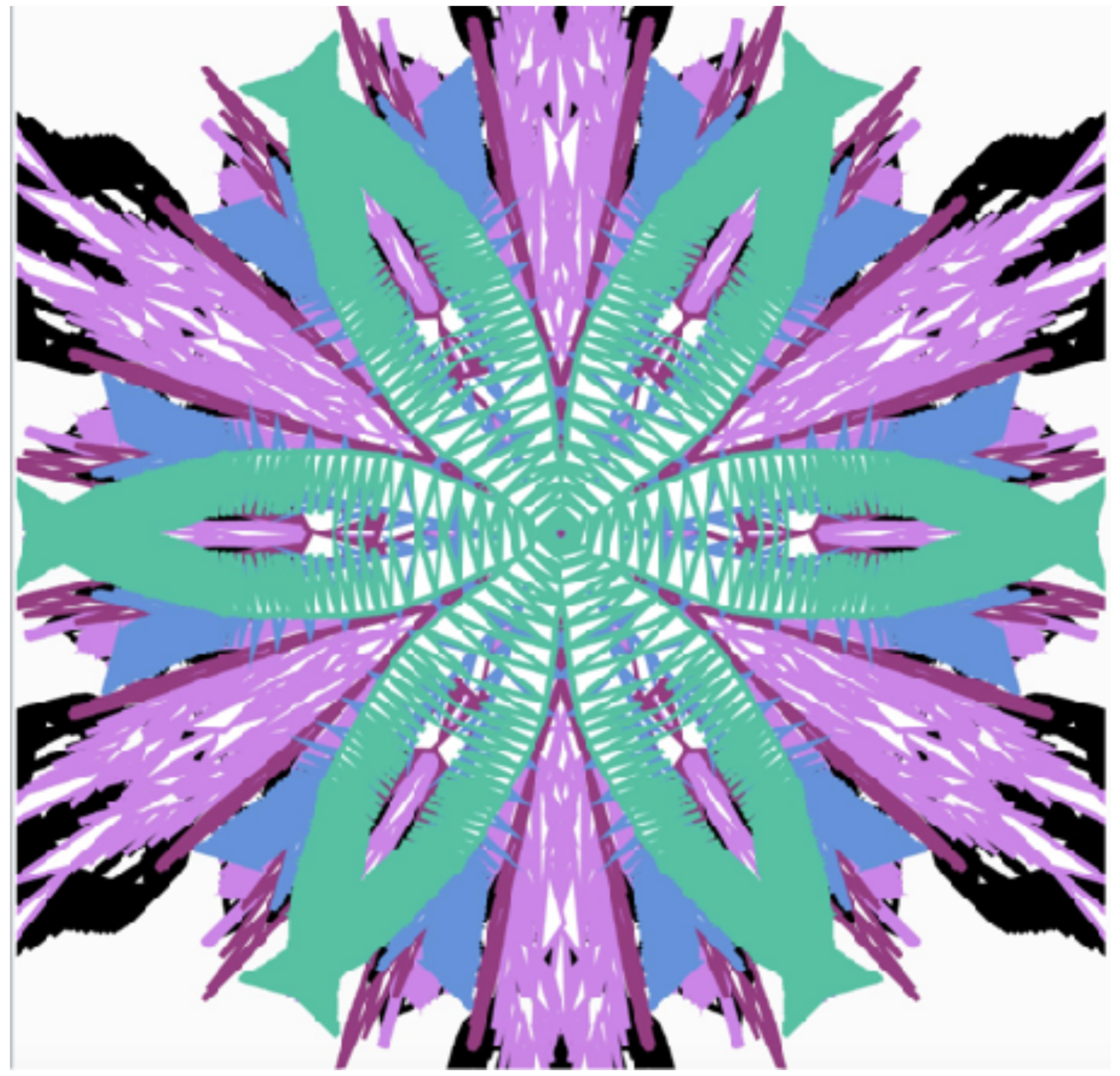
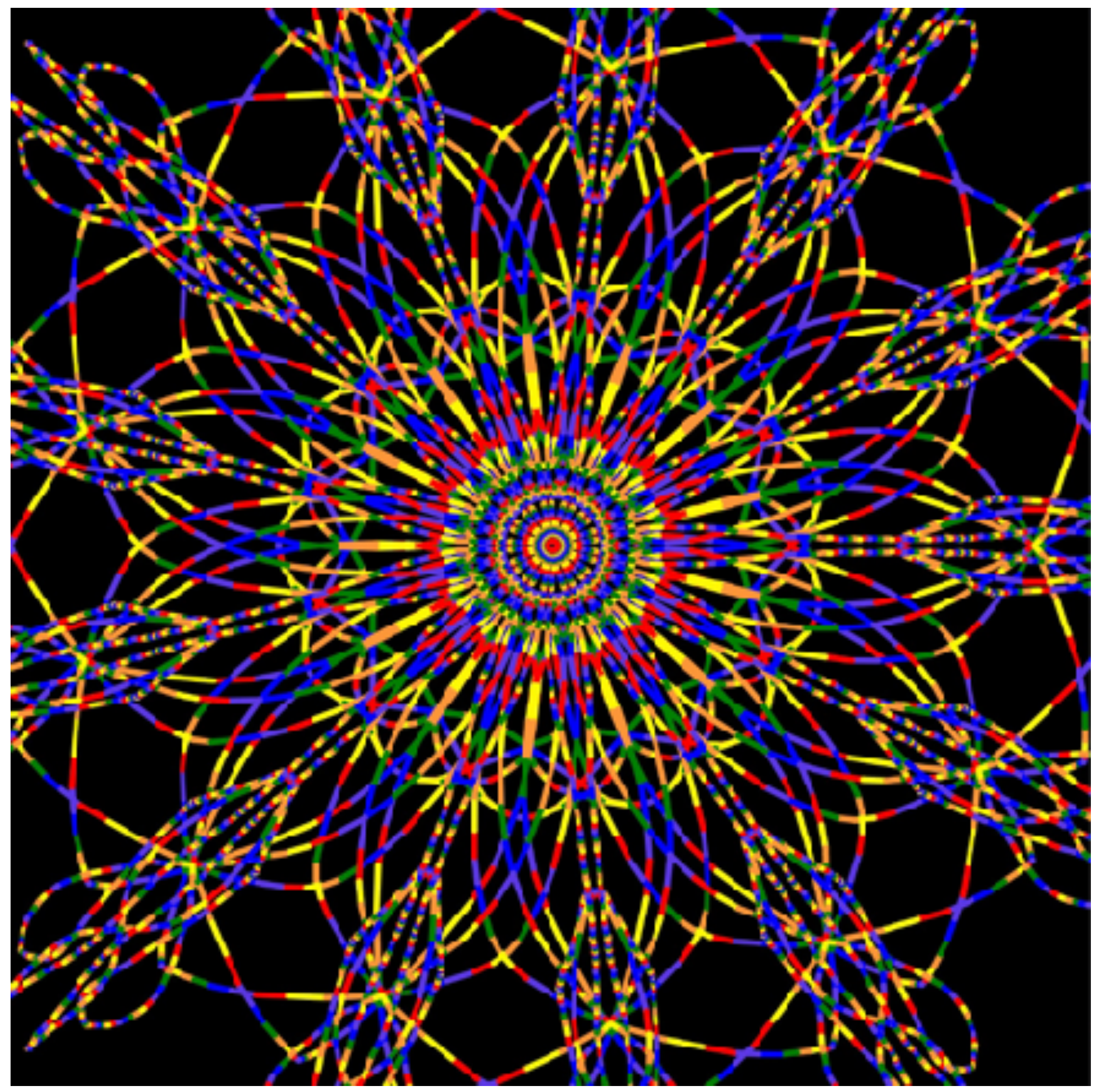
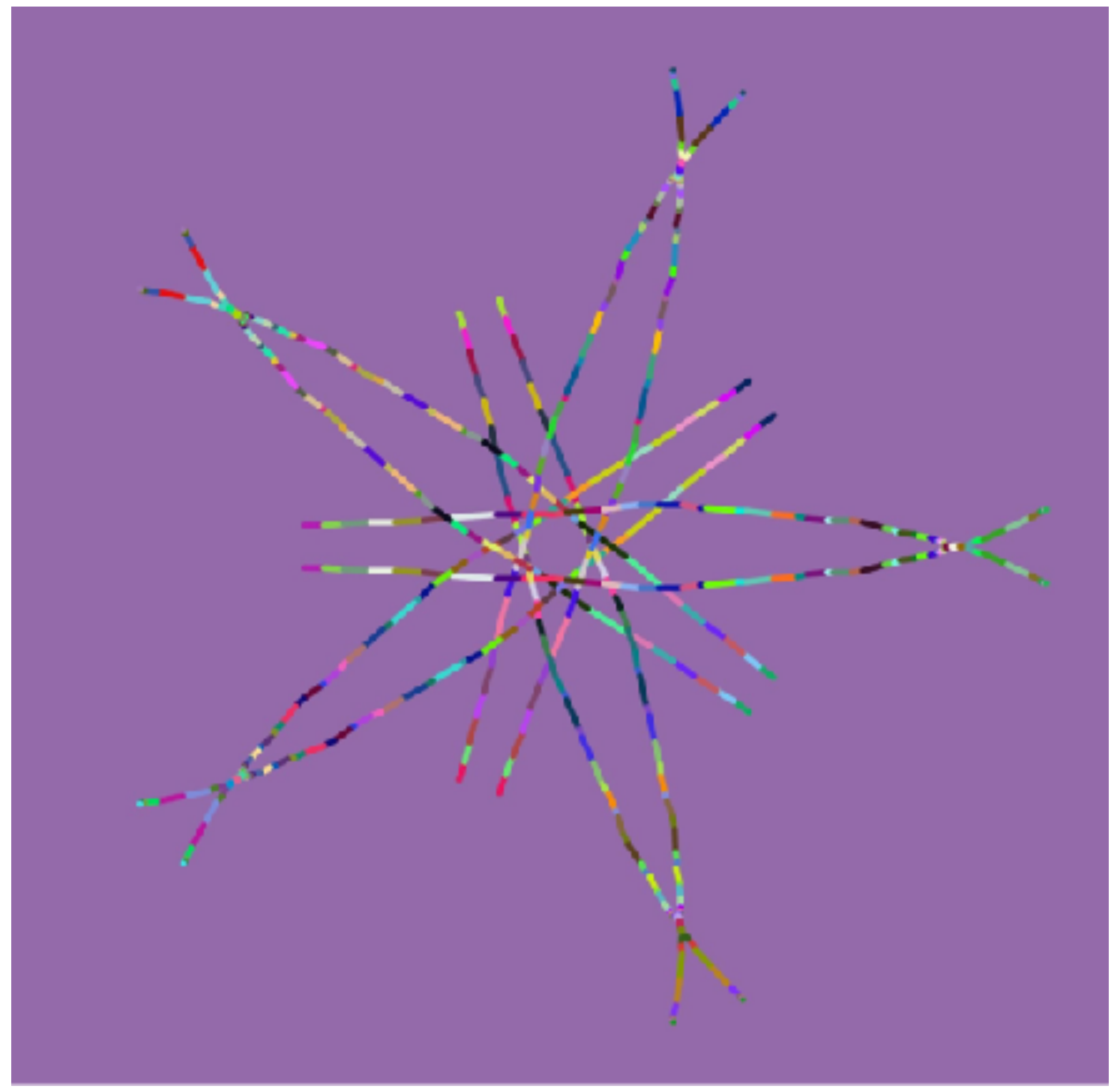
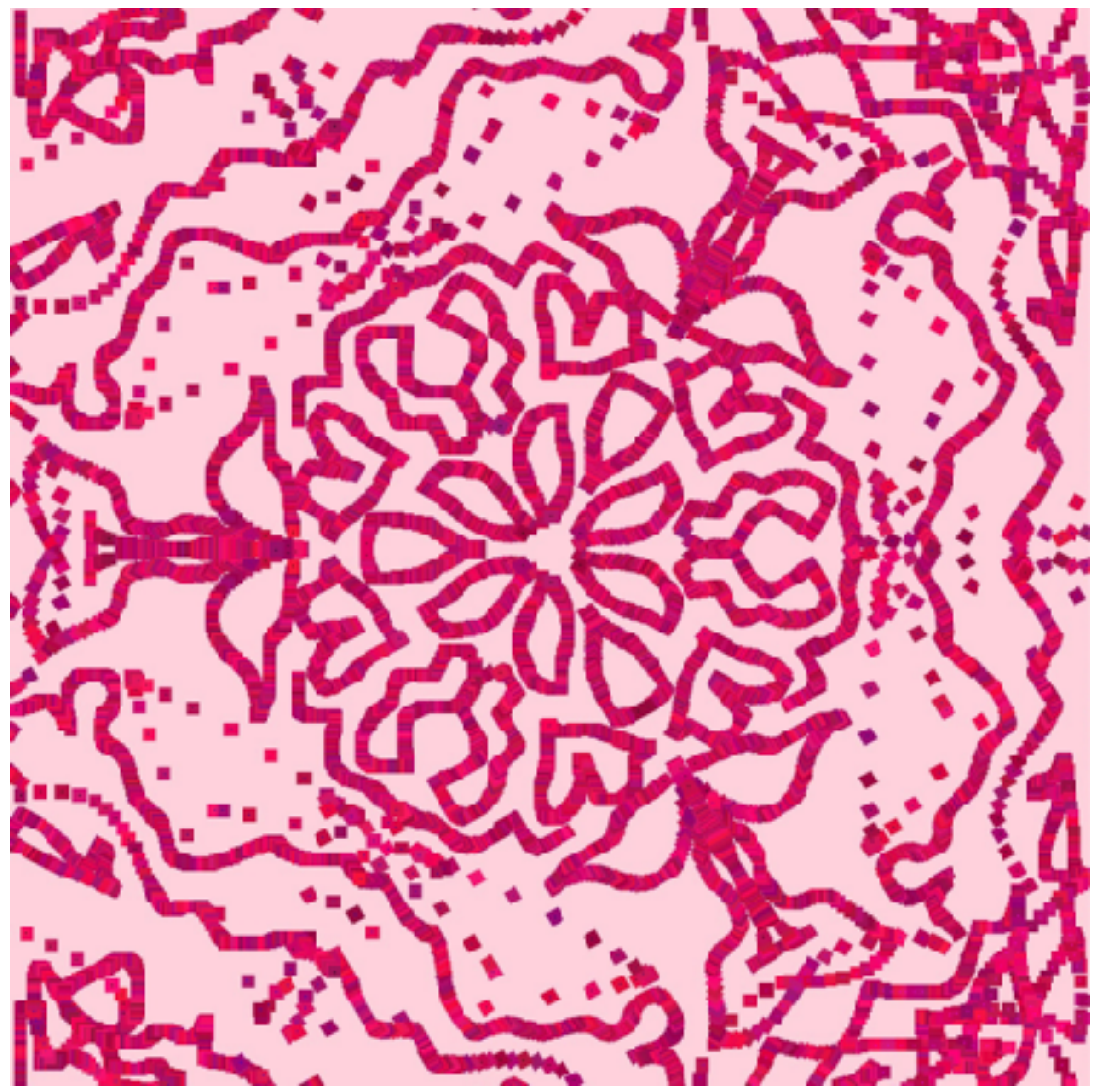
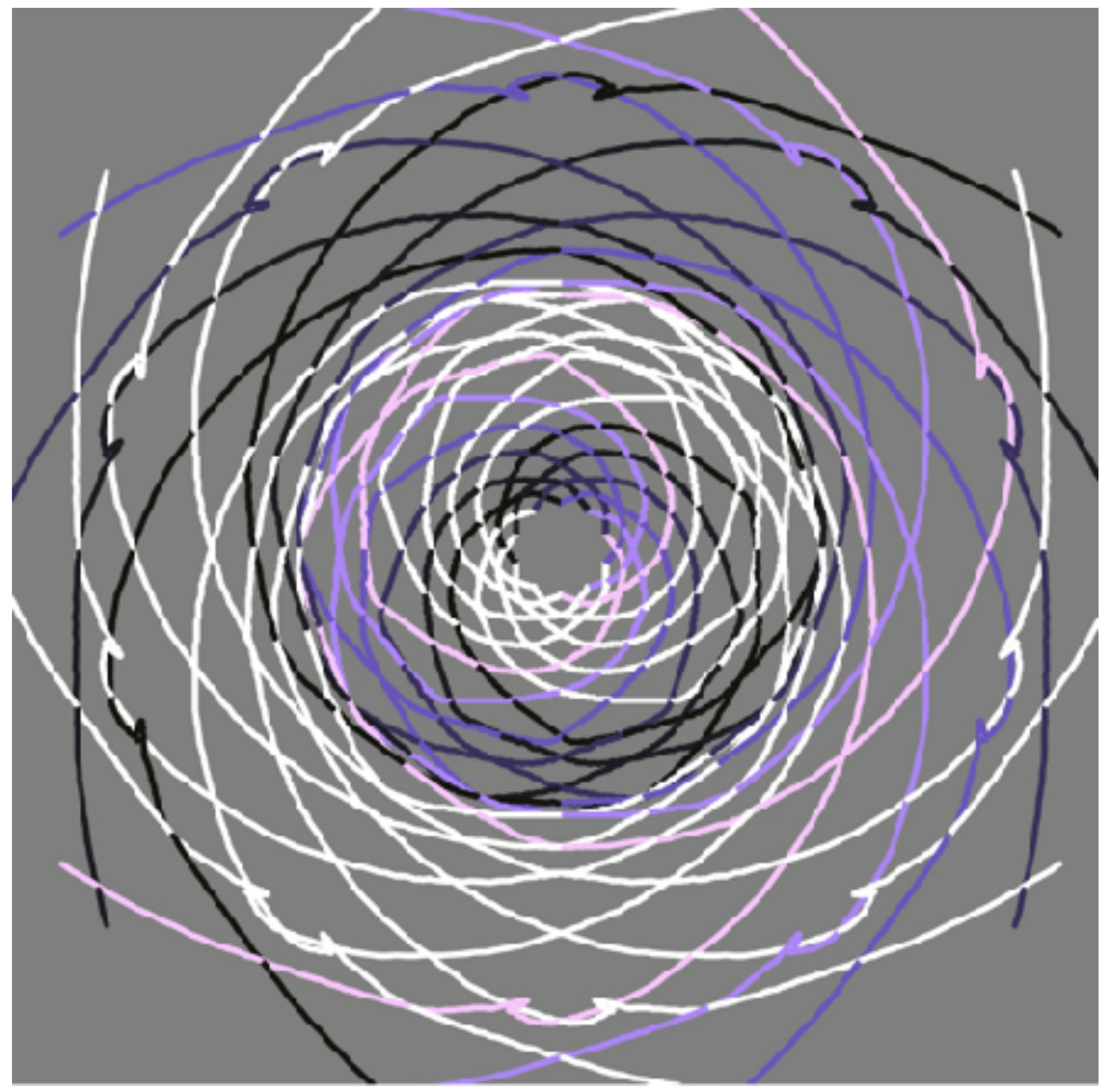
```
ArrayList<ParticleSystem> systems;

void setup() {
  size(640, 360);
  systems = new ArrayList<ParticleSystem>();
}

void drawGradient(float x, float y) {
  int radius = dim/2;
  float h = random(0, 360);
  for (int r = radius; r > 0; --r) {
    fill(h, 90, 90);
    ellipse(x, y, r, r);
    h = (h + 1) % 360;
  }
}

void draw() {
  background(0);
  for (ParticleSystem ps : systems) {
    ps.run();
    ps.addParticle();
```

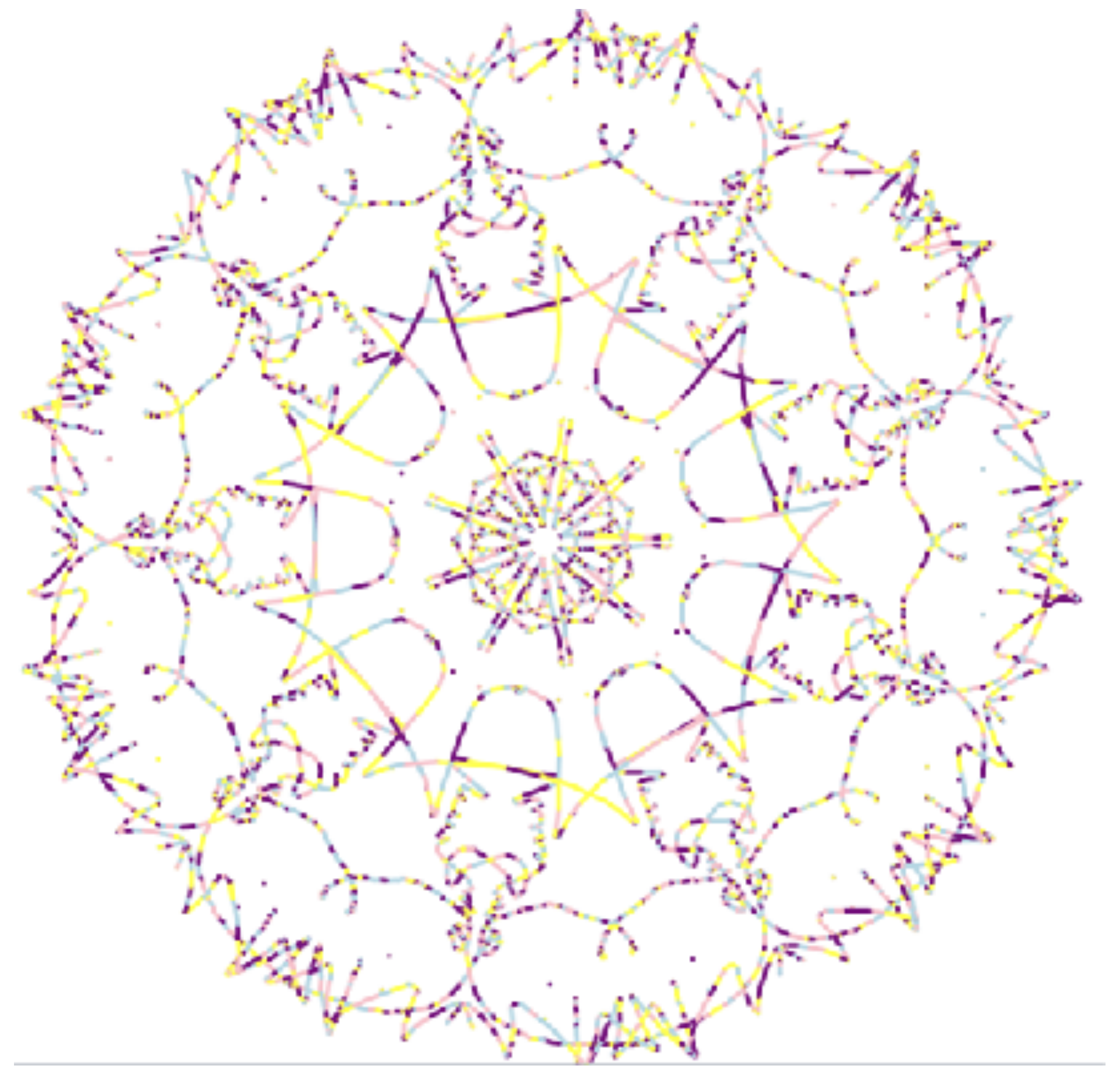
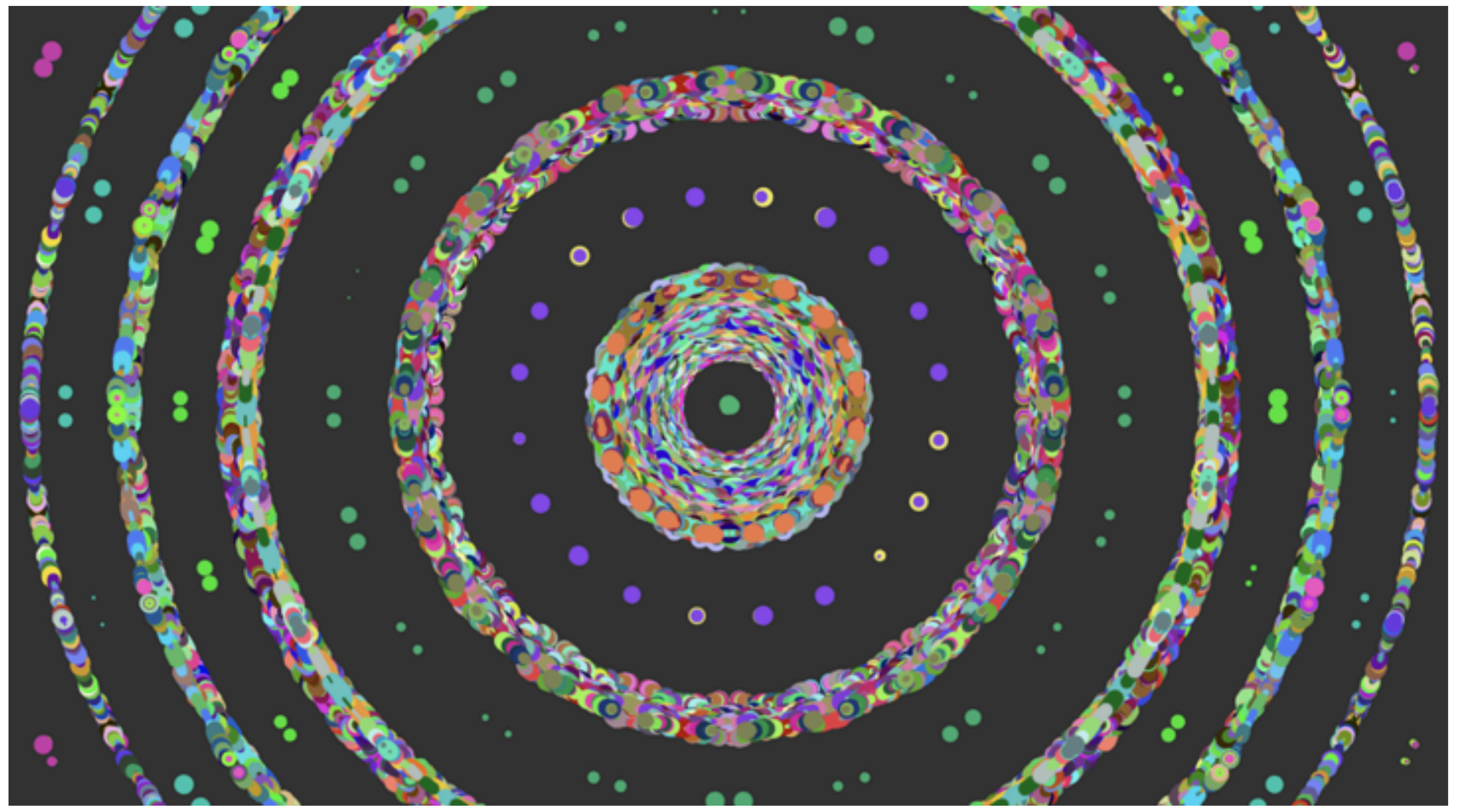
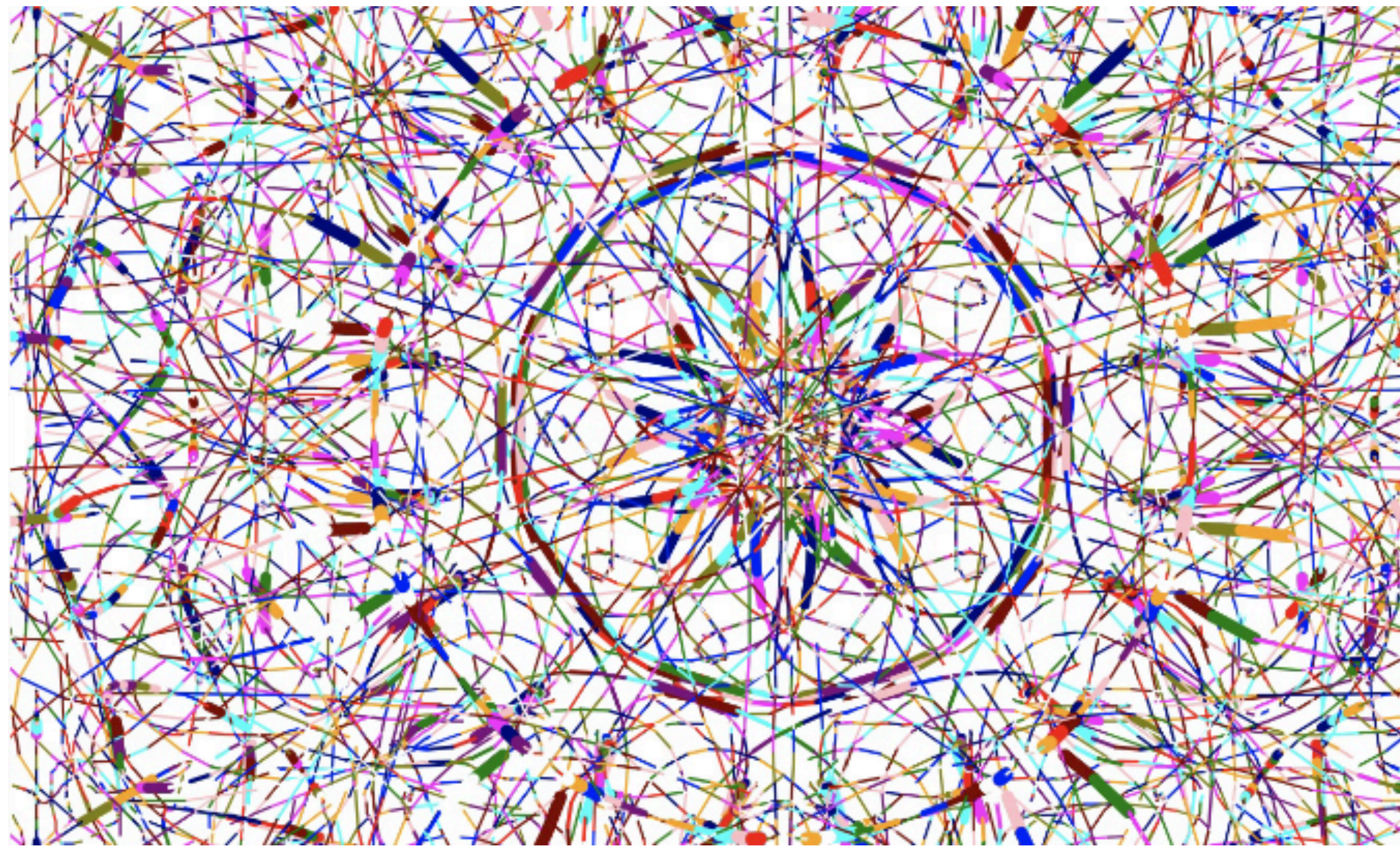
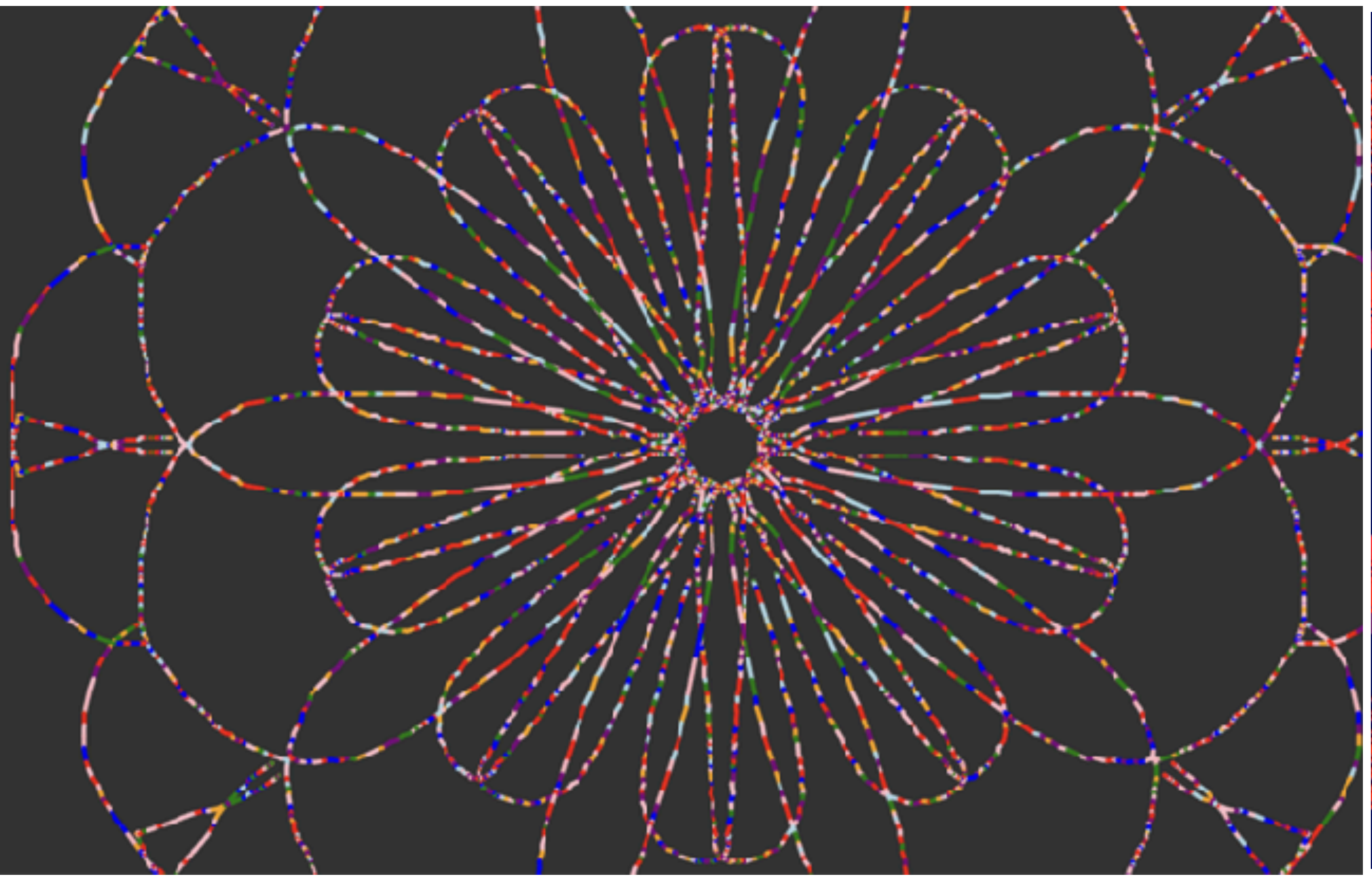
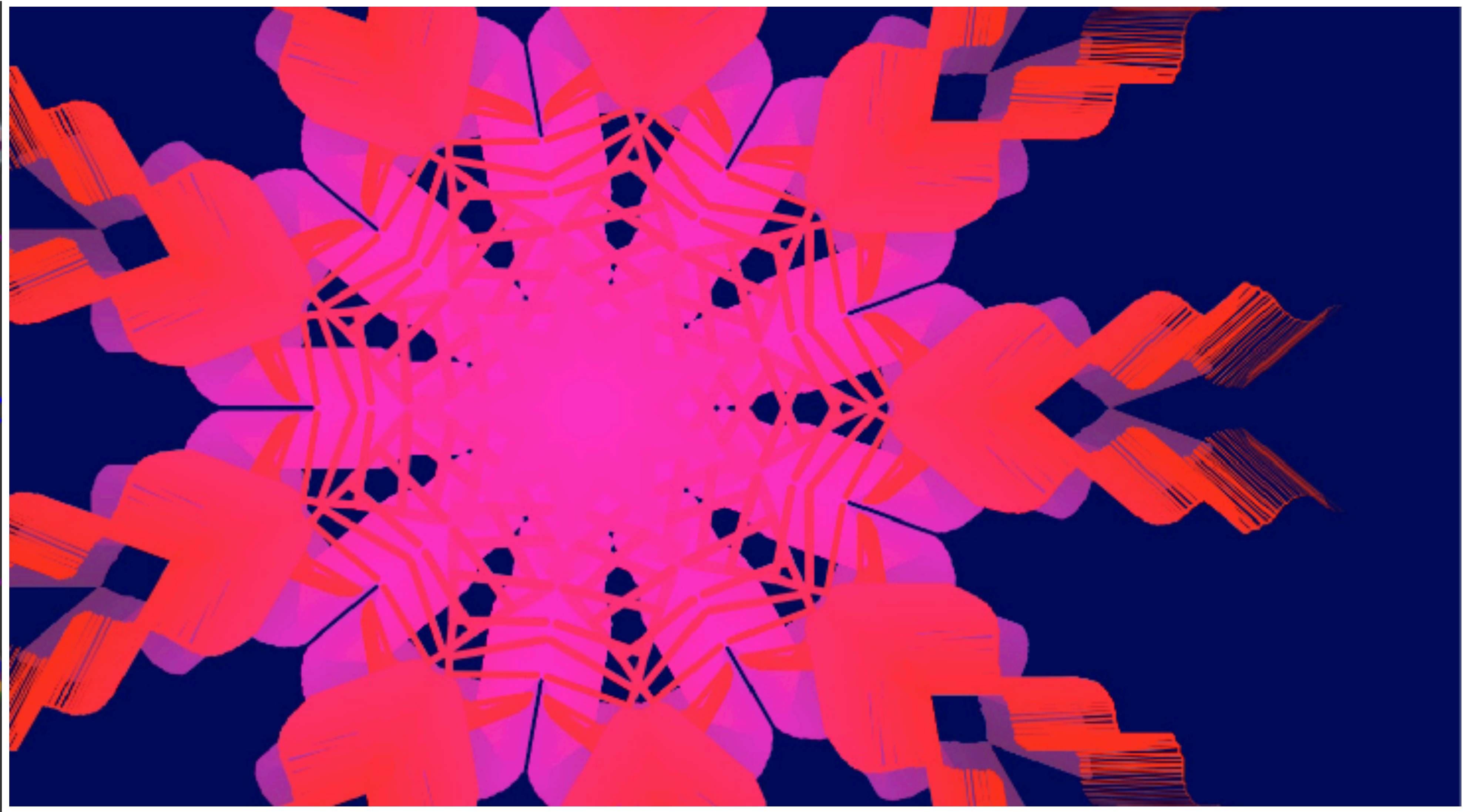
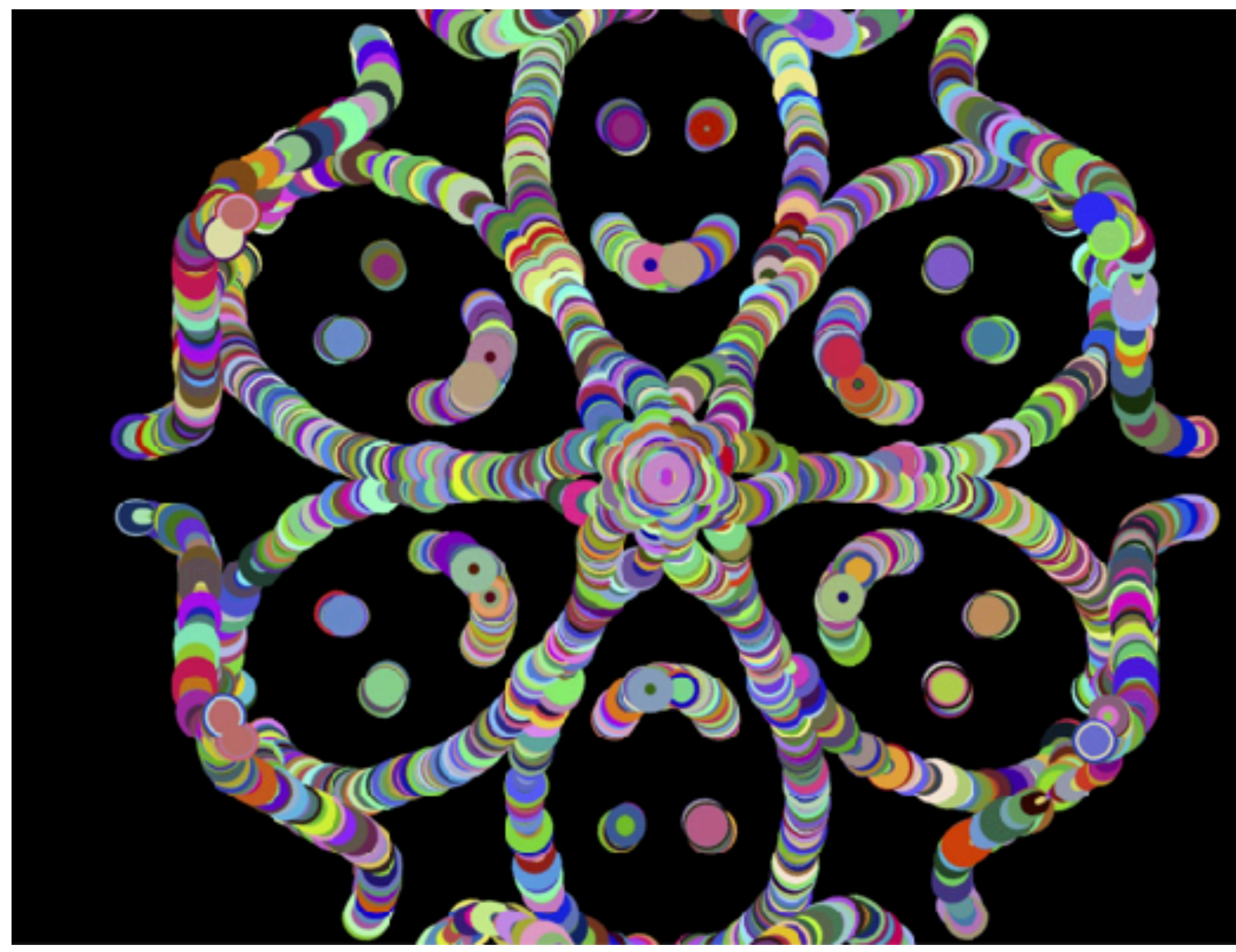
Canvas

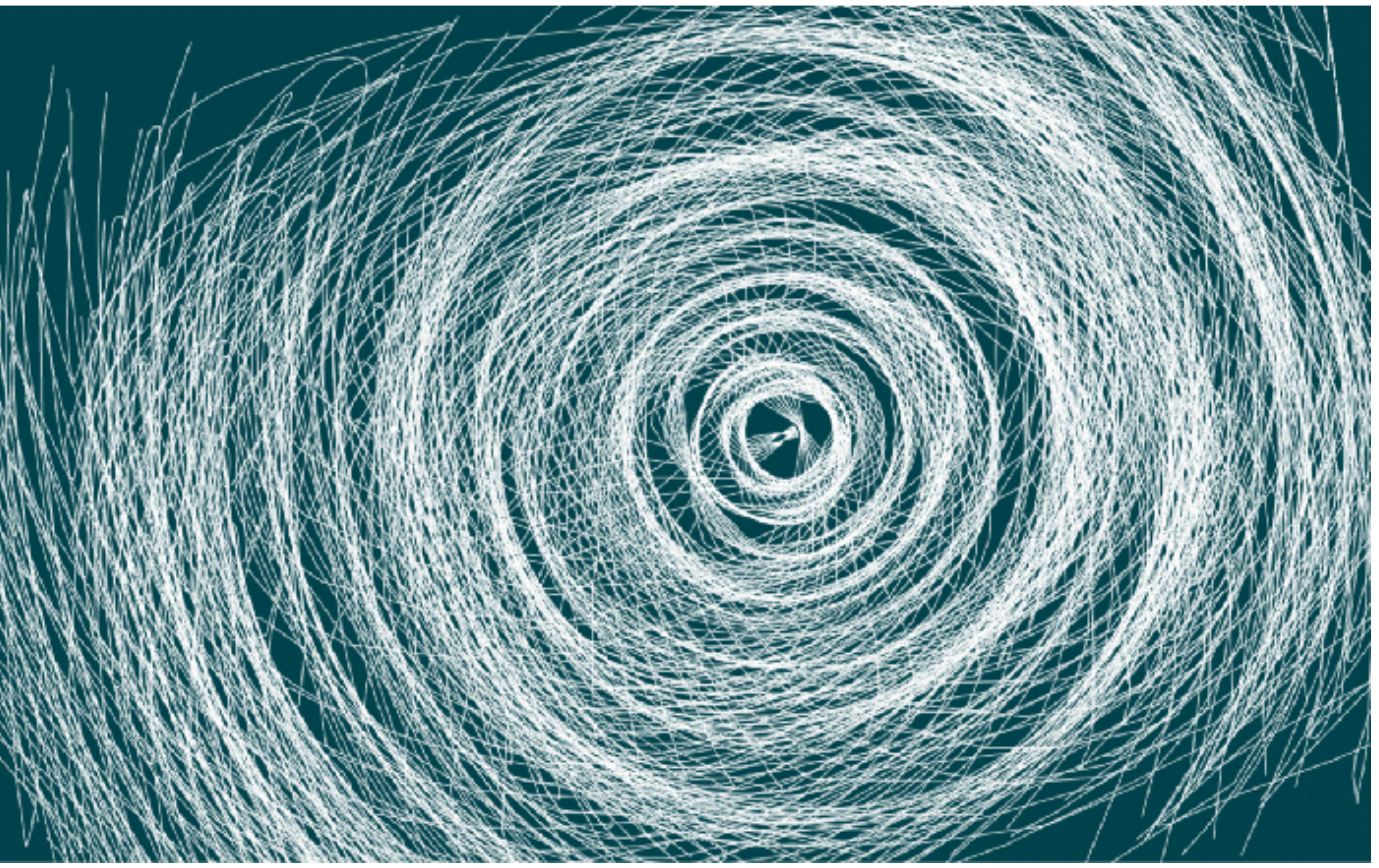
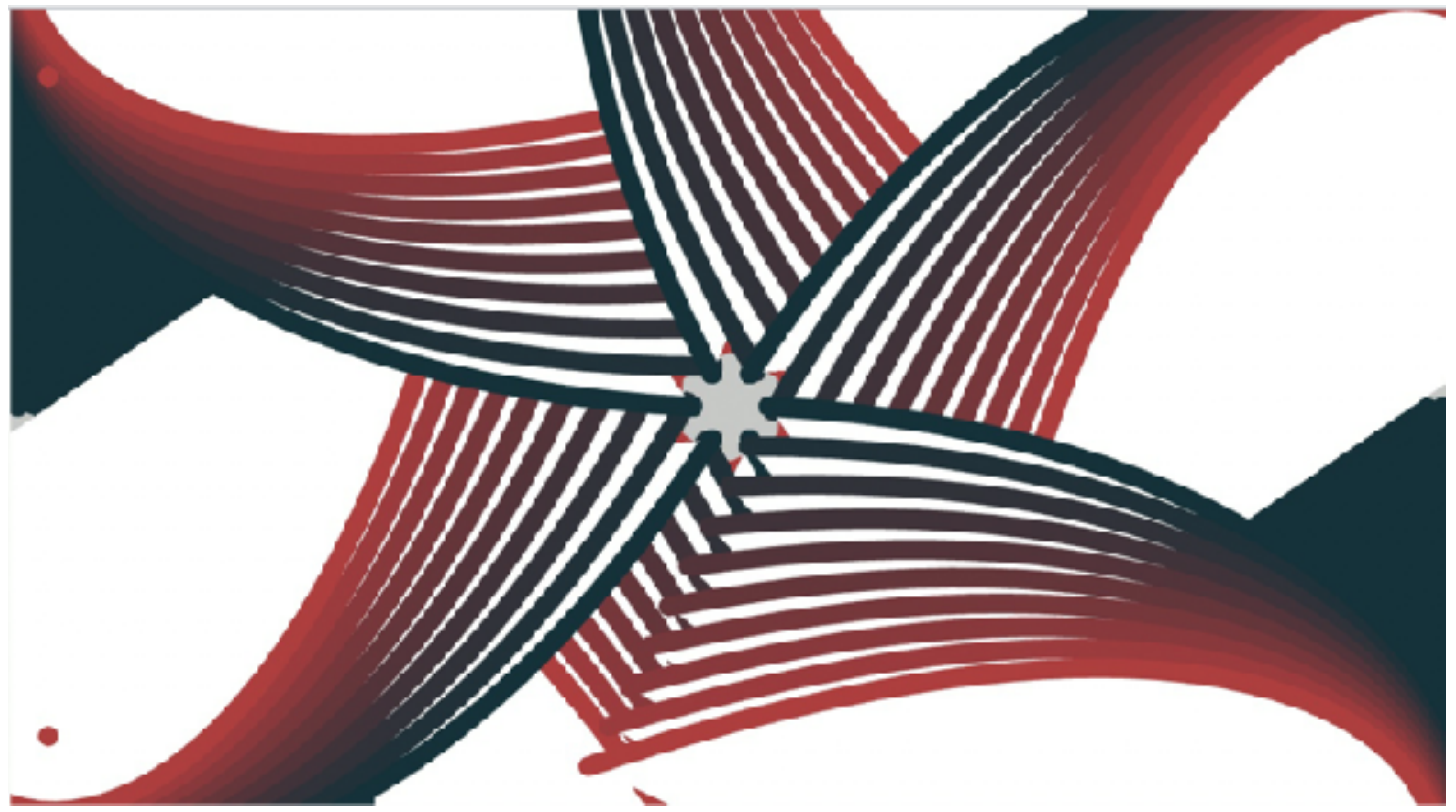
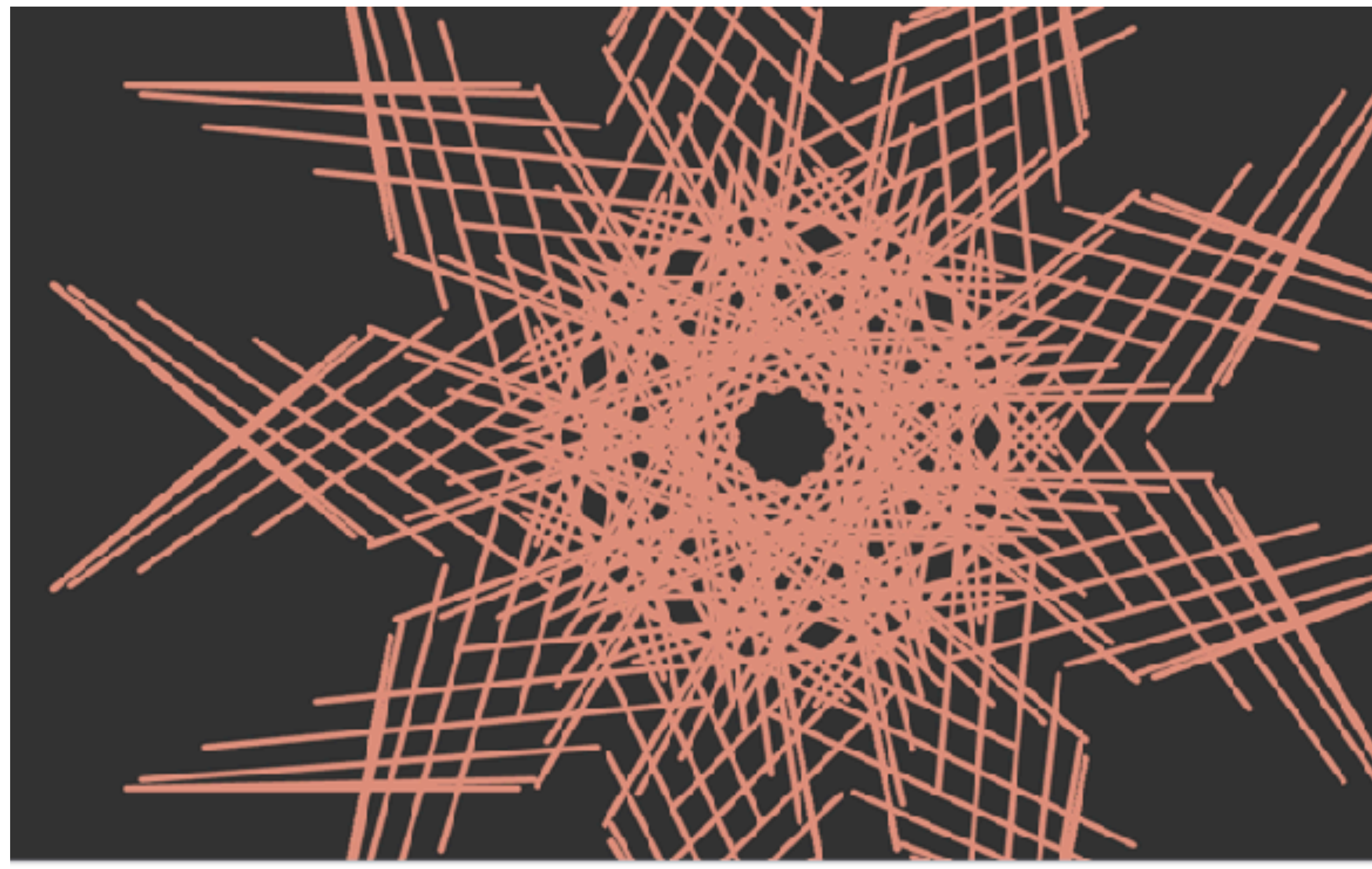
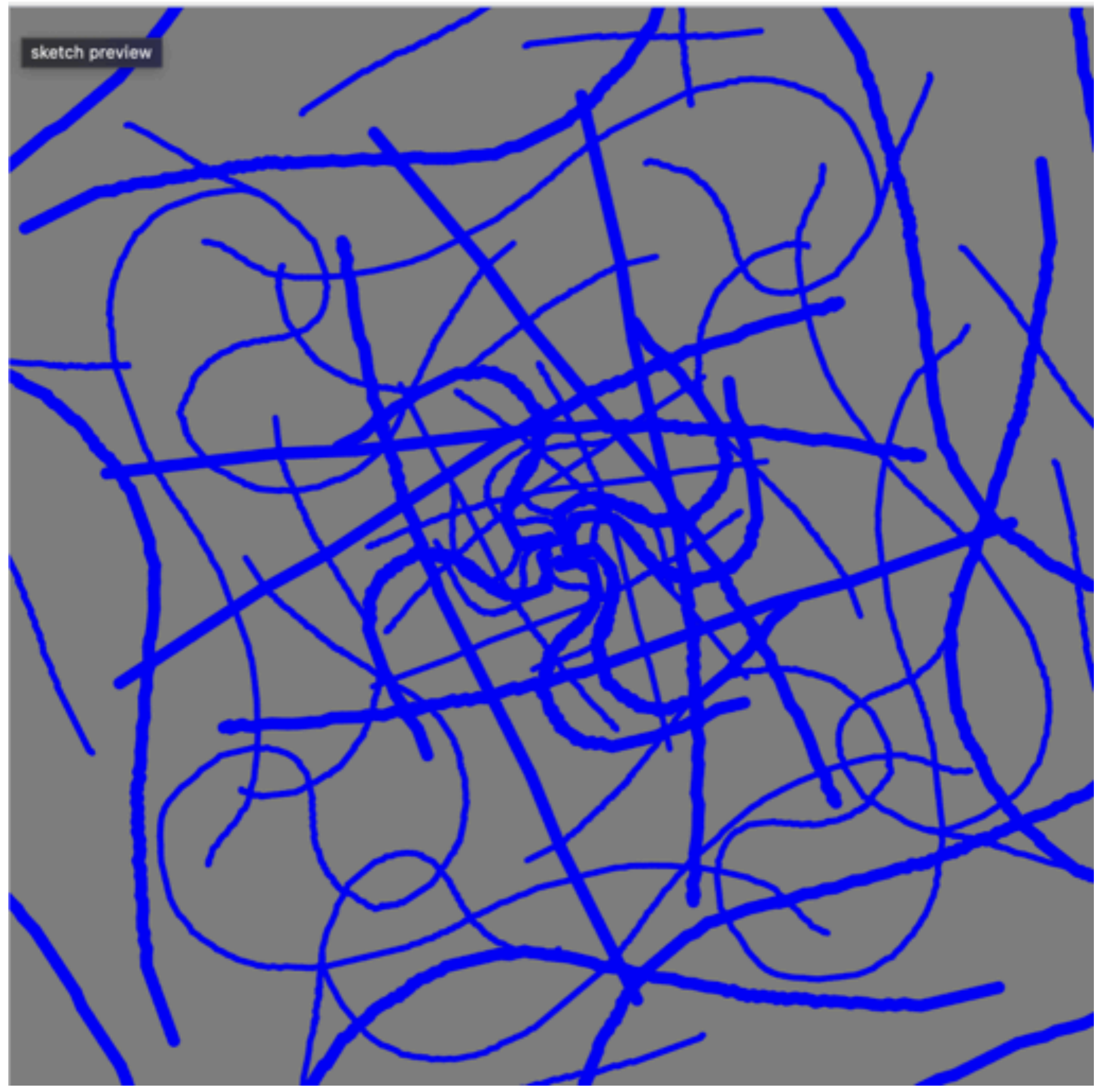
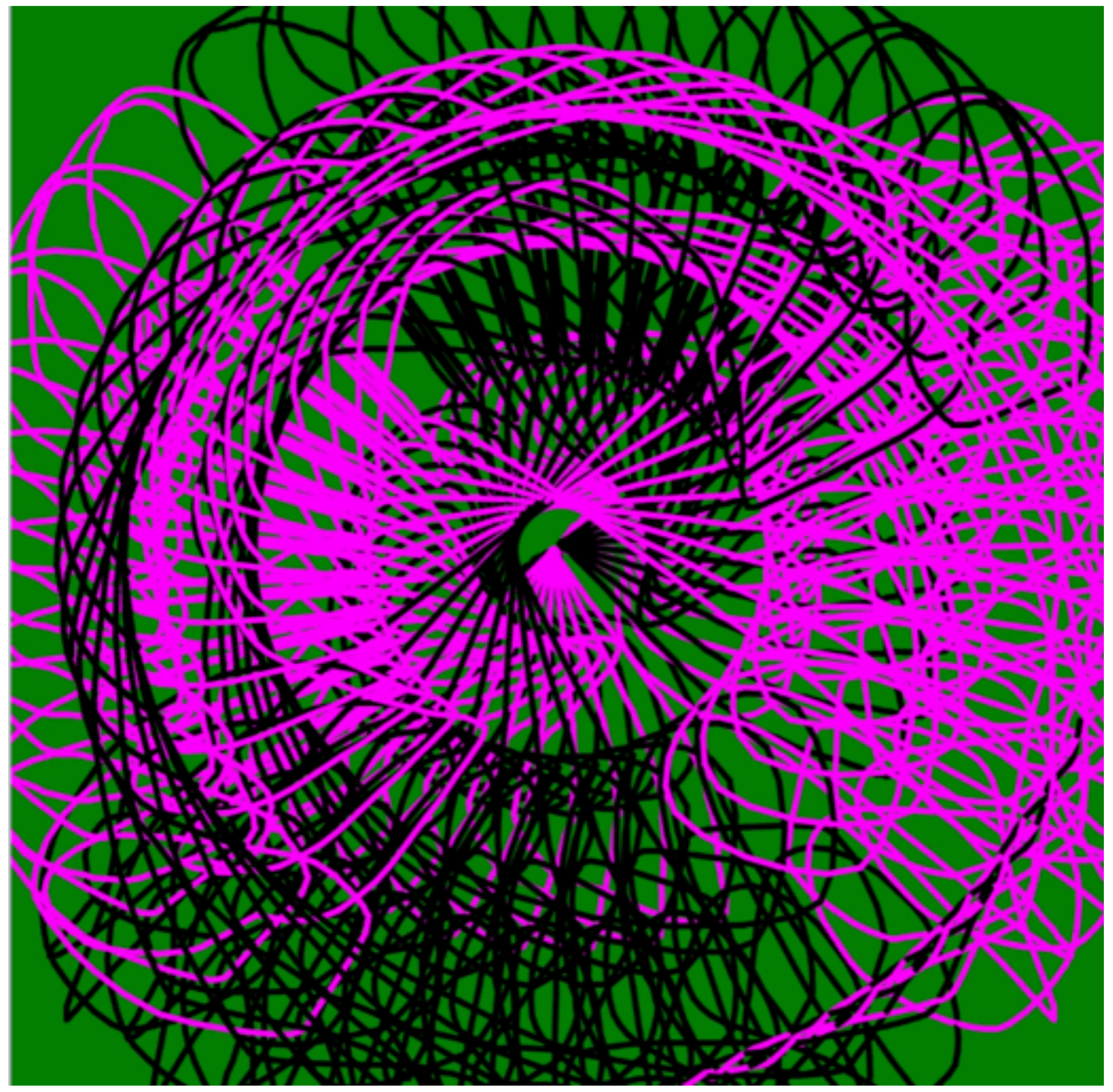
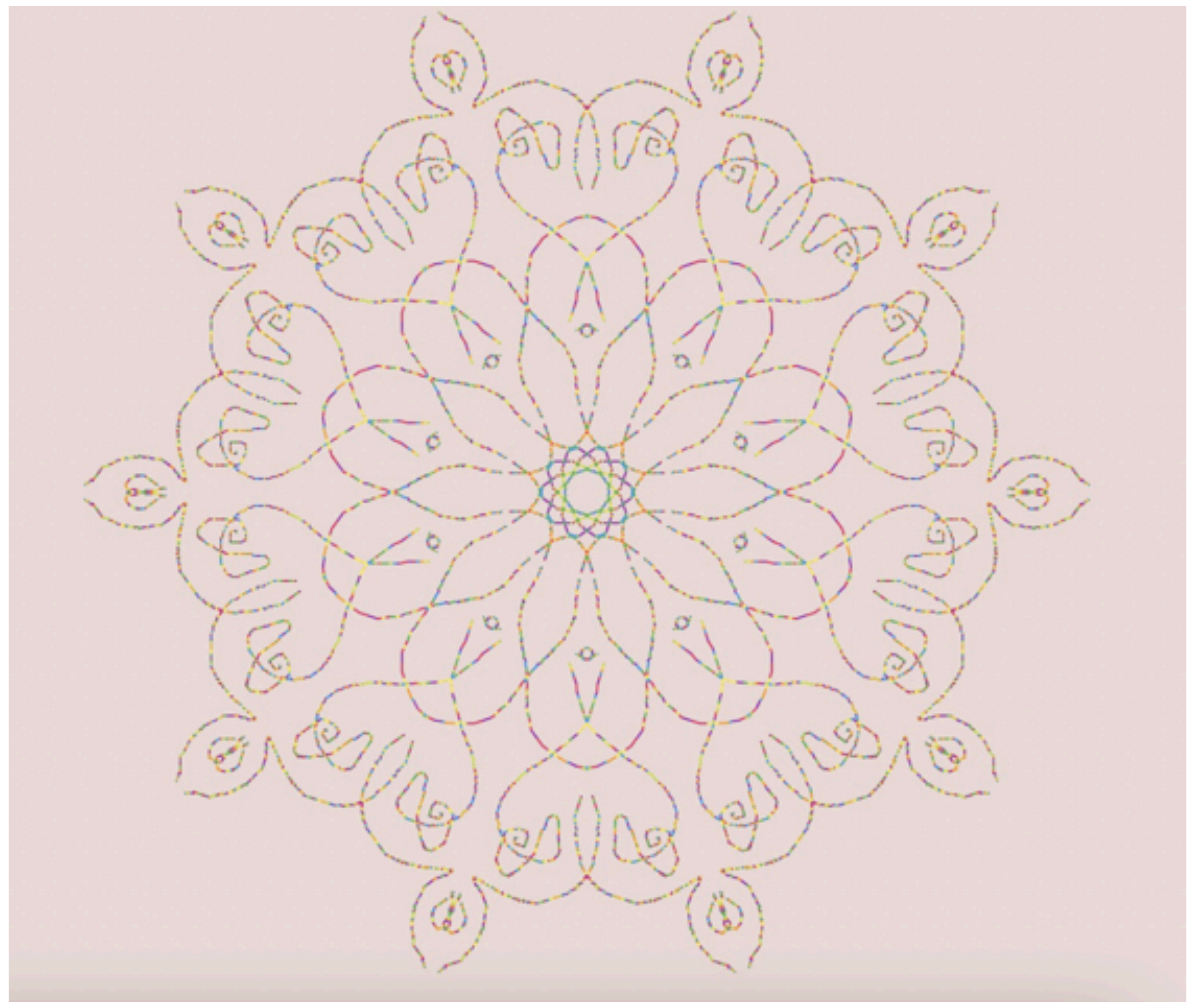
- Directly manipulate the output
- Immediately observe how actions result in changes
- Allows for open-ended exploration

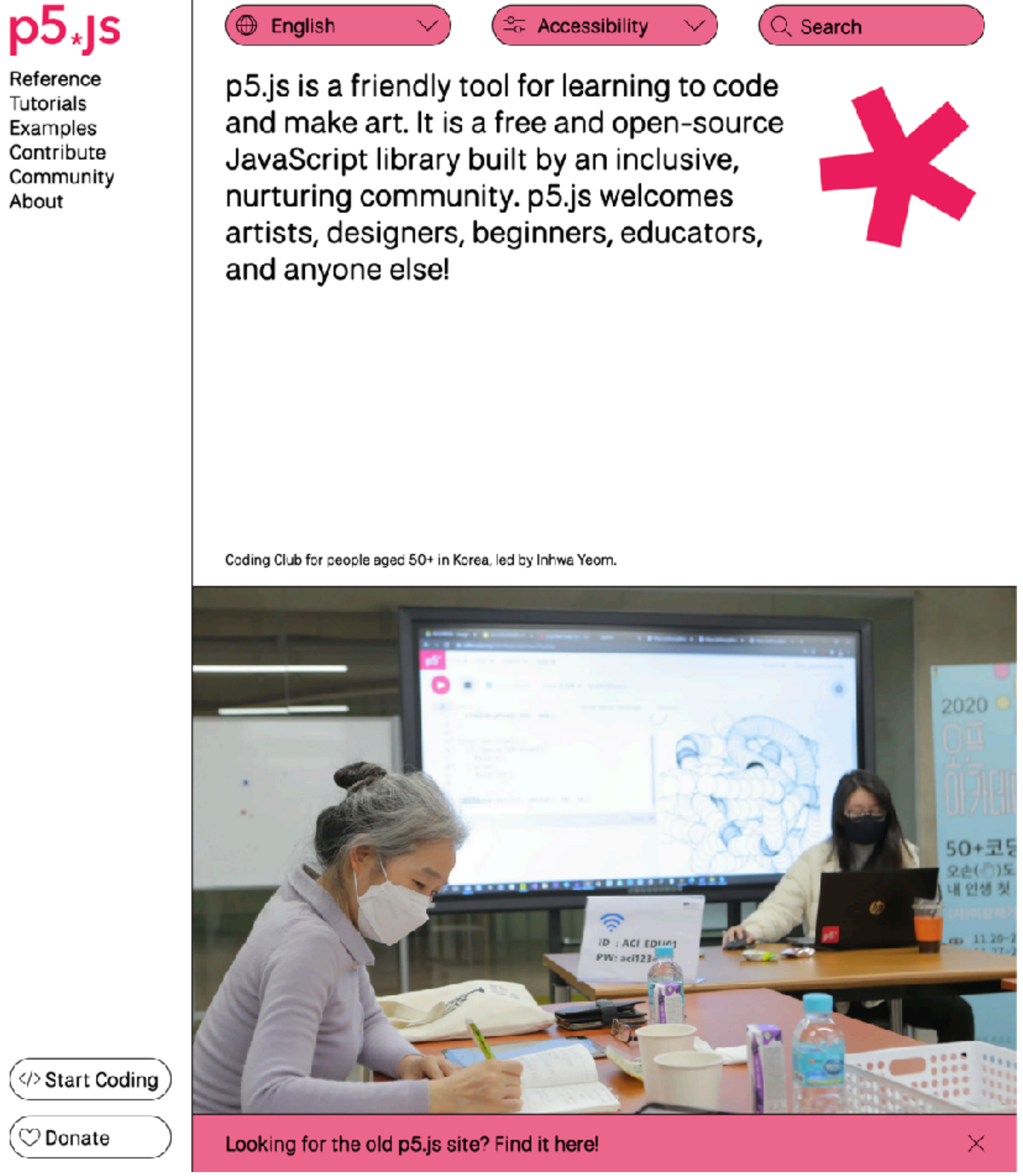
- Manipulate abstract symbols (code)
- Programming and execution are separate, unclear which pixel is caused by which line of code
- Requires more linear structure and building blocks before exploring

Your turn: p5.js studio



save
clear
Full Screen
Brush Size Slider
Reset Background Color
yellow











sketch preview



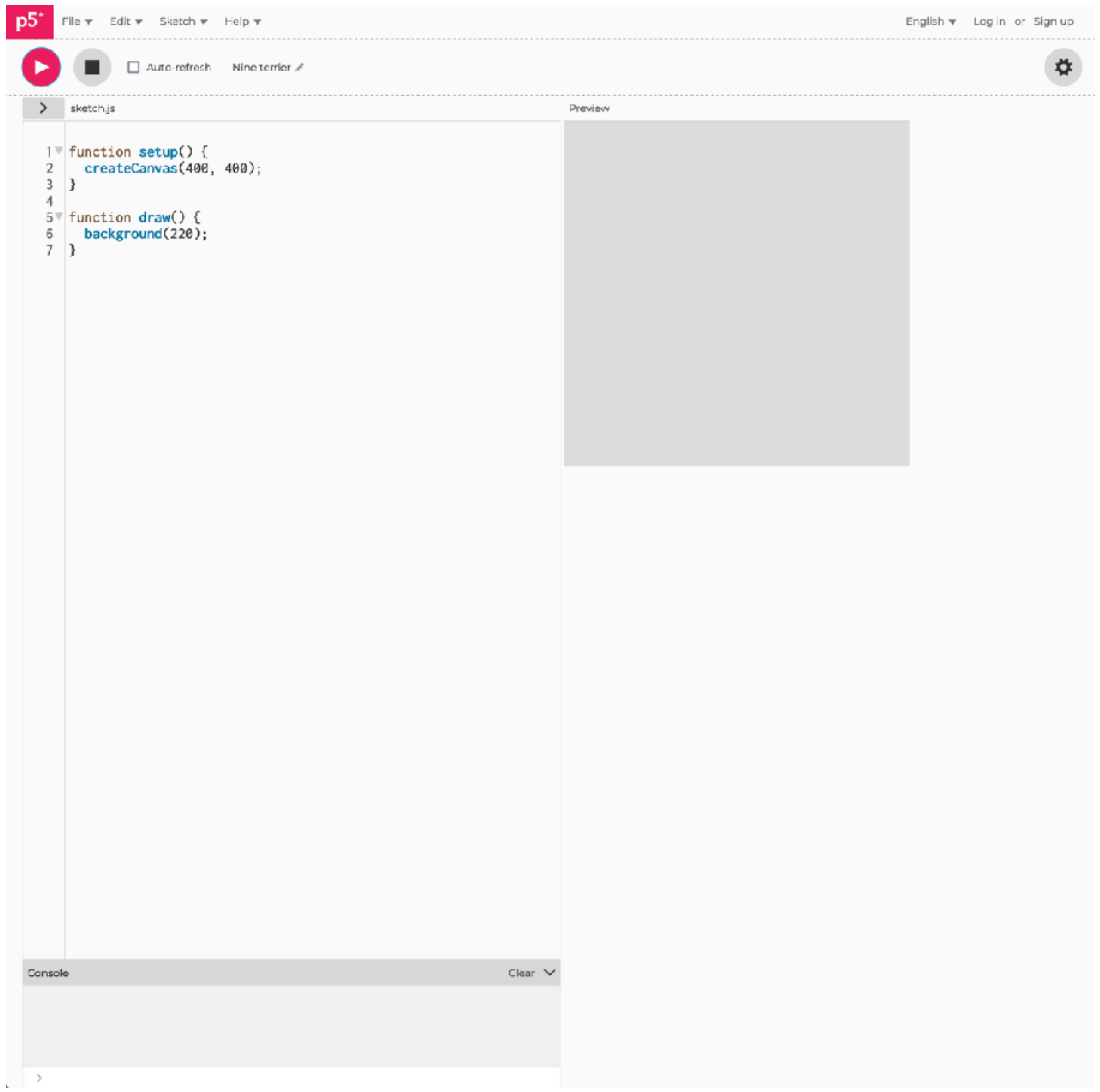


p5.js is a friendly tool for learning to code and make art. It is a free and open-source JavaScript library built by an inclusive, nurturing community. p5.js welcomes artists, designers, beginners, educators, and anyone else!

Coding Club for people aged 50+ in Korea, led by Inhwa Yeom.

Looking for the old p5.js site? Find it here!

p5js.org



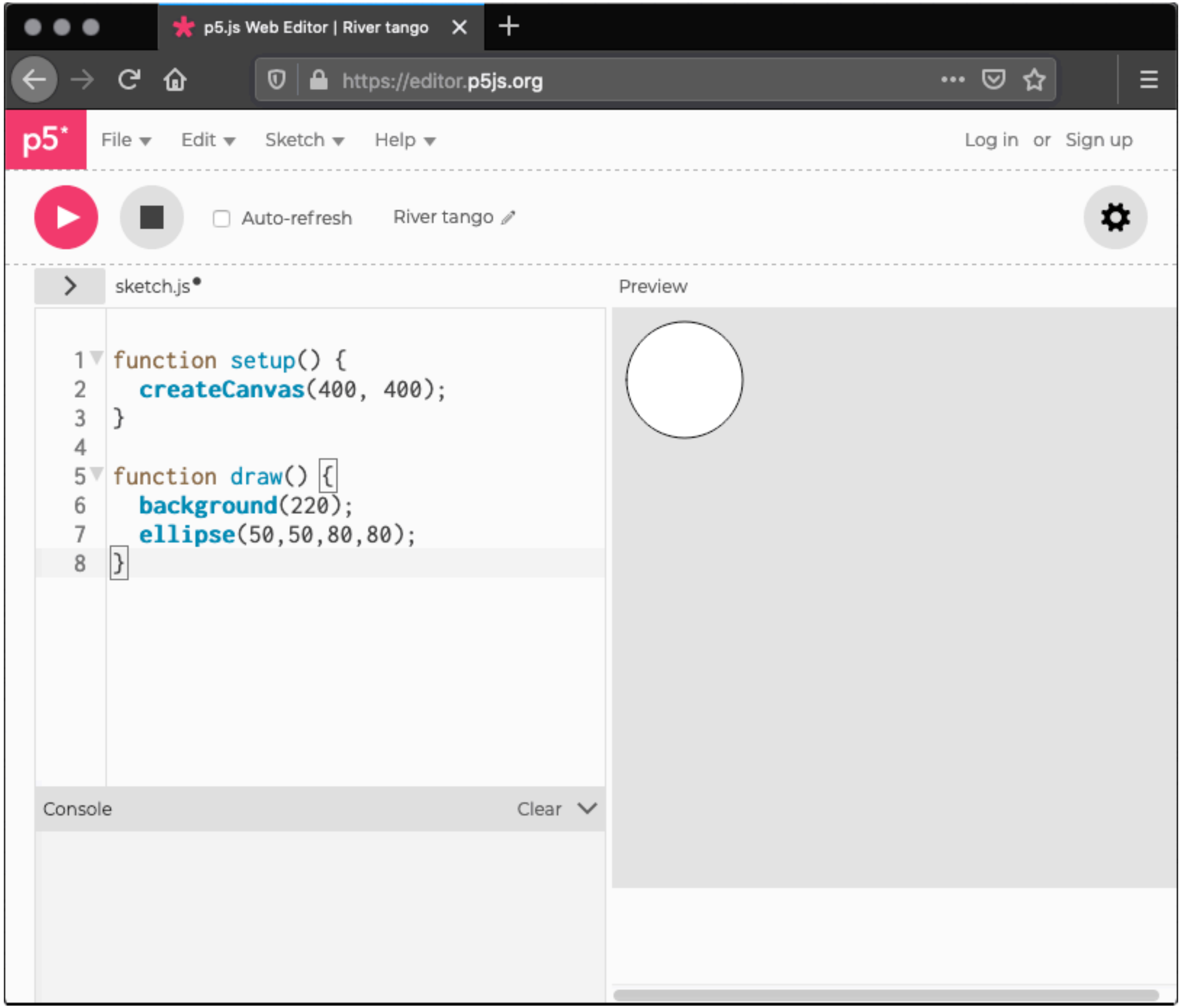
```
function setup() {
  createCanvas(400, 400);
}

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

editor.p5js.org

Javascript syntax

setup() called once at the beginning

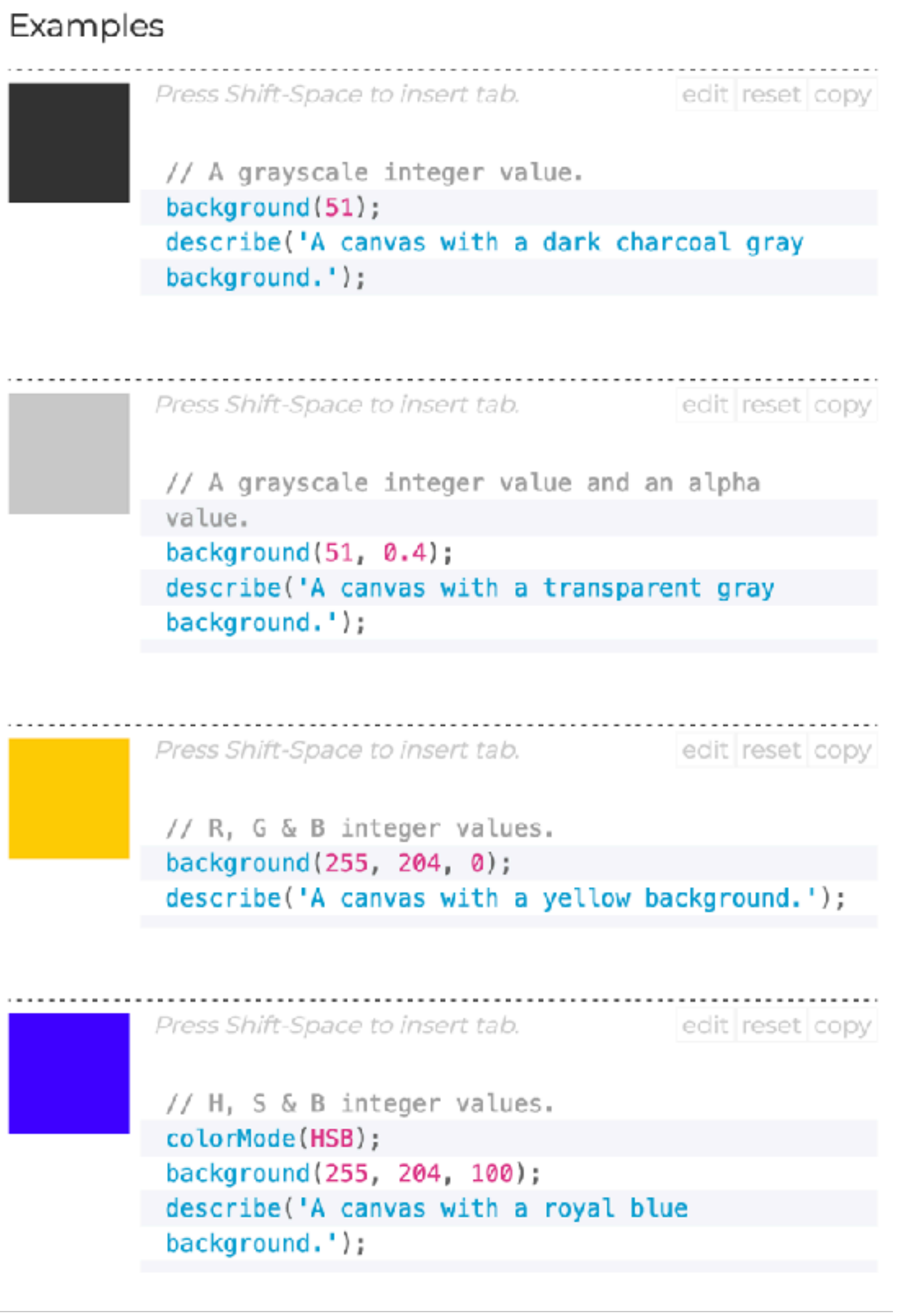
draw() called in a loop



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

function draw() {
  background(220);
  ellipse(50,50,80,80);
}
```

Useful functions

background(color)



Examples

Press Shift-Space to insert tab. edit reset copy

```
// A grayscale integer value.
background(51);
describe('A canvas with a dark charcoal gray
background.');
```

Press Shift-Space to insert tab. edit reset copy

```
// A grayscale integer value and an alpha
value.
background(51, 0.4);
describe('A canvas with a transparent gray
background.');
```

Press Shift-Space to insert tab. edit reset copy

```
// R, G & B integer values.
background(255, 204, 0);
describe('A canvas with a yellow background.');
```

Press Shift-Space to insert tab. edit reset copy

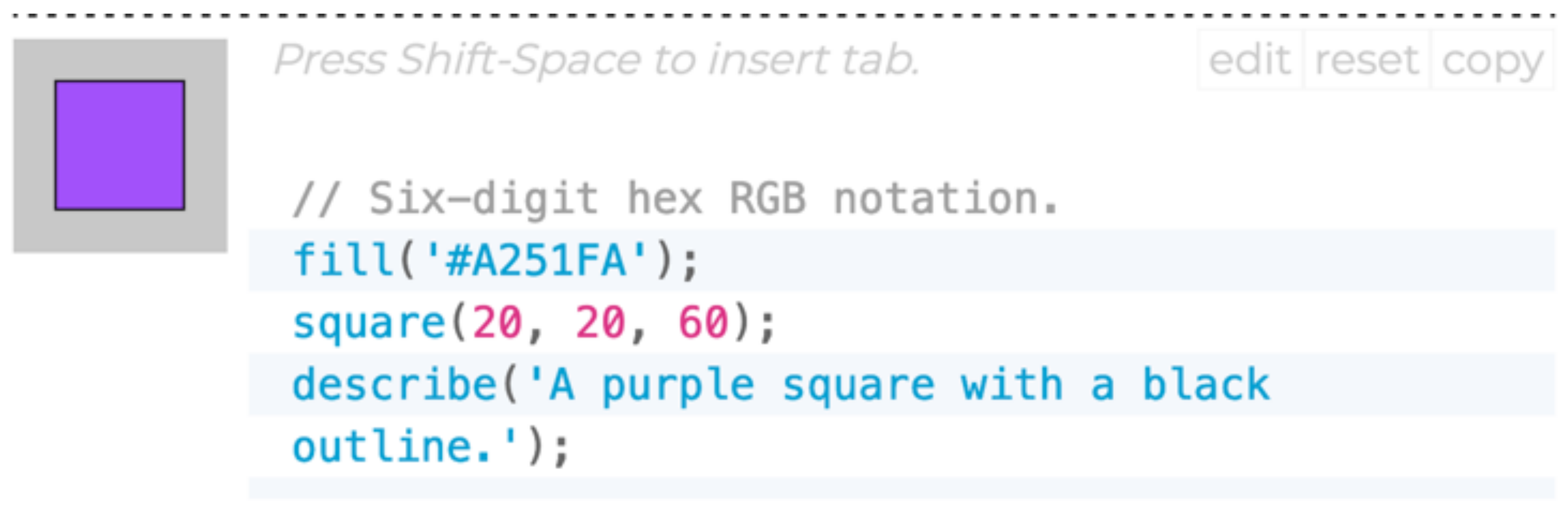
```
// H, S & B integer values.
colorMode(HSB);
background(255, 204, 100);
describe('A canvas with a royal blue
background.');
```

color can be

- 1 argument: grayscale value (0-255)
- 2 arguments: grayscale value & opacity (0-1)
- 3 arguments: (red, green, blue) (0-255)
- 3 arguments: (hue, saturation, value)
- 1 argument: hex code '#00ff00'
- 1 argument: CSS named color 'magenta'
- and more!

fill(color)

applies to all shapes after



Press Shift-Space to insert tab. edit reset copy

```
// Six-digit hex RGB notation.
fill('#A251FA');
square(20, 20, 60);
describe('A purple square with a black
outline.');
```

what kinds of shapes?

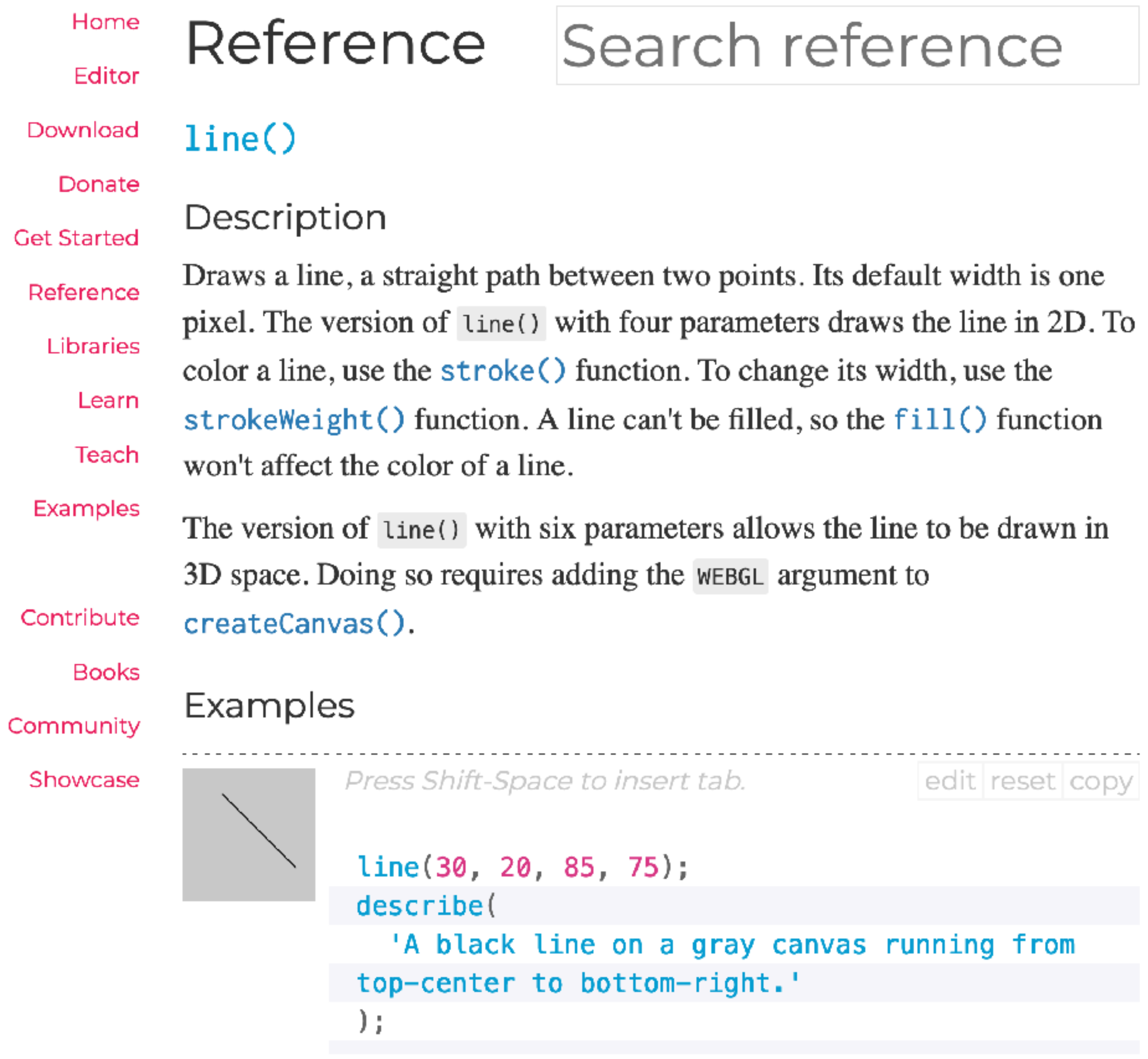
line(x1, y1, x2, y2)

ellipse(x, y, w, [h])

rect(x, y, w, [h])

...

Read the reference docs!!!



Reference

line()

Description

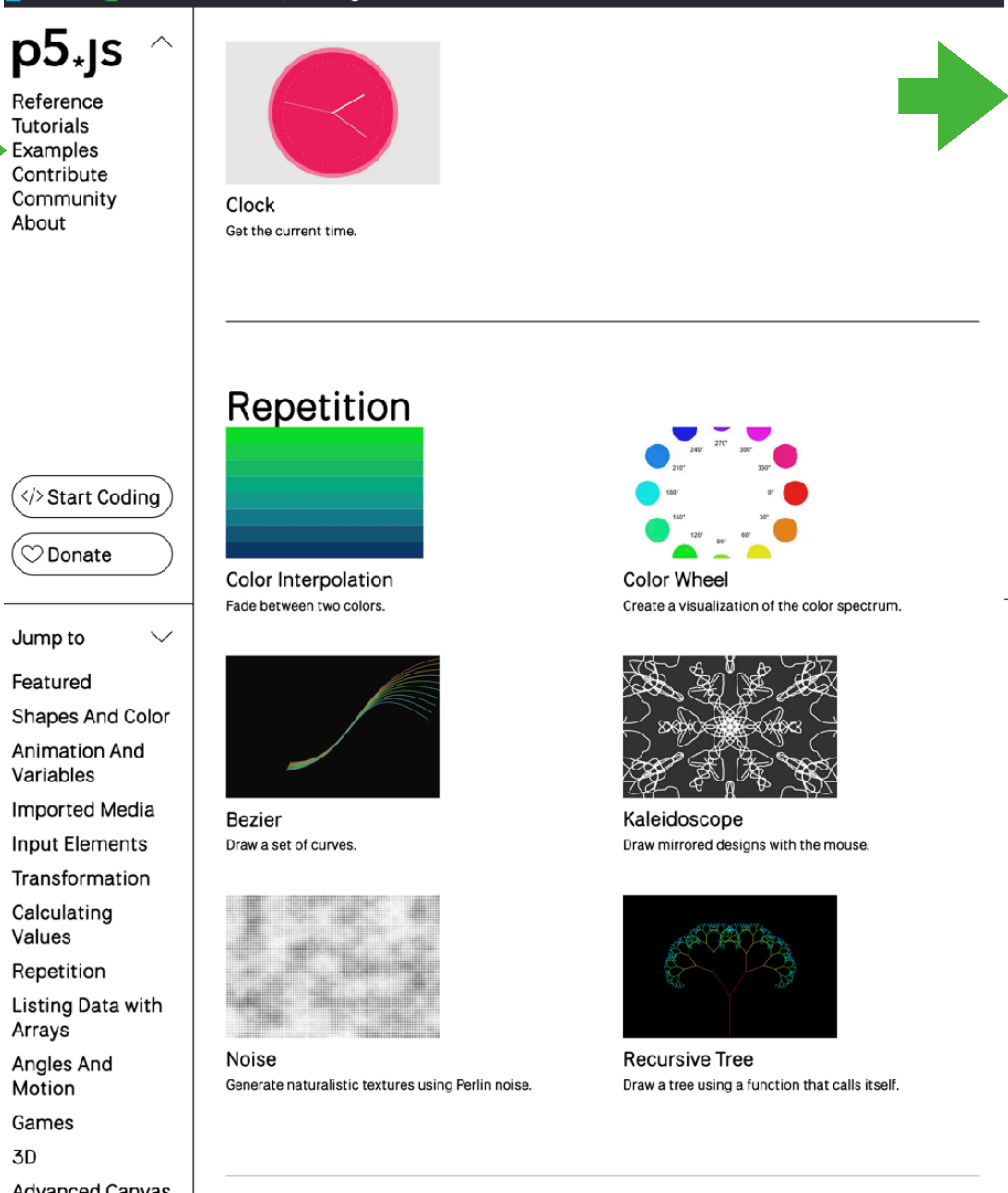
Draws a line, a straight path between two points. Its default width is one pixel. The version of `line()` with four parameters draws the line in 2D. To color a line, use the `stroke()` function. To change its width, use the `strokeWeight()` function. A line can't be filled, so the `fill()` function won't affect the color of a line.

The version of `line()` with six parameters allows the line to be drawn in 3D space. Doing so requires adding the `WEBGL` argument to `createCanvas()`.

Examples

Press Shift-Space to insert tab.

```
line(30, 20, 85, 75);
describe(
  'A black line on a gray canvas running from
top-center to bottom-right.'
);
```



p5.js

Reference
Tutorials
Examples
Contribute
Community
About

Start Coding
Donate

Jump to

Featured
Shapes And Color
Animation And Variables
Imported Media
Input Elements
Transformation
Calculating Values
Repetition
Listing Data with Arrays
Angles And Motion
Games
3D
Advanced Canvas

Clock
Get the current time.

Repetition

Color Interpolation
Fade between two colors.

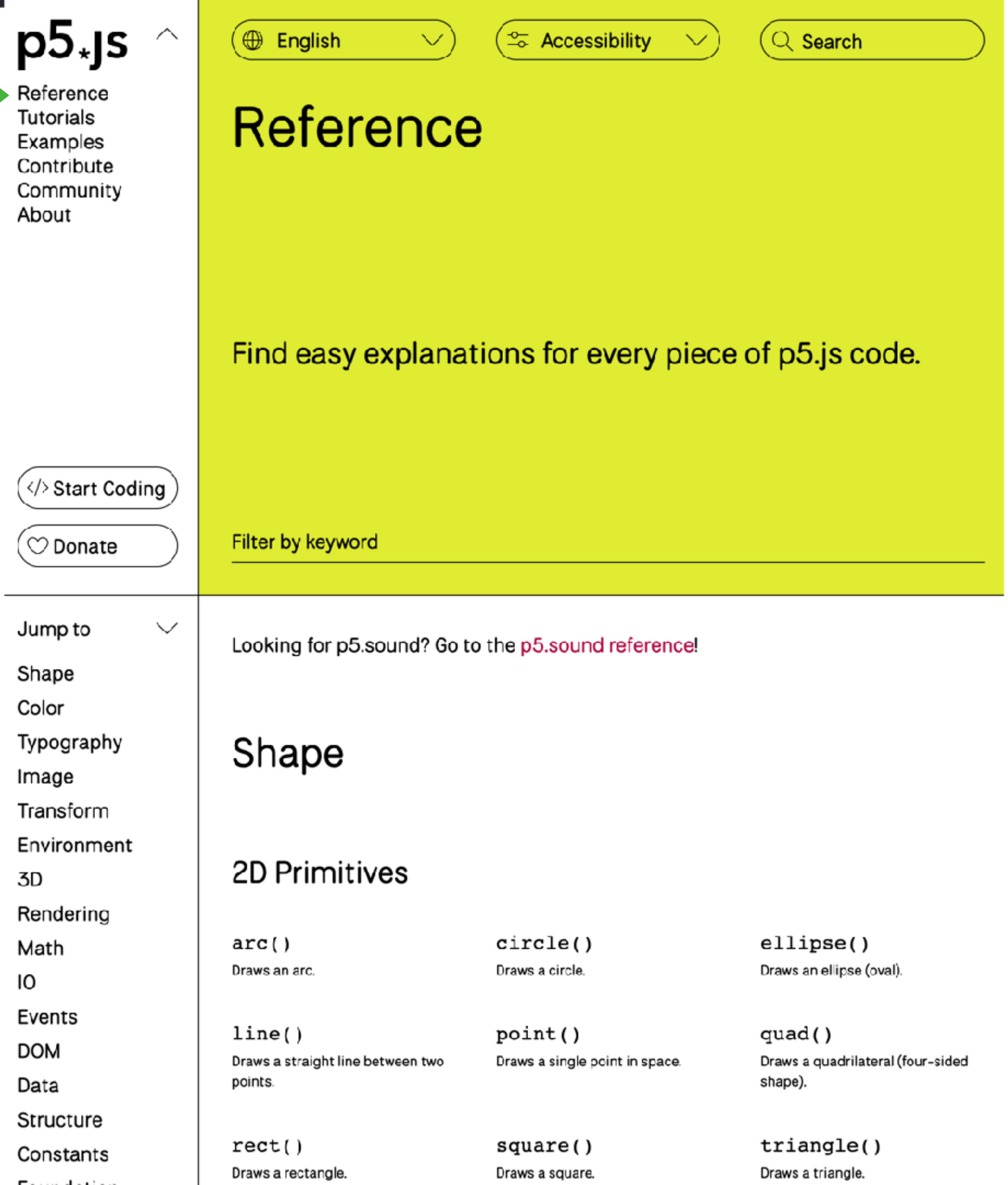
Color Wheel
Create a visualization of the color spectrum.

Bezier
Draw a set of curves.

Kaleidoscope
Draw mirrored designs with the mouse.

Noise
Generate naturalistic textures using Perlin noise.

Recursive Tree
Draw a tree using a function that calls itself.



p5.js

Reference
Tutorials
Examples
Contribute
Community
About

Start Coding
Donate

Jump to

Shape
Color
Typography
Image
Transform
Environment
3D
Rendering
Math
IO
Events
DOM
Data
Structure
Constants

English
Accessibility
Search

Reference

Find easy explanations for every piece of p5.js code.

Filter by keyword

Looking for p5.sound? Go to the p5.sound reference!

Shape

2D Primitives

`arc()`
Draws an arc.

`circle()`
Draws a circle.

`ellipse()`
Draws an ellipse (oval).

`line()`
Draws a straight line between two points.

`point()`
Draws a single point in space.

`quad()`
Draws a quadrilateral (four-sided shape).

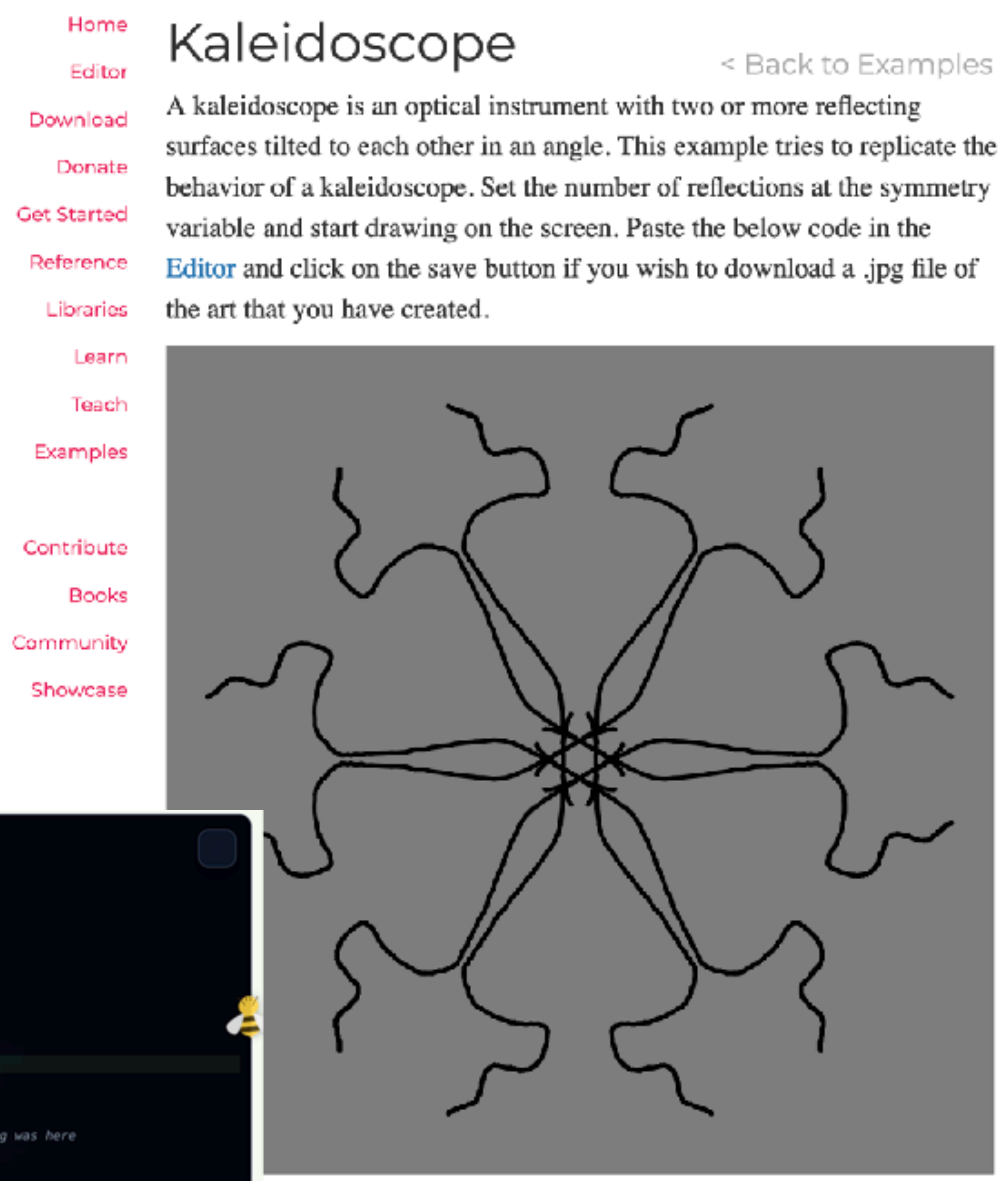
`rect()`
Draws a rectangle.

`square()`
Draws a square.

`triangle()`
Draws a triangle.

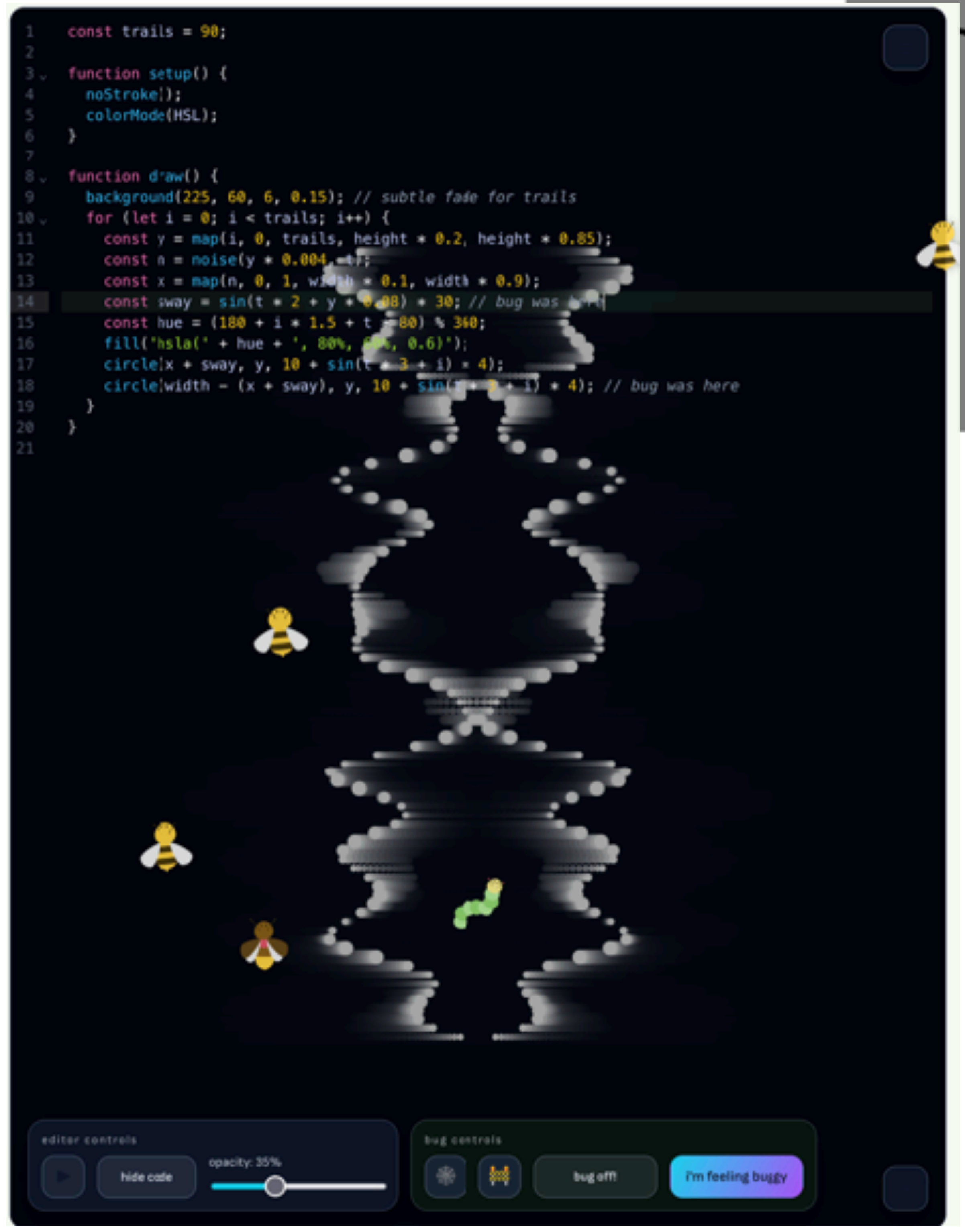
Your task: 3 way experiment

- Pair up with someone at your table and choose one of 3 conditions:
  - 1. Edit some existing p5.js code (e.g., Examples > Repetition > Kaleidoscope)
  - 2. Use Critter Canvas https://critter-canvas.pages.dev/, an AST that introduces bugs in your code
  - 3. Vibe code something via LLM



Kaleidoscope

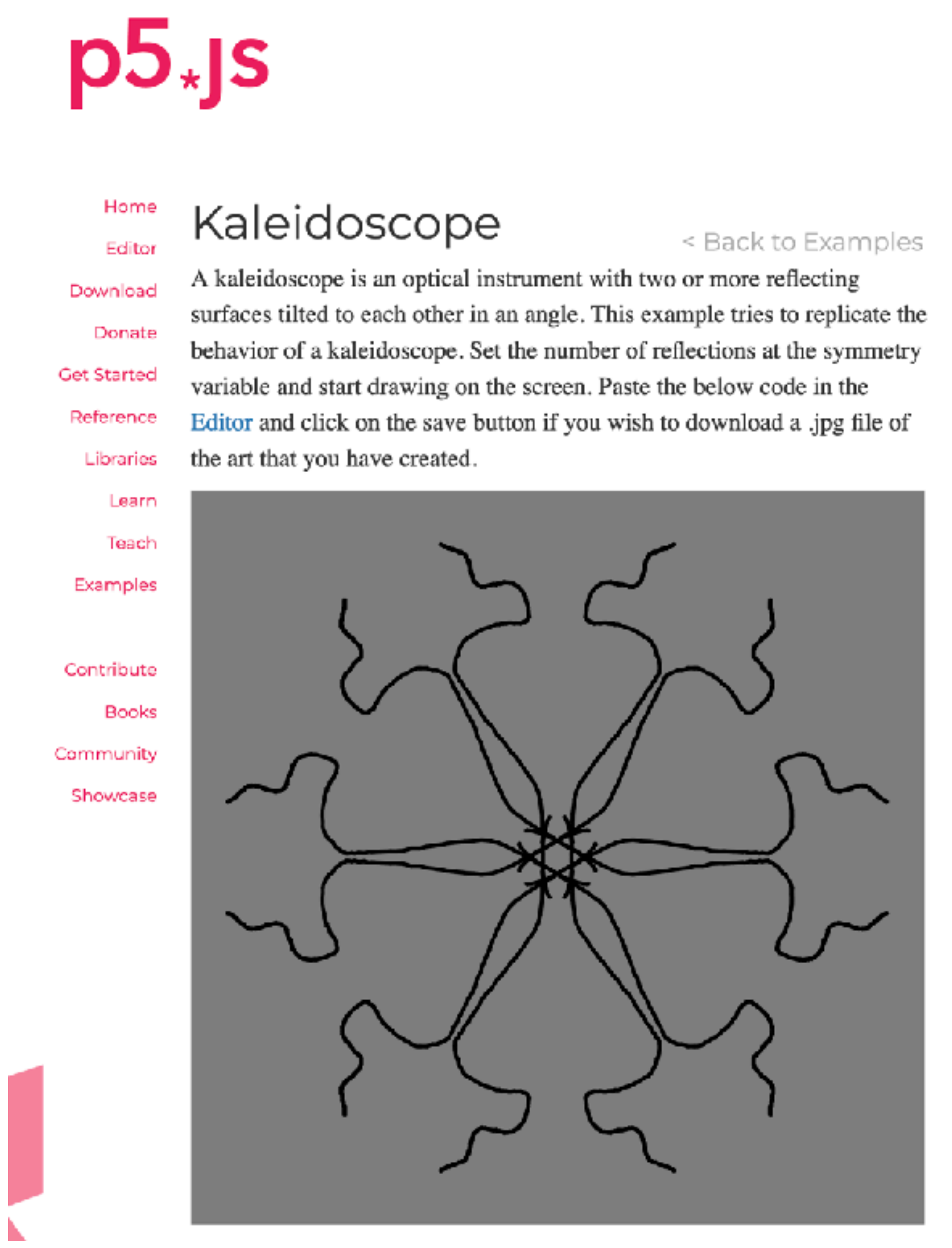
A kaleidoscope is an optical instrument with two or more reflecting surfaces tilted to each other in an angle. This example tries to replicate the behavior of a kaleidoscope. Set the number of reflections at the symmetry variable and start drawing on the screen. Paste the below code in the Editor and click on the save button if you wish to download a .jpg file of the art that you have created.



```
const trails = 90;

function setup() {
  noStroke();
  colorMode(HSL);
}

function draw() {
  background(225, 60, 6, 0.15); // subtle fade for trails
  for (let i = 0; i < trails; i++) {
    const y = map(i, 0, trails, height * 0.2, height * 0.85);
    const n = noise(y * 0.004, t);
    const x = map(n, 0, 1, width * 0.1, width * 0.9);
    const sway = sin(t * 2 + y * 0.08) * 30; // bug was here
    const hue = (180 + i * 1.5 + t * 80) % 360;
    fill('hsla(' + hue + ', 80%, 60%, 0.6)');
    circle(x + sway, y, 10 + sin(t * 3 + i) * 4);
    circle(width - (x + sway), y, 10 + sin(t * 3 + i) * 4); // bug was here
  }
}
```

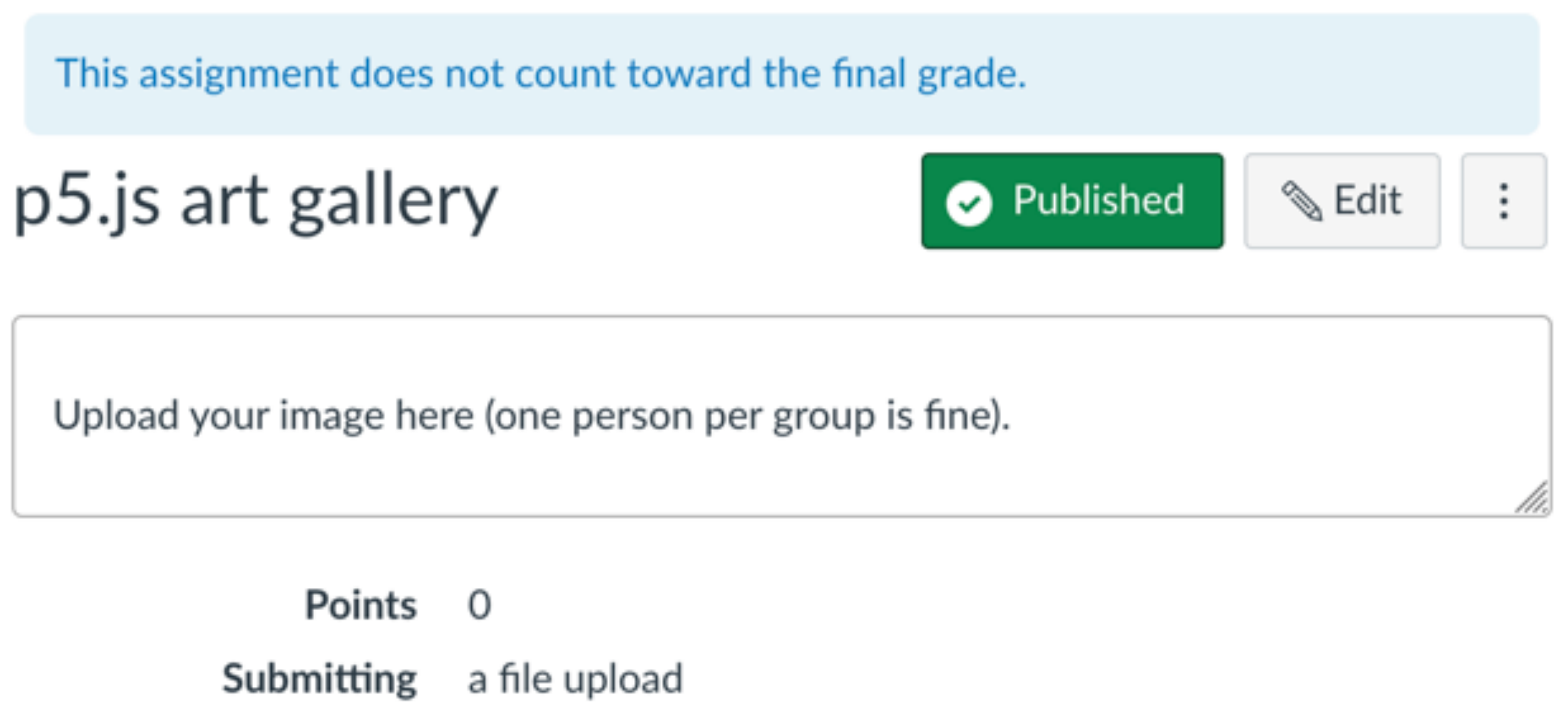
Instructions

- Learning goal: experiment with p5.js, understand creative coding process
- Make at least **3 meaningful lines of code changes** resulting in a **visually different** piece from your example/LLM generated code
- Save and **upload drawings on Canvas** (p5.js art gallery assignment, one per pair is fine)
- If you're done early, type a short reflection: How did this experience differ from other kinds of coding? What was challenging about the process of being expressive? (If applicable) how did introducing bugs or interfacing with LLMs change the experience? We'll look at reflections 12:10.



Kaleidoscope

A kaleidoscope is an optical instrument with two or more reflecting surfaces tilted to each other in an angle. This example tries to replicate the behavior of a kaleidoscope. Set the number of reflections at the symmetry variable and start drawing on the screen. Paste the below code in the Editor and click on the save button if you wish to download a .jpg file of the art that you have created.



This assignment does not count toward the final grade.

p5.js art gallery

Upload your image here (one person per group is fine).

Points 0

Submitting a file upload

Active poll

0

1. What condition did you do? 2. How did this experience differ from other kinds of coding? What was challenging about the process of being expressive? (If applicable) How did introducing bugs or using LLMs change the experience?

Join at

slido.com

#4229 125

Class 9 recap

- TODOs:
  - By EOD:
    - Recommended deadline for PM4's storyboard + making a group
  - By **Wednesday's** class:
    - ZC from Ivyer
    - RRs x 2 (remember, you can drop 4!)
    - Seminars from Leo & Bailey (design principles + epistemologies), Dualeh & Nina (evaluation of CSTs)
  - Next week
    - Monday: PM4 - 3D print for protest
    - Wednesday: **project group formation in class!**