RENDER ATL

ASMR: The Art of Scripting Media w/ Randomness

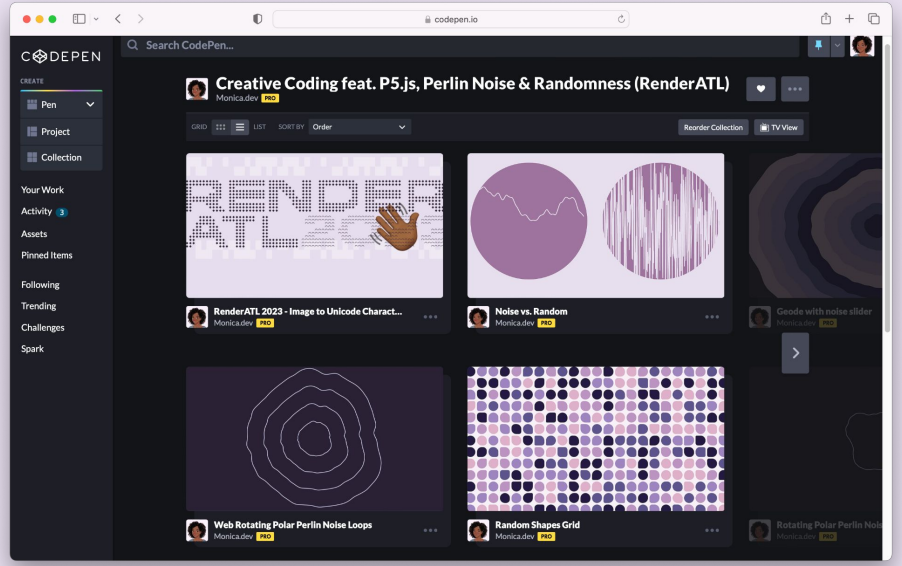
Presentation By: Monica Powell 🧕🏾

Hello, RenderATL! 👋🏾

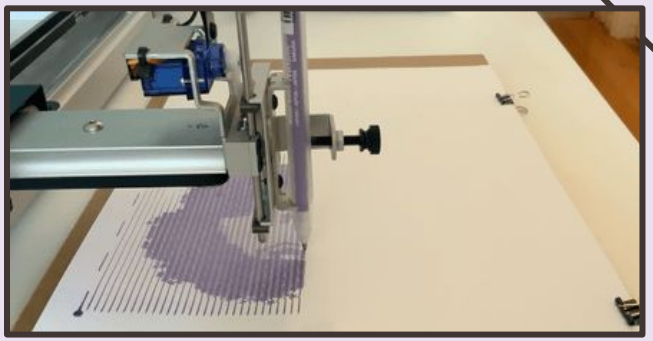
I’m Monica Powell

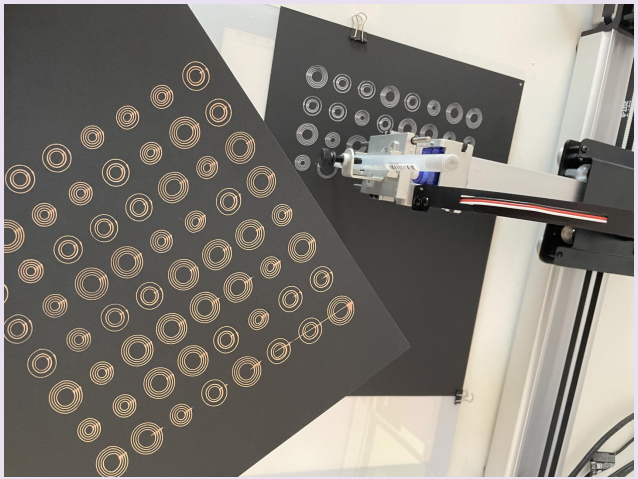
- Senior Software Engineer in EdTech at Newsela 👩🏾‍💻
- Let's chat about: Open Source and Creative Coding 👩🏾‍🎨
- React Robins Organizer (FKA React Ladies)
- GitHub Star 🌟

links.monica.dev



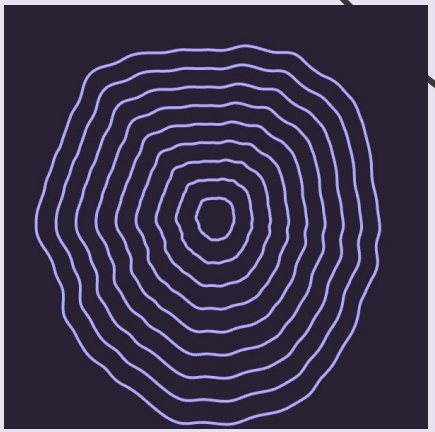
Codepen Collection

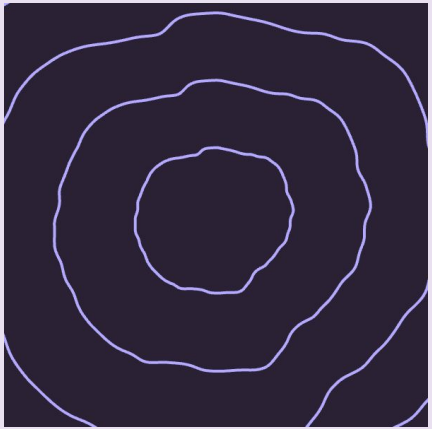




Creative Coding

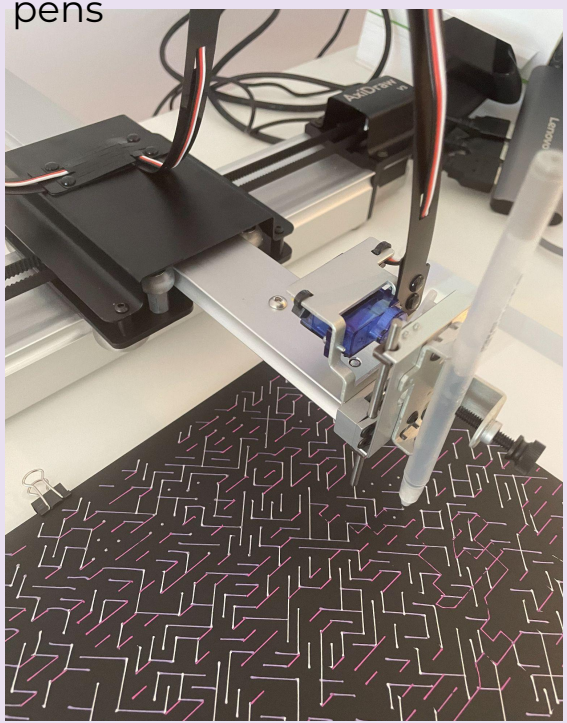
Using JavaScript (or other programming languages) to produce a variety of visual imagery that follows a predetermined set of rules or algorithms.



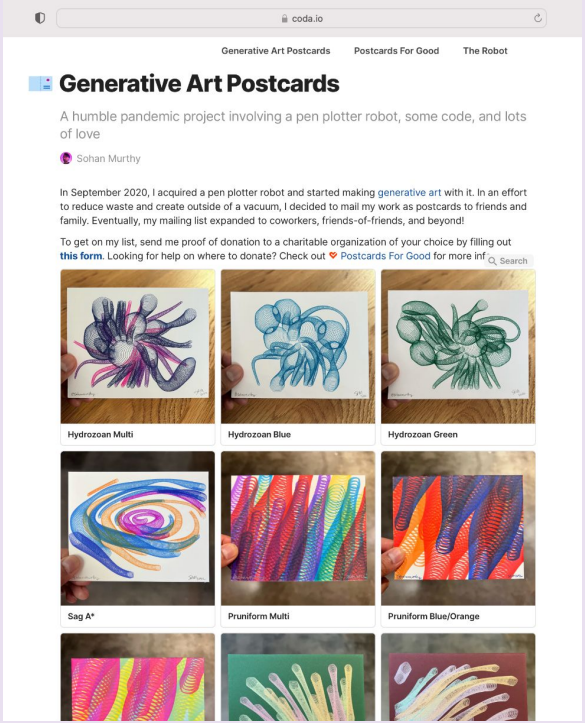


Plotter Art

me (👋🏾) plotting maze variation of 10print in multicolored gel pens



Sohan Murthy, Generative Art Postcards



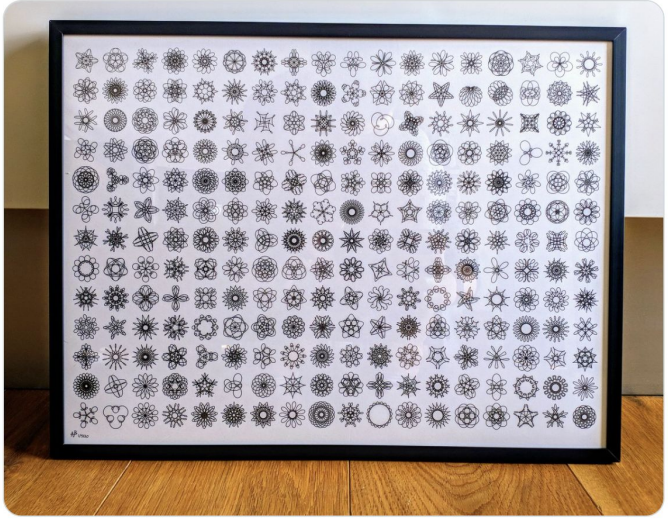
Gallery of plotter art by Adam Fuhrer

Generative Grids

Nadieh Bremer
@NadiehBremer

Framed my first pen plotter piece to hang on a wall in our house 😊😊 (the A3 tiny spirograph grid)

Having always worked in the digital realm it's a really cool feeling to see something I created come alive in the real world (on fancy paper) ^_^ #axidraw #plottertwitter



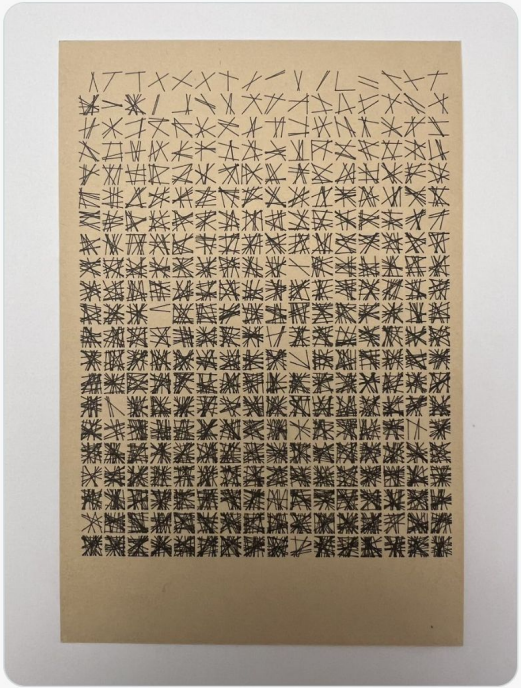
Adam genlight
@generativelight

5 years ago i spent 6 months drawing a cowboy hat every single day.

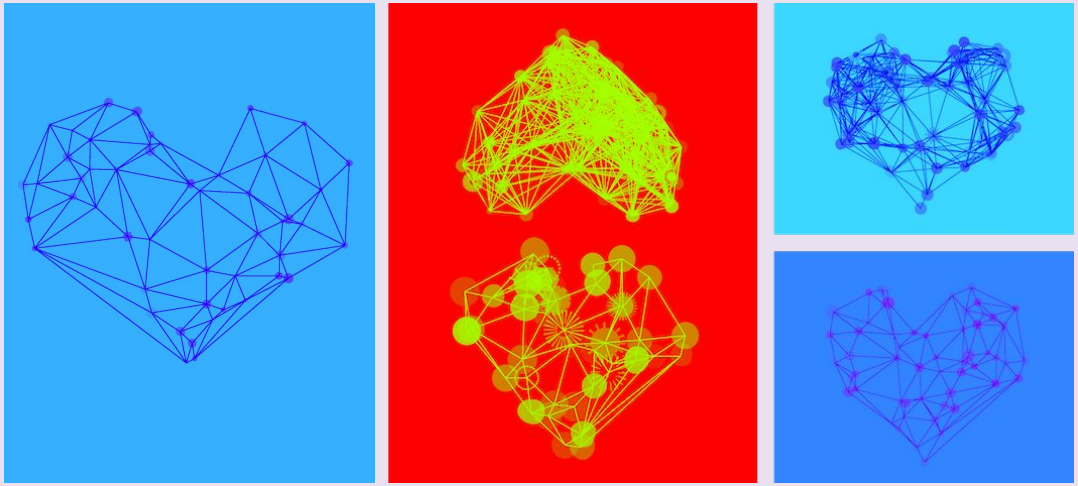
today i used p5js to make a generative art system that can instantly render 600 unique cowboy hats with every refresh.

nothing handdrawn, just plain bezier curves and lines.

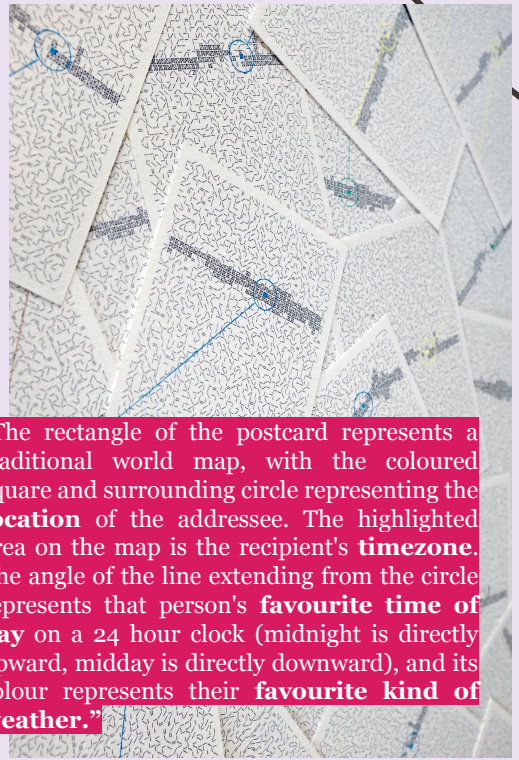
Adam Fuhrer @adamfuhrer · Apr 3
lines in a **grid** #plottertwitter



Date-Driven Generative Art



“I know how to write code. So that year I made cards utilizing data from each recipient. If they were a couple they got two hearts, if they were close to me (using zip code), etc. I made about 100 cards—each completely unique. My local digital printer hooked me up with his new HP Indigo and off they went.” - Derrick Schultz



“The rectangle of the postcard represents a traditional world map, with the coloured square and surrounding circle representing the **location** of the addressee. The highlighted area on the map is the recipient's **timezone**. The angle of the line extending from the circle represents that person's **favourite time of day** on a 24 hour clock (midnight is directly upward, midday is directly downward), and its colour represents their **favourite kind of weather.**”

Duncan Green,
Generative Art Postcards

p5.js

p5.js is a JavaScript library for **creative coding**, with a focus on making coding **accessible** and **inclusive** for artists, designers, educators, beginners, and **anyone** else!

P5.js Essential Functions

1 x

```
/* called 1x per sketch to set up, set defaults (bg color,size)*/
setup();
```

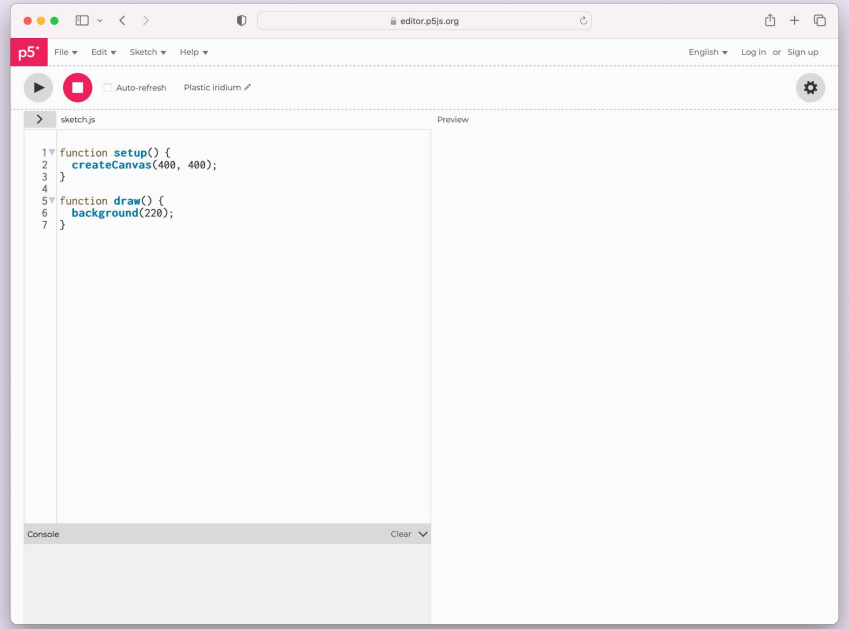
∞

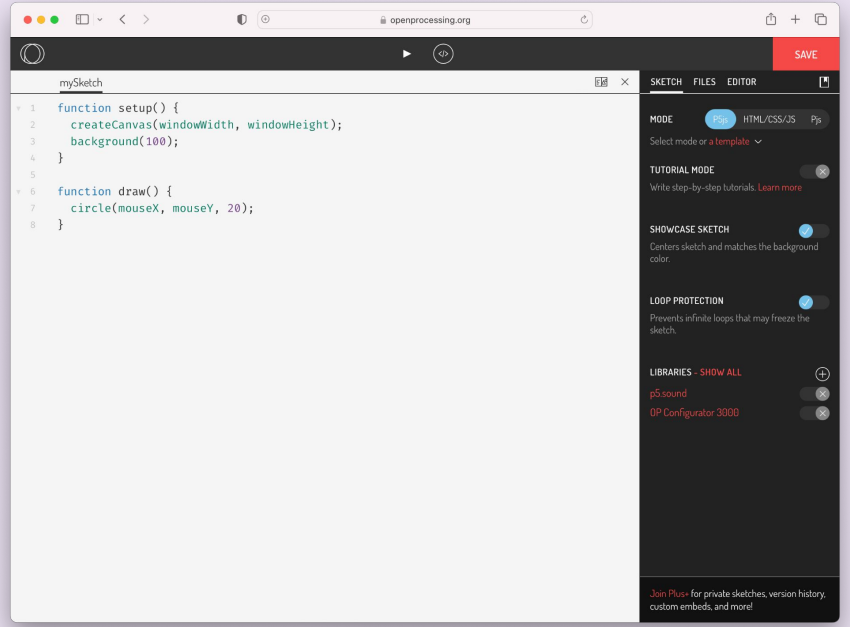
```
/* where we add new elements to sketch continuously runs */
draw();
```

Note: Beware of ∞ loops;

Default: called 60x per second

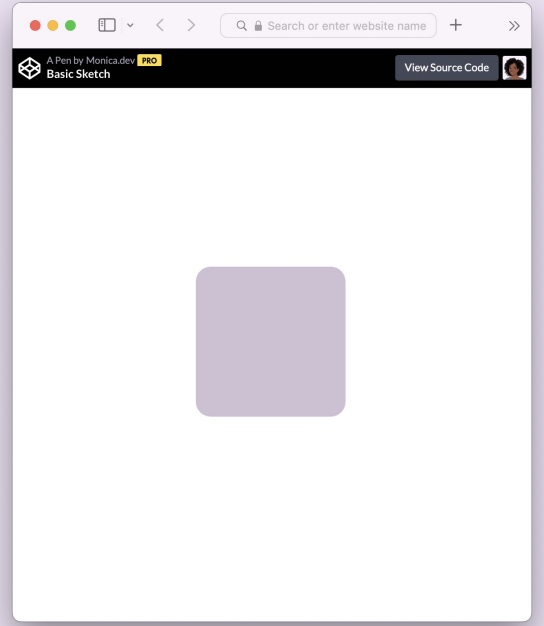
P5.js Online Editors



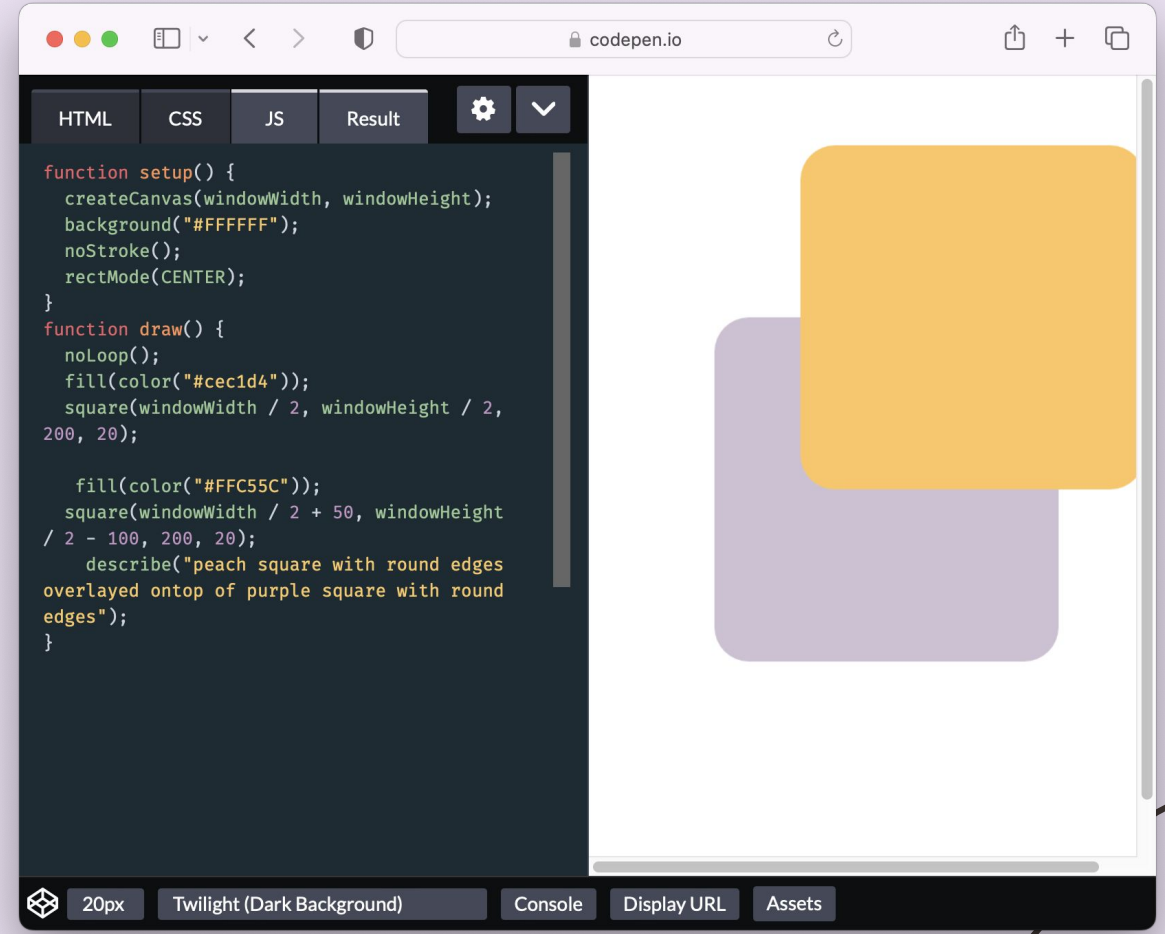


Basic Sketch

```
function setup() {
  createCanvas(windowWidth, windowHeight);
  background("#FFFFFF");
  noStroke();
  rectMode(CENTER);
}
function draw() {
  noLoop();
  fill(color("#cec1d4"));
  square(windowWidth / 2, windowHeight / 2,
200, 20);
  describe("purple square with round edges
in middle of canvas.");
}
```



Order matters when drawing in P5.js.



Unpredictable

True Random Number Generator (TRNG)

- ➔ Influenced by randomness in physical world (unknown seed value)
- ➔ Nondeterministic. Same known inputs will result in a different random sequences of numbers

Predictable

Pseudorandom Number Generator (PRNG)

- ➔ Typically computers use a PRNG algorithm to generate a series of random values
- ➔ Deterministic. Same same known inputs (including seed value) will result in the same random sequence of numbers

random.org

Home Games Numbers Lists & More Drawings Web Tools Statistics Testimonials Learn More Login

RANDOM.ORG

Search RANDOM.ORG Search

True Random Number Service

Advisory: We only operate services from the RANDOM.ORG domain. Other sites that claim to be operated by us are impostors. If in doubt, contact us.

What's this fuss about *true* randomness?

Perhaps you have wondered how predictable machines like computers can generate randomness. In reality, most random numbers used in computer programs are *pseudo-random*, which means they are generated in a predictable fashion using a mathematical formula. This is fine for many purposes, but it may not be random in the way you expect if you're used to dice rolls and lottery drawings.

RANDOM.ORG offers *true* random numbers to anyone on the Internet. The randomness comes from atmospheric noise, which for many purposes is better than the pseudo-random number algorithms typically used in computer programs. People use RANDOM.ORG for holding drawings, lotteries and sweepstakes, to drive online games, for scientific applications and for art and music. The service has existed since 1998 and was built by Dr Mads Haahr of the School of Computer Science and Statistics at Trinity College, Dublin in Ireland. Today, RANDOM.ORG is operated by Randomness and Integrity Services Ltd.

True Random Number Generator

Min: 1

Max: 100

Generate

Result:

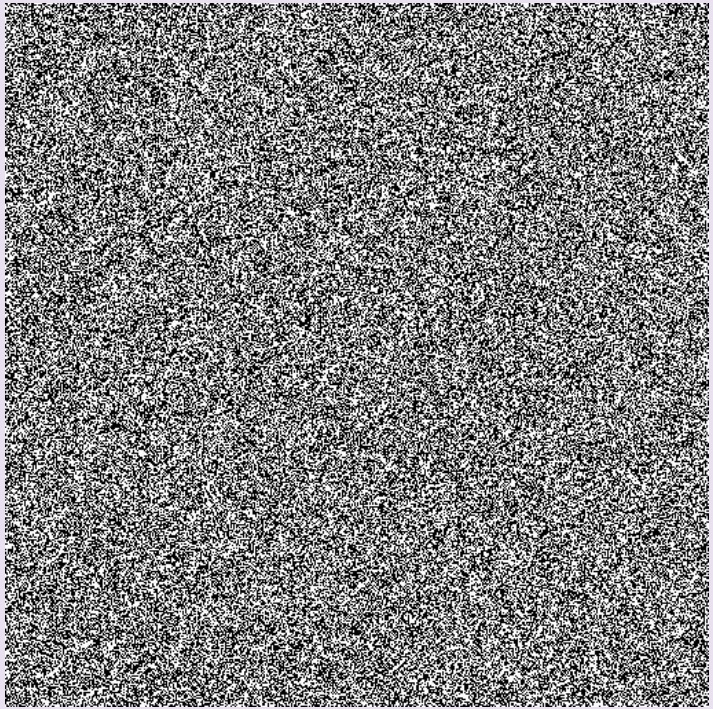
Powered by RANDOM.ORG

FREE services

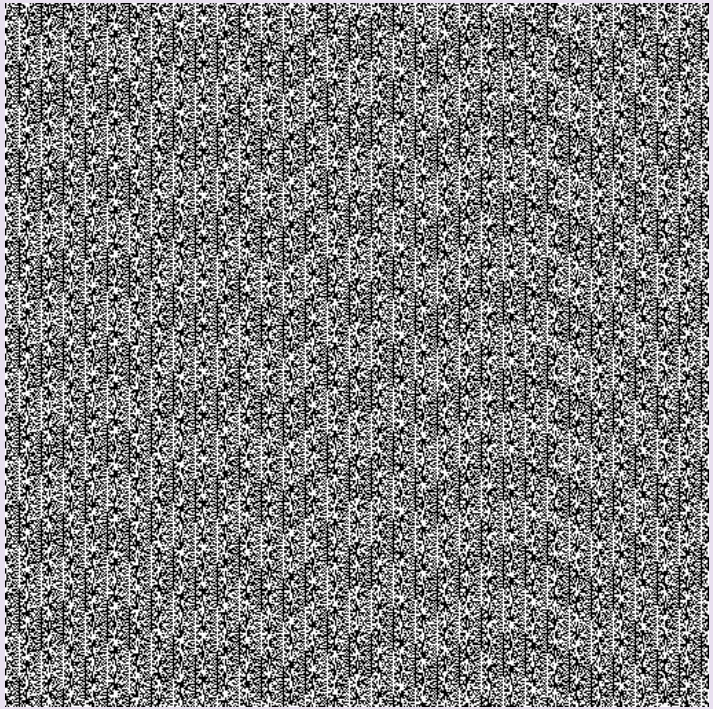
Games and Lotteries

Lottery Quick Pick is perhaps the Internet's most popular with over 280 lotteries
Keno Quick Pick for the popular game played in many countries
Coin Flipper will give you heads or tails in many currencies
Dice Roller does exactly what it says on the tin
Playing Card Shuffler will draw cards from multiple shuffled decks
Birdie Fund Generator will create birdie holes for golf courses

Random.org Bitmap Random bitmap based on atmospheric noise

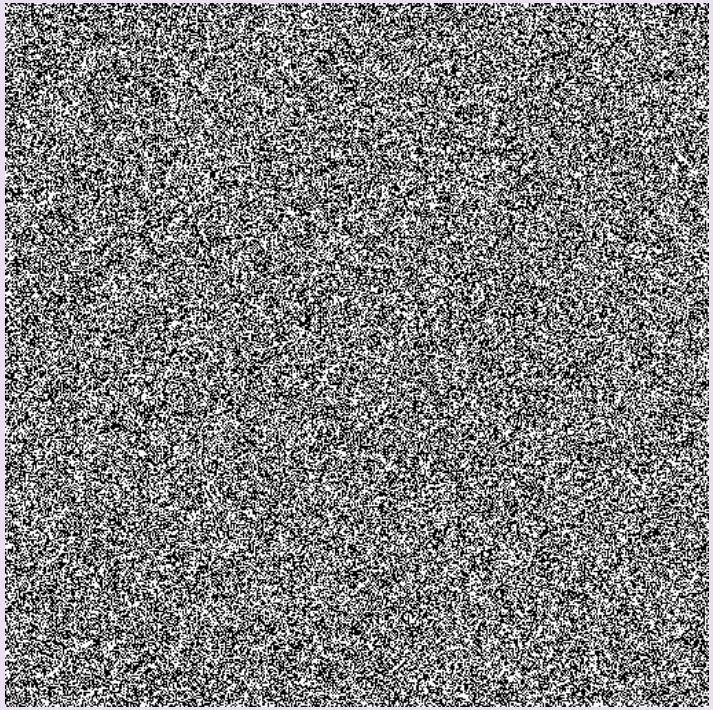


PHP rand() on Windows Bitmap



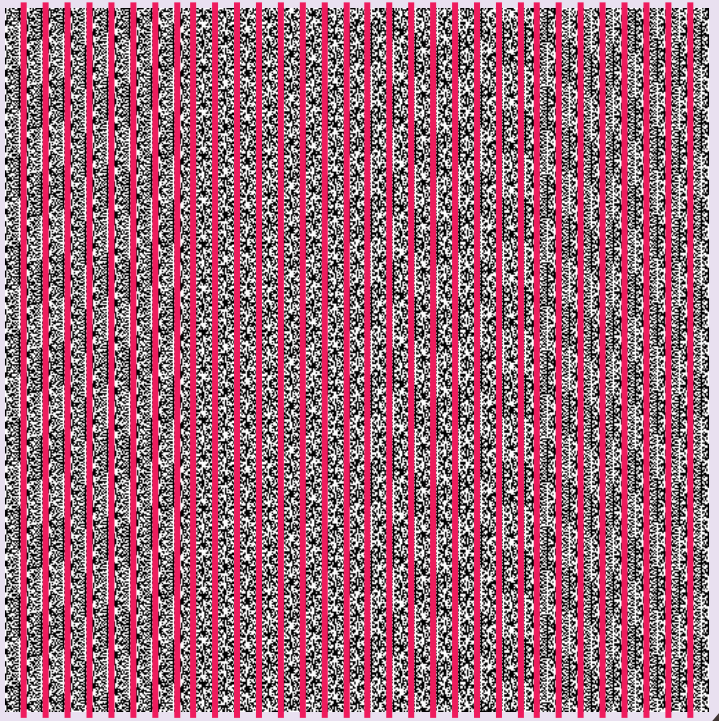
Source: https://boallen.com/random-numbers.html

Random.org Bitmap Random bitmap based on atmospheric noise



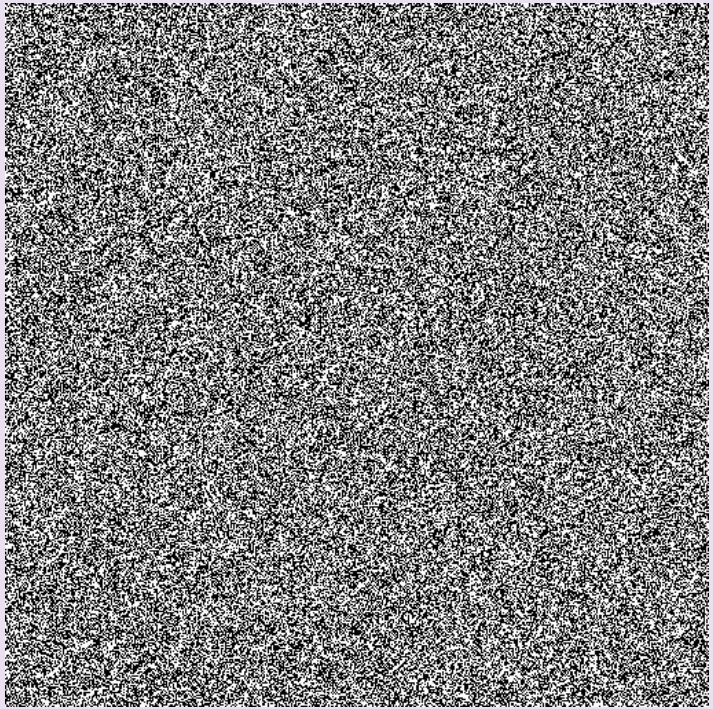
PHP rand() on Windows Bitmap

emerging pattern

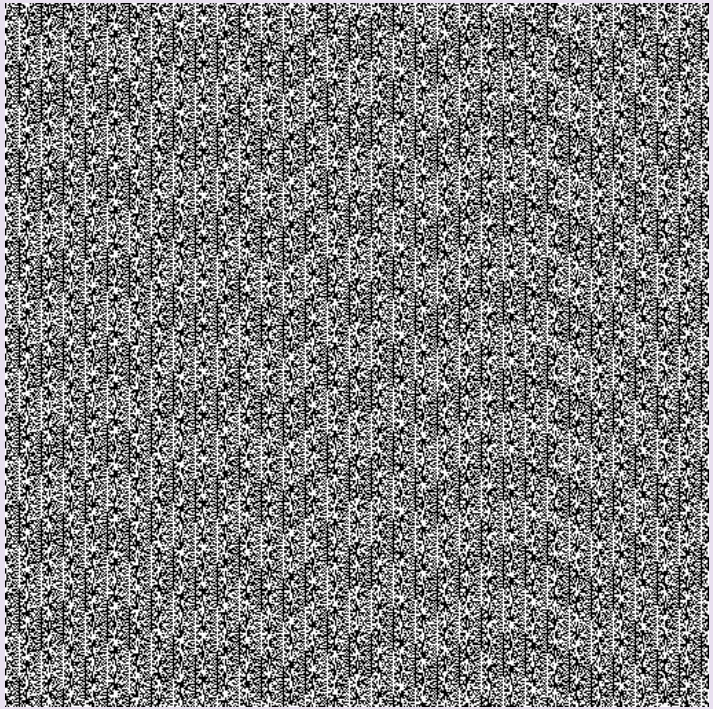


Source: https://boallen.com/random-numbers.html

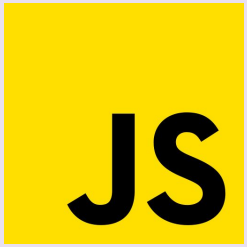
Random.org Bitmap Random bitmap based on atmospheric noise



PHP rand() on Windows Bitmap



Source: https://boallen.com/random-numbers.html



p5.js

`Math.random()`

- ➔ pseudo-random
- ➔ `number >= 0 && number < 1`
- ➔ **Initial seed *cannot* be set by user**

`random(max)`

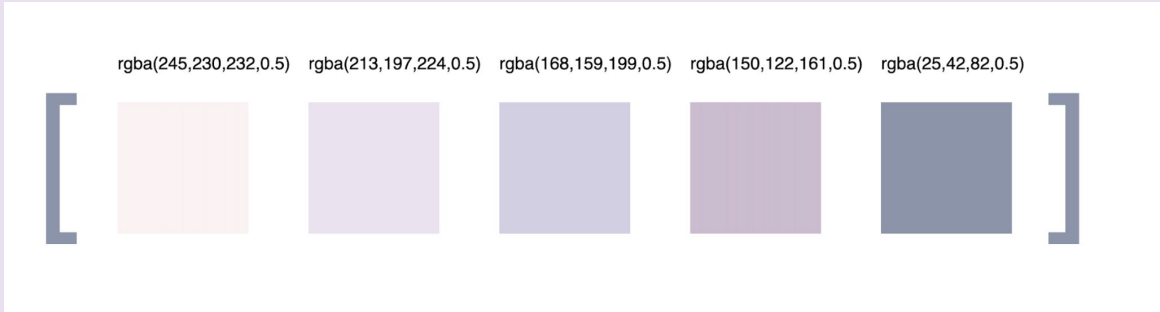
- ➔ Pseudo-random
- ➔ `number >= 0 && number < max`
- ➔ **Initial seed *can* be set by user**

Choosing Random Colors

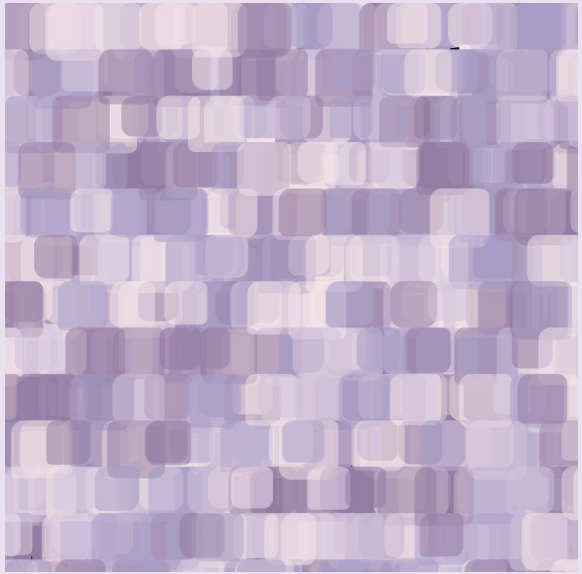
THIS OR THAT

Array of Colors

```
const colors = [
    color(352, 6, 96, opacity),
    color(275, 12, 88, opacity),
    color(254, 20, 78, opacity),
    color(283, 24, 63, opacity),
    color(222, 69, 32, opacity)
]
```

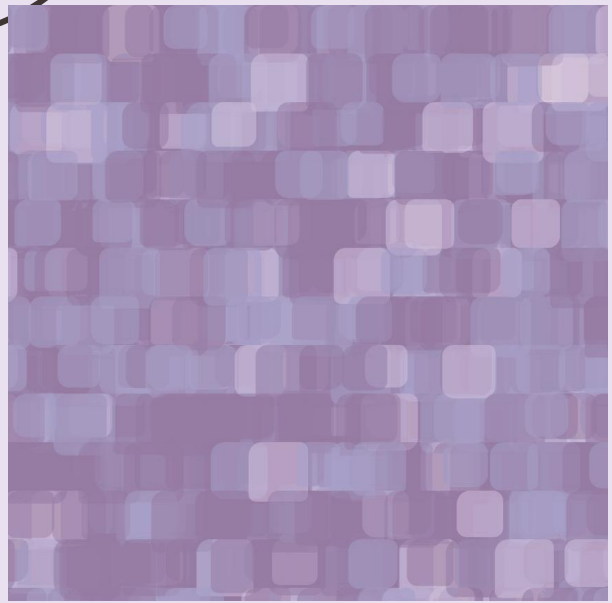


Choosing Random Color for Each Square

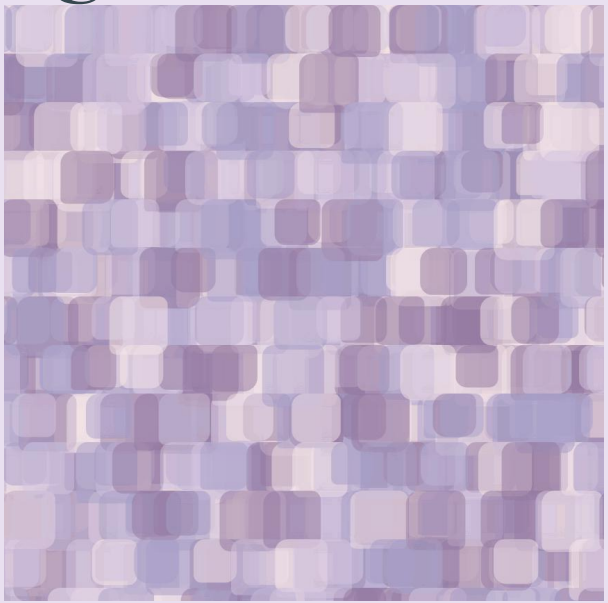


```
function draw() {
  let height = 60;
  let width = height;

  for (let row = 0; row < windowWidth / width; row++) {
    let x = row * 70;
    for (let column = 0; column < windowHeight / height;
column++) {
      let y = column * 70;
      fill(random(colors));
      square(x,y);
    }
  }
}
```

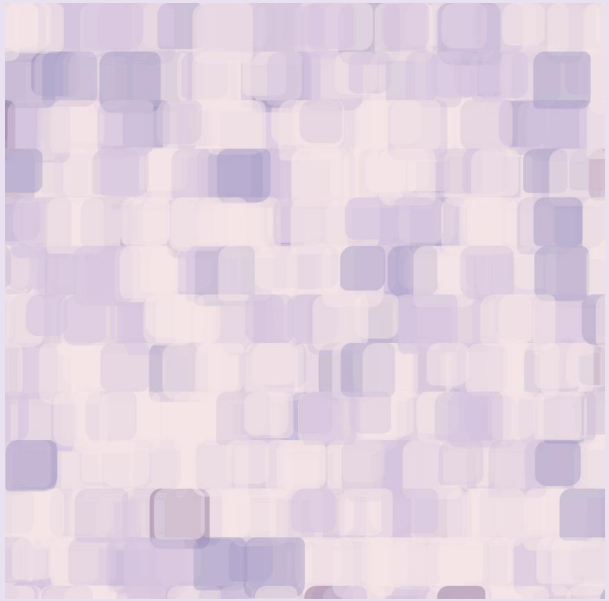
Weighted Random



➔ Random values skewed darker



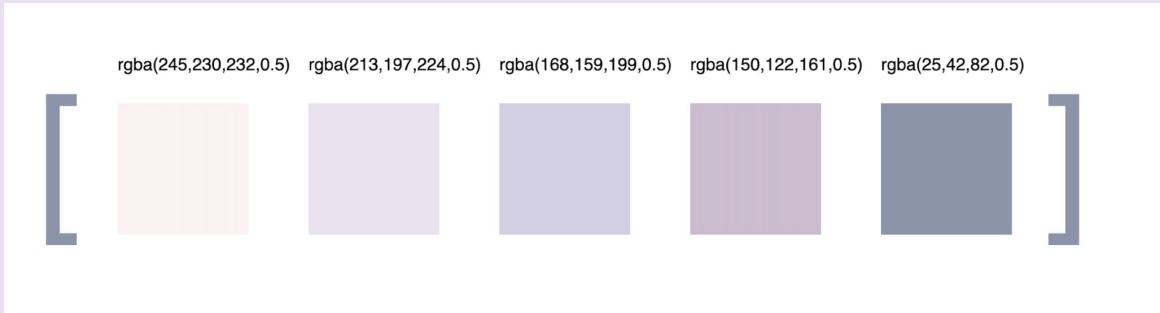
➔ Default random values



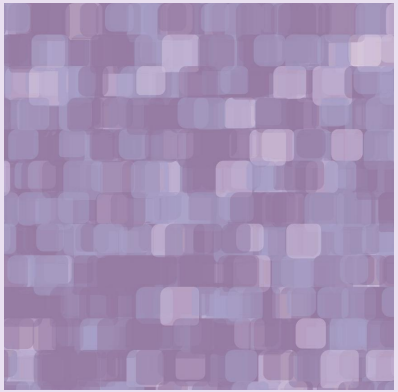
➔ Random values skewed lighter

Sorted Array of Colors

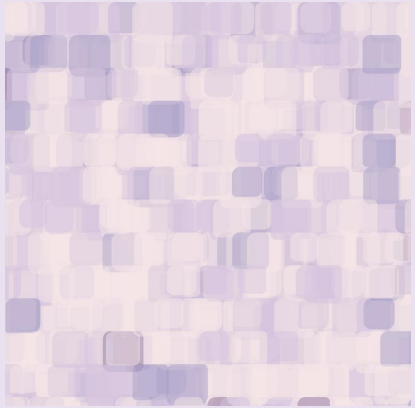
```
const colors = [
    color(352, 6, 96, opacity),
    color(275, 12, 88, opacity),
    color(254, 20, 78, opacity),
    color(283, 24, 63, opacity),
    color(222, 69, 32, opacity)
]
```



Weighted Random

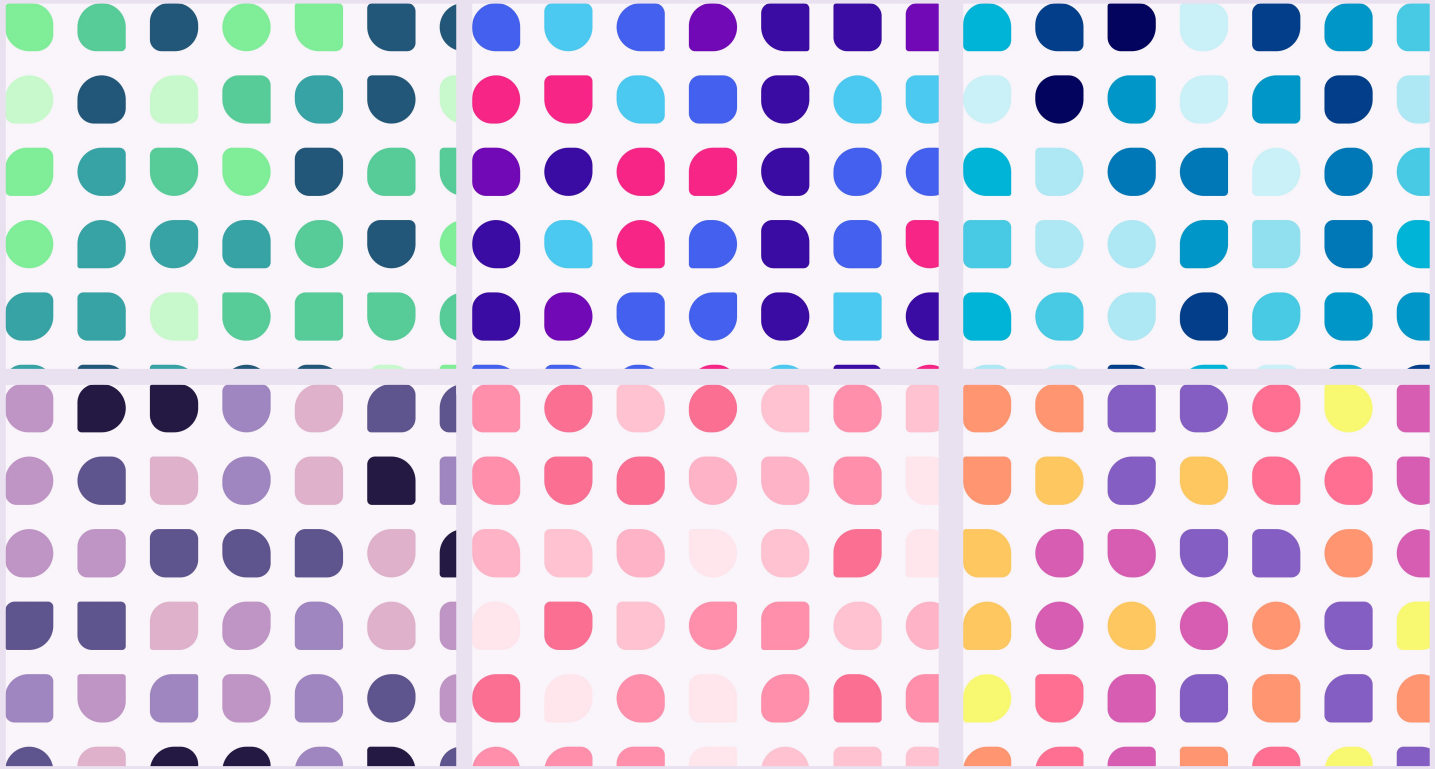


```
Math.floor(Math.max(
      random(0, colors.length - 1),
      random(0, colors.length - 1),
      random(0, colors.length - 1)))
```



```
Math.floor(Math.min(
      random(0, colors.length - 1),
      random(0, colors.length - 1),
      random(0, colors.length - 1)))
```

Re-Generate Random Output



JS

Math.random()

Seed Value

53493

71862

54103

p5.js

random(max)

Seed Value

85931

23125

85931

JS

`Math.random()`

Seed Value


53493


71862


54103

p5.js

`random(max)`

Seed Value


85931


23125


85931

p5.js random()

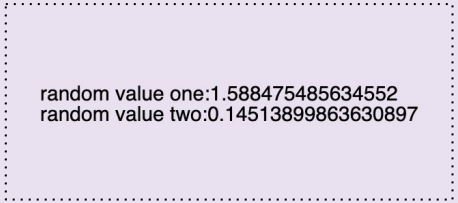
- always returns same sequence of pseudo-random numbers when randomSeed(value) is set
- seed can be: string, number, emoji, etc
- returns different sequence of pseudo-random numbers when there is no randomSeed()

Set seed value so that we always receive the same random number

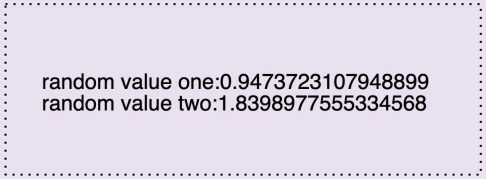
```
randomSeed("RenderATL2023🍑");

const randomNumber =
Math.floor(random(10000, 99999))
```

Generate random number between 10000 and 99999

Re-Generate Random Output



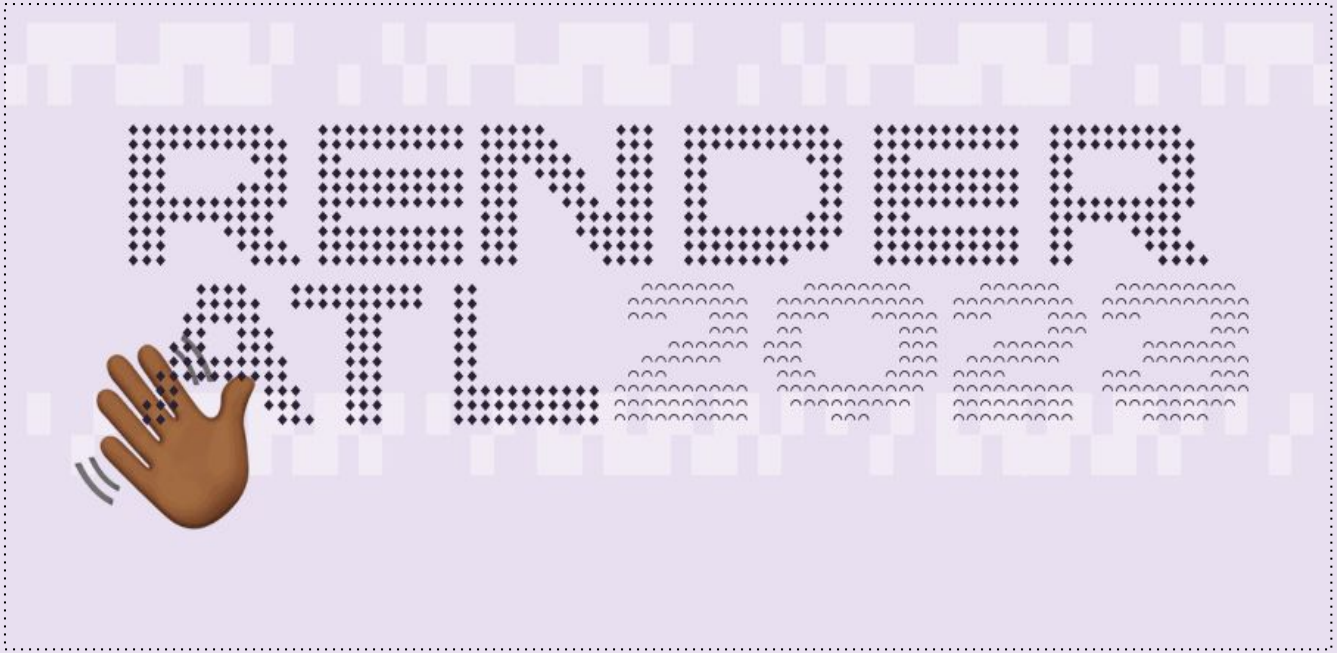

```
// randomSeed(actRandomSeed);
```



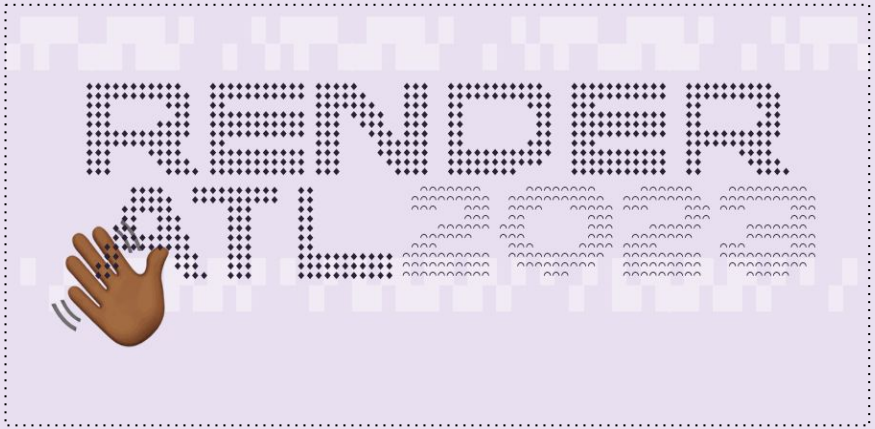

```
randomSeed(actRandomSeed);
```

```
let actRandomSeed;

function setup() {
  createCanvas(500, 500);
  actRandomSeed = int(random(0,4))
}

function draw() {
 clear();
 randomSeed(actRandomSeed);
 textSize(40);
 text(int(random(0,4)), width/2, height/2);
}
```

Anatomy of a Sketch



Anatomy of a Sketch




```
/* preload image */
...
function setup() {
  createCanvas(900, 400);
  background('#e9e0f0');
  fill(43, 33, 53);}

function draw() {
  background('#e9e0f0');
  drawBorders();
  drawHandWave();
  drawSymbols();
}
```

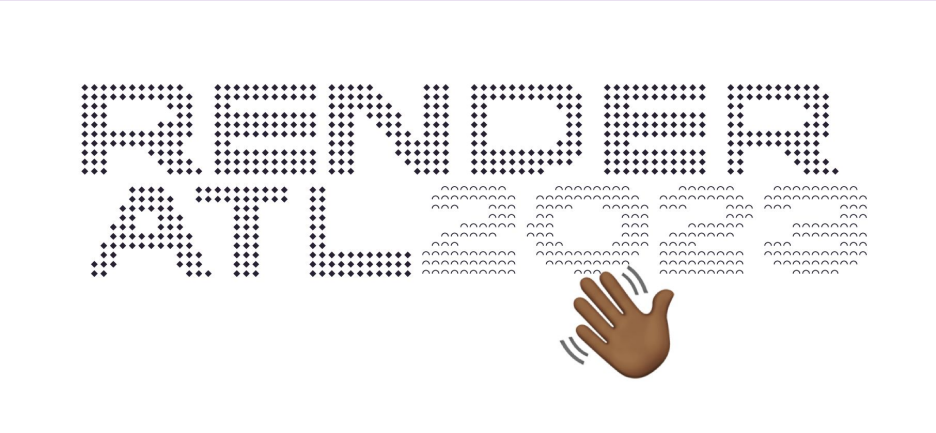
Anatomy of a Sketch

Order matters. What happens if we do not refill with background color in draw()?


```
/* preload image */
...
function setup() {
  createCanvas(900, 400);
  background('#e9e0f0');
  fill(43, 33, 53);}

function draw() {
 // background('#e9e0f0');
  drawBorders();
  drawHandWave();
  drawSymbols();
 }
```

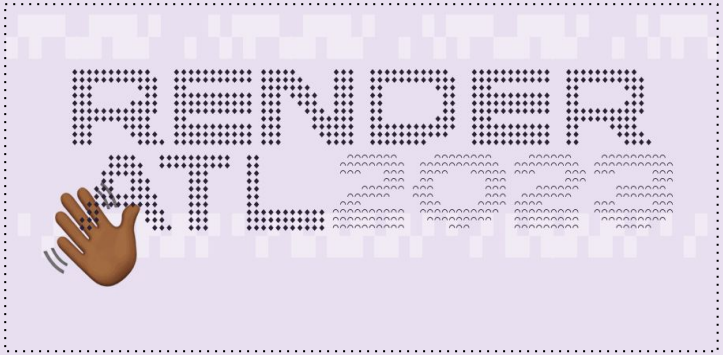
Anatomy of a Sketch

Order matters. What happens if we do not refill with background color in draw()?




```
/* preload image */
...
function setup() {
  createCanvas(900, 400);
  background('#e9e0f0');
  fill(43, 33, 53);}

function draw() {
  clear();
  drawBorders();
  drawHandWave();
  drawSymbols();
 }
```

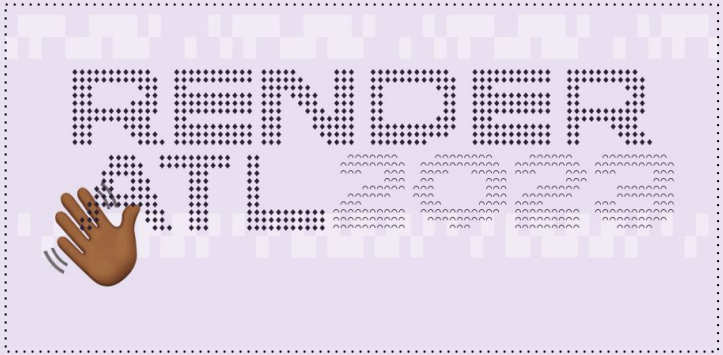
Anatomy of a Sketch



Creates a screen reader accessible description for the canvas:

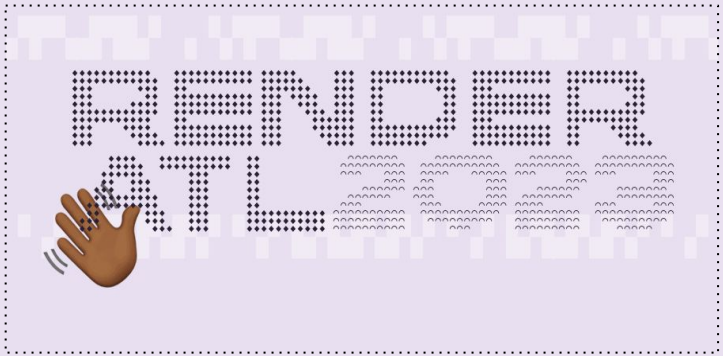
```
describe('A message that says Render ATL 2023 with Render ATL written with diamond unicode characters and 2023 written with the eyebrow shaped unicode character. There is a floating  emoji that is easing its way across the background behind the text, growing and shrinking.');
```

Anatomy of a Sketch



```
function preload() {
  img = loadImage(“imageUrl”);
}

function drawSymbols()
  for (let y = 0; y < img.height; y += 12) {
    for (let x = 0; x < img.width; x += 12) {
      const c = color(img.get(x, y));
      const bright = (red(c) + green(c) +
blue(c)) / 3;
      let mx = map(x, 0, img.height, 0, height)
      let my = map(y, 0, img.width, 0, width)

      if (bright >= 255) {
        text('◆', mx, my)
      } else if (bright >= 143) {
        text('○', mx, my)
      } else {
        // do nothing
      }
    }
  }
}
```

Anatomy of a Sketch



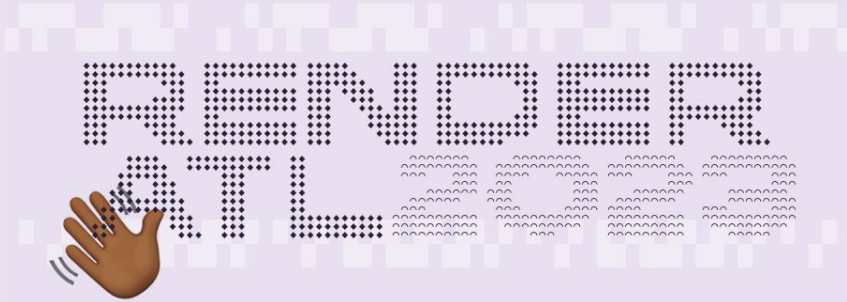
```
function drawSymbols()
for (let y = 0; y < img.height; y += 12) {
    for (let x = 0; x < img.width; x += 12) {
      const c = color(img.get(x, y));
      const bright = (red(c) + green(c) + blue(c))
...
      if (!seen[bright]) {
        seen[bright] = true;
        console.log(seen)
      }
    }
  }
```

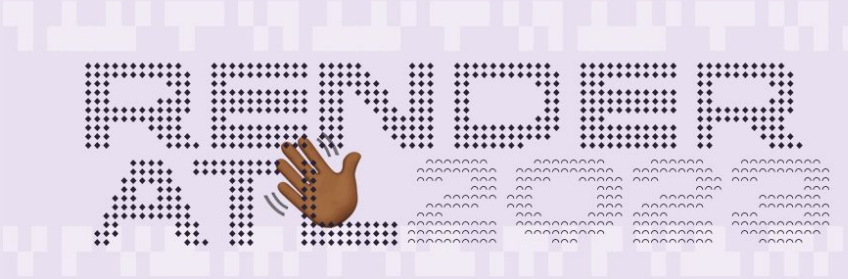
```
{
 "0": true,
 "255": true,
 "204.6": true
}
```

Noise - More Organic Movement

noise() to compute size and x,y position of 👋🏾

random() to compute size and x,y position of 👋🏾





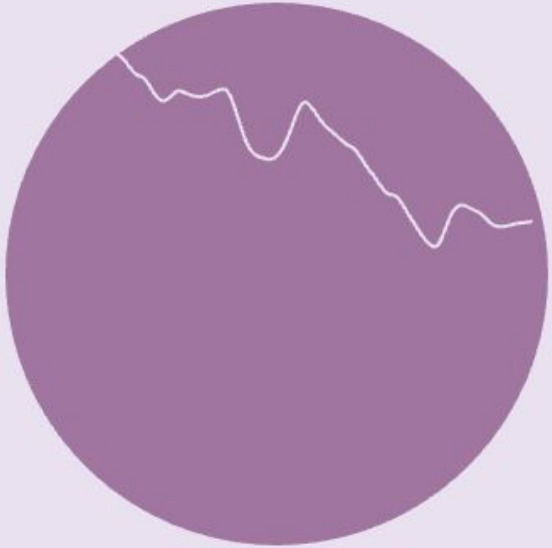
Perlin Noise 🏆
An Academy Award Winning Algorithm

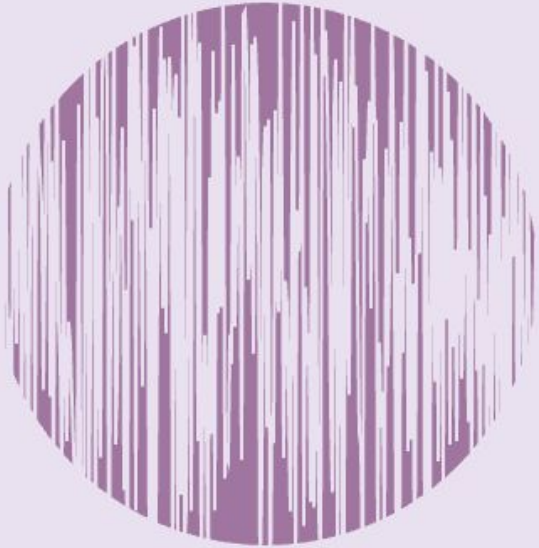
In 1997, Perlin was awarded an **Academy Award for Technical Achievement** for creating the (Perlin Noise) algorithm, the citation for which read:

> To Ken Perlin for the development of Perlin Noise, a technique used to produce natural appearing textures on computer generated surfaces for motion picture visual effects. The development of Perlin Noise has allowed computer graphics artists to better represent the complexity of natural phenomena in visual effects for the motion picture industry

Source: https://en.wikipedia.org/wiki/Perlin_noise

Noise vs. Random Distribution

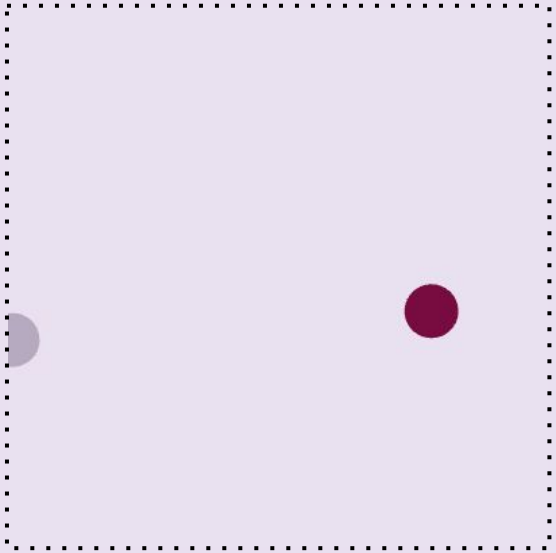




➔ Line drawn with noise()

➔ Line drawn with random()

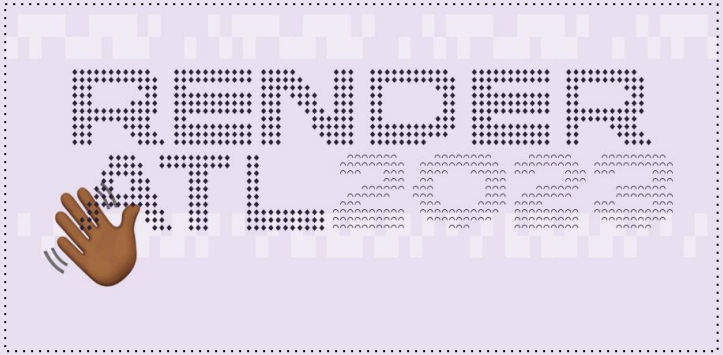
Noise - More Organic Movement



Uses random() to determine x,y position

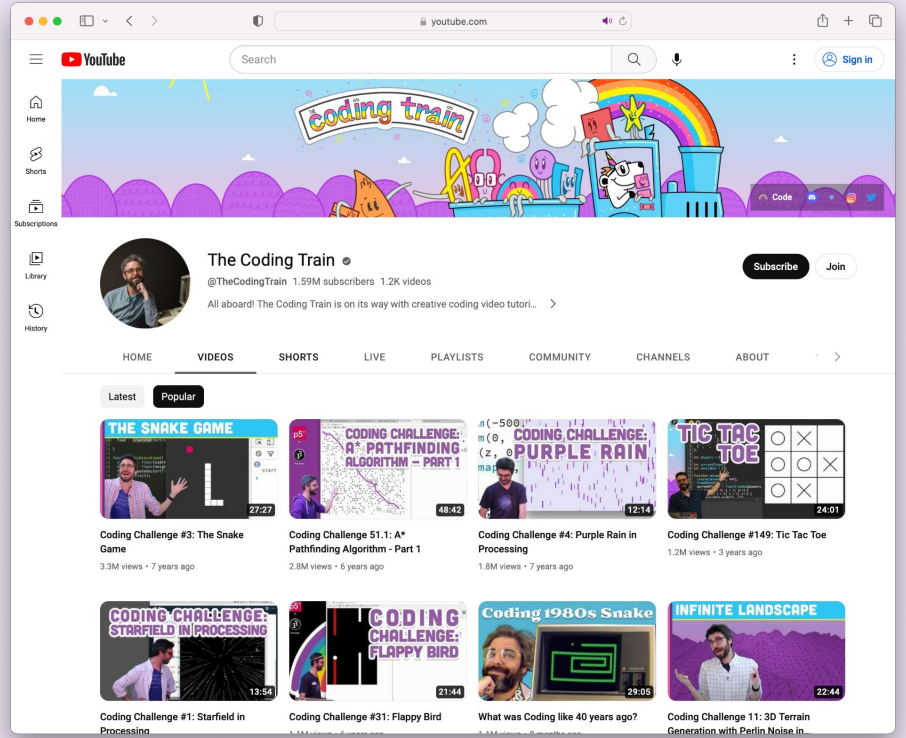
Uses noise() to determine x,y position

Anatomy of a Sketch




```
let xoff1 = 0;
let xoff2 = 10000;
let sizeoff = 2;
...
function drawHandWave() {
  const x = map(noise(xoff1), 0, 1, 0, width);
  const y = map(noise(xoff2), 0, 1, 200, 400);
  const size = map(noise(sizeoff), 0, 1, 75, 200)

  xoff1 += 0.01;
  xoff2 += 0.02;
  sizeoff += 0.01

  textSize(size)
  text("👋🏾", x, y)
}
```

The Coding Train



THANK YOU!

SLIDES + EXAMPLES:
LINKS.MONICA.DEV