PV259
Generative Design Programing

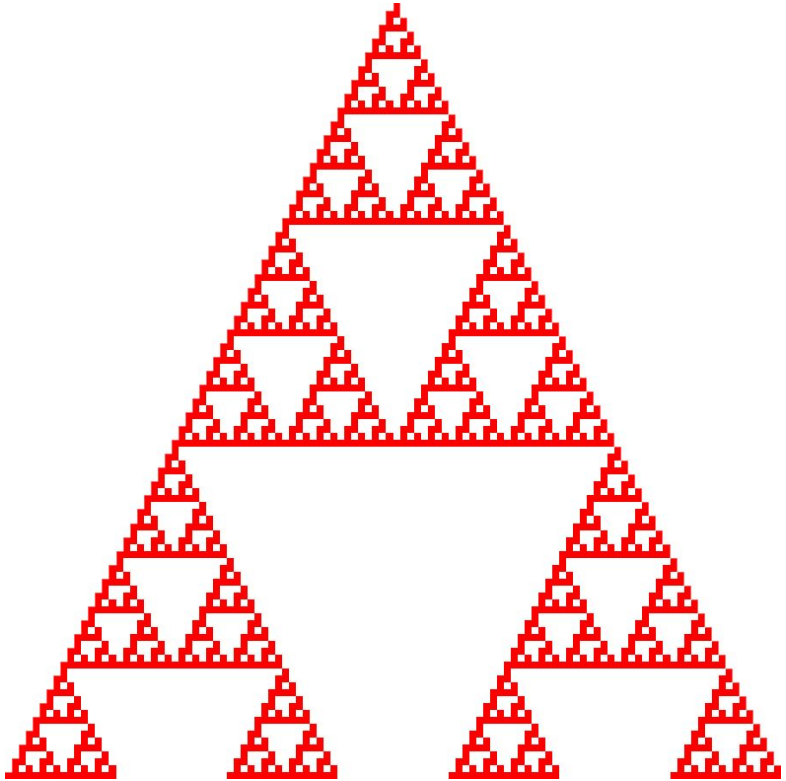
Week 6

Agentic systems

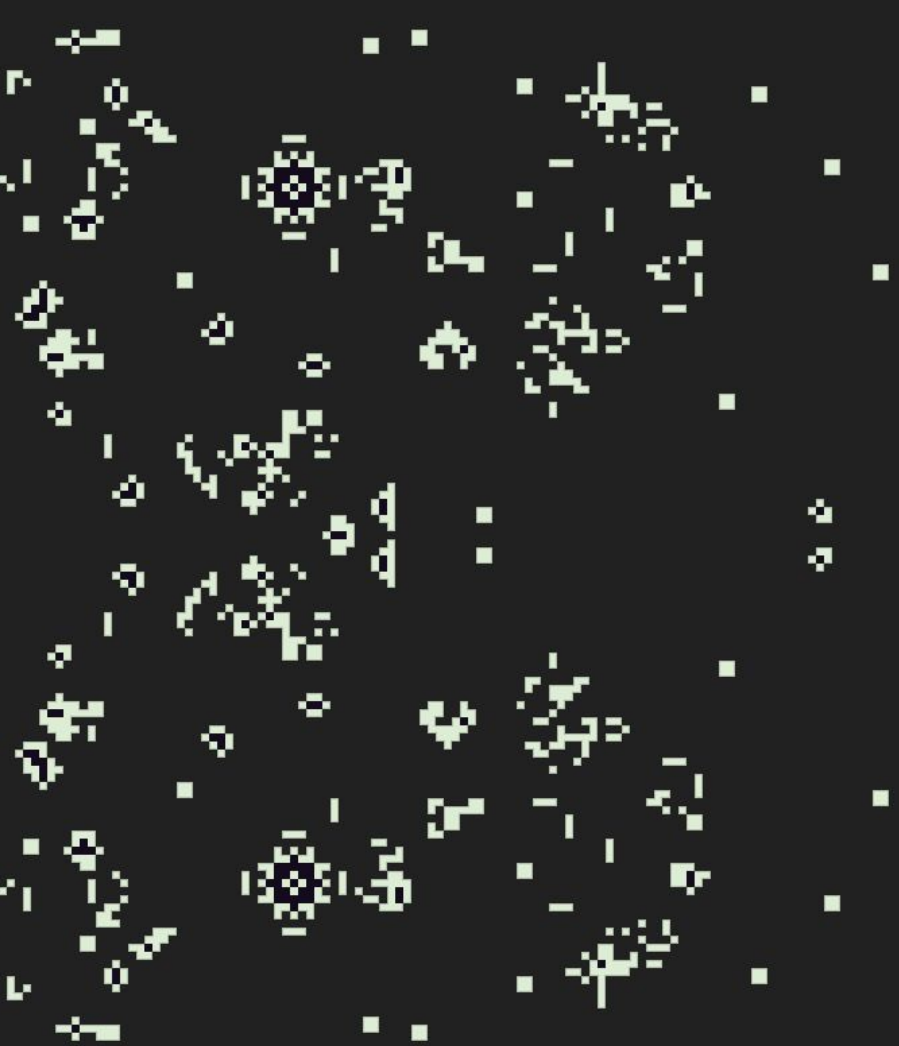
MUNI FI

Marko Řeháček
rehacek@mail.muni.cz

Cybernetics



Rule 90 generating Sierpinski triangle fractal



→ A-LIFE PROGRAM

Game of life

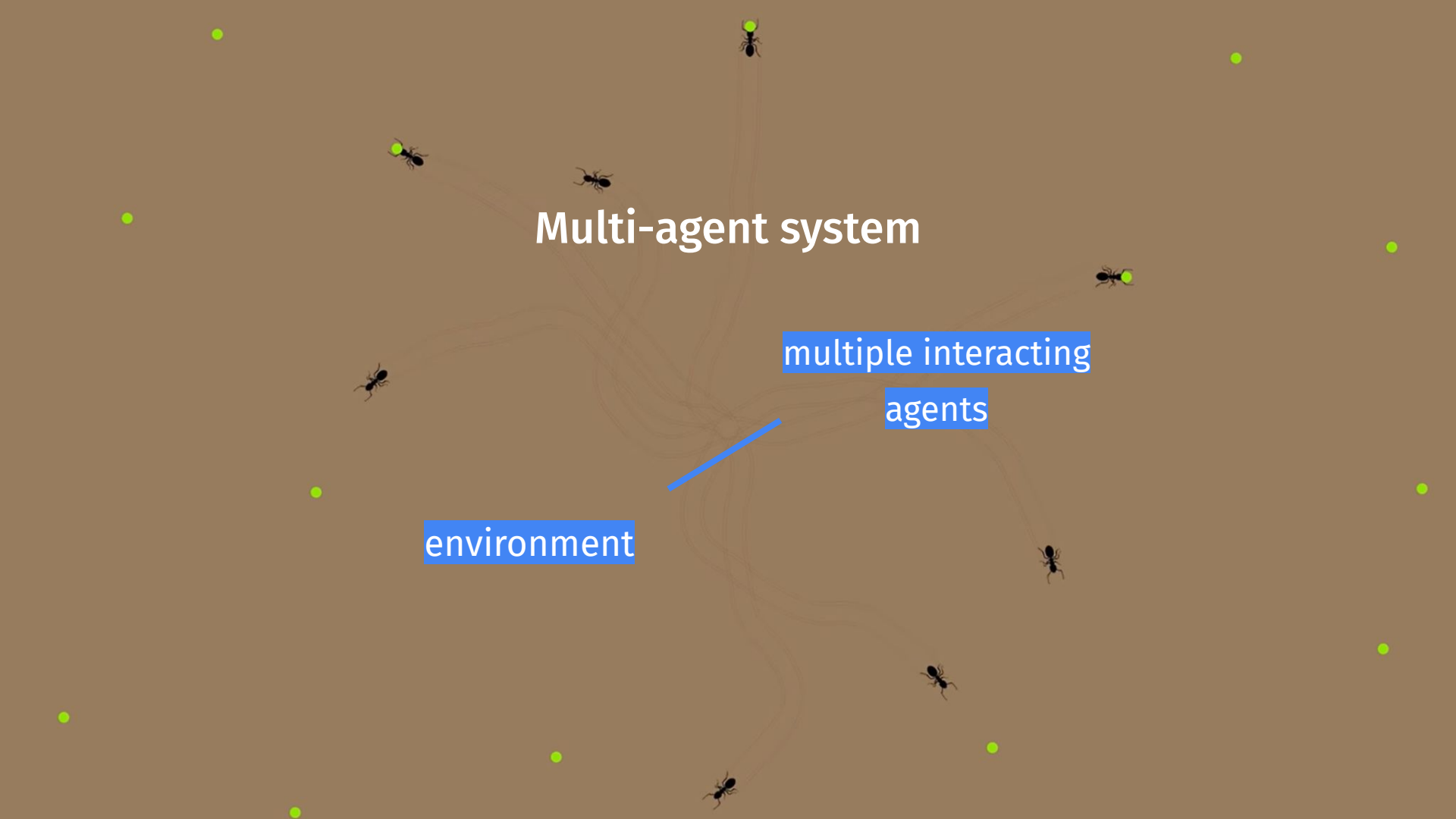
John Horton Conway, 1970

Most popular cellular automaton. It takes place on an **infinite two-dimensional grid** in which **cells** can be **alive or dead**, and is defined by a **set of rules** that jointly determine the state of a cell given the state of its neighbours. Following specification of an initial configuration, patterns evolve over time across the grid requiring no further user input.

The rules

1. Any live cell with two or three live neighbours survives.
2. Any dead cell with three live neighbours becomes a live cell.
3. All other live cells die in the next generation. Similarly, all other dead cells stay dead.

Example: alexandrunst.github.io/Game-of-Life/
Youtube: GoL in GoL
Wikipedia


Multi-agent system
multiple interacting agents
environment

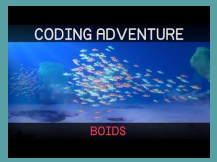
→ A-LIFE PROGRAM

Boids

Craig Reynolds, 1986

Reynolds invented algorithmic steering behaviors for animated characters. These behaviors allowed individual elements to navigate their digital environments in a “lifelike” manner with strategies for fleeing, wandering, arriving, pursuing, evading.

Boids is an A-life program simulating the flocking behaviour of birds. A genre of ***swarm art*** emerged.



Coding Adventure: Boids, by Lague
https://www.youtube.com/watch?v=bqtqltqcQhw
Algorithm explanation

Example ->

PROJECTION

Inside a Flock / Uvnitř Hejna

Tomáš Janoušek, 2022

Projection simulates flocking of birds with a spectator inside. Birds fly in groups, they also separate and form new groups. Spectator can scatter them by walking to them. Installation extends boids structures by Craig Reynolds from the year 1986 using multiple layers and interactivity.

MOVIE

An Autonomous Agent Choreo

Toni Mitjanit, 2016

Attraction-repulsion.

https://filmfreeway.com/aaac

Recap

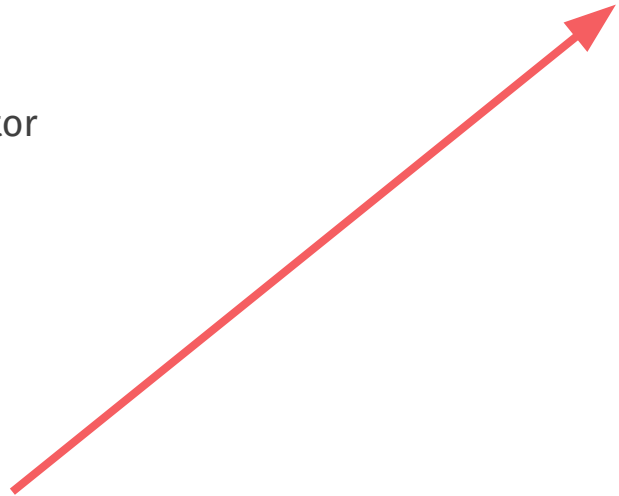
Vectors

```
const v = createVector(x, y, [z])
      // v.x returns x coordinate
      // v.y returns y
```

Many methods available

- **set()** to change x, y without creating new vector
- **add()** to add two vectors, or add scalar
- **mult()** to multiply fields by scalar
- **array()** access as array instead of object
- **random2D()** create random vector

... reference page with more methods



OOP recap

Class vs. Object only

```
class Car {
    constructor(whose) {
        this.mileage = 340671;
        this.owner = whose;
    }
    rent() {
        this.mileage = 121020;
    }
}

const obj = new Car(“Peder”);
```

```
let car = {
    mileage: 340671,
    owner: “Peder”,
    array :  [
        “some”,
        “text”,
        “for”,
        “example”
    ]
}
car.owner = “Jano”;
car[‘owner’] = “”;
```

Example ->

Example 2 ->

Autonomous agents

Autonomous agent...

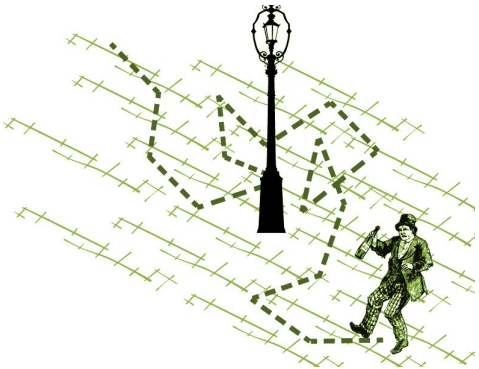
- has limited ability to perceive environment
- processes the information from its environment and calculates an action
- has no leader

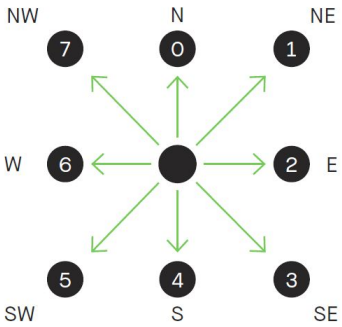
Dumb agent

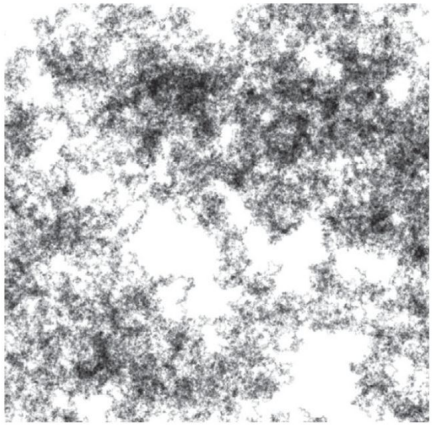
1. write a class representing Agent
   a. store it’s position (**createVector()**)
   b. create draw method, which makes the visual representation of it
2. it does nothing, just sits in center of screen

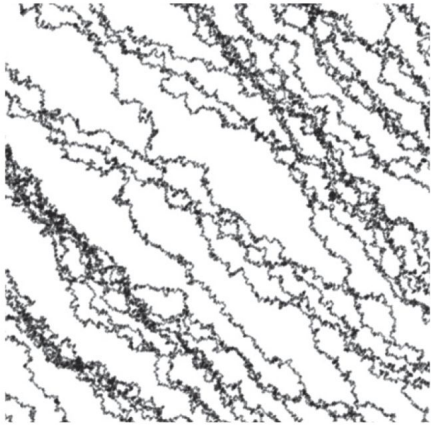
Random walk

Move and in each step, choose direction randomly.









Examples from the book *Generative Design: Visualize, Program, and Create with JavaScript in p5.js* (Benedikt Groß, Hartmut Bohnacker, Julia Laub)

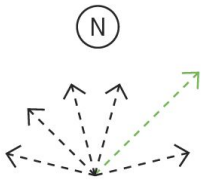
4am Brno

1. implement random walk (https://editor.p5js.org/mrehacek/sketches/BpWZUDqJe)
2. spawn more agents

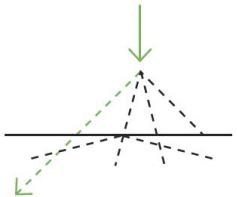
Code: single agent

Code: complete

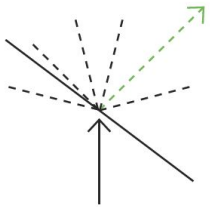
More intelligent movement



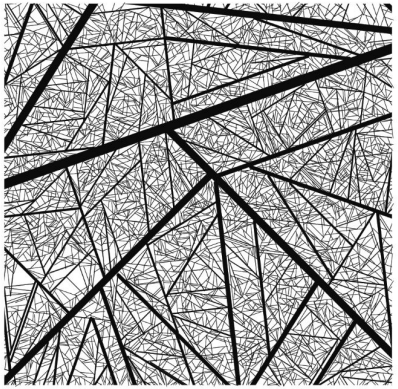
possible angles of the agents moving north

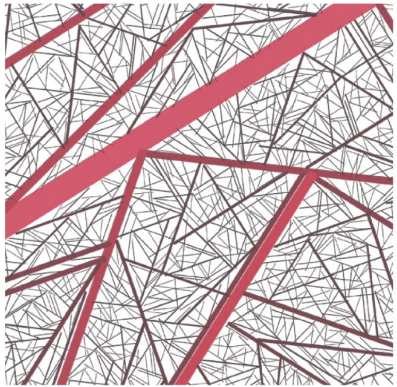


agent reaches border of drawing canvas

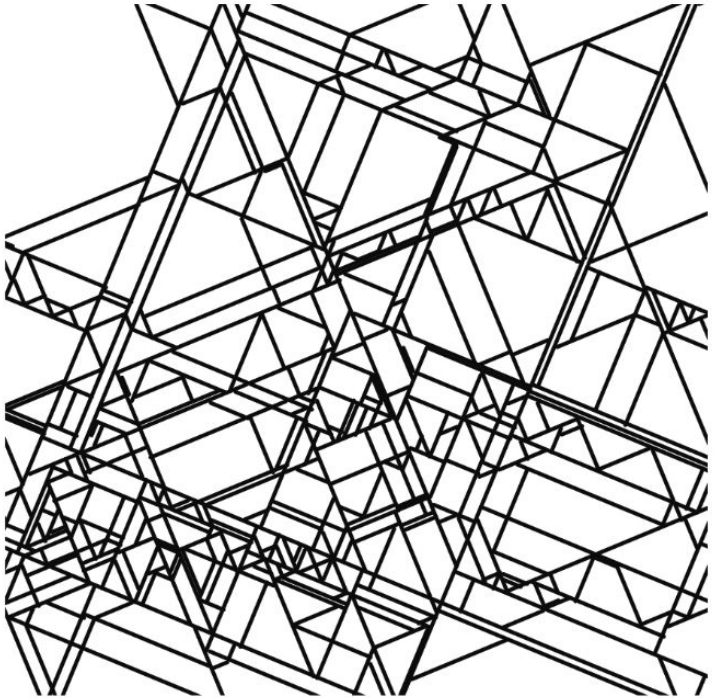


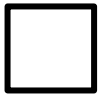
agent crosses its own path.





Examples from the book *Generative Design: Visualize, Program, and Create with JavaScript in p5.js* (Benedikt Groß, Hartmut Bohnacker, Julia Laub)





Drunk but intelligent agent

1. bounce of the edge – implement edge collisions

Code: complete

Physical simulations

Let’s try to simulate some apple falling. What are the laws?
A force is a vector that causes an object with mass to accelerate.

Newton’s first

An object at rest stays at rest and an object in motion stays in motion at a constant speed and direction unless acted upon by an unbalanced force.

→ **velocity** vector

Newton’s second

Force equals mass times acceleration. F=M×A

→ **acceleration** vector

Gravity

Make agent fall with gravity and bounce of the canvas edge. Use velocity and acceleration. Force, in our case downward acting gravity, will change acceleration, and acceleration will add speed of the object.

Code: complete

p5.Vector syntax vs. semantics

Let assume:

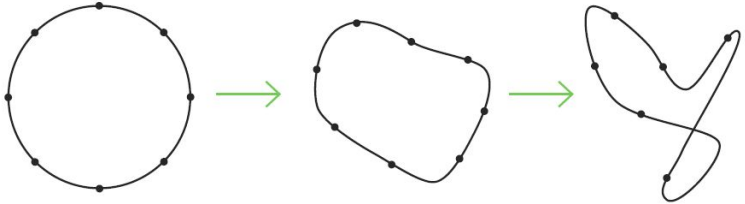
```
…
this.pos = createVector(x, y);
this.vel = createVector(s, t);
…
```

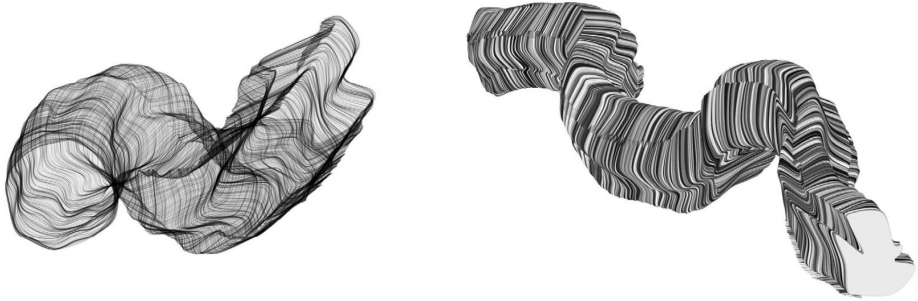
Then:

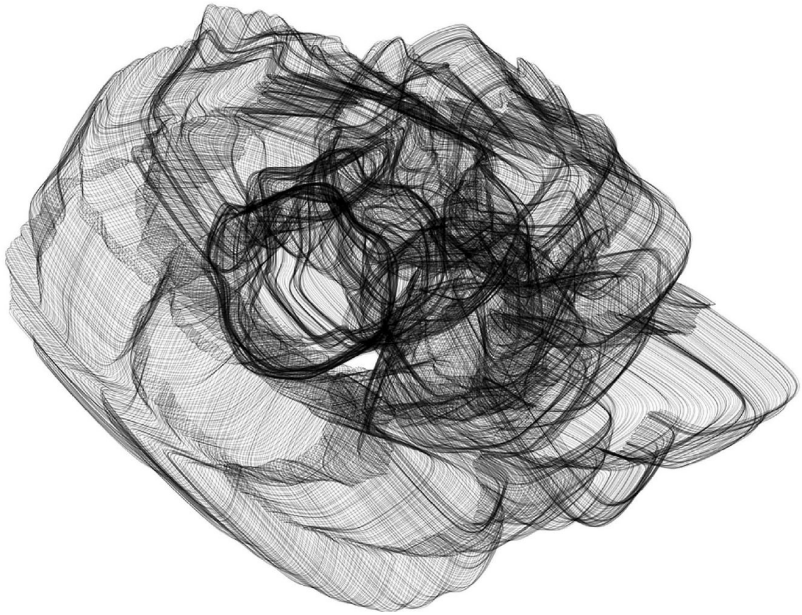
```
this.pos.add(this.vel);   //  OK

this.pos += this.vel;     // NOK - we cannot add or concatenate 2 objects

this.pos.x += this.vel.x; //  OK - we add 2 numbers, not objects
```

But be careful. All the expressions above are syntactically correct!
Thus, in the second case the error does not arise (due to automatic javascript type conversion).

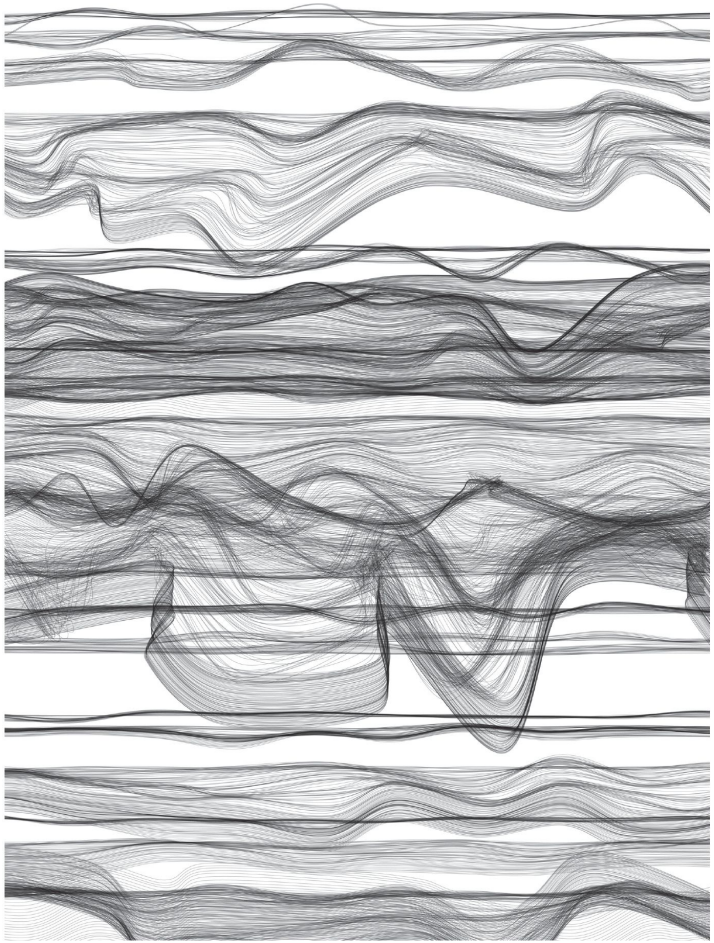
Shapes and noise

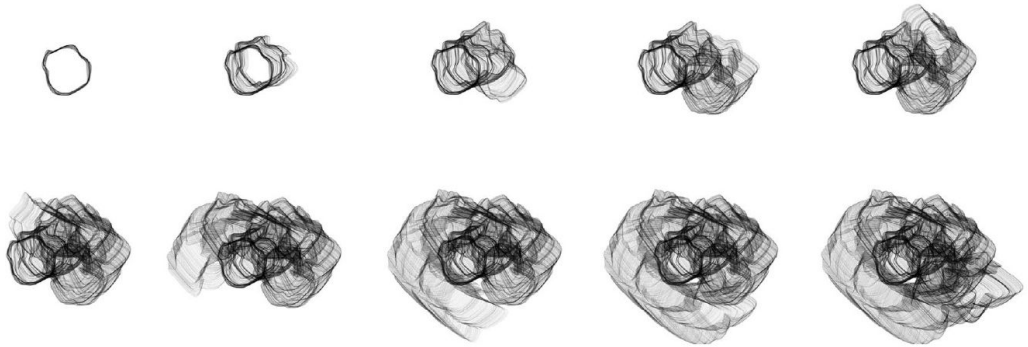






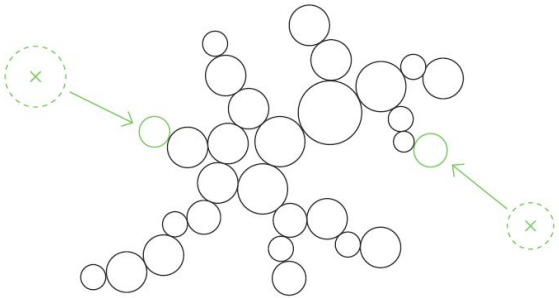
Examples from the book *Generative Design: Visualize, Program, and Create with JavaScript in p5.js* (Benedikt Groß, Hartmut Bohnacker, Julia Laub)

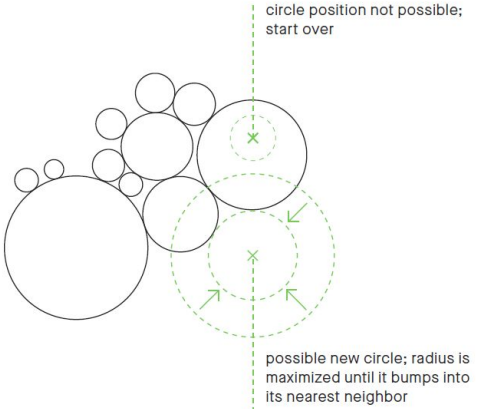


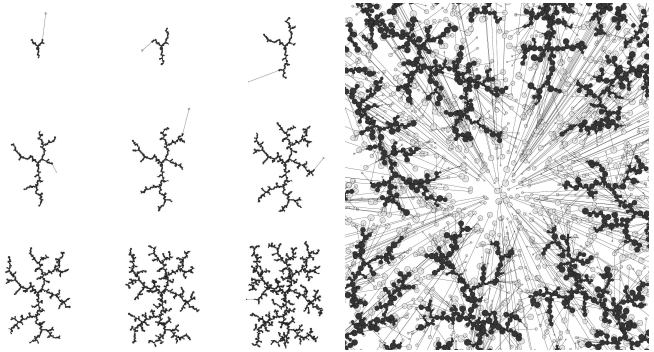


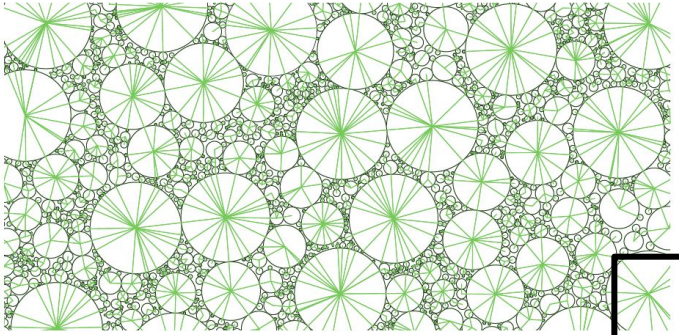
Examples from the book *Generative Design: Visualize, Program, and Create with JavaScript in p5.js* (Benedikt Groß, Hartmut Bohnacker, Julia Laub)

Growth structures





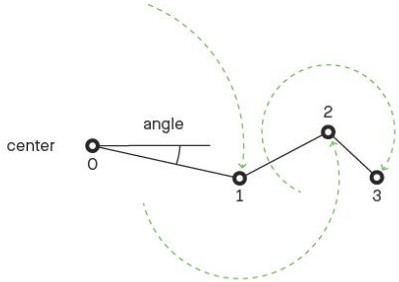


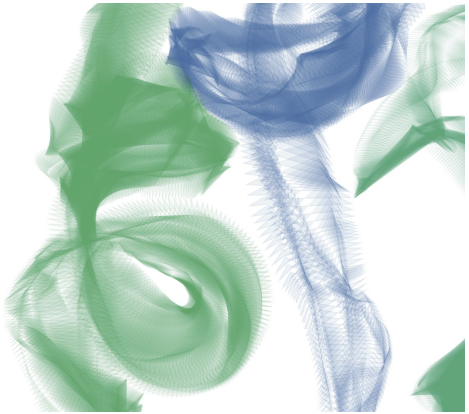


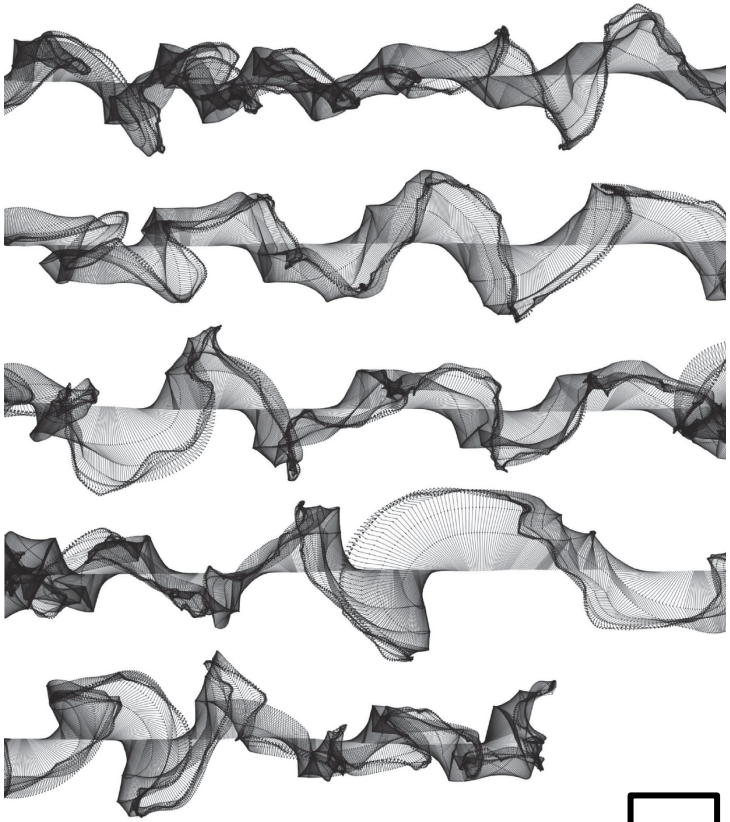
Examples from the book *Generative Design: Visualize, Program, and Create with JavaScript in p5.js* (Benedikt Groß, Hartmut Bohnacker, Julia Laub)

Pendulums

The agents form a chain of pendulums. To determine the individual positions of the agents, we begin at the center. The angle of rotation and the length of the pendulum determine the position of the first agent. This is also the rotation center for the next pendulum. The farther out in the chain, the shorter the pendulum becomes and the faster it turns. Every other pendulum turns in the opposite direction.

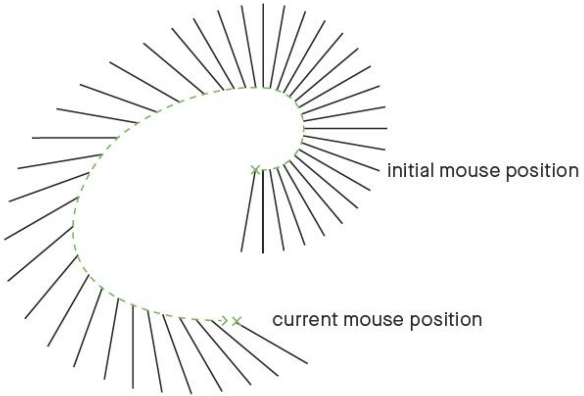


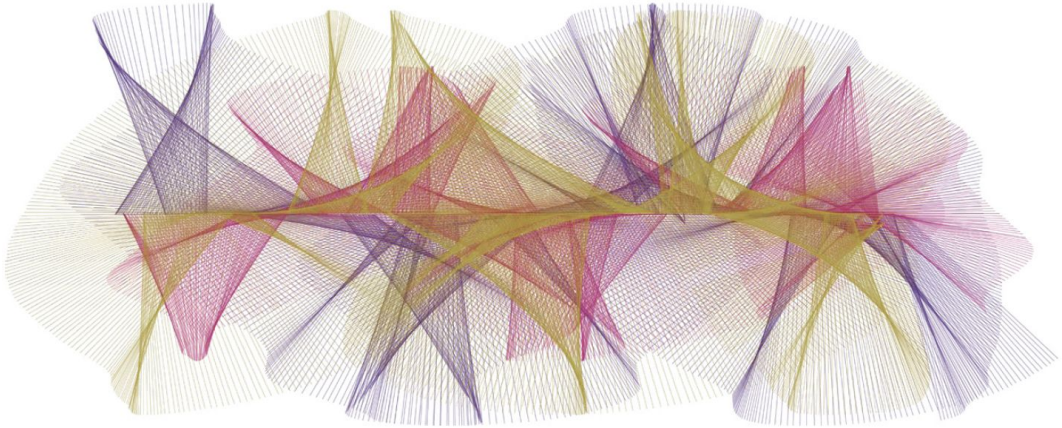




Examples from the book *Generative Design: Visualize, Program, and Create with JavaScript in p5.js* (Benedikt Groß, Hartmut Bohnacker, Julia Laub)

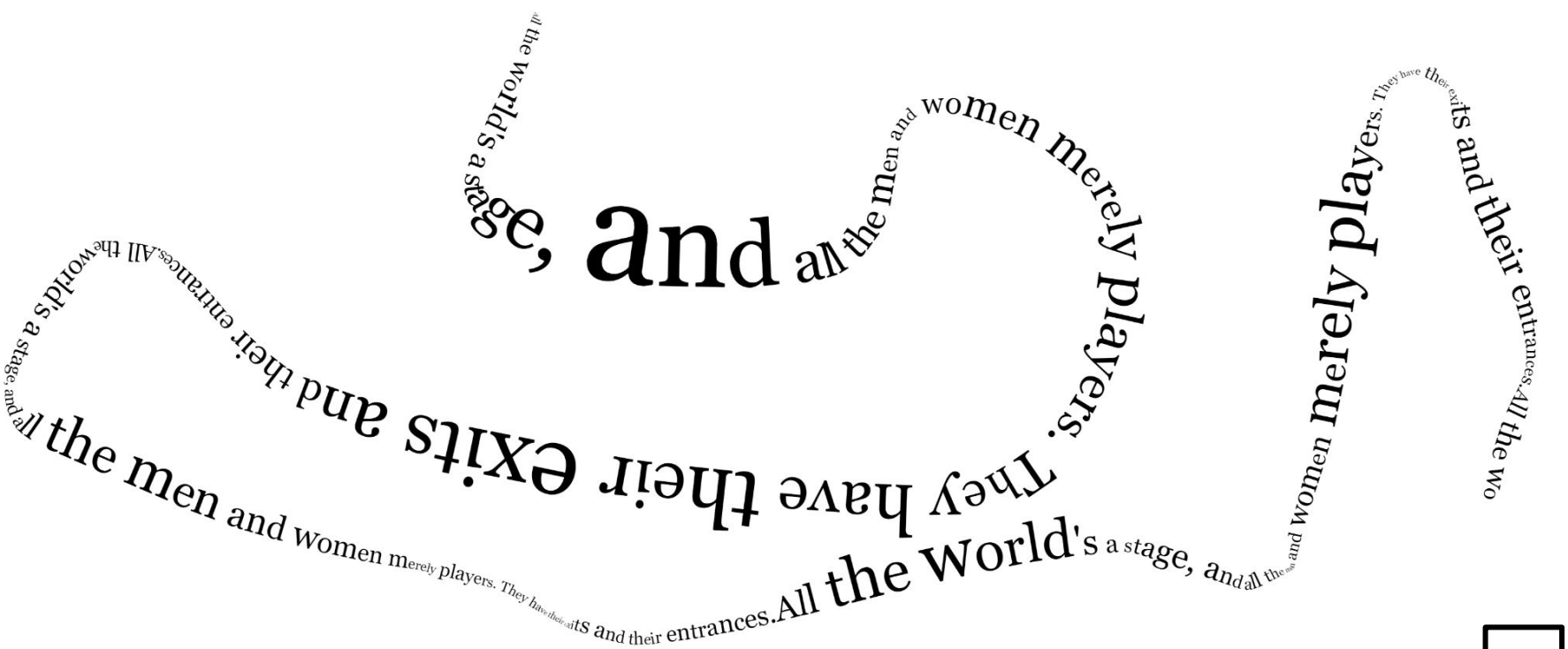
Drawing with mouse





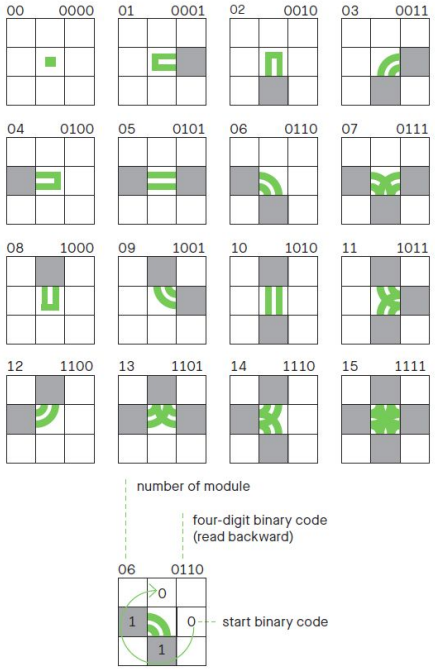
Examples from the book *Generative Design: Visualize, Program, and Create with JavaScript in p5.js* (Benedikt Groß, Hartmut Bohnacker, Julia Laub)

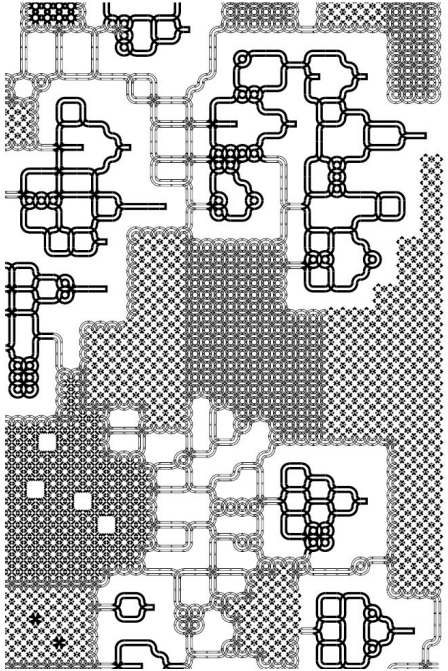
Drawing type



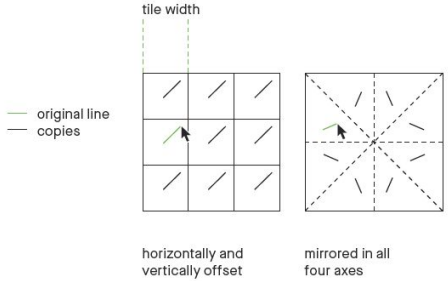
Examples from the book *Generative Design: Visualize, Program, and Create with JavaScript in p5.js* (Benedikt Groß, Hartmut Bohnacker, Julia Laub)

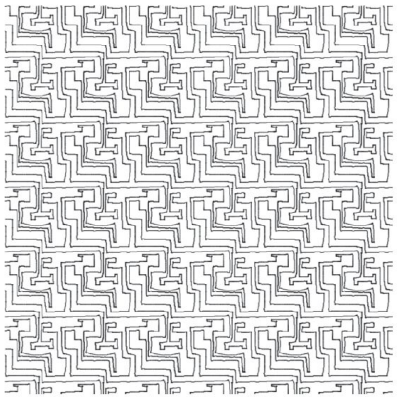
Cells

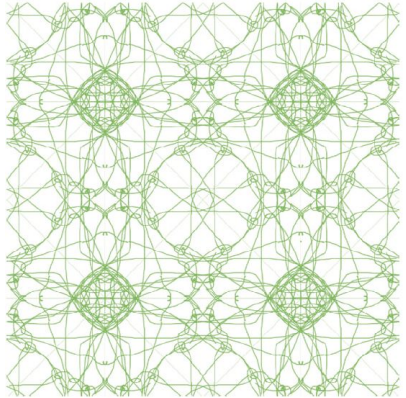




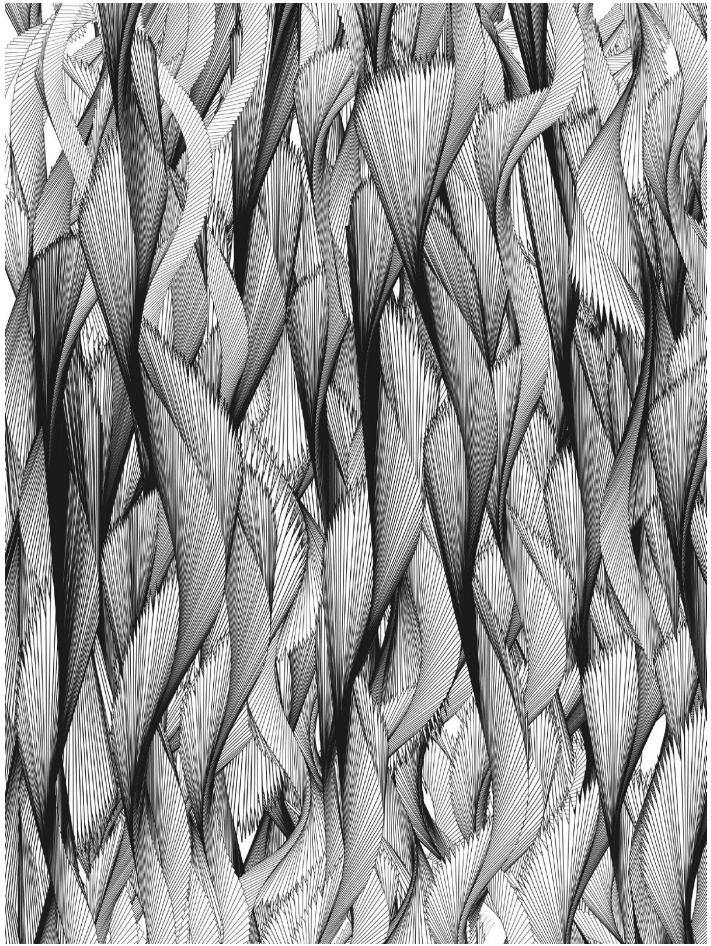
Symmetries

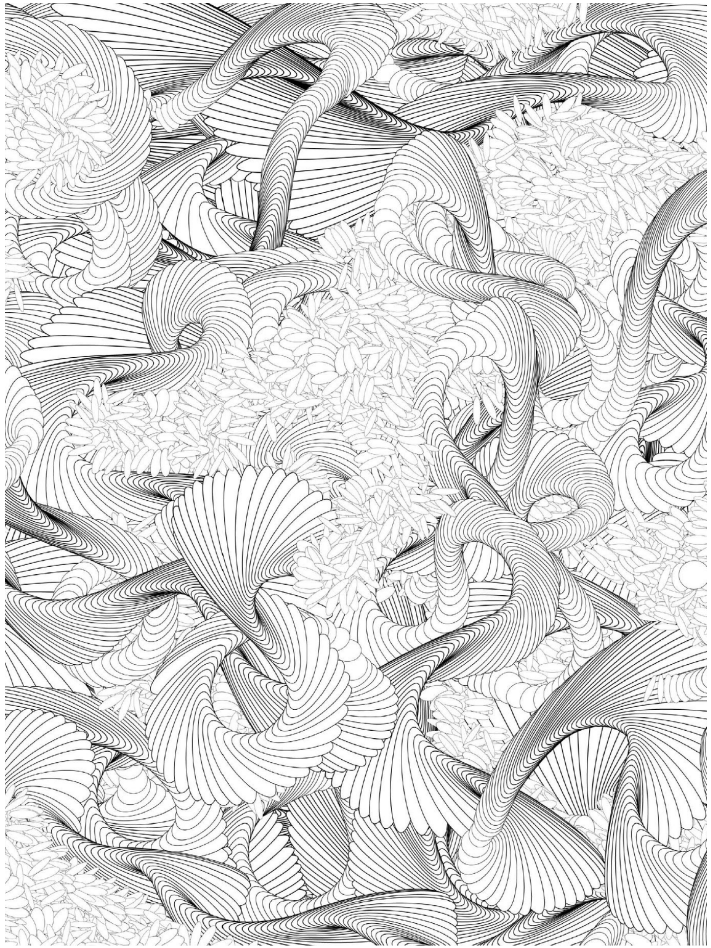




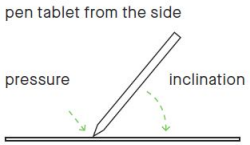


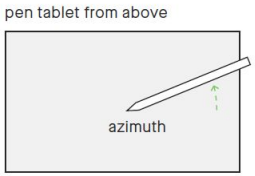
Examples from the book *Generative Design: Visualize, Program, and Create with JavaScript in p5.js* (Benedikt Groß, Hartmut Bohnacker, Julia Laub)

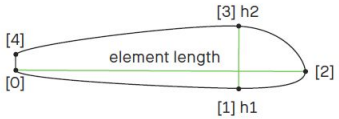


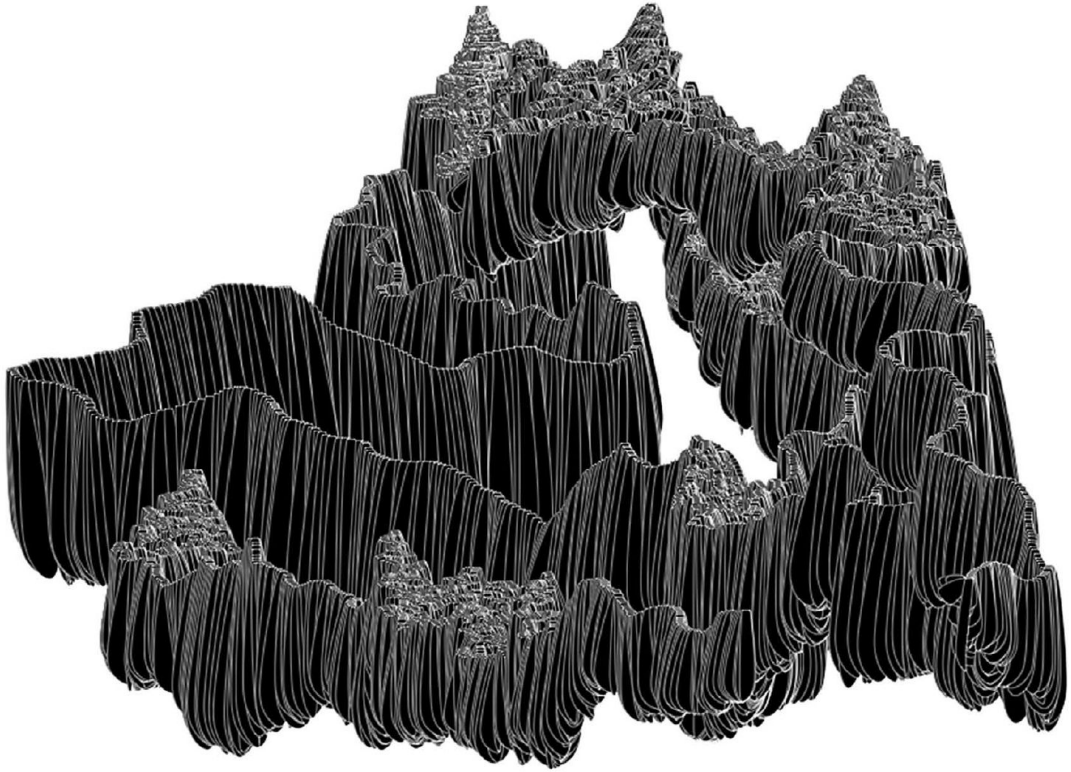


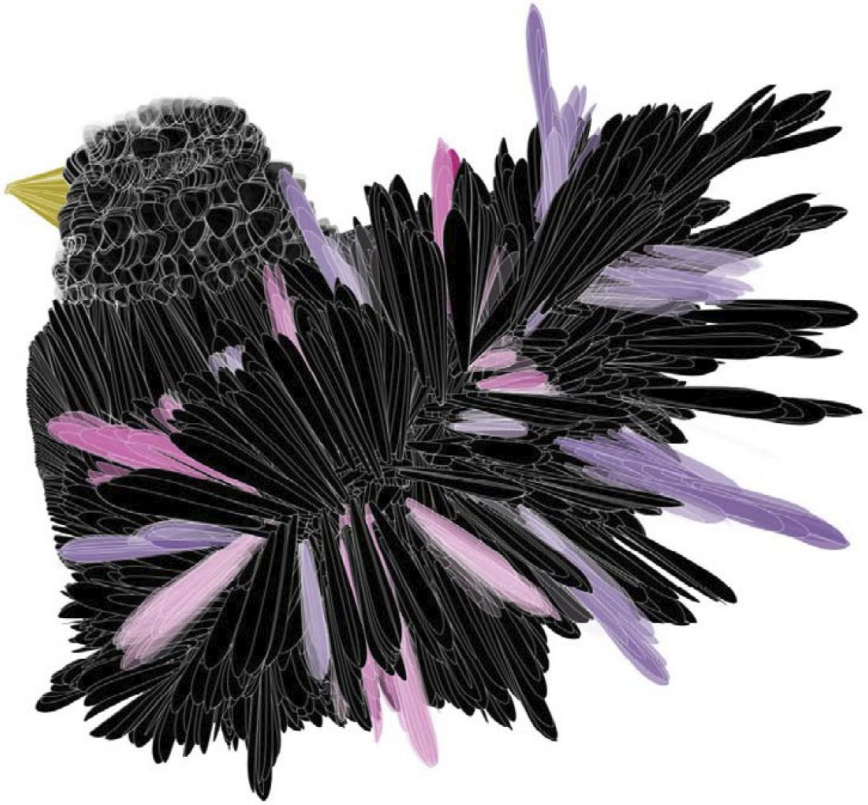
Drawing tablet











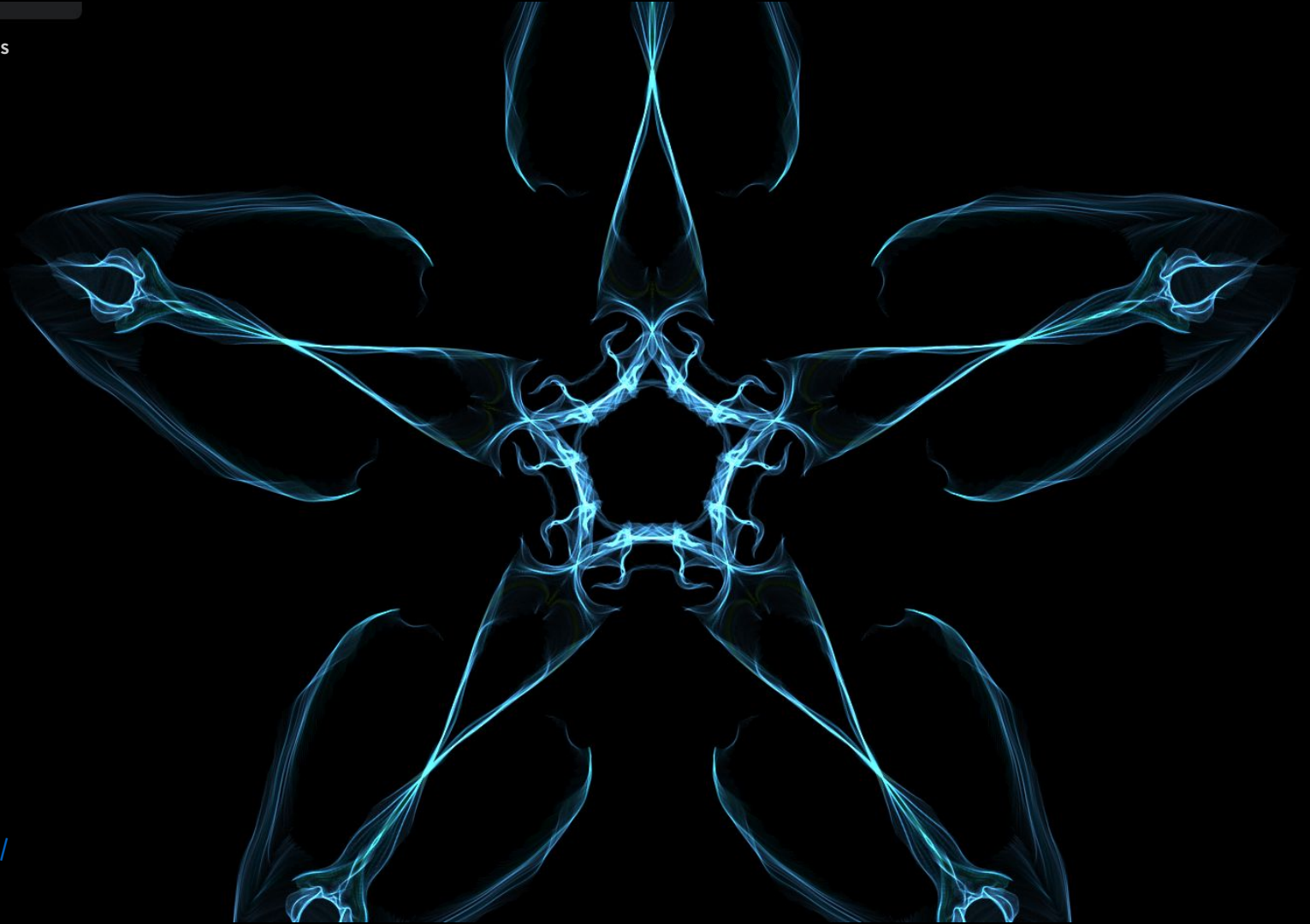
Animated brush

→ TUTORIAL

→ INTERACTIVE APP

Weavesilk

Yuri Vishnevsky

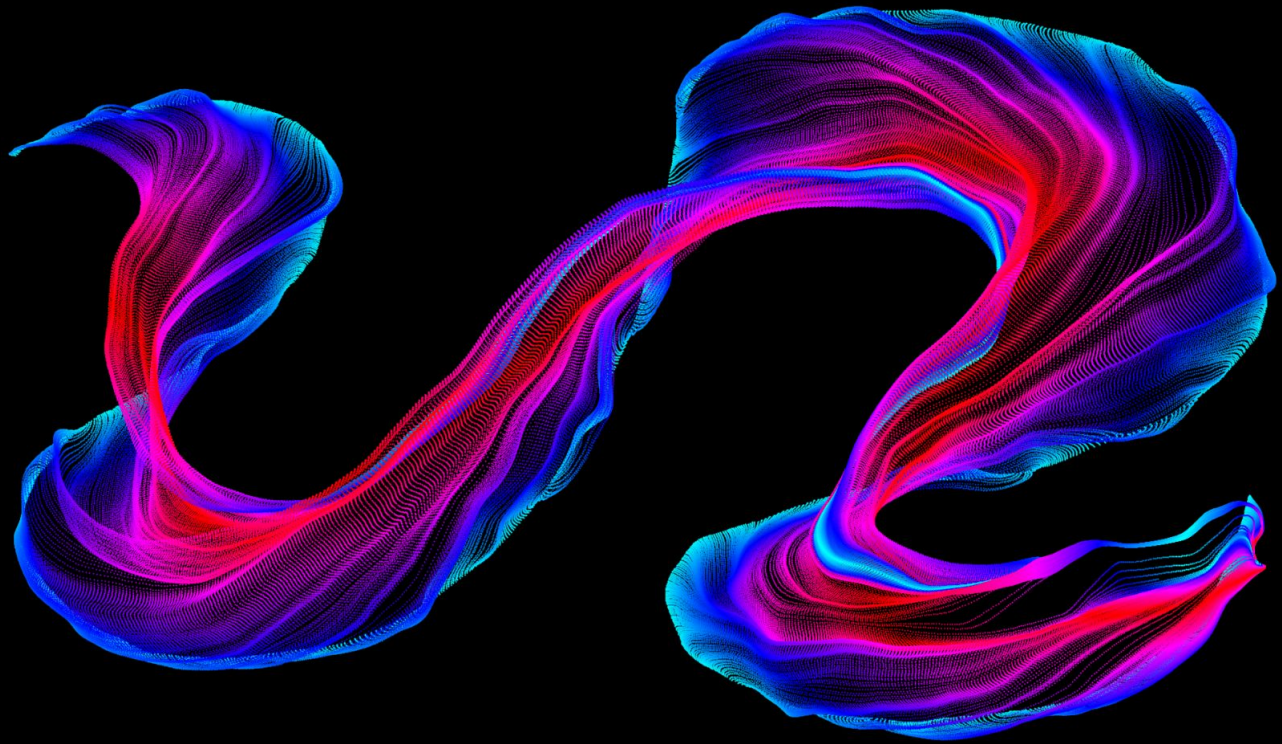


http://weavesilk.com/

→ IMAGE

Unnamed

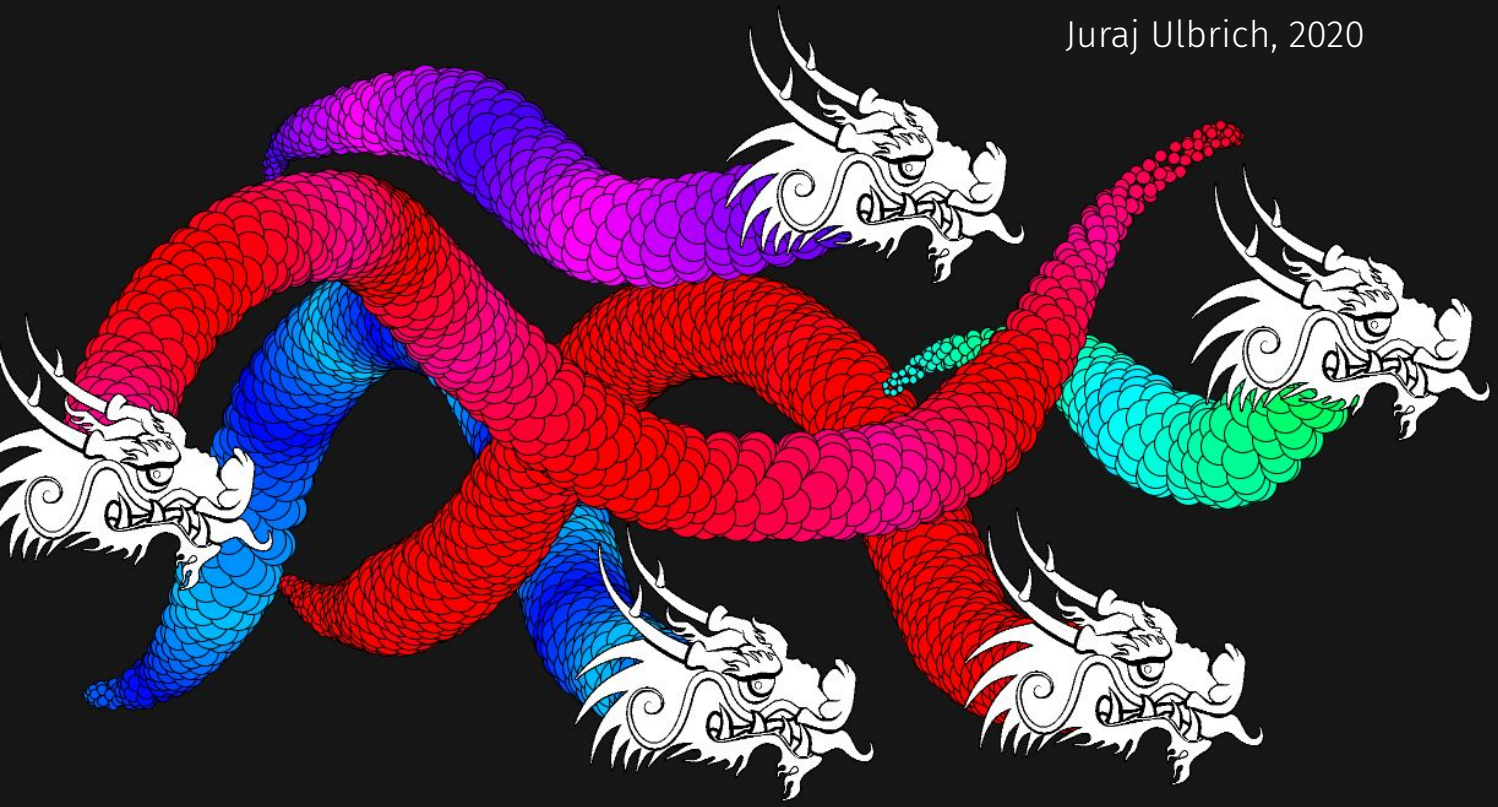
Hana Tokárová, 2020



→ IMAGE

Dragon brush

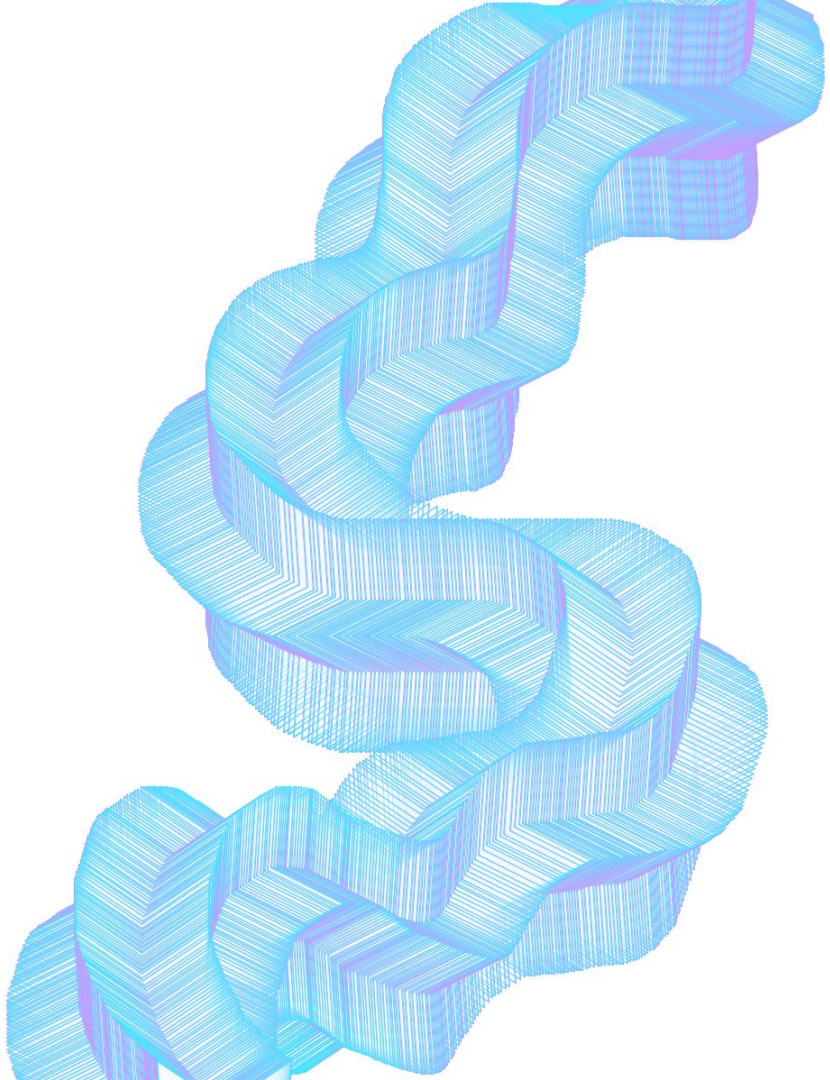
Juraj Ulbrich, 2020



→ IMAGE

Star brush

Tamara Babálová, 2020



Animated brush

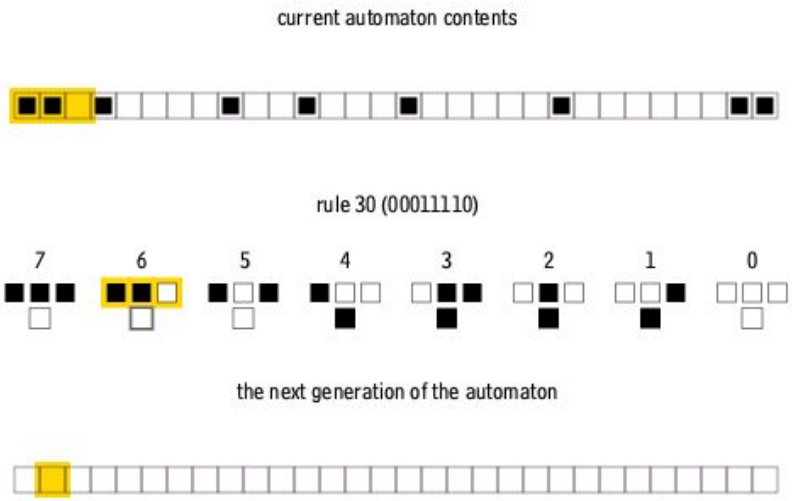
1. write a class representing Agent
   a. store it's position (**createVector()**)
   b. create draw method, which makes the visual representation of it
2. instantiate several agents on random positions (**new** `Agent( ... )`) and store them in an array
3. inside the sketch's draw(), call draw() on each agent, so the agents can draw themselves
   a. remember, the agents are autonomous (don't make the sketch's draw() function the leader)
4. create agents on mouse click (**mouseClicked() + mouseX + mouseY**)
5. move the agents
6. implement some interesting behavior (inspiration in slides)
   a. move in random/noise direction
   b. cellular life – die after being drawn X times
   c. bounce from the boundaries of the sketch

Cellular automata

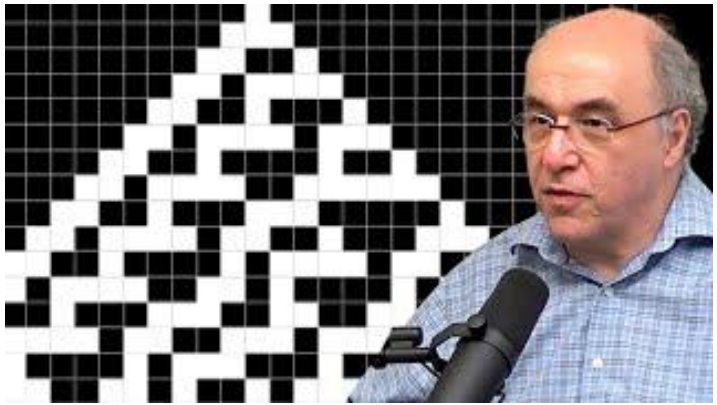
Nature of code chapter

1-dimensional cellular automata

1. You have row of cells, cells have state on and off (1 or 0)
2. Based on the row, you calculate new row using rules. For example, if we look at first 3 cells, and they are all “on”, then turn off the middle cell in new row.
3. Repeat infinitely, usually draw rows below each other to show the progress.

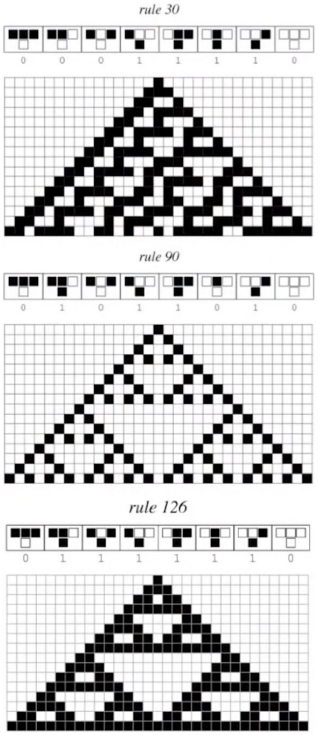


Rule 30



Lex Fridman + Stephen Wolfram podcast

“They seem to be uncrackable … they show irreducible computation”
“Can you prove it will never repeat? We were never able to show that“
“Are there equal number of 0,1 in center column?”



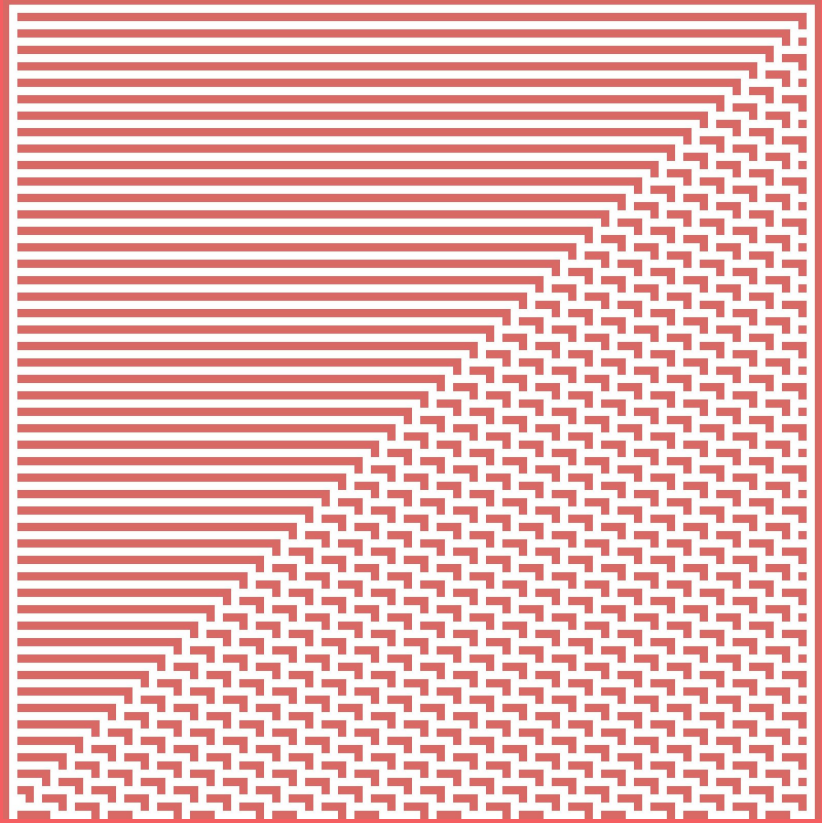
A Conus textile shell similar in appearance to Rule 30.

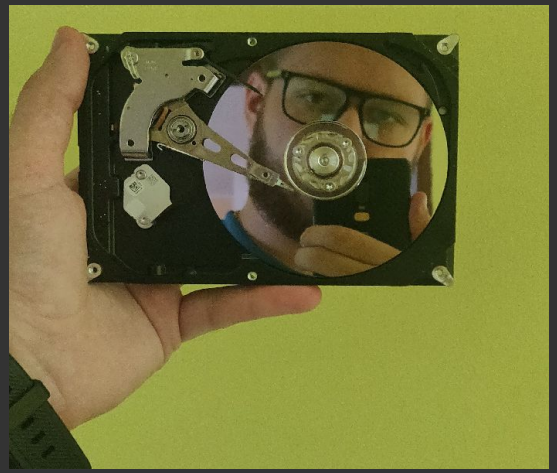
Rule 110

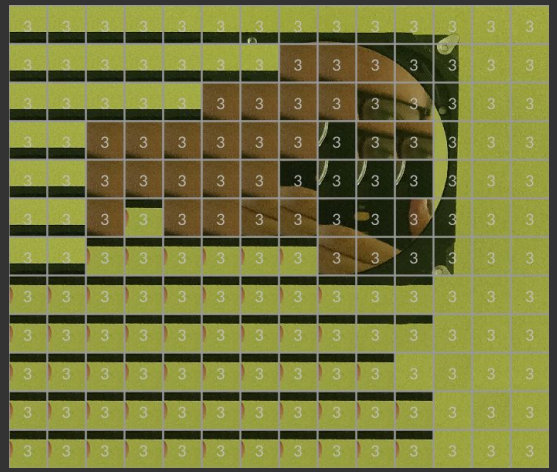
In 2004, Matthew Cook published a proof that Rule 110 with a particular repeating background pattern is Turing complete, i.e., capable of universal computation, which Stephen Wolfram had conjectured in 1985.

Recreate it.

Finished code



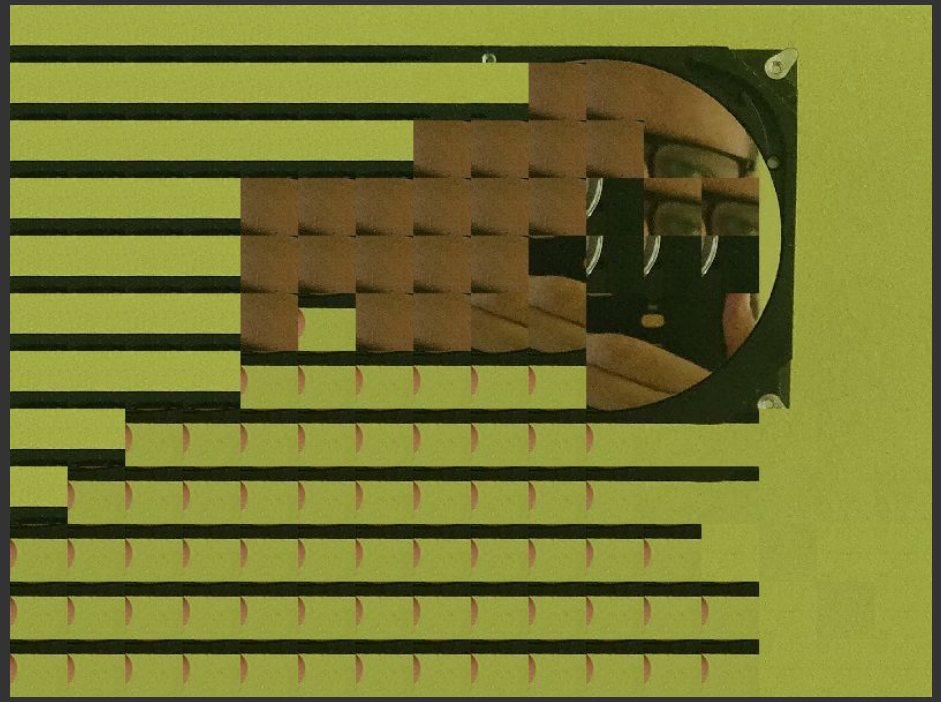




→ IMAGE, A-LIFE PROGRAM

Game of image

Ján Popeláš, 2020

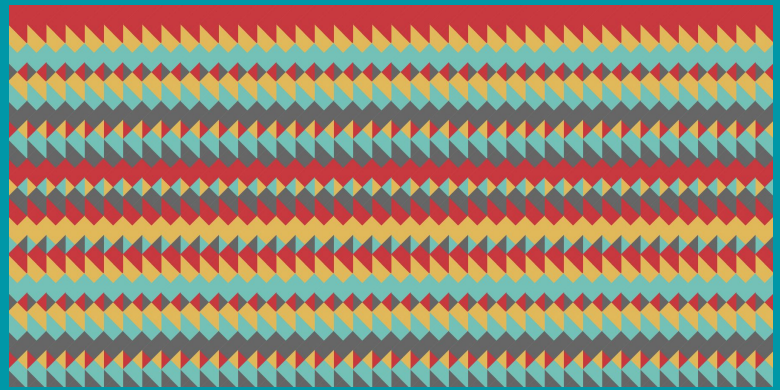


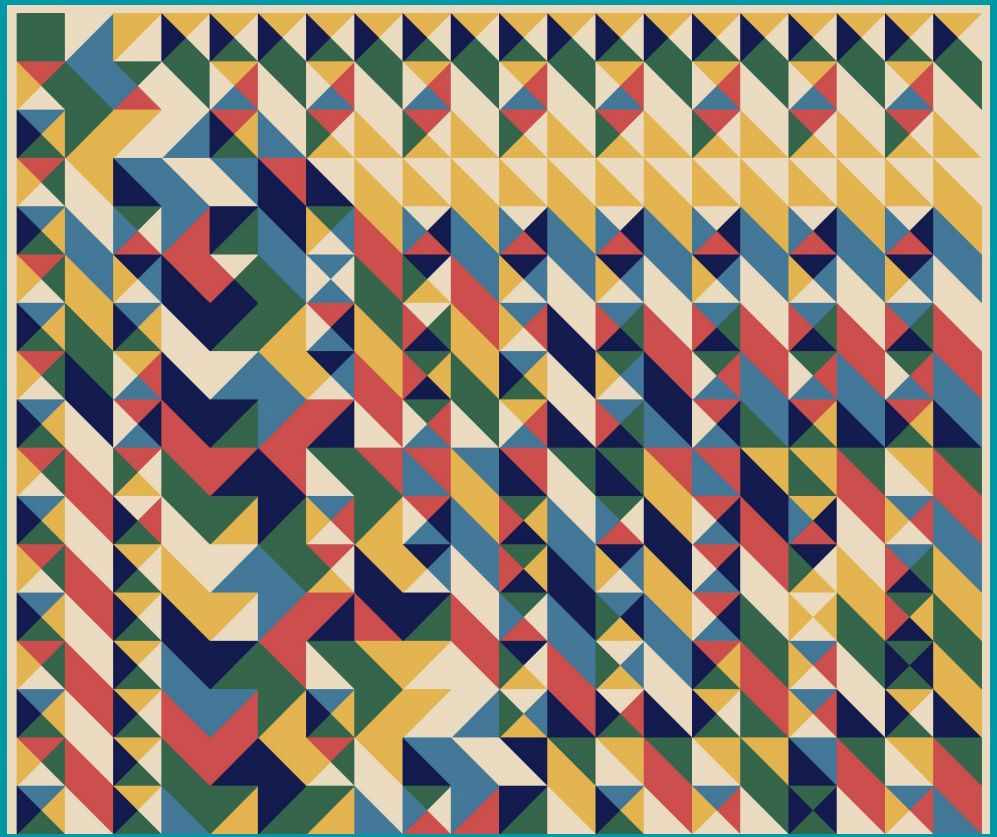
→ IMAGE, A-LIFE PROGRAM

Crosshatch Automata

Kjetil Golid, 2020

Isometric cellular automaton. Four rules makes up a pattern, each rule determining the behaviour of one line direction (horizontal, vertical, descending- and ascending diagonally).



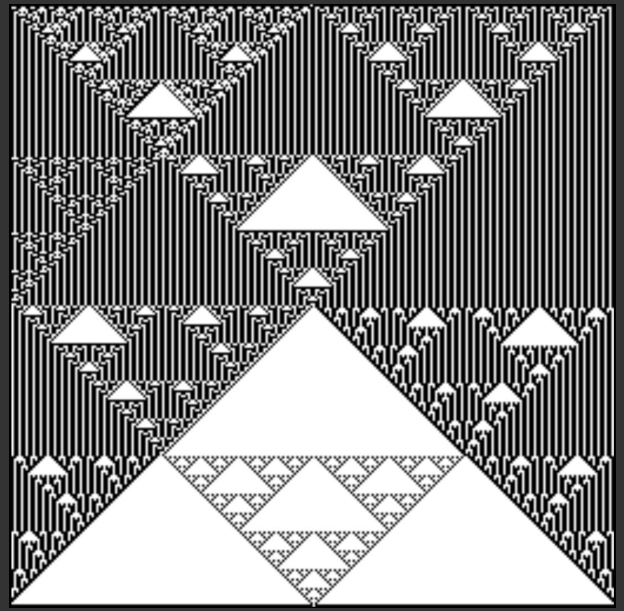


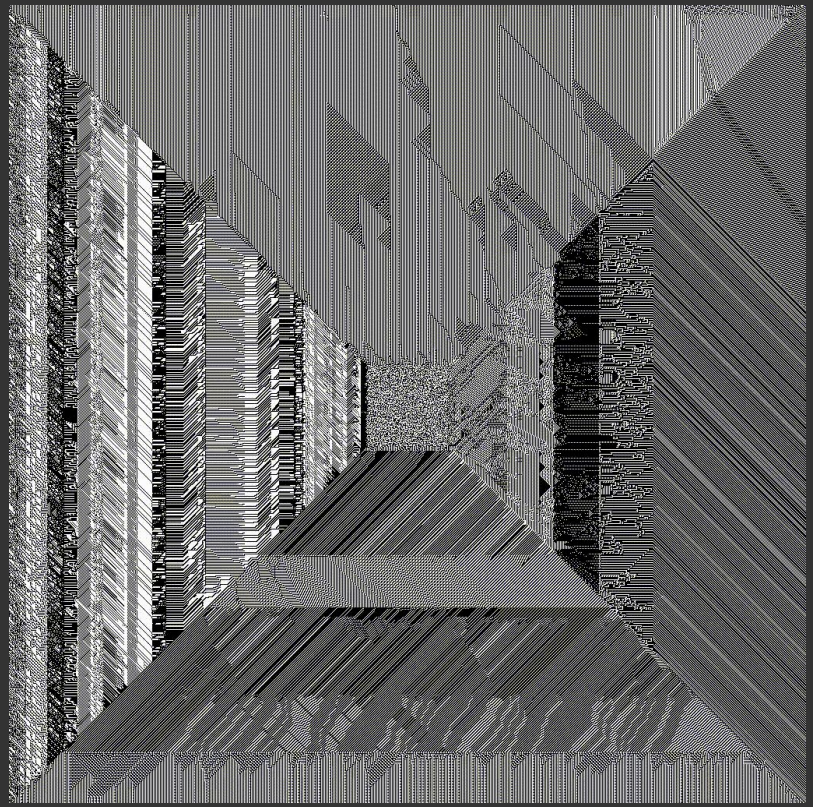
https://generated.space/sketch/crosshatch-automata/

→ IMAGE, PROJECTION

Roko’s heart

Santiago .



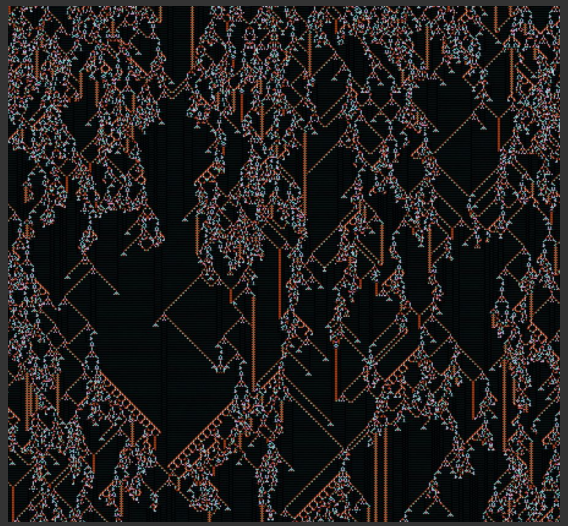


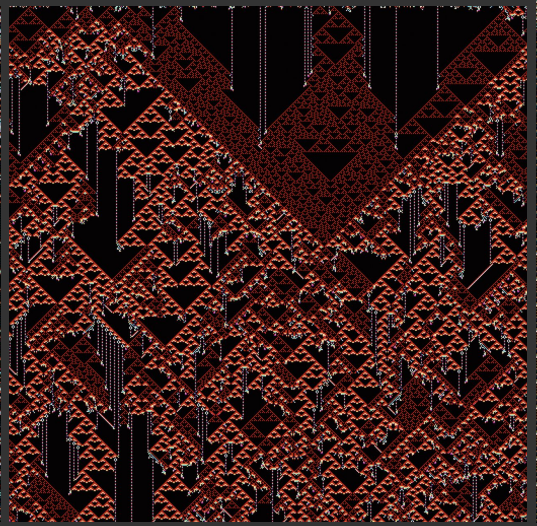
Project + Code ->

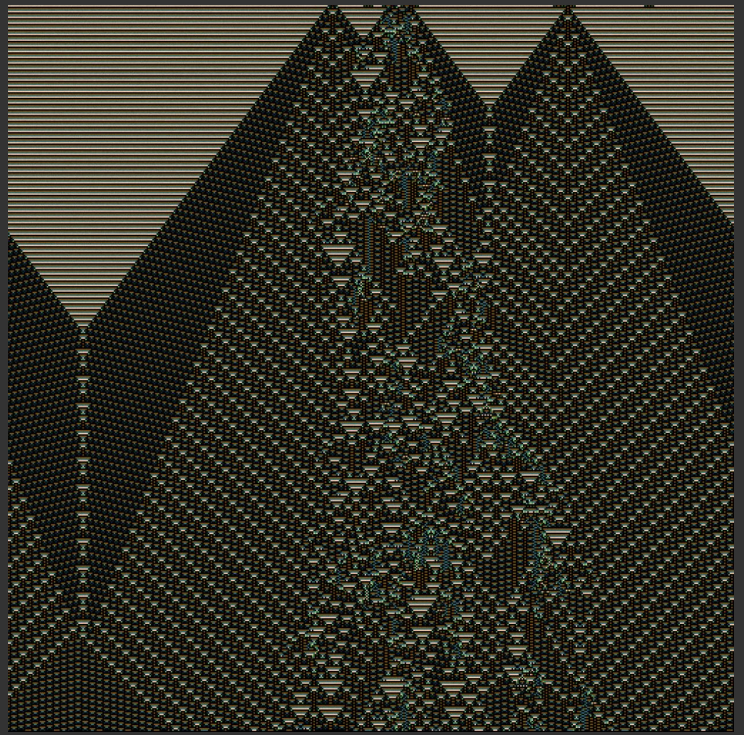
→ IMAGE

Edge of chaos

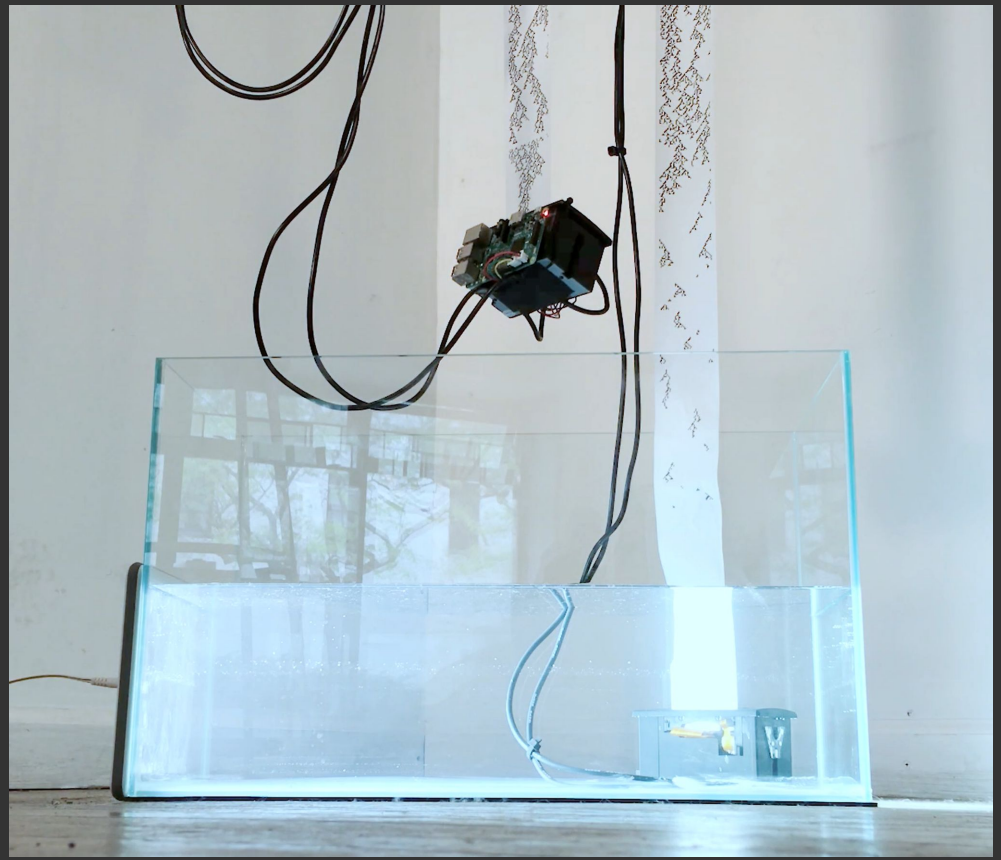
Tim Zarki, 2022







Behance



→ INTERACTIVE INSTALLATION

The Emergence and Decay of Computation

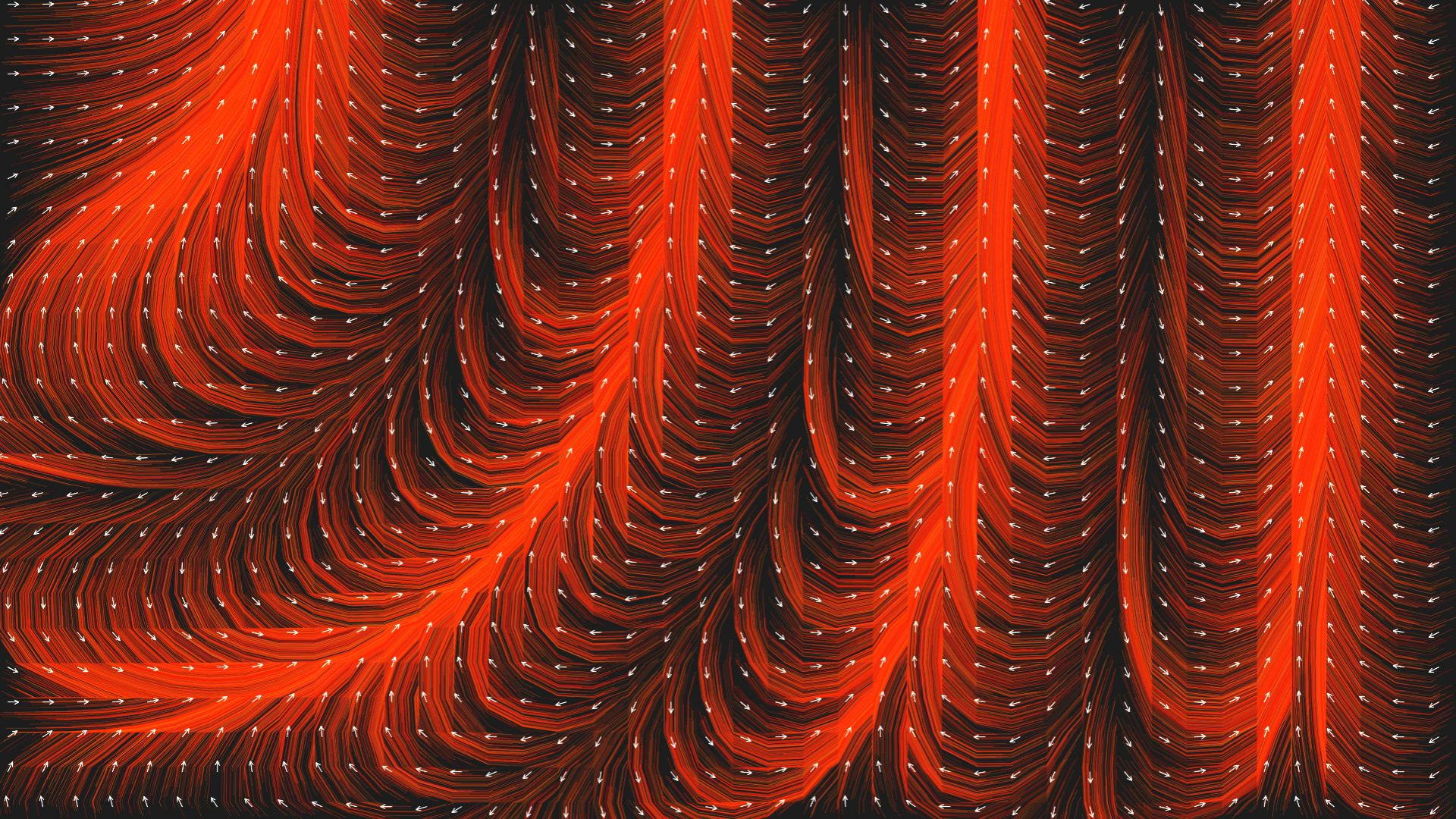
Alexander Miller, 2019

Receipt printers that hang from their own receipts above a basin of water and slowly print themselves to death over the course of three days. The printers output rows of a cellular automaton — a mathematical simulation that generates emergent patterns from simple rules. Entering the water short circuits the printer, permanently killing it. The output of the cellular automaton simulation is directly responsible for the destruction of the device computing the simulation.

SFPC student showcase 2019

Flowfields

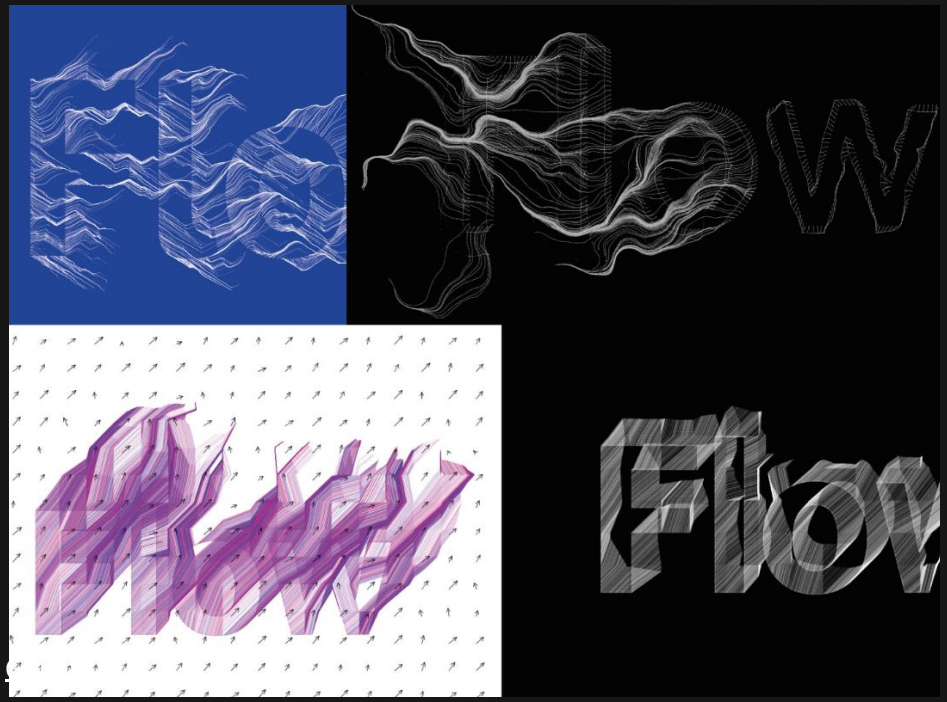
Explanation in Nature of Code, video series

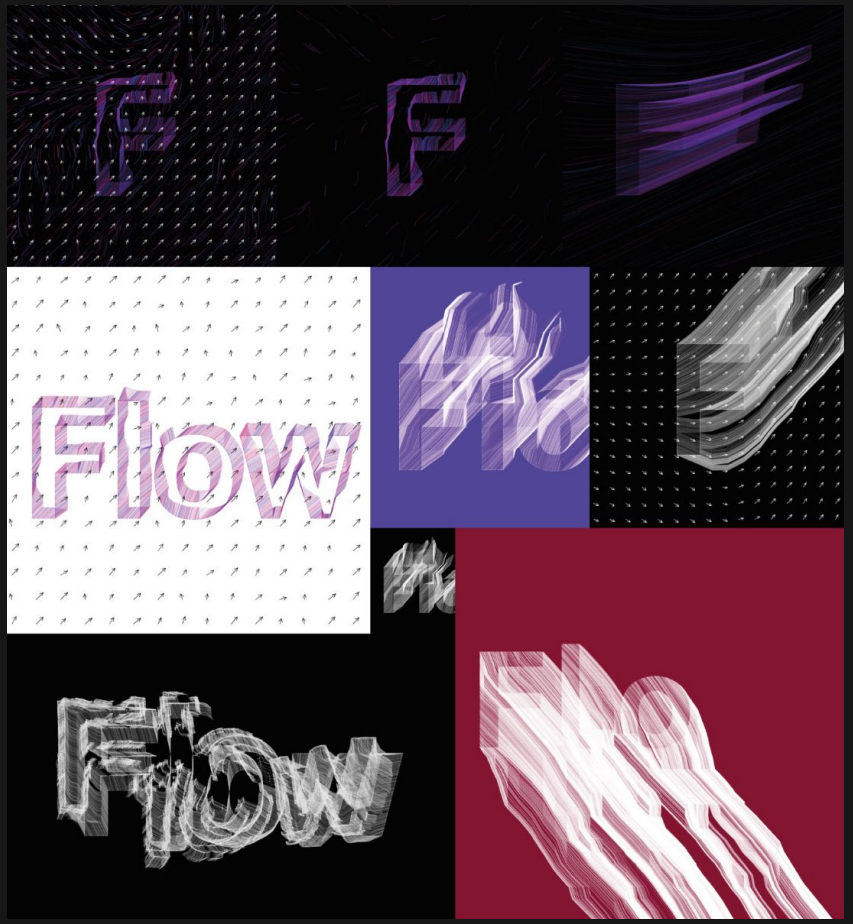


→ INTERACTIVE APP

Typoflow

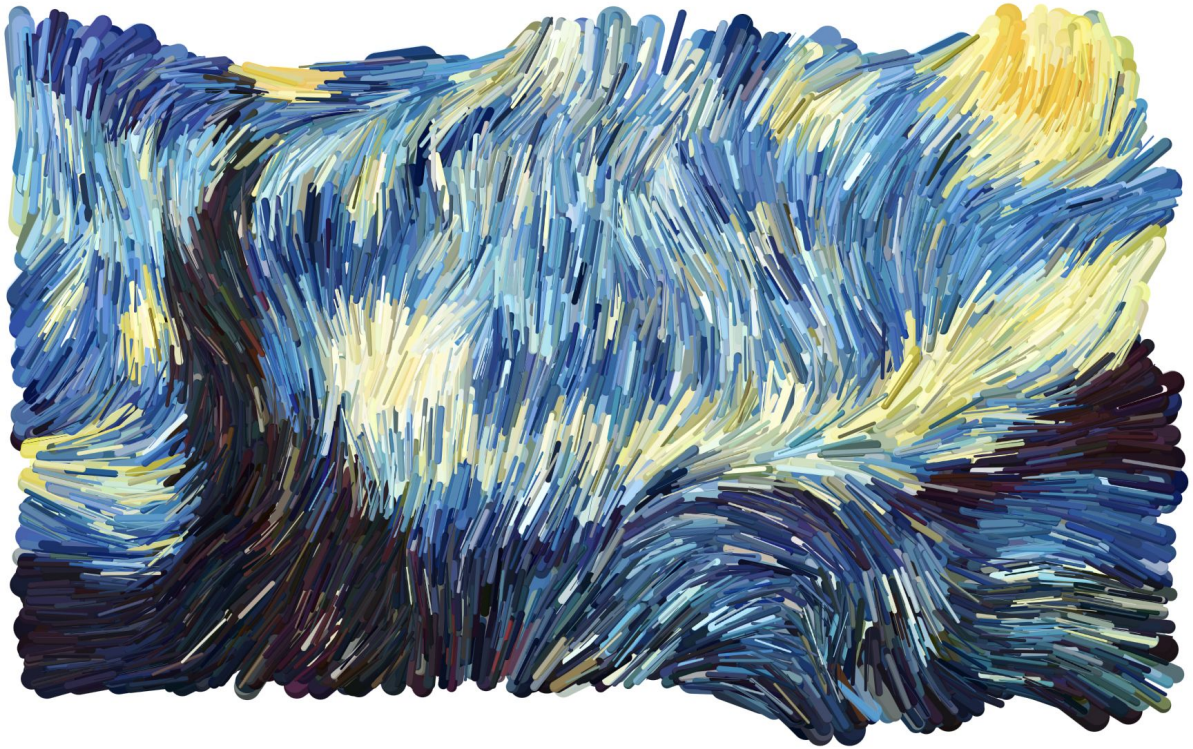
Marko Řeháček, 2019





Tyler Hobbs — Loxodography 0.26

Tyler Hobbs — Loxodography 0.26



Jason Labe - Noise Flow Field Painter

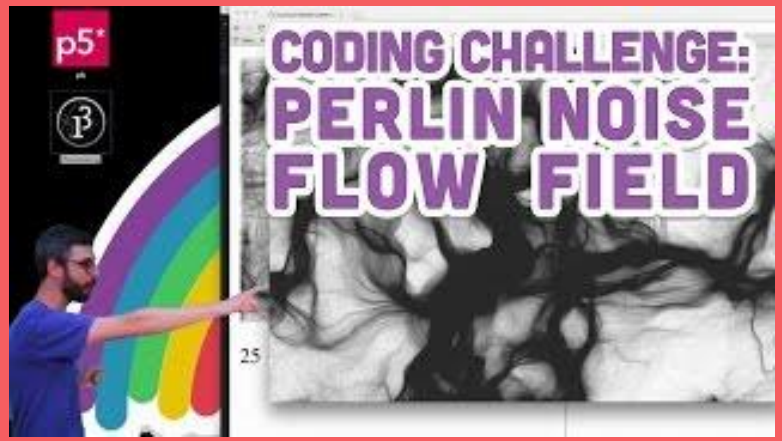
→

Daniel Shiffman

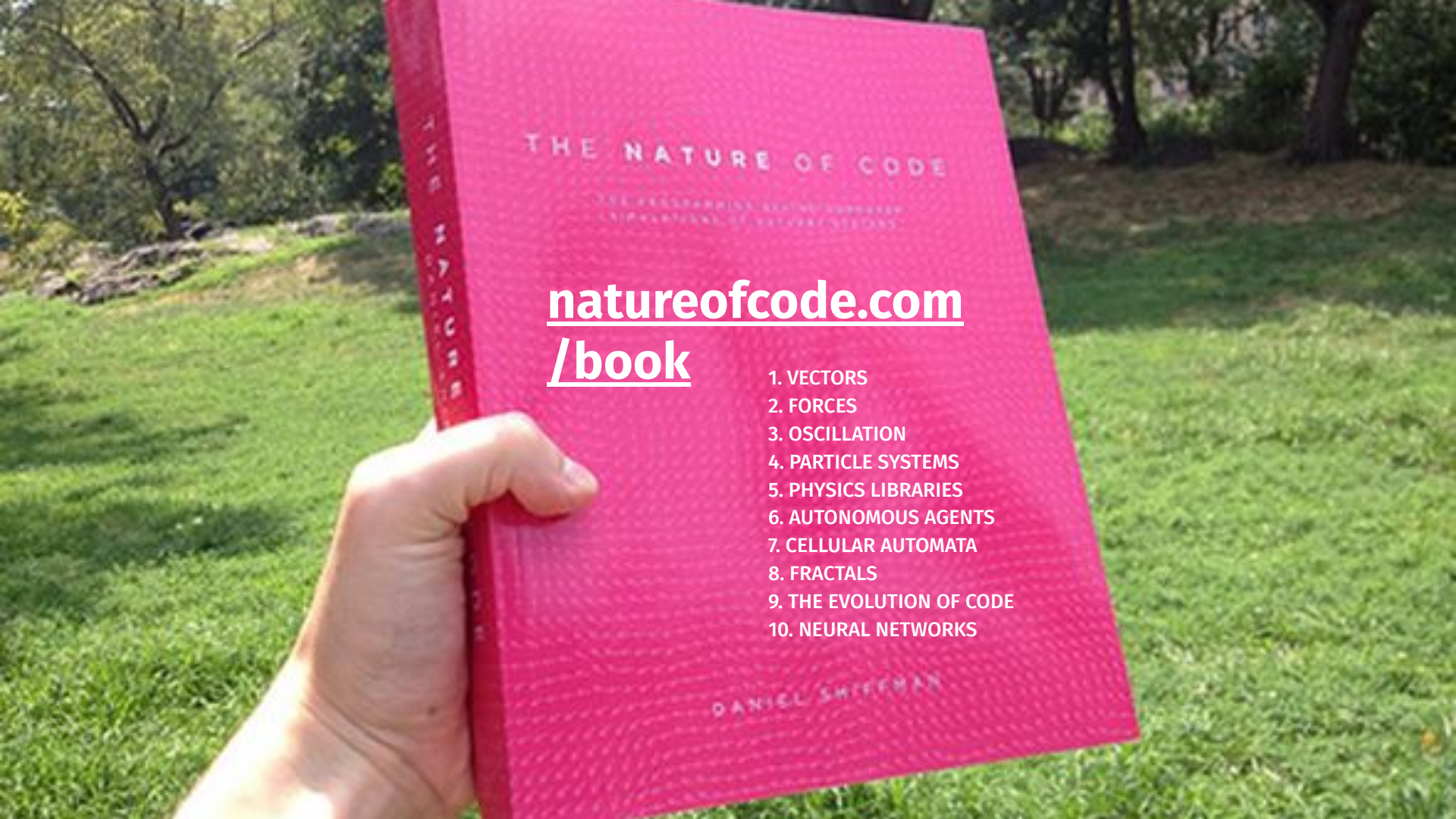
Flowfield

Create a grid of Perlin noise values. Move agent in the direction of the values (follow higher ones) or create more complicated movement.

Example from class



Other sources



natureofcode.com/book

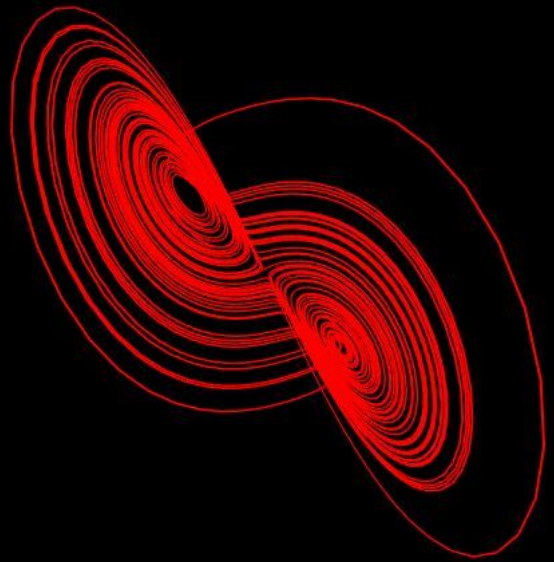
1. VECTORS
2. FORCES
3. OSCILLATION
4. PARTICLE SYSTEMS
5. PHYSICS LIBRARIES
6. AUTONOMOUS AGENTS
7. CELLULAR AUTOMATA
8. FRACTALS
9. THE EVOLUTION OF CODE
10. NEURAL NETWORKS

→

Lorenz attractor

System of ordinary differential equations first studied by mathematician and meteorologist Edward Lorenz.

https://editor.p5js.org/GenJamGuy/sketches/TUOEwXvty



https://en.wikipedia.org/wiki/Lorenz_system

Homework 3: MAS

Create artwork using the principles from this week's class. It can be a simulation of paint brush, a unique cellular automaton, or a dancing choreography of agents. If you incorporate ***emergent*** behavior or properties, that's great accomplishment!

Look in the slides for inspiration. The work needs to show skills learnt from last two weeks of classes and be more than was made during live-coding.

Technical information regarding the submission and deadline will be in interactive syllabus in IS.

Deadline 11.11.2024 (including).