PV259
Generative Design Programing

Week 4

Randomness & noise

MUNI FI

Marko Řeháček & Megi Kejstová
rehacek@mail.muni.cz

Between **order and chaos**

is a sweet spot we try to achieve.

Observed in a sequence, when order and patterns seem to be lacking.

A crucial element to the generative process.

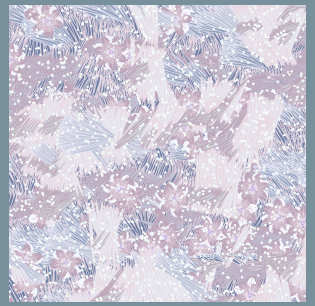
→ IMAGE

Fragments of an Infinite Field

Monica Rizzolli

Collection of 1024 generative artworks of “potentially infinite field of foliage” sold for 1,623 ether (around $5.38 million) through a sale on Art Blocks, the non-fungible token (NFT) platform. P5.js.

https://www.theblockcrypto.com/post/117605/art-blocks-hit-generative-artist-5-38-million



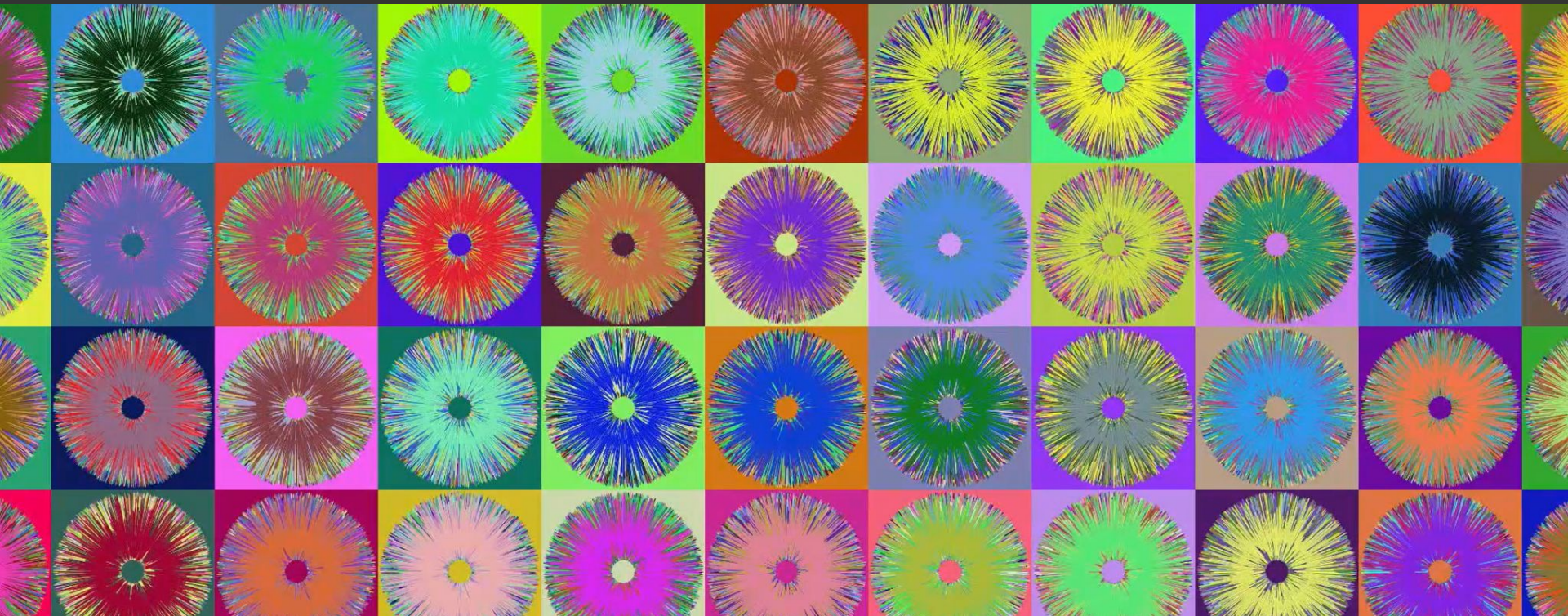
https://artblocks.io/project/159

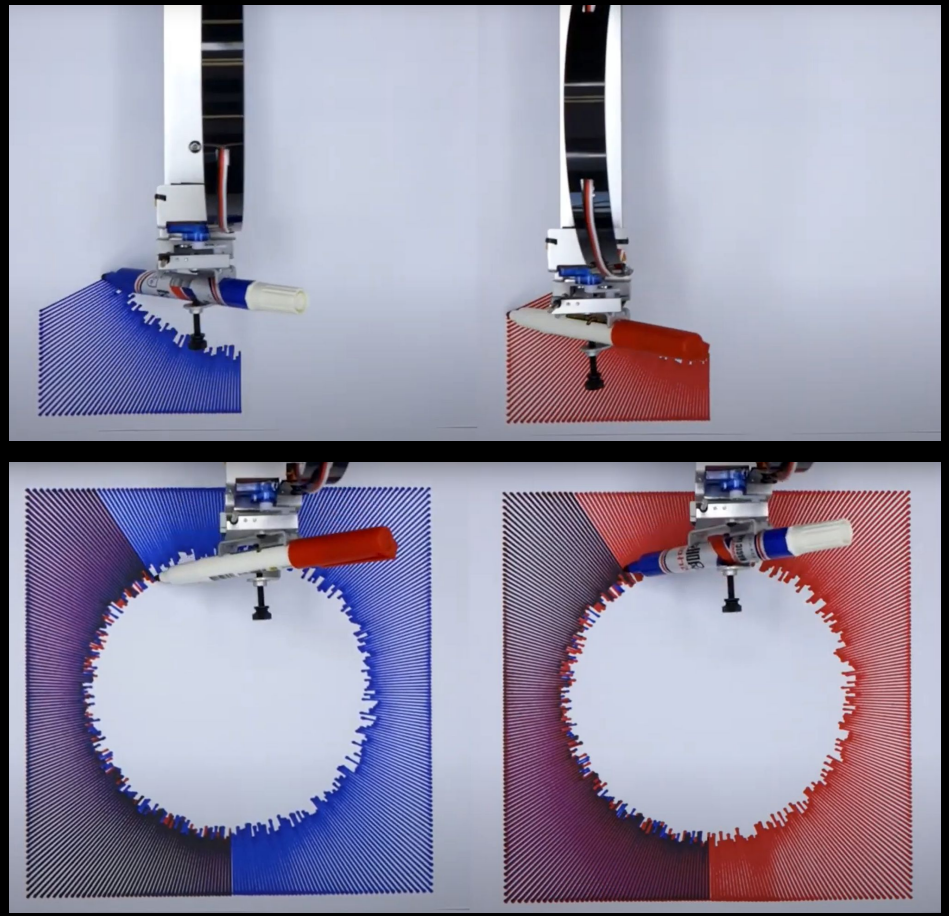
→ IMAGES

Generative circle

David Mrugala / thedotisblack

Watch the whole process.
https://www.youtube.com/watch?v=UZoVBMgzULk





→ IMAGE + MACHINE

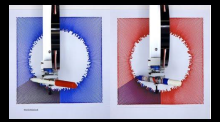
DRAWING MACHINE 04

David Mrugala / thedotisblack, 2014

This art and craft of a pen plotter is made with the AxiDraw V3/A3 drawing machine. The circle is made up by lines with randomized end points. Once the first drawing was drawn, the drawings and color pens switched and were drawn on top of the other drawing.

Pen plotter drawings have a unique quality that no algorithm can recreate - it's the interaction of the pen with the surface of the paper that creates unique and unexpected outcomes.

www.youtube.com/watch?v=sAWXLWsjEF4



PlasmaFractal

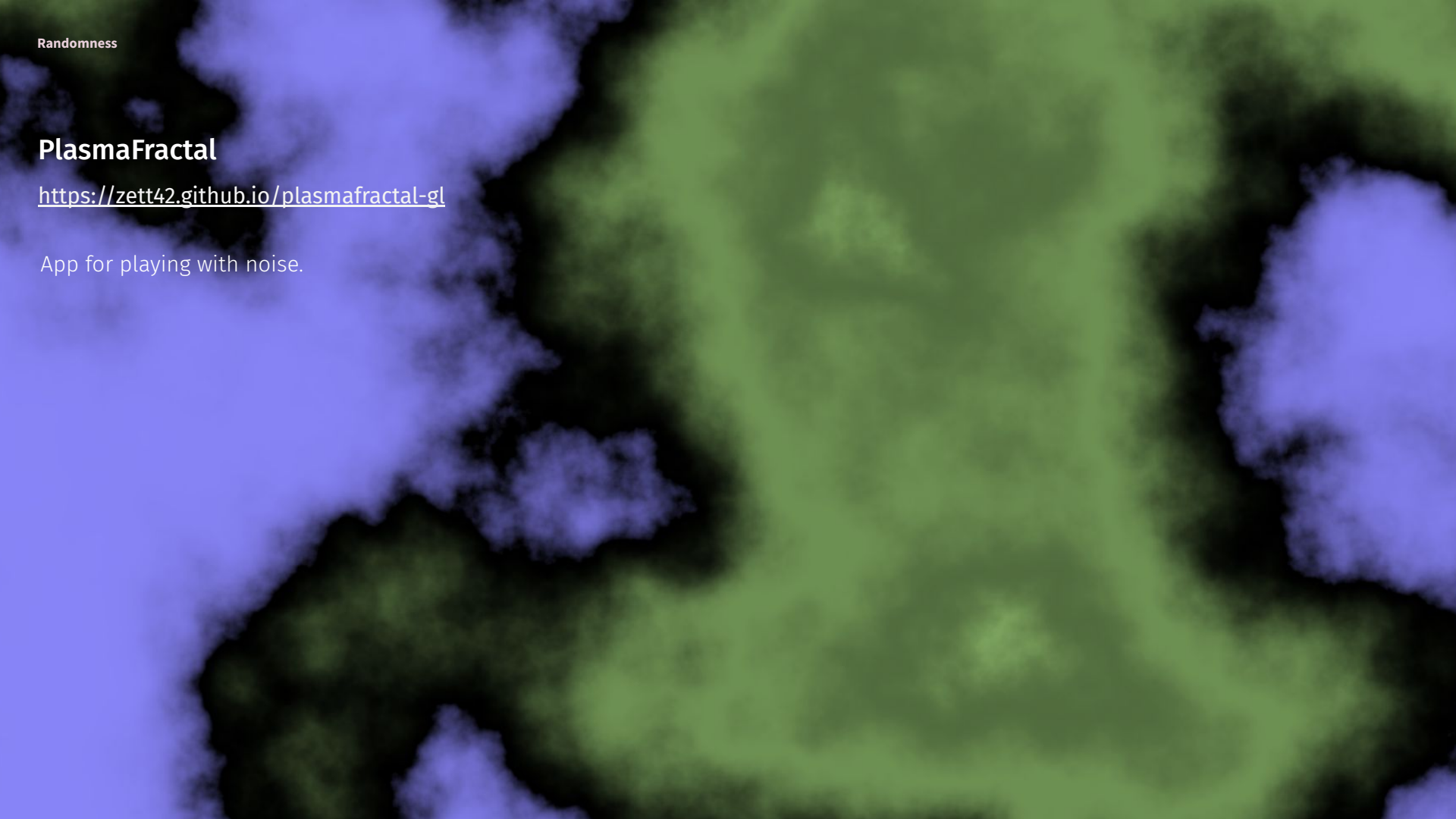
https://zett42.github.io/plasmafractal-gl

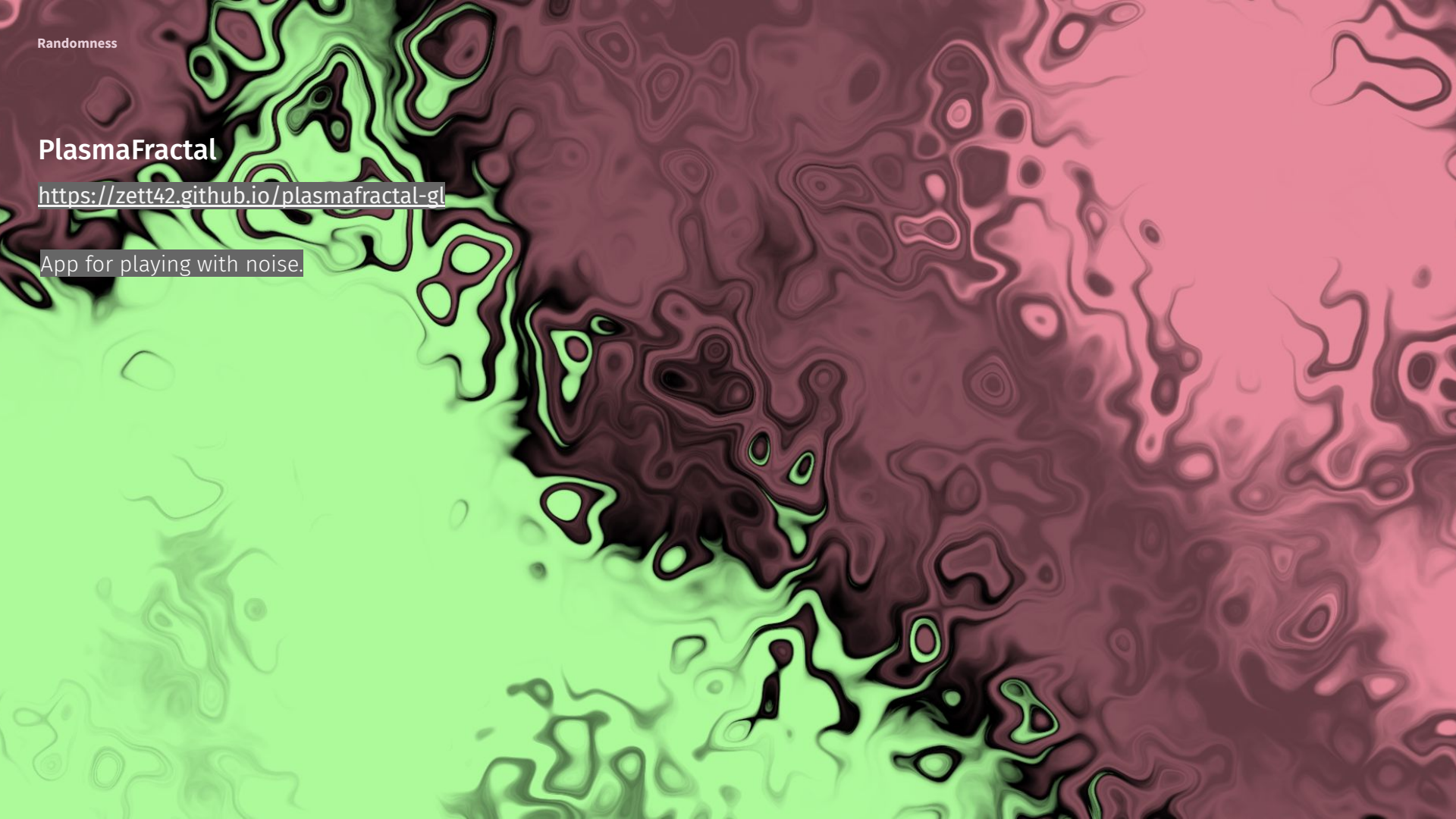
App for playing with noise.

PlasmaFractal

https://zett42.github.io/plasmafractal-gl

App for playing with noise.





PlasmaFractal

https://zett42.github.io/plasmafractal-gl

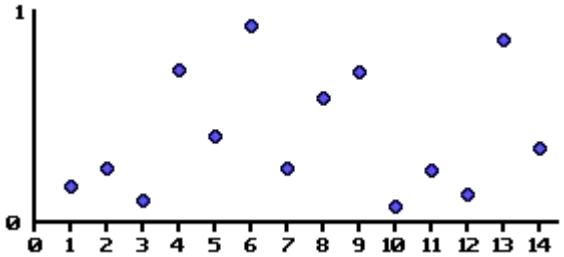
App for playing with noise.

Generating noise

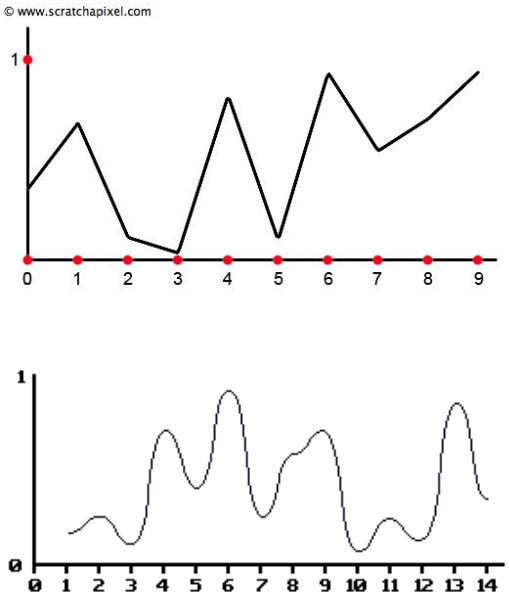
Coding Train

Noise functions

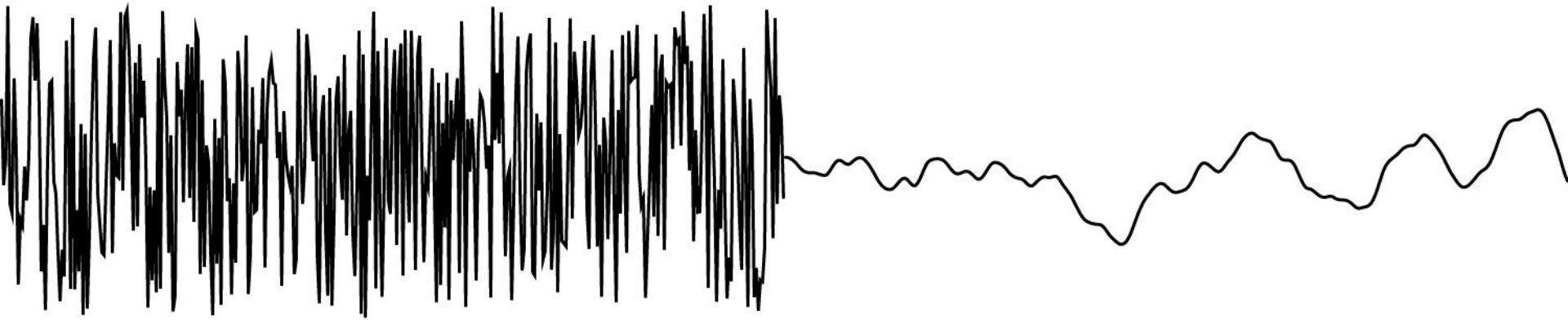
Generate sequence of random numbers



interpolate to get continuous function



Noise types



1. white noise – random numbers

- p5: ***random()***

code comparison

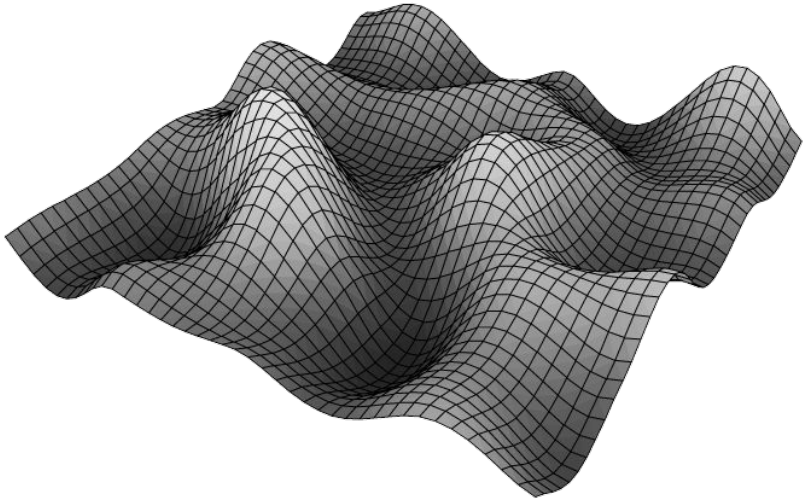
2. gradient noise

- successive random numbers are close to each other
- p5: ***noise()***

Perlin noise

Algorithm for generating gradient noise.

Invented by Ken Perlin in 1981 to break machine-like (solid-shaded) procedural textures for CGI in the film Tron.



P5 functions

Generate random numbers in range [min, max].

```
random(max)

random(min, max)

random(array)
// select random element
```

Generate random numbers in range [0, 1]. The produced numbers are more naturally ordered and close to each other.

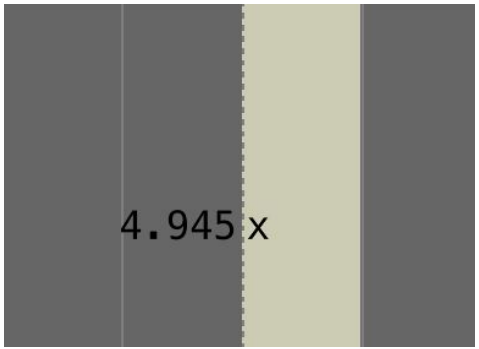
```
noise(x)
// 1-dimensional

noise(x, y)
// 2-dimensional

noise(x, y, z)
// 3-dimensional
```

p5 reference

Visualization of dimensions

We have many different ways to visualize different dimensions of noise

1D noise
One value



2D noise

2D visualization

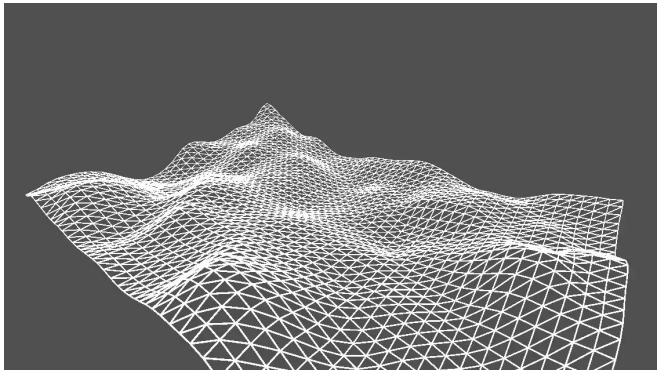
1D visualization
(second dimension is replaced by pixel colour)

p5 reference

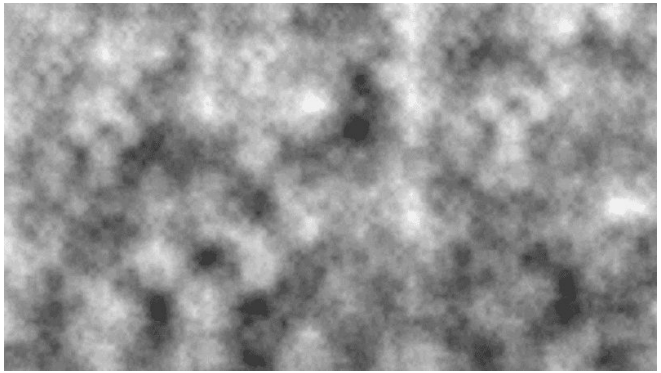
Visualization of dimensions

We have a many different ways to visualize different dimensions of noise

3D noise



3D visualization



2D visualization
(third dimension is replaced by pixel colour)

p5 reference

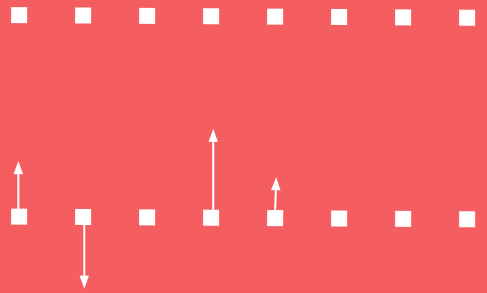
Sketch: squiggly line

1. draw points organized in a line

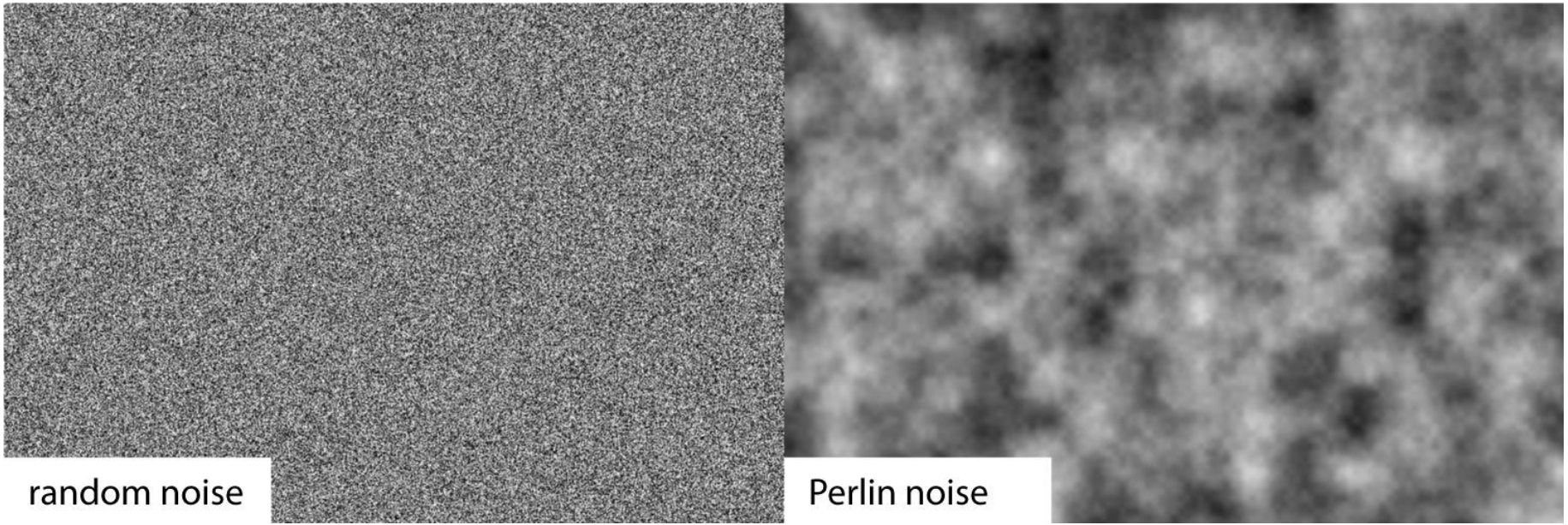
2. displace the points from the center line
   a. using **random( )**
   b. use **noise( )**
   c. try **millis( )** – milliseconds passed since start
   d. extra: use random like running sum (*random walk* technique)

3. create lines using beginShape-vertex-endShape

```
beginShape(POINTS | LINES)
     vertex( x, y )
endShape()

Create a complex shape by
connecting a series of vertices.
```



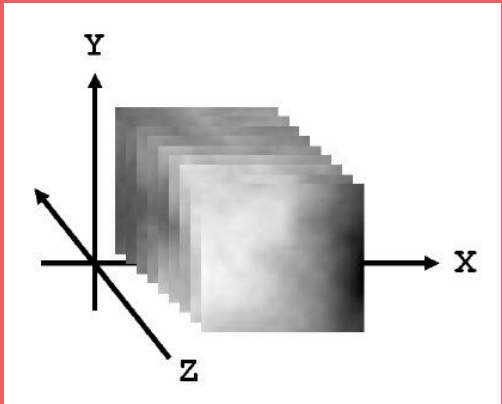
Code ->


random noise
Perlin noise

Noisy grid

1. make a grid of squares
2. use noise() function to color the squares
3. explore the noise "map"
   - scale the noise by scalar
   - offset x, y (panning)
   - hook the map coordinates to cursor position
   - at what level of zoom does the noise look random again?
   - are there any artifacts?
4. animate the noise using z-coordinate and time

Animation using time as a z-dimension, then slicing in xy plane. Picture from Classification of solid textures using 3D mask patterns, Suzuki et. al

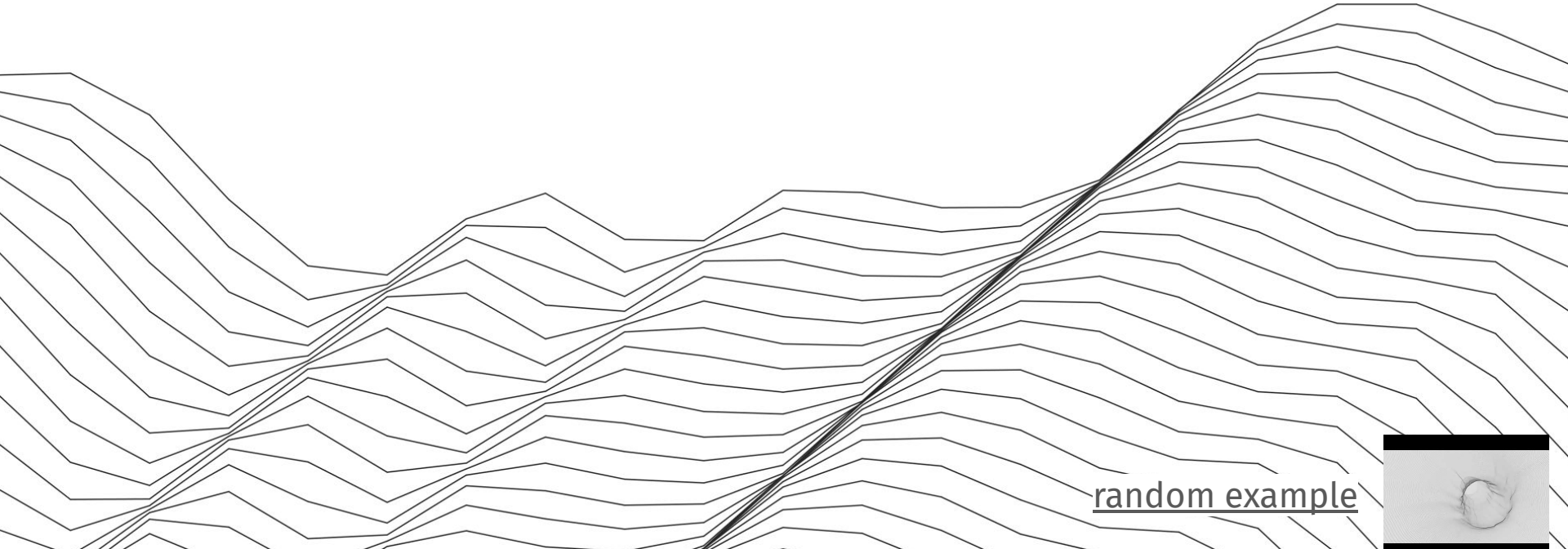


Code ->

How is it used?

- shape deformations
- illusion of a flow
- terrain generation

Picture from
Kristína Zákopčanová

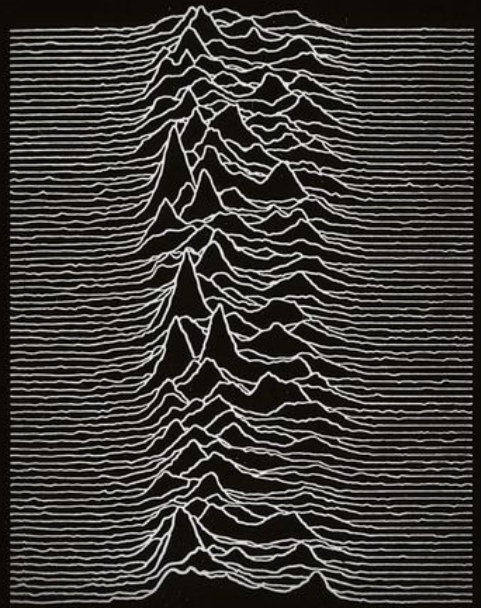


random example

Examples

→ ALBUM ART

Joy Division



Creative constraints

http://creative-constraints.surge.sh/#/en

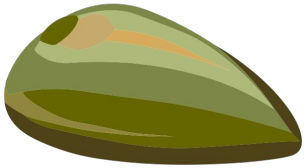
The grid

1. Create a 2D grid of shapes you like / open your old sketch
2. Add just enough chaos to maintain equilibrium with order

Ideas:
move shapes around, rotate them, scale them, choose random shape, color from some range, etc.

Functions:
random (min, max **)**
noise (x, y **)**
push(), pop(), translate(x,y**), rotate(**rad**), scale(**val**)**

More control



Set ***seed*** to always generate the same pseudo-random sequences

```
randomSeed(seed)
```

```
noiseSeed(seed)
```

Control the quality of the noise

```
noiseDetail(lod, falloff)
```

ref

Advanced stuff

- Perlin invented **Simplex noise** in 2001
  - improvement over Perlin noise in artefacts
  - patented, look for **OpenSimplex**
- libraries for noise generation (including OpenSimplex):

  JS/P5: https://github.com/josephg/noisejs

  https://github.com/jackunion/tooloud (not sure how fast)

  Java/Processing: https://github.com/KdotJPG/Noise-Extras
- great video about programming with noise: https://www.youtube.com/watch?v=CHZtK-keEvU

  which explains more advanced techniques, such as:
  - fractal noise - summing up noise functions (layering the noise)
  - turbulence noise
  - domain warping

HW1 reminder

Sketches from all classes available at
https://editor.p5js.org/mrehacek/collections/Y7yY_s7PN