

PV259

Generative Design Programming

Week 4

Randomness & noise

MUNI
FI

Marko Řeháček
rehacek@mail.muni.cz

Between **order and chaos**
is a sweet spot we try to achieve.



R a d n e S
n

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

o m s

A crucial element to the generative process.

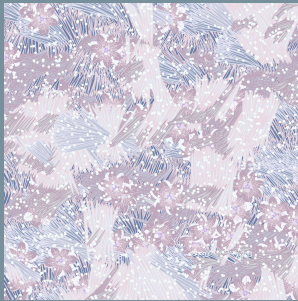
→ IMAGE

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>

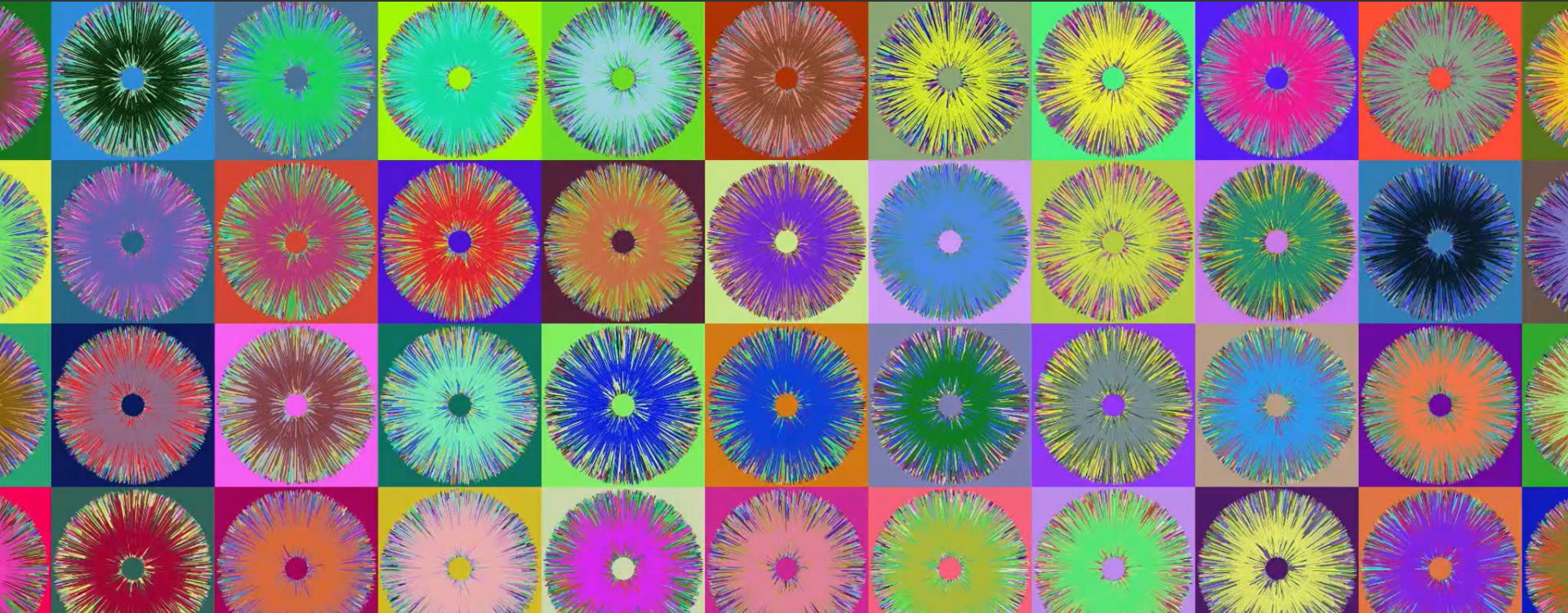
→ IMAGES

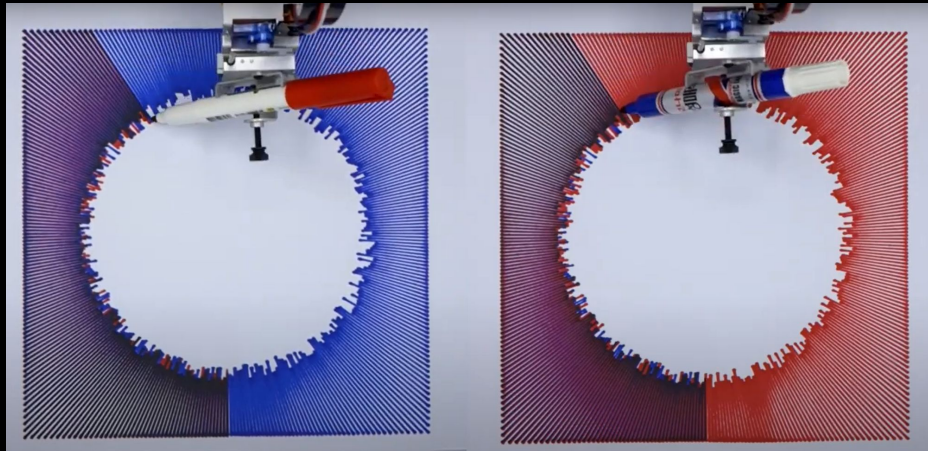
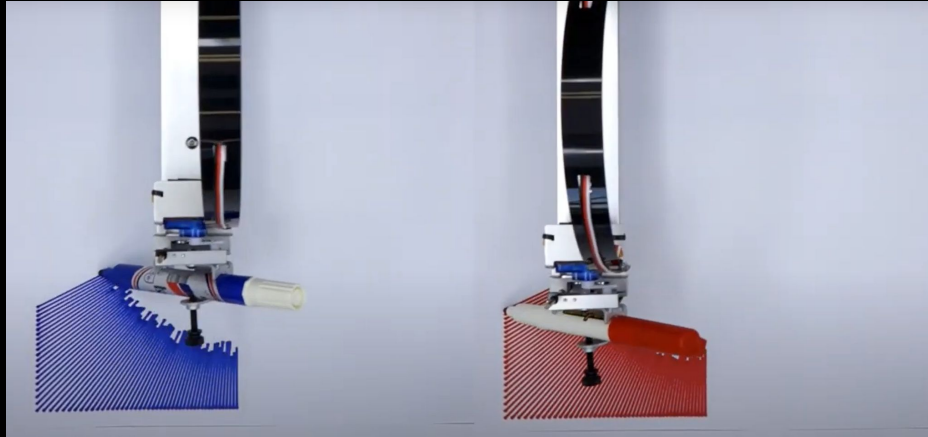
Generative circle

David Mrugala / thedotisblack

Watch the whole process.

<https://www.youtube.com/watch?v=UZoVBMezULk>





→ IMAGE + MACHINE

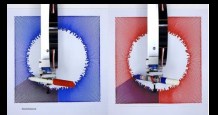
DRAWING MACHINE 04

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.

Randomness

PlasmaFractal

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

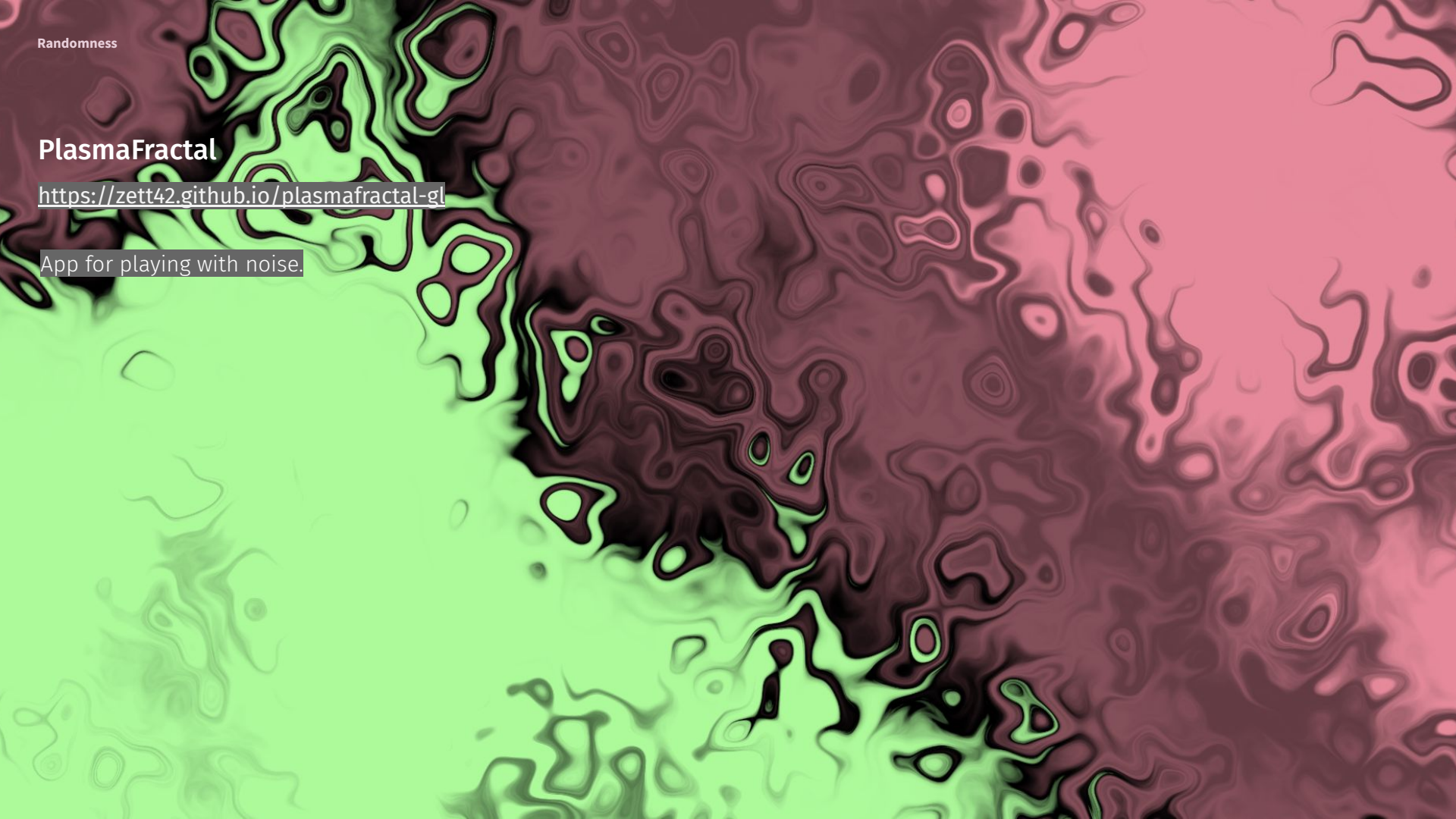
App for playing with noise.

Randomness

PlasmaFractal

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

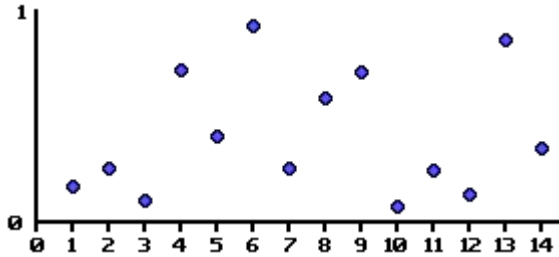
App for playing with noise.



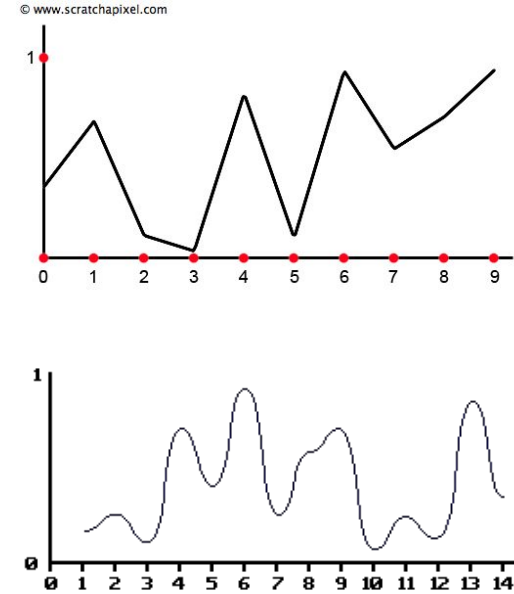
Generating noise

Noise functions

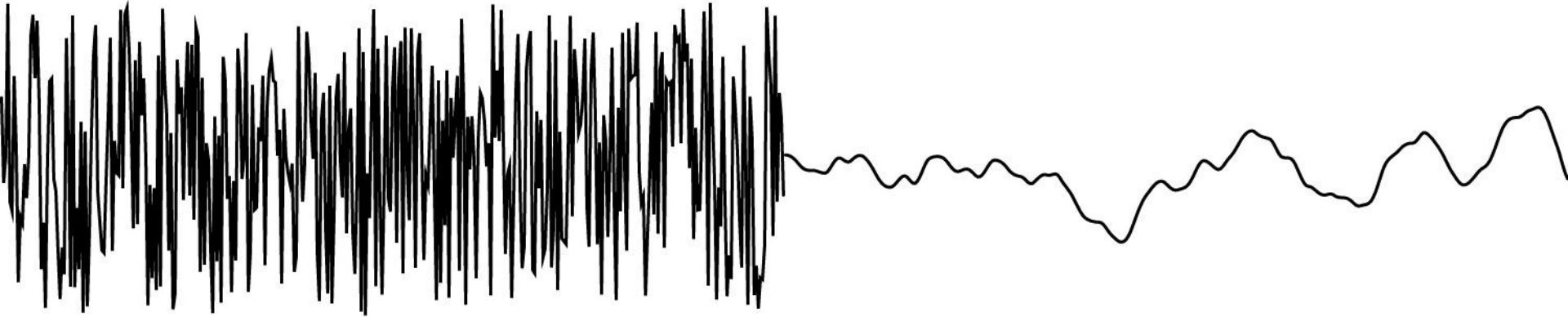
Generate sequence of random numbers



interpolate to get continuous function



Noise types



1. white noise – random numbers

- ❑ p5: *random()*

code comparison

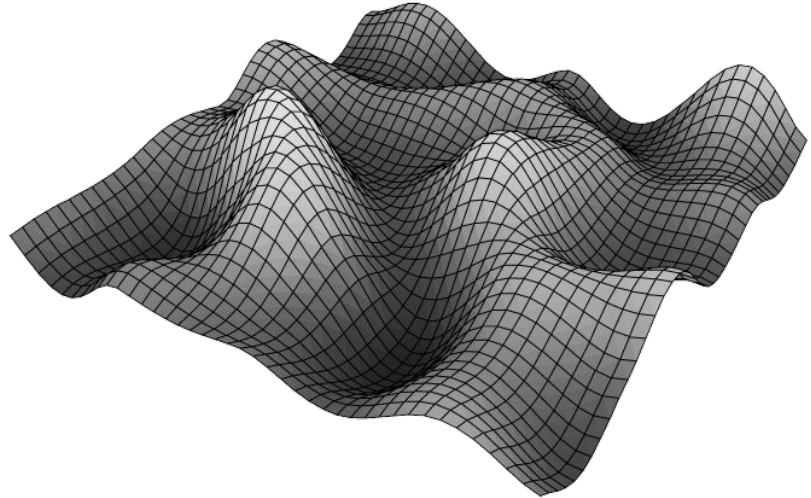
2. gradient noise

- ❑ 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)
```

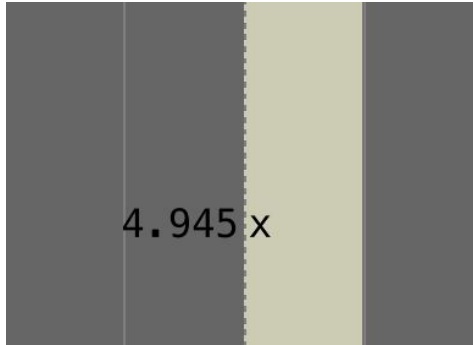
```
// 3-dimensional
```

Visualization of dimensions

We have many different ways to visualize different dimensions of noise

1D noise

One value



2D noise



2D visualization

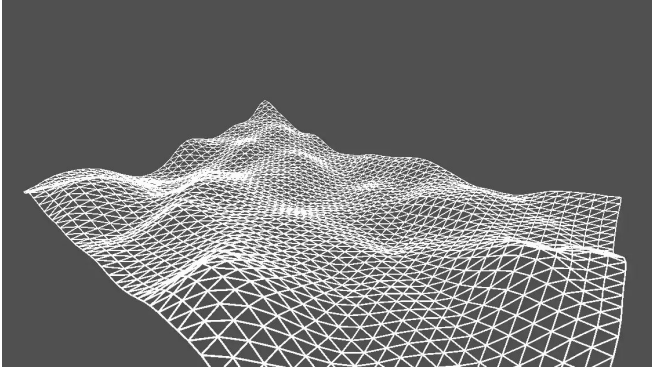
1D visualization
(*second dimension is replaced by pixel colour*)

[p5 reference](#)

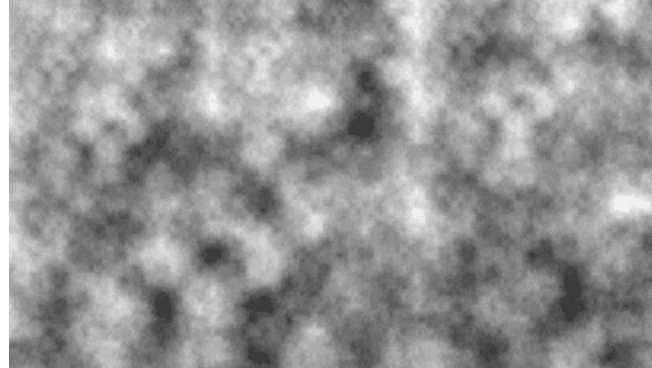
Visualization of dimensions

We have many different ways to visualize different dimensions of noise

3D noise



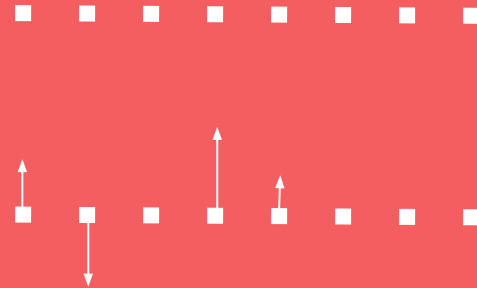
3D visualization



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

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()  
  vertex( x, y )  
endShape()
```

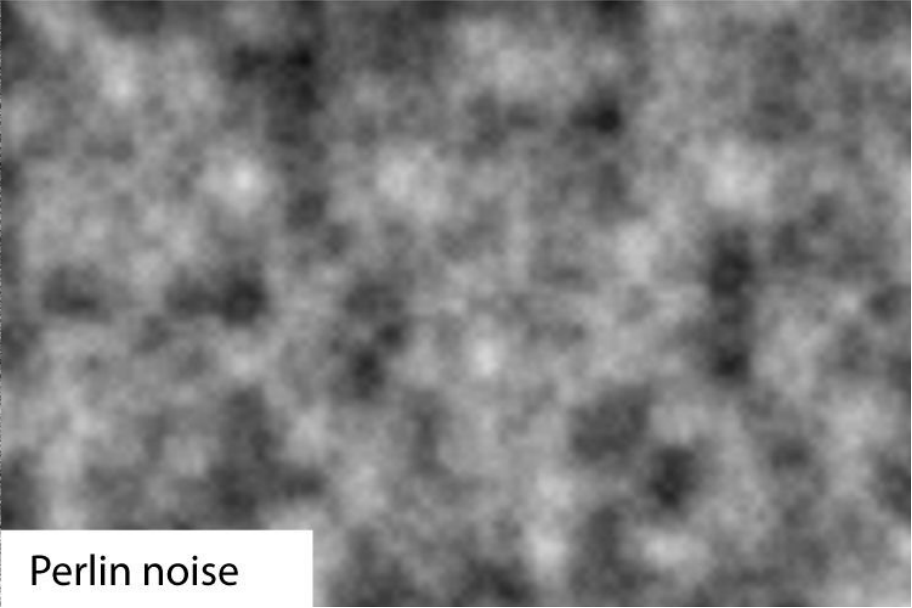
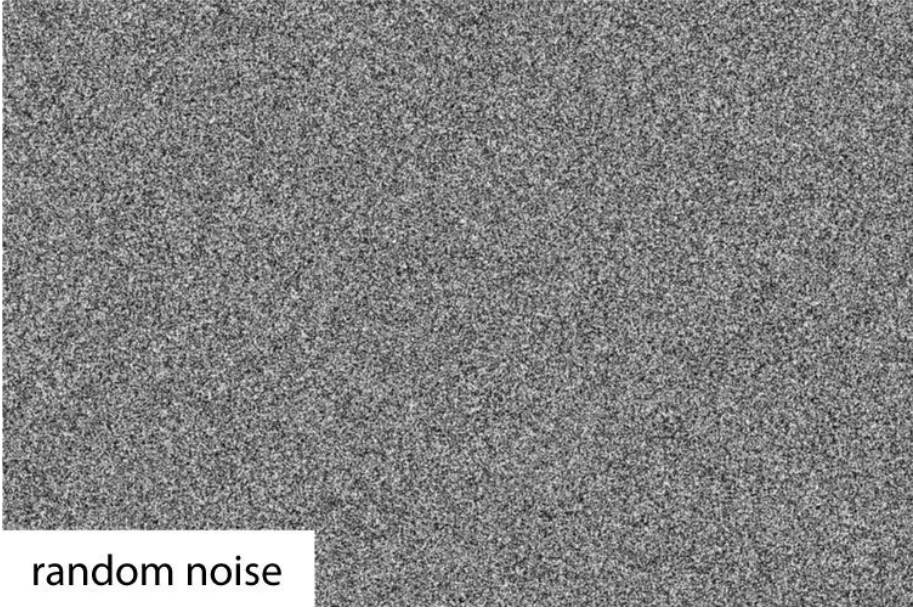
Create a complex shape by connecting a series of vertices.

Bonus: noise circles

[recommended for graphic designers]

1. Use the knowledge you have to create abstract noisy circles. Curves should be thin with a noticeable gap between them, they cannot cross.
2. You can choose from 2 variations (or do both):
 - a. black bg with white lines
 - b. white bg with black lines
3. Zoom in - resize the fingerprint so it will contain the entire canvas.
4. If done, call one of us. Then wait for a surprise ;)

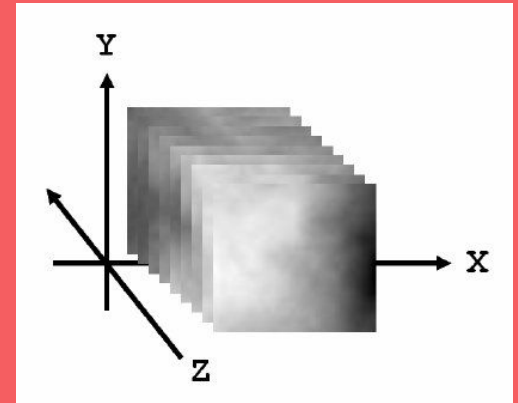




Noisy grid

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](#)

->

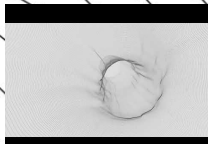


How is it used?

- ❑ shape deformations
- ❑ illusion of a flow
- ❑ terrain generation

Picture from
Kristína Zákopčanová

random example



Examples

A leap into graphic design

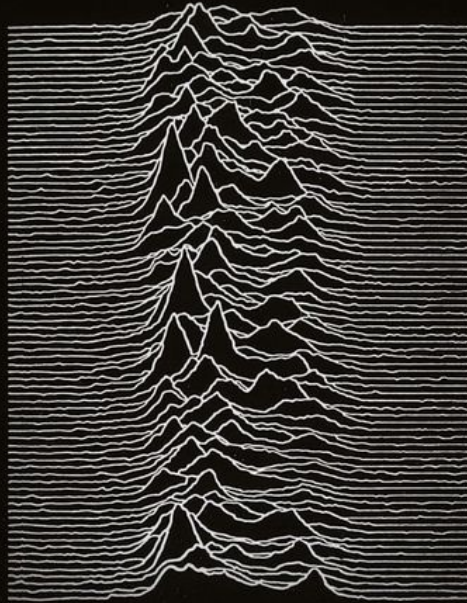


- ❑ notino manual:
https://cdn.notinoimg.com/files/1/logo-pdf/notino-visual-identity-manual_2022-02-28.pdf
- ❑ code for p5.js notino visual:
<https://editor.p5js.org/Berti/sketches/IQEFGGpZV>



→ ALBUM ART

Joy Division

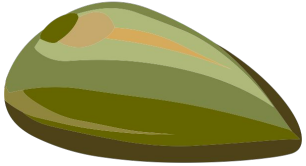


Think-pair-act

1. Go through the topics explained in this lecture. What technique interests you? Are you inspired by a particular example? Either write down or formulate in your head an outline of a simple composition, the only requirement is that it uses something we've learned today. (Example: "Moire patterns created with noisy shapes.") **3 minutes**
2. Find a partner. A person sitting next to you, or someone else, it's up to you.
3. Exchange ideas with your partner. Can your concepts fit together? Agree on what you'll create, either one of the original concepts or something else you come up with together. Consult the steps required to implement your composition. **2 minutes**
4. Create your artwork in pairs. One writes the code, the other observes, asks questions, looks up references, existing code snippets. Try to explain the intention of what you're doing to one another. **swap after 15 minutes** (we'll signal).
5. Depending on excess time, we'll repeat steps 3 and 4, giving you space to elaborate on your creations.

Creative constraints

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

Creative constraints

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

Sketches from all classes available at
<https://editor.p5js.org/mrehacek/collections/LLLIfXiAP>

Homework 2: Curved/noisy creations in motion

Stay abstract. Paint using curves or other complex shapes. Use Perlin or other interesting noise to introduce aesthetic randomization, explore its effects. Put everything in motion, slow and barely visible, or fast.

Look in the slides for inspiration: landscapes from perlin noise, some blob artistry, explore interesting mathematical concepts like Max Cooper. Look for some projects on instagram or the libraries we've shown you. Sky's the limit.

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.

