

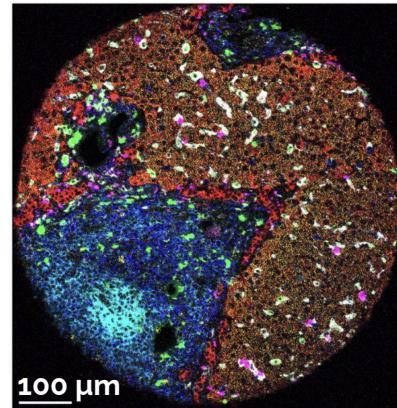
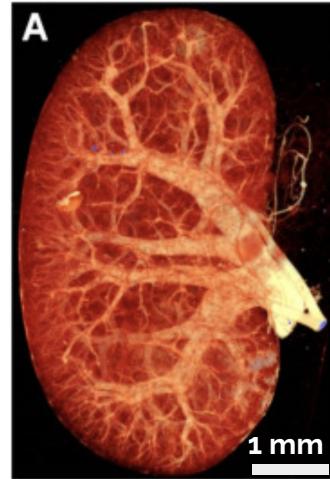
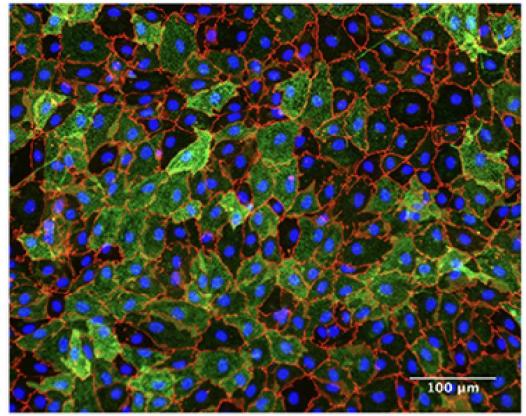
# AICP: Image module

## Day 3

Guillaume Witz  
Data Science Lab, University of Bern

**DSL**

My usual world...



Starting from left, clockwise:

Castro Dias M. et al. *J Cell Sci* 2021

Odriozola A. et al. *bioRxiv* 2017

Hlushchuk R. et al. *Vasc. Pharmacol.* 2019

Mi H. et al. *Front. in Immun.* 2022

Knüsel S. et al. *Mol Microbiol.* 2022

# Program: the technical aspect

## Overall goals

- Get an intuition of how models actually work, focusing on image models
- Get access to other model types and mix models, processing etc.
- Learn how to automate tasks

## Content

- Images and algorithmic art
  - Handling and processing images with code
  - Algorithmic image and animation generation (Python, p5.js)
- Deep Learning
  - Deep Learning concepts for images, example with convolutional networks
  - Applications of DL in Computer Vision (segmentation, tracking etc)
  - Running models from the API

# Generative Art

**Generative art** is post-conceptual art that has been created (in whole or in part) with the use of an autonomous system. An *autonomous system* in this context is generally one that is non-human and can independently determine features of an artwork that would otherwise require decisions made directly by the artist. In some cases the human creator may claim that the generative system represents their own artistic idea, and in others that the system takes on the role of the creator.

"Generative art" often refers to algorithmic art (algorithmically determined computer generated artwork) and synthetic media (general term for any algorithmically generated media), but artists can also make generative art using systems of chemistry, biology, mechanics and robotics, smart materials, manual randomization, mathematics, data mapping, symmetry, and tiling.

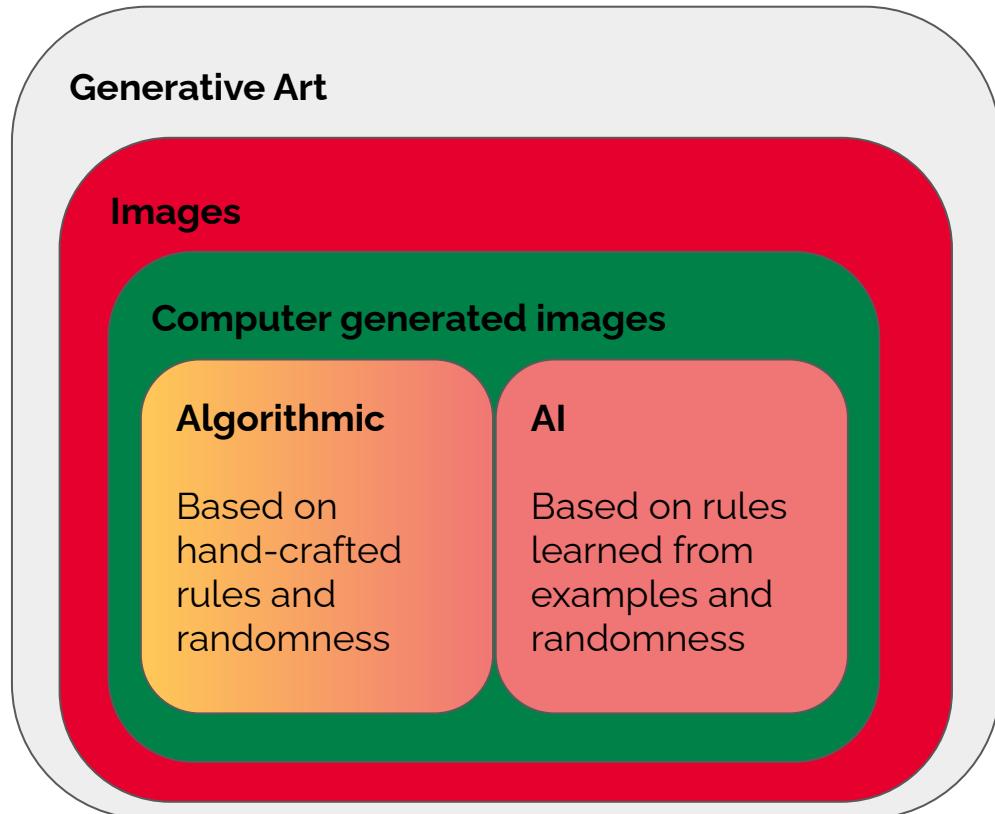
[https://en.wikipedia.org/wiki/Generative\\_art](https://en.wikipedia.org/wiki/Generative_art)

# Generative Art

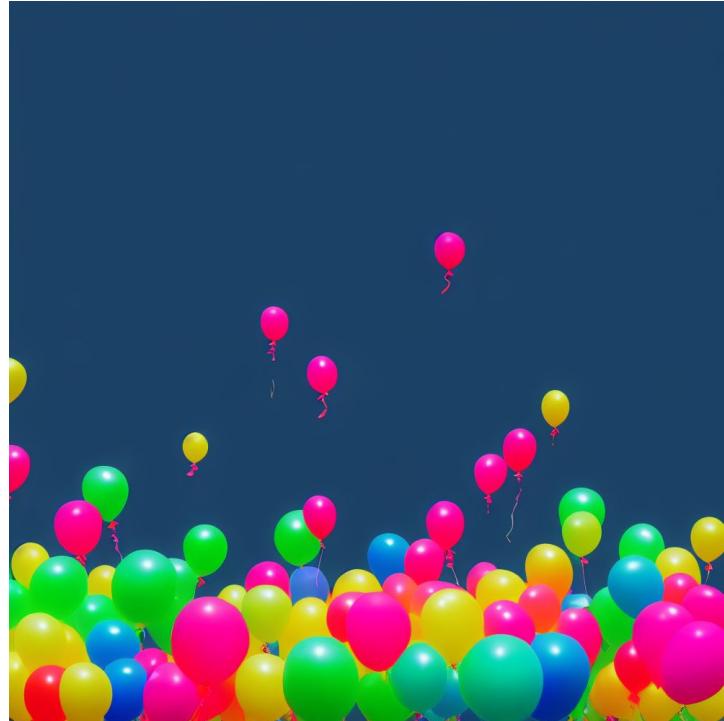
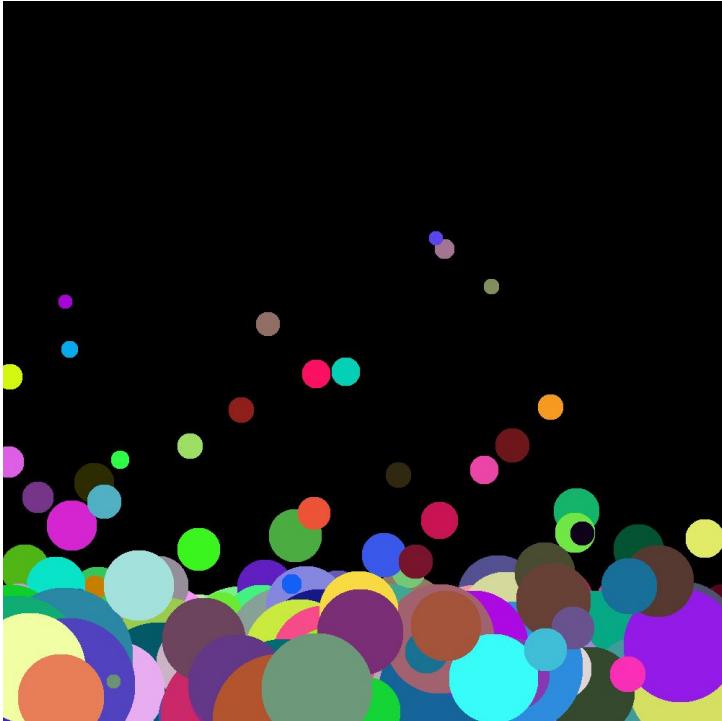
Generative art refers to any art practice where the artist uses a system, such as a set of natural language rules, a computer program, a machine, or other procedural invention, which is set into motion with **some degree of autonomy** contributing to or resulting in a completed work of art.

What is Generative Art? [Complexity Theory as a Context for Art Theory](#), Philip Galanter

# Generative Art: images

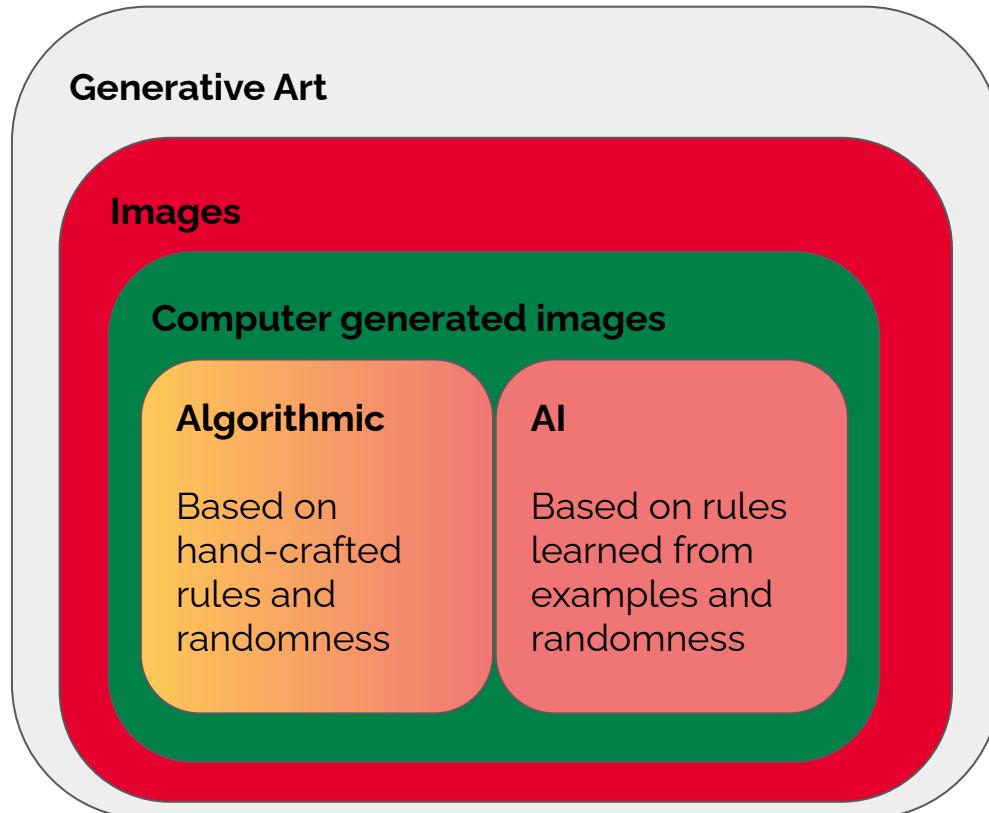


# "Classical" and AI generative art interaction



Pix2Pix model: "Turn into balloons"

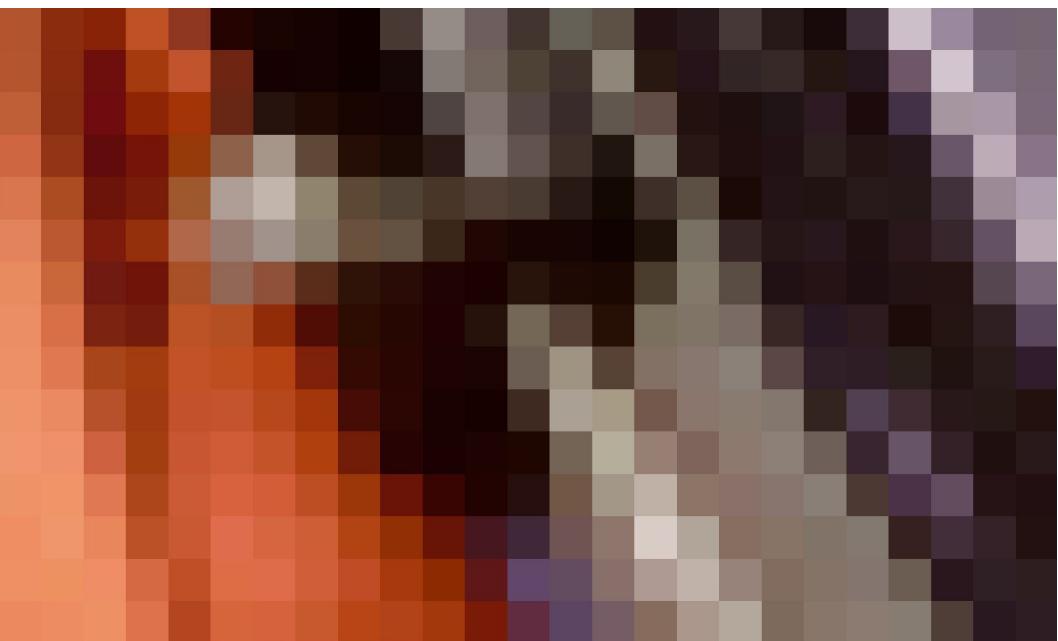
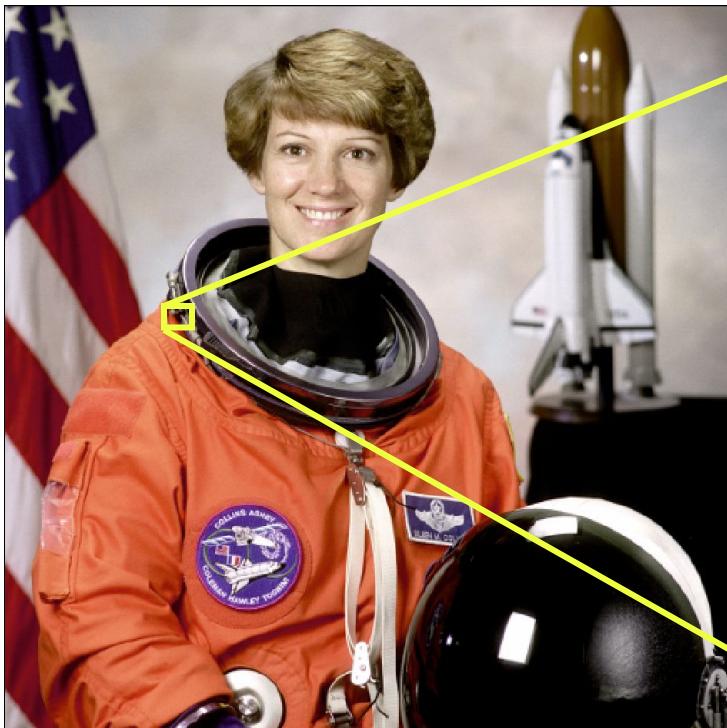
# Generative Art: images



Before or after generating images, we often need to do some image handling and processing!

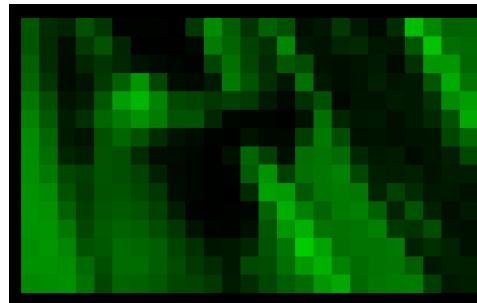
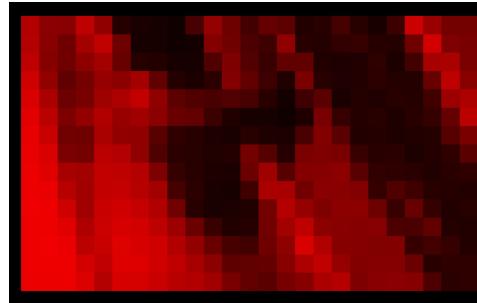
# What are digital images?

# What is an image?

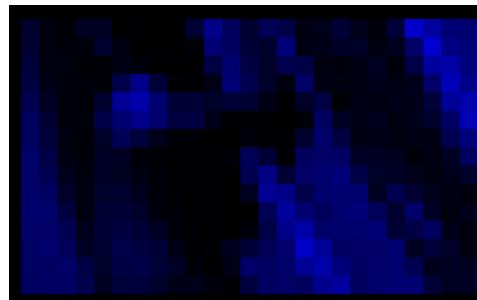


An image is just a 2D list of pixels!

# What is an image?



A natural image is composed of three 2D lists of pixels for red, green and blue.



# What is an image?

178	139	137	190	144	36	26	22	17	73	149	110	66	102	93	34	41	70	39	24	61	205	154	116	117
179	137	110	164	193	110	23	22	16	21	132	113	78	63	144	41	37	50	54	38	36	111	210	126	120
191	134	112	142	163	104	39	34	24	22	80	127	83	57	98	97	34	30	32	46	28	67	166	169	122
206	146	97	118	150	141	165	96	36	28	44	132	98	62	33	122	41	32	34	44	36	37	105	189	137
218	171	109	121	159	175	194	145	92	80	72	82	73	40	20	62	91	27	36	32	40	38	64	156	173
227	187	125	148	177	152	161	139	105	99	59	33	24	22	16	31	121	54	38	41	31	41	55	100	187
232	201	113	112	169	147	144	89	48	41	31	27	41	31	28	74	131	90	34	37	30	35	36	86	122
235	216	124	117	187	179	145	79	45	39	33	38	116	86	38	123	128	122	57	40	45	29	36	48	91
237	224	168	163	194	190	181	127	53	41	31	28	108	159	88	132	135	140	91	49	47	43	32	40	48
239	234	183	161	195	196	183	163	71	44	27	23	63	171	168	115	138	137	132	52	80	64	40	38	34
240	238	206	162	201	208	197	176	113	40	27	29	32	116	181	151	126	141	141	110	57	103	52	32	41
238	239	224	174	203	215	210	189	156	106	56	34	38	114	163	191	140	137	135	137	77	75	98	37	33
238	240	234	187	201	223	214	206	178	147	104	70	65	112	143	217	177	135	135	129	131	54	66	53	35
238	238	238	212	192	222	222	208	192	168	141	95	98	100	136	175	193	151	130	133	133	107	41	47	46
236	237	237	221	179	214	213	200	185	178	162	121	98	91	118	134	172	180	126	136	139	135	78	42	46

Each pixel just represents a number. We can choose any color to represent those numbers!

# What is an image?

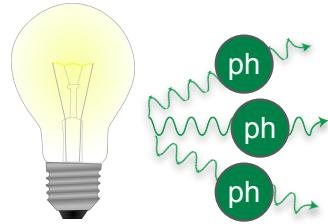
178	139	137	190	144	36	26	22	17	73	149	110	66	102	93	34	41	70	39	24	61	205	154	116	117
179	137	110	164	193	110	23	22	16	21	132	113	78	63	144	41	37	50	54	38	36	111	210	126	120
191	134	112	142	163	104	39	34	24	22	80	127	83	57	98	97	34	30	32	46	28	67	166	169	122
206	146	97	118	150	141	165	96	36	28	44	132	98	62	33	122	41	32	34	44	36	37	105	189	137
218	171	109	121	159	175	194	145	92	80	72	82	73	40	20	62	91	27	36	32	40	38	64	156	173
227	187	125	148	177	152	161	139	105	99	59	33	24	22	16	31	121	54	38	41	31	41	55	100	187
232	201	113	112	169	147	144	89	48	41	31	27	41	31	28	74	131	90	34	37	30	35	36	86	122
235	216	124	117	187	179	145	79	45	39	33	38	116	86	38	123	128	122	57	40	45	29	36	48	91
237	224	168	163	194	190	181	127	53	41	31	28	108	159	88	132	135	140	91	49	47	43	32	40	48
239	234	183	161	195	196	183	163	71	44	27	23	63	171	168	115	138	137	132	52	80	64	40	38	34
240	238	206	162	201	208	197	176	113	40	27	29	32	116	181	151	126	141	141	110	57	103	52	32	41
238	239	224	174	203	215	210	189	156	106	56	34	38	114	163	191	140	137	135	137	77	75	98	37	33
238	240	234	187	201	223	214	206	178	147	104	70	65	112	143	217	177	135	135	129	131	54	66	53	35
238	238	238	212	192	222	222	208	192	168	141	95	98	100	136	175	193	151	130	133	133	107	41	47	46
236	237	237	221	179	214	213	200	185	178	162	121	98	91	118	134	172	180	126	136	139	135	78	42	46

Each pixel just represents a number. We can choose any color to represent those numbers!

# Two types of images: recorded vs generated

The pixel values of an image either represent a physical measurement or are created by an algorithm (which in certain cases can attempt to mimic a physical process).

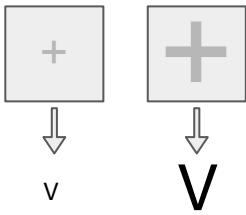
# Recorded



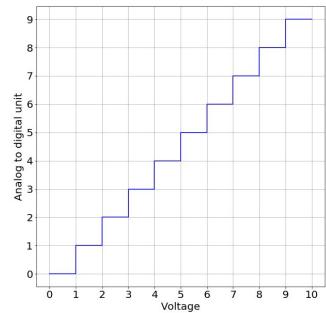
1. Fluorescent protein emits light



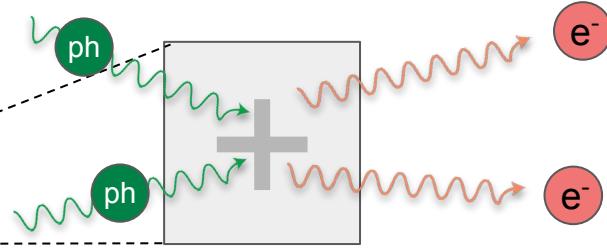
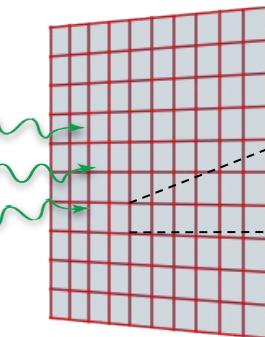
2. Emitted light goes through optics



5. Pixel charge translated into voltage



3. Light lands on camera detector composed of pixels



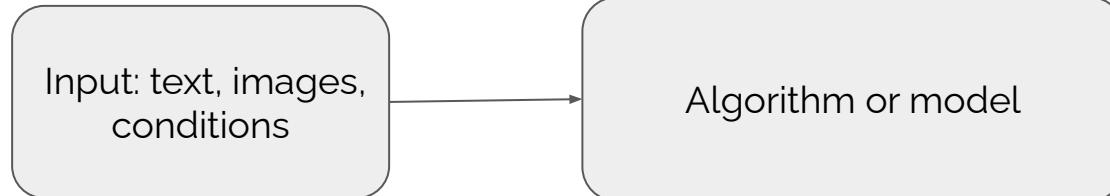
4. Each pixel reacts to light through a physical effect (e.g. photoelectric effect where pixel gets charged)

1	0	1	3	2	0	0	0	0	0
1	2	2	2	2	1	2	0	0	0
0	2	3	3	5	0	1	0	1	1
2	4	4	4	4	4	3	2	0	0
1	3	5	9	6	4	1	2	0	0
3	6	7	7	7	6	4	2	0	0
2	4	6	6	7	4	1	2	0	0
2	4	6	4	5	3	1	0	0	0
0	2	3	2	4	3	2	0	0	1
0	2	2	2	2	1	1	1	0	0

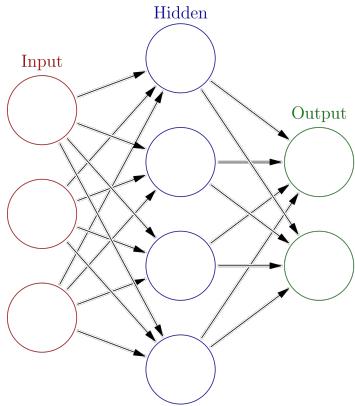
6. Voltage of each pixel is digitized, i.e. assigned a number depending on the voltage range to which it belongs.

7. For the human eye, values represented as colors.

# Generated



$$f(x, y) = \dots$$

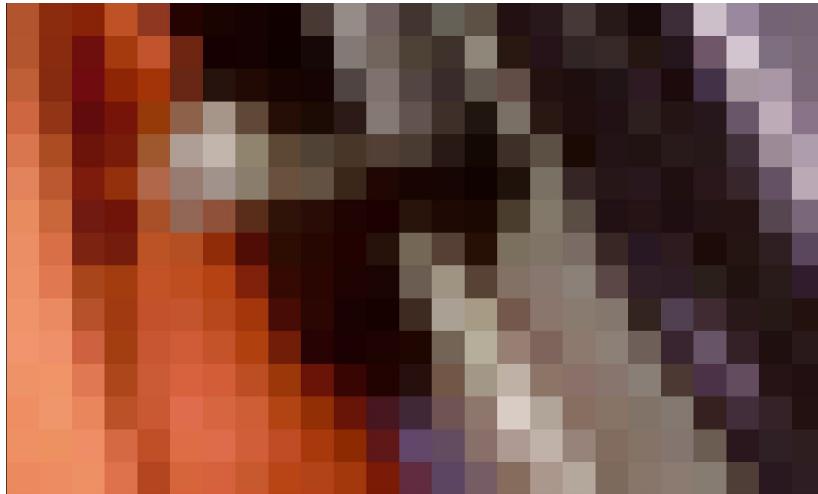


# Images are large arrays of numbers



178	139	137	190	144	36	26	22	17	73	149	110	66	102	93	34	41	70	39	24	61	205	154	116	117
179	137	110	164	193	110	23	22	16	21	132	113	78	63	144	41	37	50	54	38	36	111	210	126	120
191	134	112	142	163	104	39	34	24	22	80	127	83	57	98	97	34	30	32	46	28	67	166	169	122
206	146	97	118	150	141	165	96	36	28	44	132	98	62	33	122	41	32	34	44	36	37	105	189	137
218	171	109	121	159	175	194	145	92	80	72	82	73	40	20	62	91	27	36	32	40	38	64	156	173
227	187	125	148	177	152	161	139	105	99	59	33	24	22	16	31	121	54	38	41	31	41	55	100	187
232	201	113	112	169	147	144	89	48	41	31	27	41	31	28	74	131	90	34	37	30	35	36	86	122
235	216	124	117	187	179	145	79	45	39	33	38	116	86	38	123	128	122	57	40	45	29	36	48	91
237	224	168	163	194	190	181	127	53	41	31	28	108	159	88	132	135	140	91	49	47	43	32	40	48
239	234	183	161	195	196	183	163	71	44	27	23	63	171	168	115	138	137	132	52	80	64	40	38	34
240	238	206	162	201	208	197	176	113	40	27	29	32	116	181	151	126	141	141	110	57	103	52	32	41
238	239	224	174	203	215	210	189	156	106	56	34	38	114	163	191	140	137	135	137	77	75	98	37	33
238	240	234	187	201	223	214	206	178	147	104	70	65	112	143	217	177	135	135	129	131	54	66	53	35
238	238	238	212	192	222	222	208	192	168	141	95	98	100	136	175	193	151	130	133	133	107	41	47	46
236	237	237	221	179	214	213	200	185	178	162	121	98	91	118	134	172	180	126	136	139	135	78	42	46

# Images are large arrays of numbers



178	139	137	190	144	36	26	22	17	73	149	110	66	102	93	34	41	70	39	24	61	205	154	116	117
179	137	110	164	193	110	23	22	16	21	132	113	78	63	144	41	37	50	54	38	36	111	210	126	120
191	134	112	142	163	104	39	34	24	22	80	127	83	57	98	97	34	30	32	46	28	67	166	169	122
206	146	97	118	150	141	165	96	36	28	44	132	98	62	33	122	41	32	34	44	36	37	105	189	137
218	171	109	121	159	175	194	145	92	80	72	82	73	40	20	62	91	27	36	32	40	38	64	156	173
227	187	125	148	177	152	161	139	105	99	59	33	24	22	16	31	121	54	38	41	31	41	55	100	187
232	201	113	112	169	147	144	89	48	41	31	27	41	31	28	74	131	90	34	37	30	35	36	86	122
235	216	124	117	187	179	145	79	45	39	33	38	116	86	38	123	128	122	57	40	45	29	36	48	91
237	224	168	163	194	190	181	127	53	41	31	28	108	159	88	132	135	140	91	49	47	43	32	40	48
239	234	183	161	195	196	183	163	71	44	27	23	63	171	168	115	138	137	132	52	80	64	40	38	34
240	238	206	162	201	208	197	176	113	40	27	29	32	116	181	151	126	141	141	110	57	103	52	32	41
238	239	224	174	203	215	210	189	156	106	56	34	38	114	163	191	140	137	135	137	77	75	98	37	33
238	240	234	187	201	223	214	206	178	147	104	70	65	112	143	217	177	135	135	129	131	54	66	53	35
238	238	238	212	192	222	222	208	192	168	141	95	98	100	136	175	193	151	130	133	133	107	41	47	46
236	237	237	221	179	214	213	200	185	178	162	121	98	91	118	134	172	180	126	136	139	135	78	42	46

Both types of images can be further processed.  
To train a DL model, often images need pre-processing

# Operate on Images

# Processing existing images

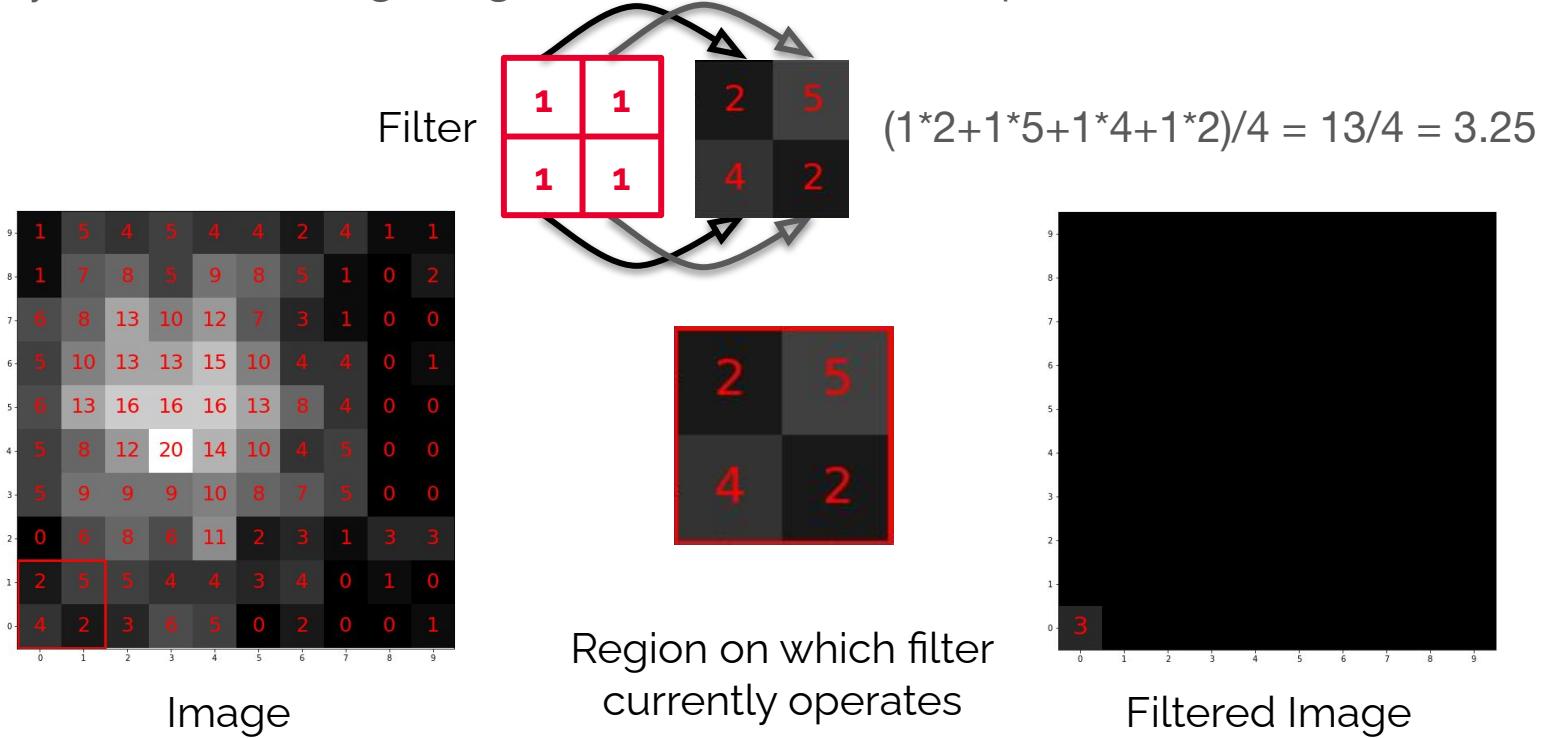
Images can be modified in many ways:

- cropping, flipping, rotating
- adjusting the color rendering (tone, brightness, contrast etc.)
- filtering: modify each pixel value considering the surrounding pixels

We cover these points in an interactive notebook!

# Filtering

Filtering consist in applying a mathematical operation, for example a sum, locally across an image, e.g. in an area of size 2x2 pixels.



# Filtering

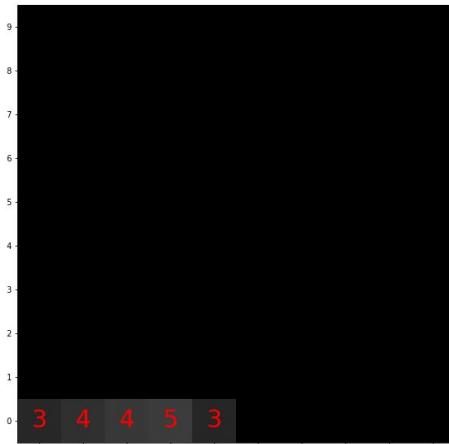
Filtering consist in applying a mathematical operation, for example a sum, locally across an image, e.g. in an area of size 2x2 pixels.

9	1	5	4	5	4	4	2	4	1	1
8	1	7	8	5	9	8	5	1	0	2
7	6	8	13	10	12	7	3	1	0	0
6	5	10	13	13	15	10	4	4	0	1
5	6	13	16	16	16	13	8	4	0	0
4	5	8	12	20	14	10	4	5	0	0
3	5	9	9	9	10	8	7	5	0	0
2	0	6	8	6	11	2	3	1	3	3
1	2	5	5	4	4	3	4	0	1	0
0	4	2	3	6	5	0	2	0	0	1

Image



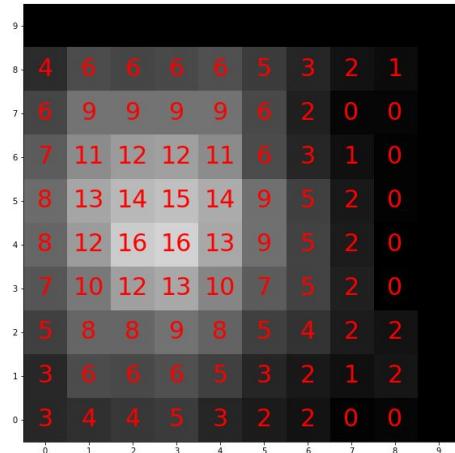
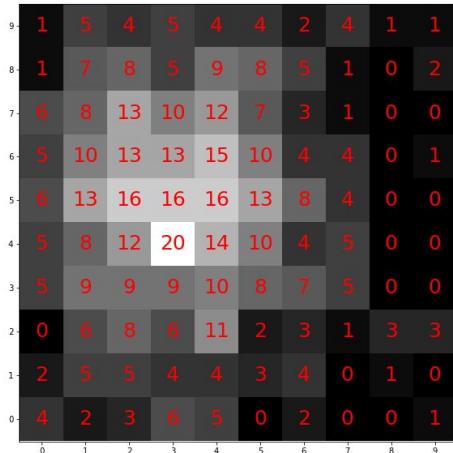
Region on which filter currently operates



Filtered Image

# Filtering

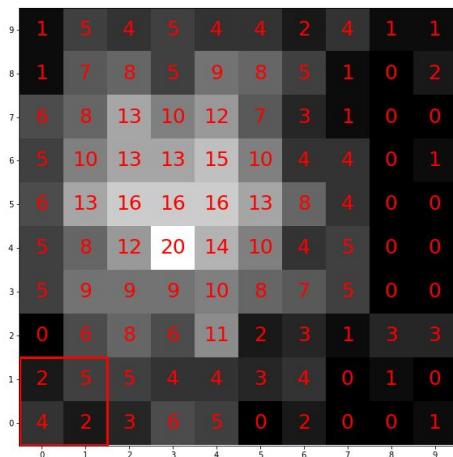
Filtering consist in applying a mathematical operation, for example a sum, locally across an image, e.g. in an area of size  $2 \times 2$  pixels.



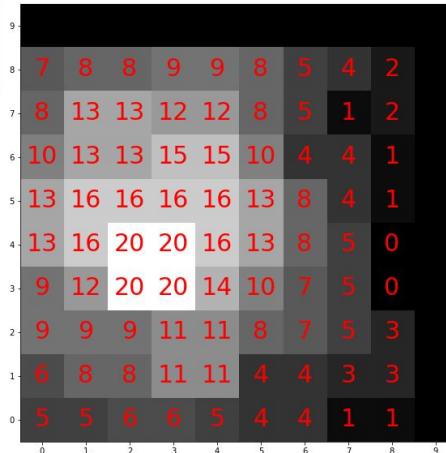
# Non-linear Filtering

Instead of a sum of pixels, one can also apply a specific function locally. For example we can use a maximum filter which replaces each pixel by the largest neighbouring value.

$$\max \left( \begin{array}{cc} 2 & 5 \\ 4 & 2 \end{array} \right) \quad \max (2,5,4,2) = 5$$



Image



Filtered Image

# Algorithmic generation: Rules and Randomness

# Randomness

One overly simple but useful definition is that generative art is art programmed using a computer that intentionally introduces randomness as part of its creation process. This often brings up two common but misguided viewpoints that hold people back from appreciating the beauty and nuance of generative art.

**Myth One:** The artist has complete control and the code is always executed exactly as written. Therefore, generative art lacks the elements of chance, accident, discovery, and spontaneity that often makes art great, if not at least human and approachable.

**Myth Two:** The artist has zero control and the autonomous machine is randomly generating the designs. The computer is making the art and the human deserves no credit, as it is not really art.

The truth is that generative artists skillfully control both the magnitude and the locations of randomness introduced into the artwork.

# Origins: breaking down art to geometry



Paul Klee

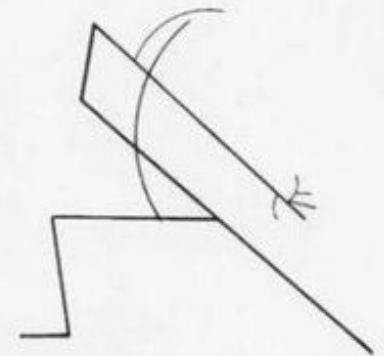
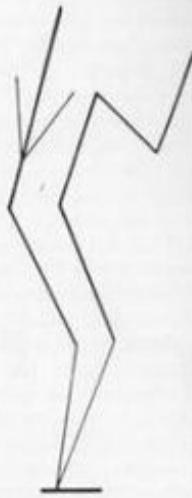
Static Dynamic  
Gradation, 1923



Vasily Kandinsky

Several circles,  
1926

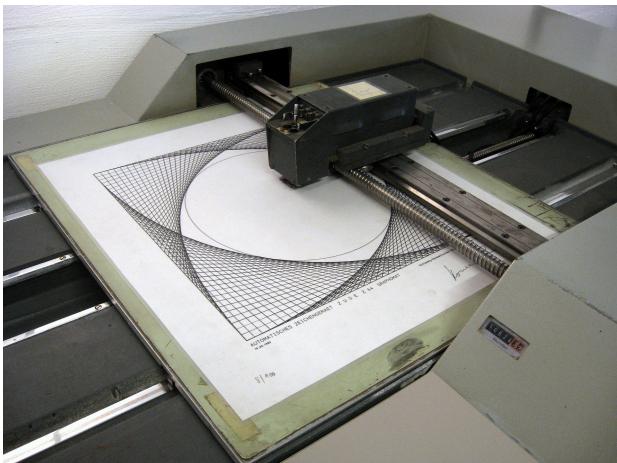
# Origins:



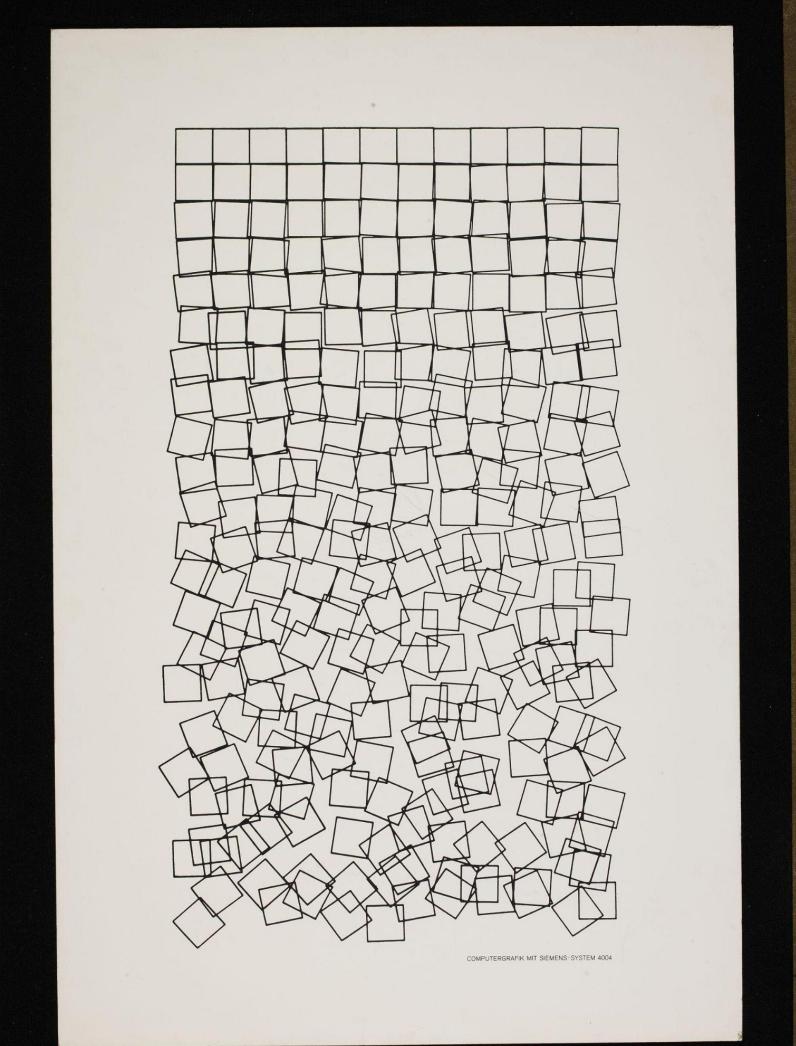
Vasily Kandinsky, Dance curves. 1926

# Georg Nees (1926-2016)

- One of the founders of generative art
- Produced images directly with plotters



Gravel, 1968



# Frieder Nake (1939-)

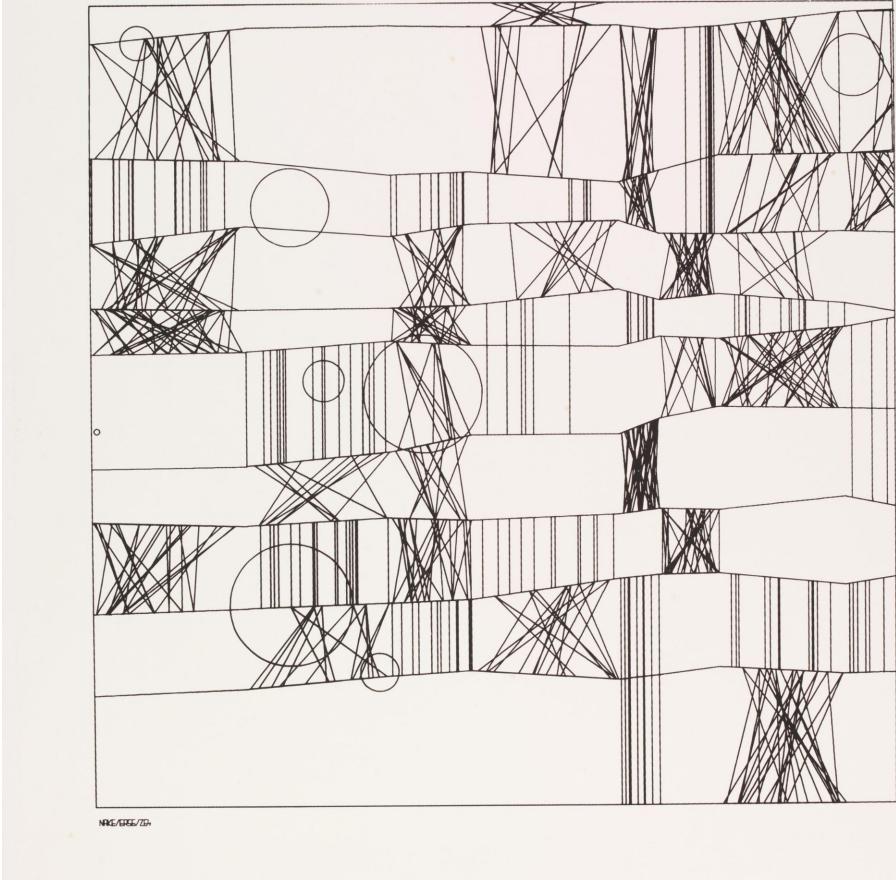
Computer graphics are two-dimensional visual aesthetic objects created in cooperation between humans and computers. In this case the human works together with the electronic computer system telefunken tr4 and the punched tape controlled drawing machine zuse z64. This cooperation works as follows: man invents a program that the computer should process. He still has certain freedoms. **The barriers to these freedoms are set by people as fixed or variable in the program. They can be arbitrarily narrow or arbitrarily wide.**



Matrix Multiplications, 1967

# Frieder Nake

Computer graphics are two-dimensional visual aesthetic objects created in cooperation between humans and computers. In this case the human works together with the electronic computer system telefunken tr4 and the punched tape controlled drawing machine zuse z64. This cooperation works as follows: man invents a program that the computer should process. He still has certain freedoms. **The barriers to these freedoms are set by people as fixed or variable in the program. They can be arbitrarily narrow or arbitrarily wide.**



Hommage à Paul Klee, 1965

# Michael Noll (1939-)

- Interest in animation, 3D computer graphics and objects
- Interest for human-machine interaction

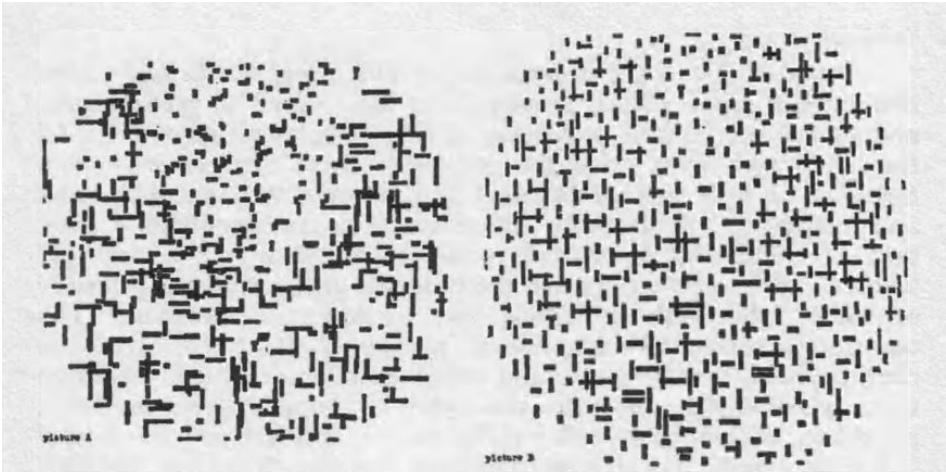
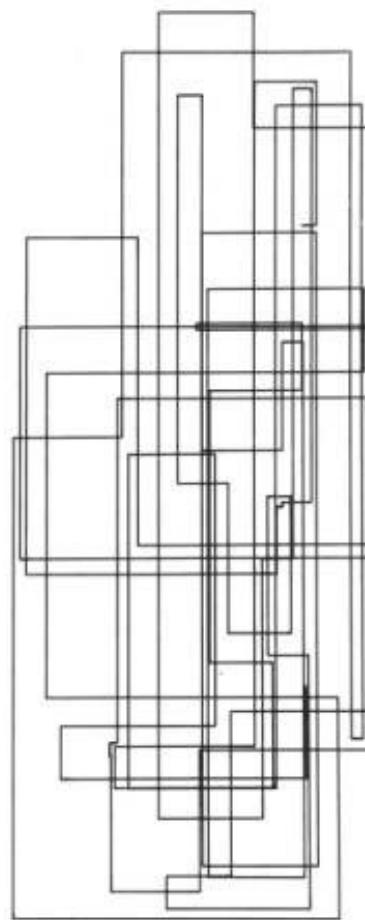


Fig. 4 Picture pair as presented on separate sheets to subjects. The original microfilm and Mondrian photograph were copied xerographically and then reproduced so that both pictures were identical in quality. (© A. Michael Noll 1965 and © Rijkmuseum Kröller-Müller.)

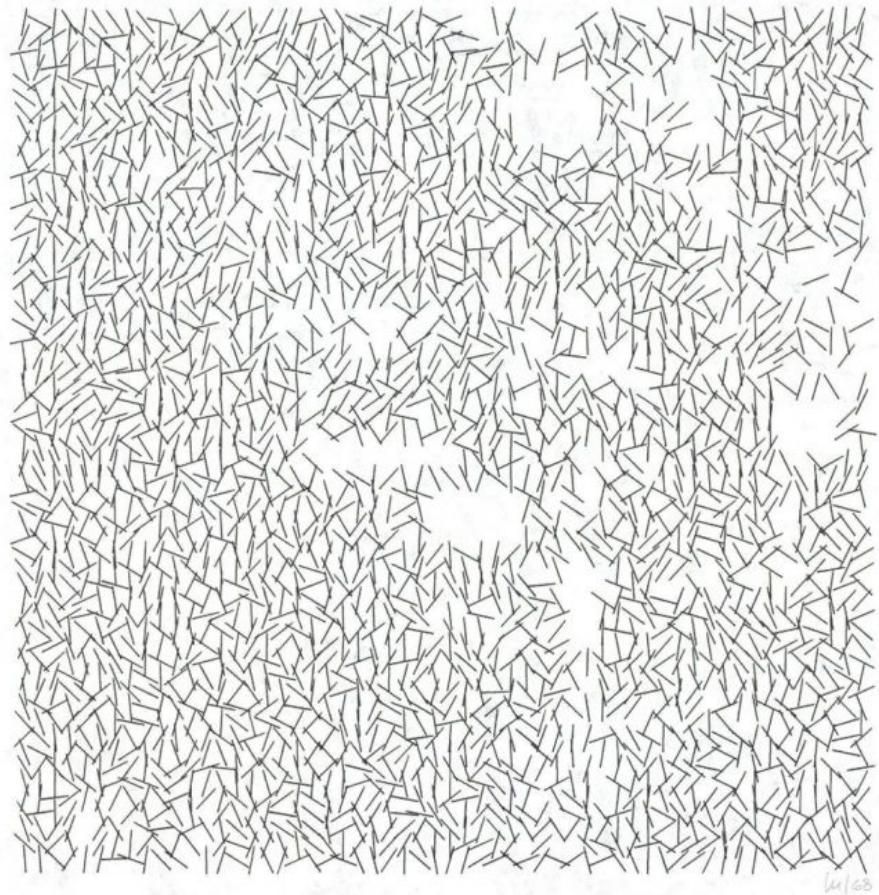


© AMN 1965

VERTICAL-HORIZONTAL NUMBER THREE (1964)  
BY A. MICHAEL NOLL

# Vera Molnár (1924-2023)

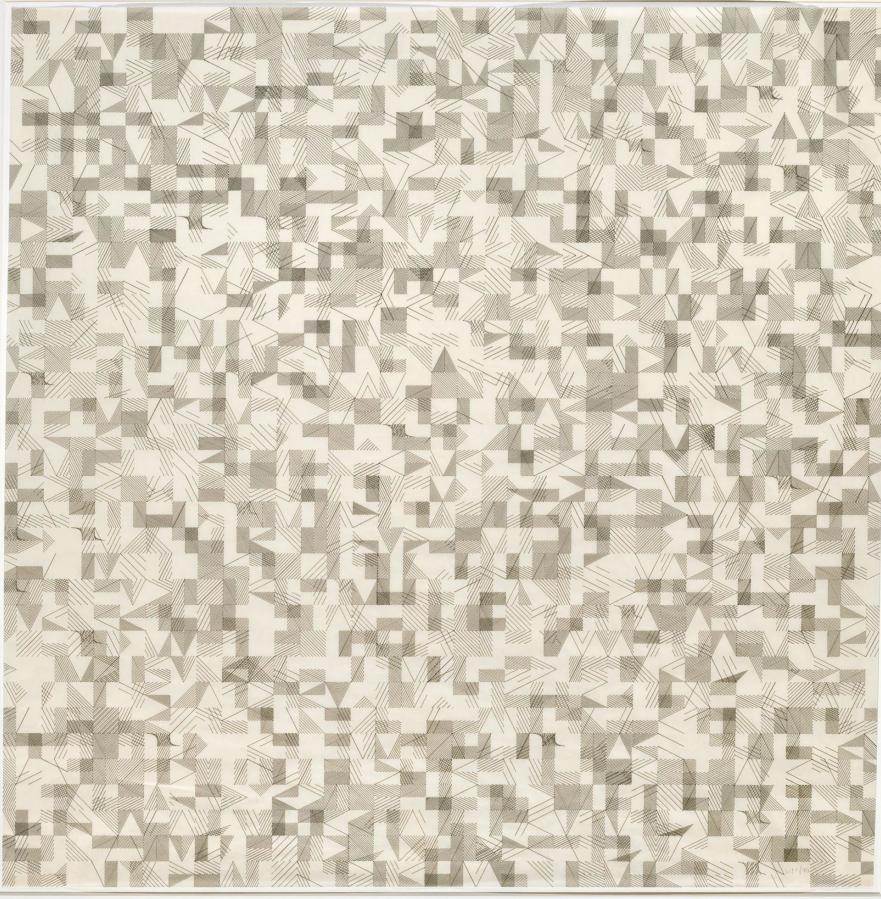
- First woman in generative arts and pioneer of the movement
- Started with hand-drawn algorithmic images
- Moved to computer/plotter images on early computers



Interruptions, 1965

# Vera Molnár

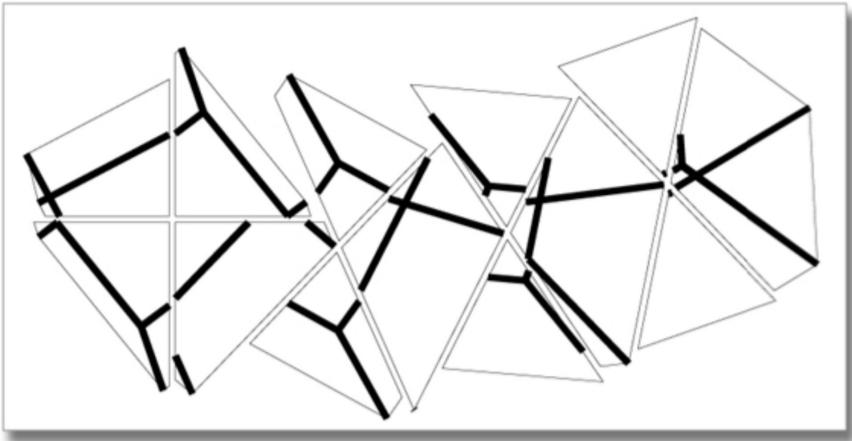
- First woman in generative arts and pioneer of the movement
- Started with hand-drawn algorithmic images
- Moved to computer/plotter images on early computers



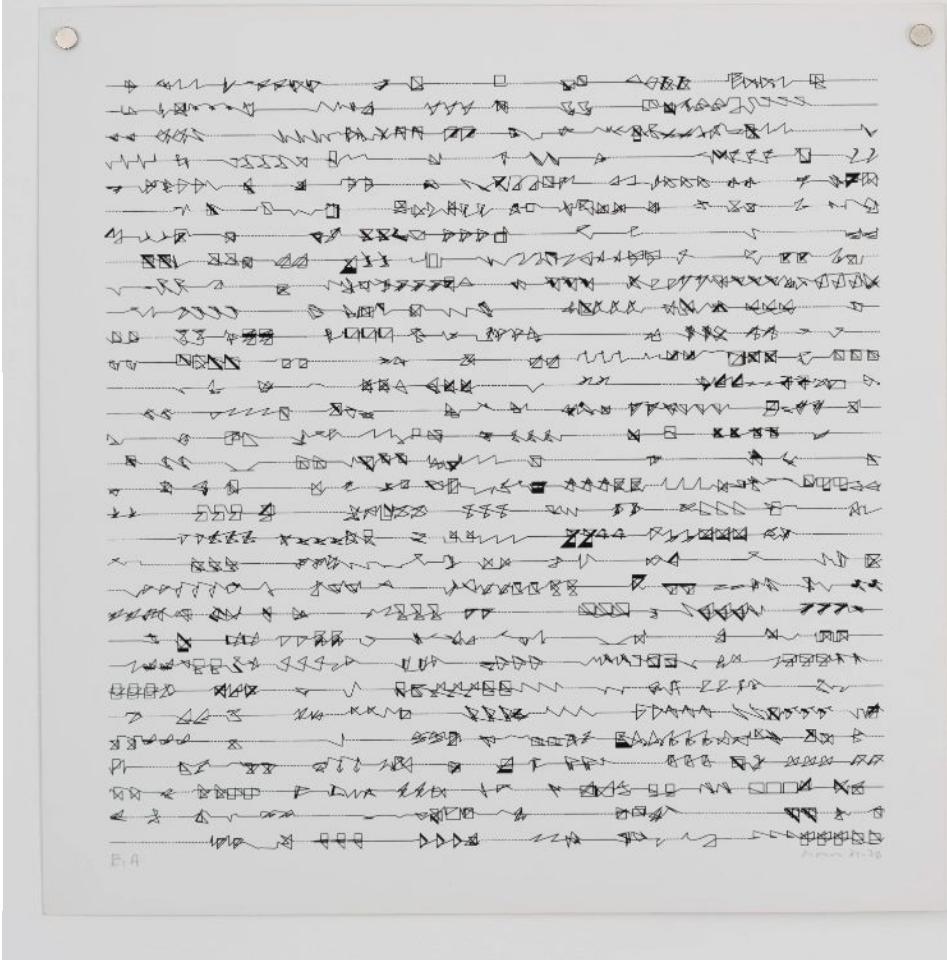
A la recherche de Paul Klee, 1970

# Manfred Mohr (1938-)

- Exploration of geometric abstraction
- Influence from music



P-360-F {FF}, 1984



P026, 1970

# Manolo Gamboa Naon (1978-)



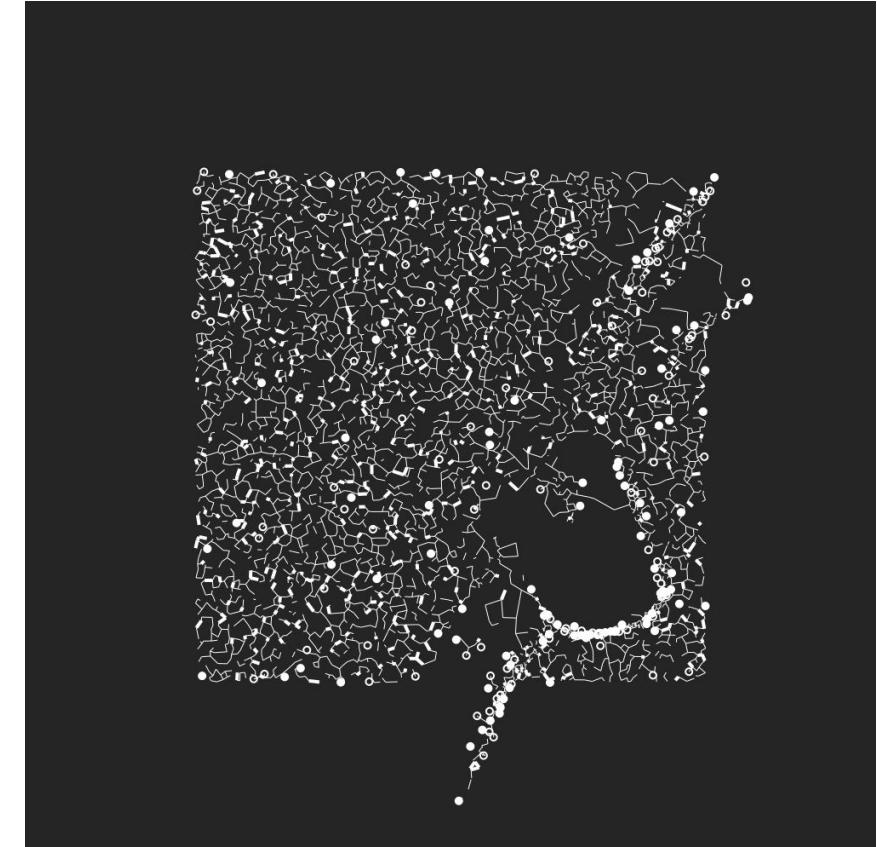
cllm 2020



Arbolito 2018

# Anders Hoff (1976-)

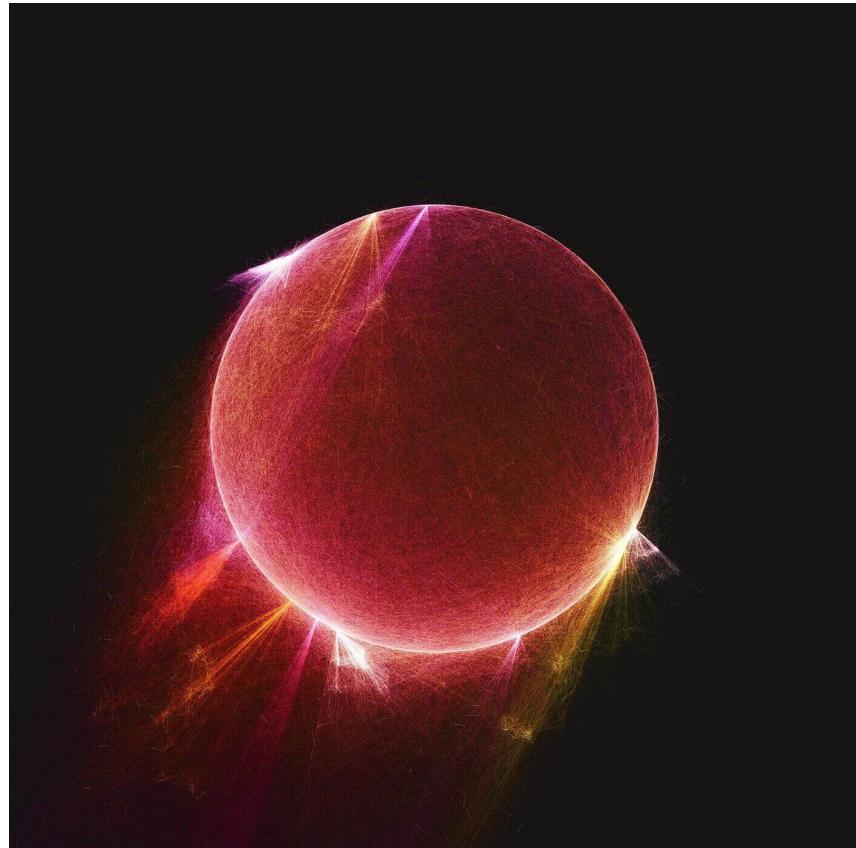
"Randomness can also be used as the initial structured aspect. Here is an example where the initial structure is a graph drawn from uniformly distributed nodes. The uniform distribution has the additional limitation that no two nodes can be closer than a certain minimum distance. Chaos is introduced by having a random walker push the nodes around the canvas."



inconvergent

# Anders Hoff (1976-)

"Randomness can also be used as the initial structured aspect. Here is an example where the initial structure is a graph drawn from uniformly distributed nodes. The uniform distribution has the additional limitation that no two nodes can be closer than a certain minimum distance. Chaos is introduced by having a random walker push the nodes around the canvas."

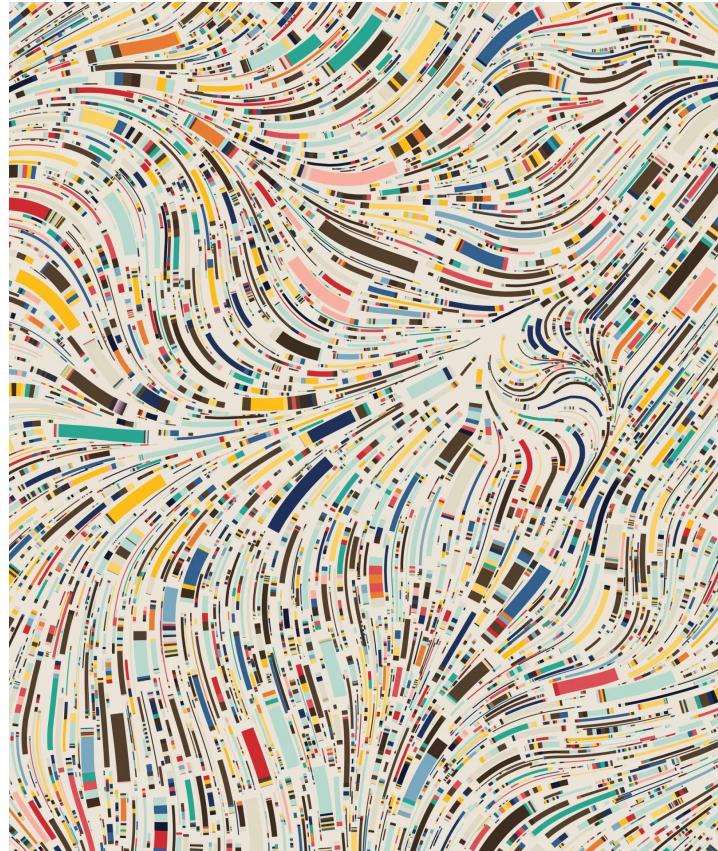


inconvergent

# Tyler Hobbs (1987-)

Interesting interview on the specific process of generative art, in particular the “curation” step:

<https://www.tylerxhobbs.com/about#Bio>



Fidenza, 2021

# Jared S. Tarbell (1973-)



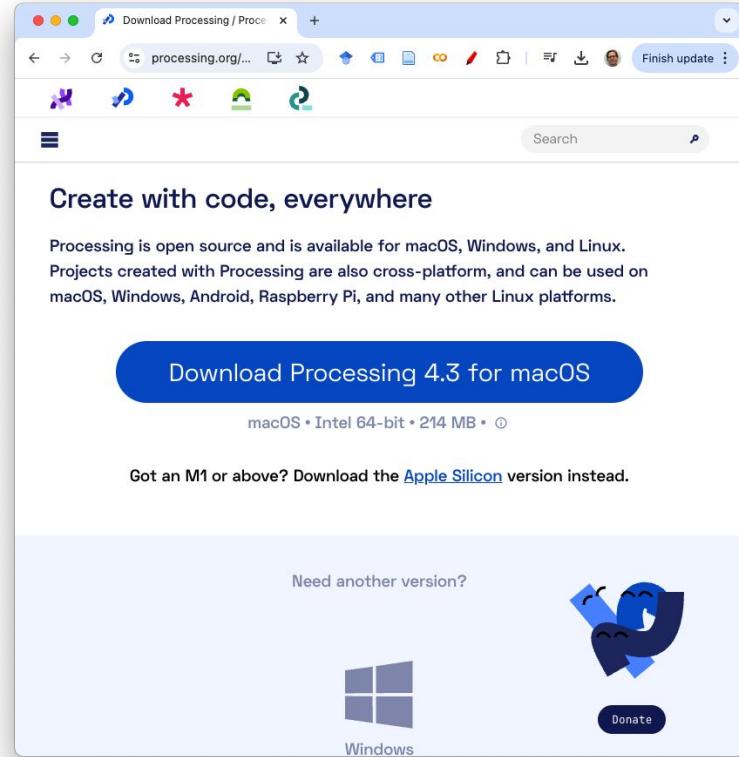
<http://www.complexification.net/gallery/machines/substrate/index.php>

# Processing

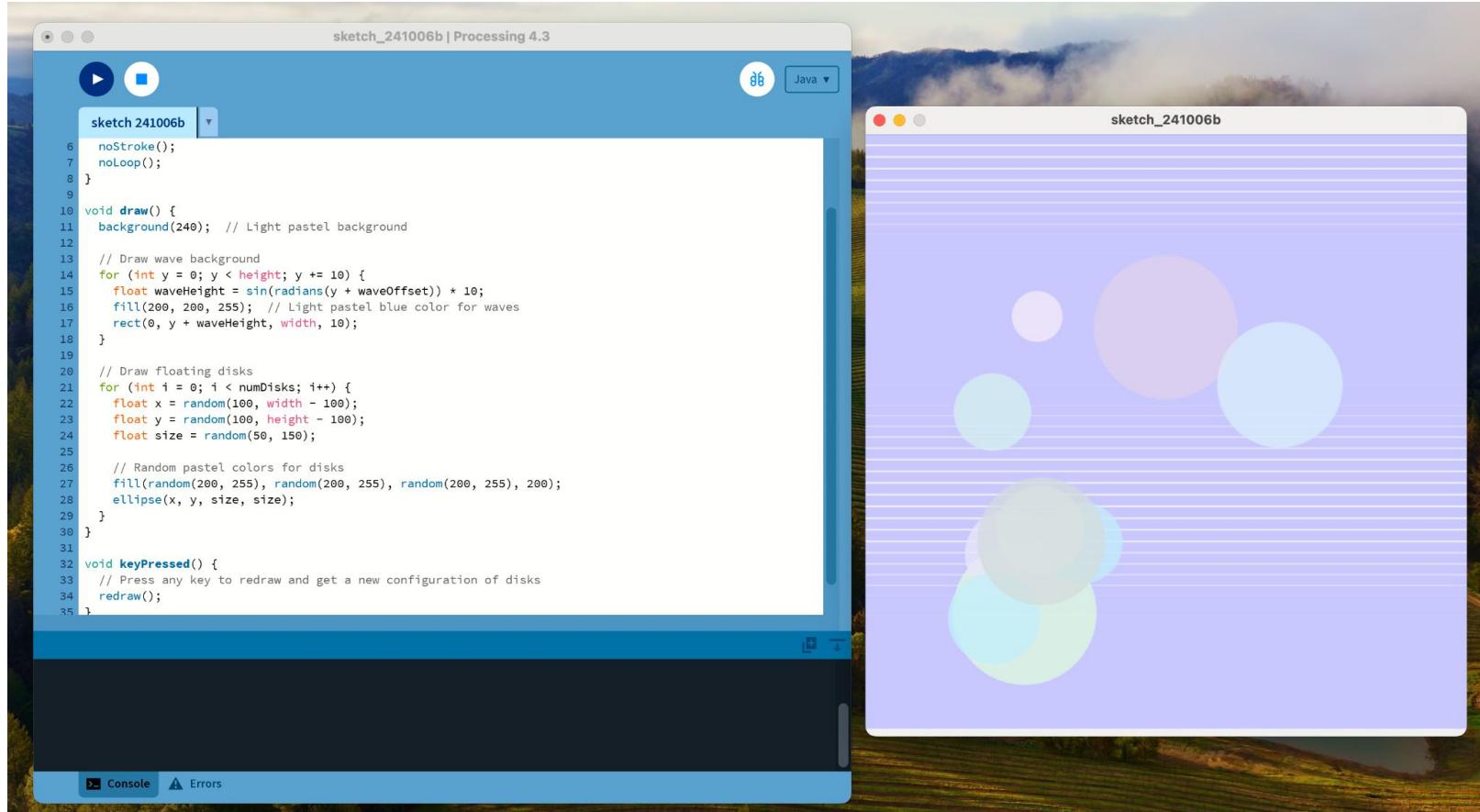
- Created by Casey Reas and Ben Fry
- Popular software for digital artists
- Relatively simple language
- Also available in Python
- A bit outdated but lots of examples

To download here:

<https://processing.org/download>



# Processing



# Jared S. Tarbell

Source code available and can be re-run in Processing

GitHub:

<https://github.com/jaredstarbell>

Interview:

<https://www.rightclicksave.com/article/the-interview-jared-s-tarbell-generative-art-etsy>

More examples:

<https://www.infinite.center/list/>



The screenshot shows a Processing sketch window with the title bar 'complexification.net/gallery/machines/substrate/appletm/substrate\_m.pde'. The status bar indicates 'Not Secure' and the URL 'complexification.net/gallery/machines/substrate/appletm/substrate\_...'. The code itself is a PDE script for a generative art piece titled 'Substrate Watercolor' by j.tarbell from June 2004, Albuquerque, New Mexico, on complexification.net. It uses Processing 0085 Beta syntax updated in April 2005. The code initializes dimensions (dimx=500, dimy=500), a counter (num=0), and a maximum value (maxnum=150). It defines arrays for cracks (cgrid, cracks) and sand painters (sands). It sets color parameters (maxpal=512, numpal=0, goodcolor=new color[maxpal]). The main methods section includes setup() and draw(). setup() initializes the canvas size (500x500, P3D), background (255), and takes a color from 'pollockShimmering.gif'. It creates the crack grid and initializes cracks. draw() iterates through all cracks and moves them.

```
// Substrate Watercolor
// j.tarbell June, 2004
// Albuquerque, New Mexico
// complexification.net

// Processing 0085 Beta syntax update
// j.tarbell April, 2005

int dimx = 500;
int dimy = 500;
int num = 0;
int maxnum = 150;

// grid of cracks
int[] cgrid;
Crack[] cracks;

// color parameters
int maxpal = 512;
int numpal = 0;
color[] goodcolor = new color[maxpal];

// sand painters
SandPainter[] sands;

// MAIN METHODS

void setup() {
    size(500,500,P3D);
    // size(dimx,dimy,P3D);
    background(255);
    takecolor("pollockShimmering.gif");

    cgrid = new int[dimx*dimy];
    cracks = new Crack[maxnum];
}

void draw() {
    // crack all cracks
    for (int n=0;n<num;n++) {
        cracks[n].move();
    }
}
```

Missing file:

<https://github.com/nottombrown/drivetrain/blob/master/processing-js-0.6-examples/examples/custom/data/pollockShimmering.gif>

# Using chatGPT as helper

ChatGPT 4o ▾

Share



Can you make a picture of floating disks with pastel colours on a background of waves in that language



Certainly! Here's a Processing sketch that creates an image of floating disks with pastel colors on a wavy background:

```
java

int numDisks = 10;
float waveOffset = 0;

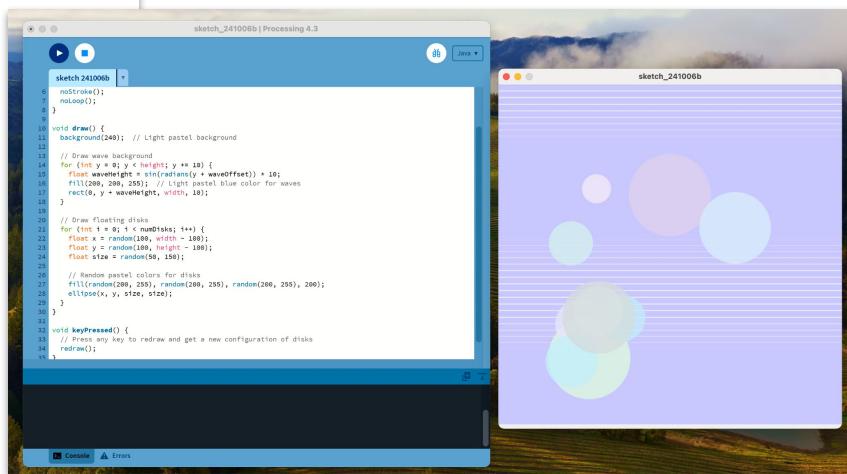
void setup() {
    size(600, 600);
    noStroke();
    noLoop();
}

void draw() {
    background(240); // Light pastel background

    // Draw wave background
    for (int y = 0; y < height; y += 10) {
        float waveHeight = sin(radians(y + waveOffset)) * 10;
        fill(200, 200, 255); // Light pastel blue color for waves
        rect(0, y + waveHeight, width, 10);
    }
}
```

How to animate disks?

Can I save images?



Message ChatGPT



# p5.js successor to Processing

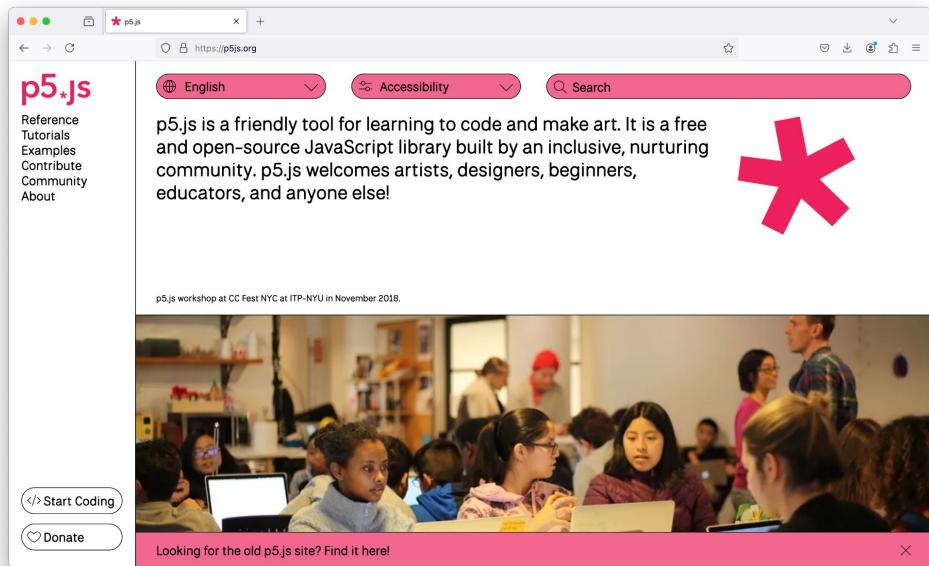
Project page: <https://p5js.org/>

A interface directly in the browser:  
<https://editor.p5js.org/>

Simple syntax but in JavaScript

Example on the Generative Design site:

<http://www.generative-gestaltung.de/2/>



# p5.js successor to Processing

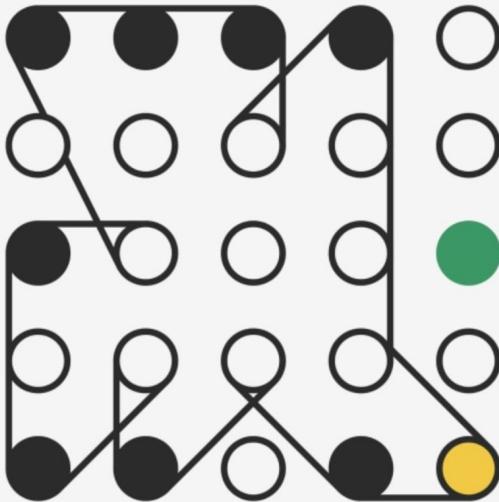
PHILLIPS

AUCTIONS CALENDAR DEPARTMENTS EXHIBITIONS PERPETUAL DROPSHOP BUY & SELL EDITORIAL ABOUT US

LOGIN

ENGLISH ▲ Q

Ex-Machina: A History of Generative Art Online Auction 13-20 July 2022



<

8

>

THIS LOT IS A NON-FUNGIBLE TOKEN  
(NFT)

8

Dmitri Cherniak

*Ringers #29*

Token ID: 13000029

Contract Address: [0xa7d8...D270](#)

Non-Fungible Token: ERC-721

PNG: 219 KB (225,006 bytes), 2400 x 2400 pixels

The HTML text is 8KB (8,285 bytes),  
resolution agnostic, p5js.

Minted on 31 January 2021, this work is  
unique and not part of an edition.

[Additional Conditions of Sale applicable  
to NFT Lots](#)

[Full Cataloguing](#)

**Estimate**

£1,100,000 - 1,300,000

**Contact Specialist**

Benjamin Kandler  
Project Lead for Digital Art, Phillips  
+44 7393 141 146  
bkandler@phillips.com

p5js

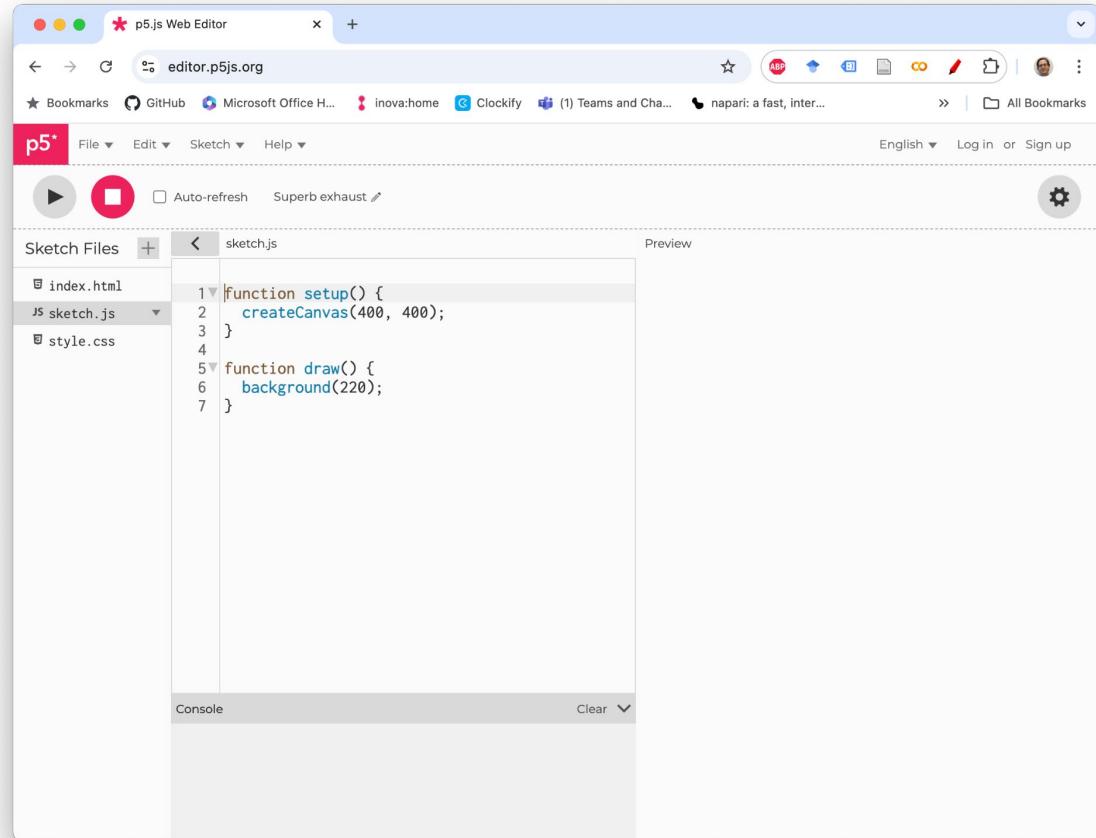
# First project: <https://editor.p5js.org/>

Minimal project:

**index.html**: creates  
the page displaying  
the image

**sketch.js**: actually  
contains your  
instructions

**style.css**: no need to  
touch



The screenshot shows the p5.js Web Editor interface. At the top, there's a toolbar with icons for file operations, sketch controls (play, stop, auto-refresh), and help. Below the toolbar is a menu bar with 'File', 'Edit', 'Sketch', and 'Help'. On the right side, there are language and user account settings. The main workspace is divided into sections: 'Sketch Files' on the left listing 'index.html', 'sketch.js' (which is currently selected), and 'style.css'; a central code editor window containing the following P5.js code; and a 'Preview' window on the right showing a blank canvas. At the bottom, there's a 'Console' tab and a 'Clear' button.

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

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

# Using LLMs to help you create generative art

The screenshot shows a ChatGPT interface in a web browser. The user has asked for a script to draw colorful squares that gradually become circles and change colors over time. ChatGPT has provided a p5.js script in JavaScript code:

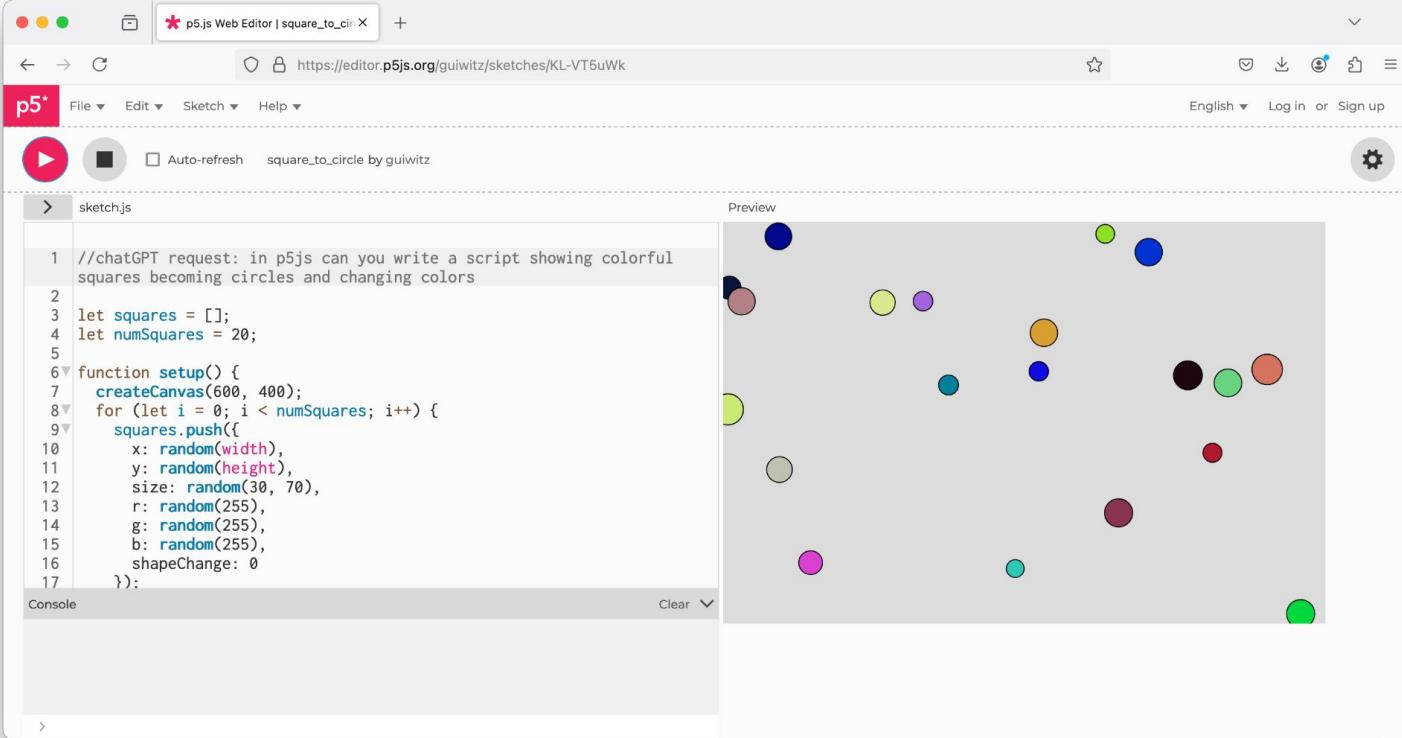
```
javascript
let squares = [];
let numSquares = 20;

function setup() {
  createCanvas(600, 400);
  for (let i = 0; i < numSquares; i++) {
    squares.push({
      x: random(width),
      y: random(height),
      size: random(30, 70),
      r: random(255),
      g: random(255),
      b: random(255).
    });
  }
}

function draw() {
  background(255);
  for (let i = 0; i < numSquares; i++) {
    let s = squares[i];
    if (s.size < 0) {
      s.r = random(255);
      s.g = random(255);
      s.b = random(255);
      s.size = random(30, 70);
      s.x = random(width);
      s.y = random(height);
    } else {
      s.size -= 1;
    }
    fill(s.r, s.g, s.b);
    rect(s.x, s.y, s.size, s.size);
  }
}
```

At the bottom of the screen, there is a footer note: "ChatGPT can make mistakes. Check important info."

# Using LLMs to help you create generative art



The screenshot shows the p5.js Web Editor interface. The title bar says "p5\* sketch.js". The code editor contains the following JavaScript code:

```
1 //chatGPT request: in p5js can you write a script showing colorful
2 //squares becoming circles and changing colors
3 let squares = [];
4 let numSquares = 20;
5
6 function setup() {
7   createCanvas(600, 400);
8   for (let i = 0; i < numSquares; i++) {
9     squares.push({
10       x: random(width),
11       y: random(height),
12       size: random(30, 70),
13       r: random(255),
14       g: random(255),
15       b: random(255),
16       shapeChange: 0
17     });
18   }
19 }
```

The preview window shows a collection of colorful circles scattered across a gray canvas. The circles vary in size and color, including shades of blue, green, red, yellow, and purple. Some circles overlap, creating a dynamic visual effect.

<https://editor.p5js.org/guiwitz/sketches/KL-VT5uWk>

# ml5js: Machine Learning for p5js

The screenshot shows a web browser window with the URL <https://ml5js.org/> in the address bar. The page title is "ml5.js" and the tab title is "ml5.js | Friendly Machine Learning for the Web". The header includes links for "Getting Started", "Reference", "Learn", "Community", and "About". The main content area features a central message: "Welcome to our new website! We've launched a new version of ml5.js with breaking changes. If you encounter errors such as '`... is not a function`' or need features that are no longer available, please refer to [our FAQ](#) for how to access the previous version and documentation." Below this, the heading "Friendly Machine Learning for the Web" is displayed, followed by the subtext "A neighborly approach to creating and exploring artificial intelligence in the browser." A "Get Started" button is visible. The background is white with various hand-drawn illustrations of AI-related concepts like a lightbulb, a rocket, a brain, a person, and a cat.

<https://ml5js.org/>

# Models

ml5js proposes multiple ready-to-use models:

- handpose: <https://editor.p5js.org/ml5/sketches/QGH3dwJ1A>
- body segmentation: <https://editor.p5js.org/ml5/sketches/ruoyal-RC>
- etc.

With adjustments:

- saving hand coordinates as csv (to be re-used e.g. in Python):  
<https://editor.p5js.org/guiwitz/sketches/iFtroo8Qv>
- saving body segmentation:  
<https://editor.p5js.org/guiwitz/sketches/cfgstlMOo>

# Resources

<https://natureofcode.com/physics-libraries/#why-use-a-physics-library>

<http://www.generative-gestaltung.de/2/>