PYCON US 2020

The Joy of Creating Art with Code.

Presented by Neeraj Pandey

@NEERAJP99

NEERAJ PANDEY

@NEERAJP99

ASHOKA UNIVERSITY

Sophomore student at Ashoka University.

Full Stack Developer, Generative Art and
Data Science Enthusiast

POINTS FOR DISCUSSION

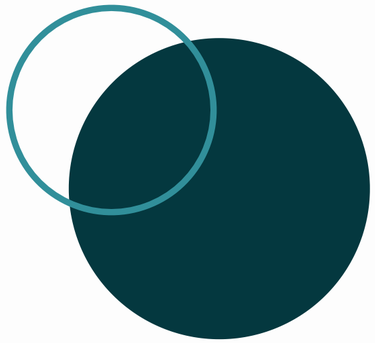
Generative Art - Principles and Elements

History behind Generative Art

Geometry, Algorithms and Randomness

Examples using Processing, PyCairo

Genetic Algorithms and GAN's



@NEERAJP99

Art created through the use of an autonomous system.

ALGORITHMS, MATHEMATICS, GENETIC SEQUENCES

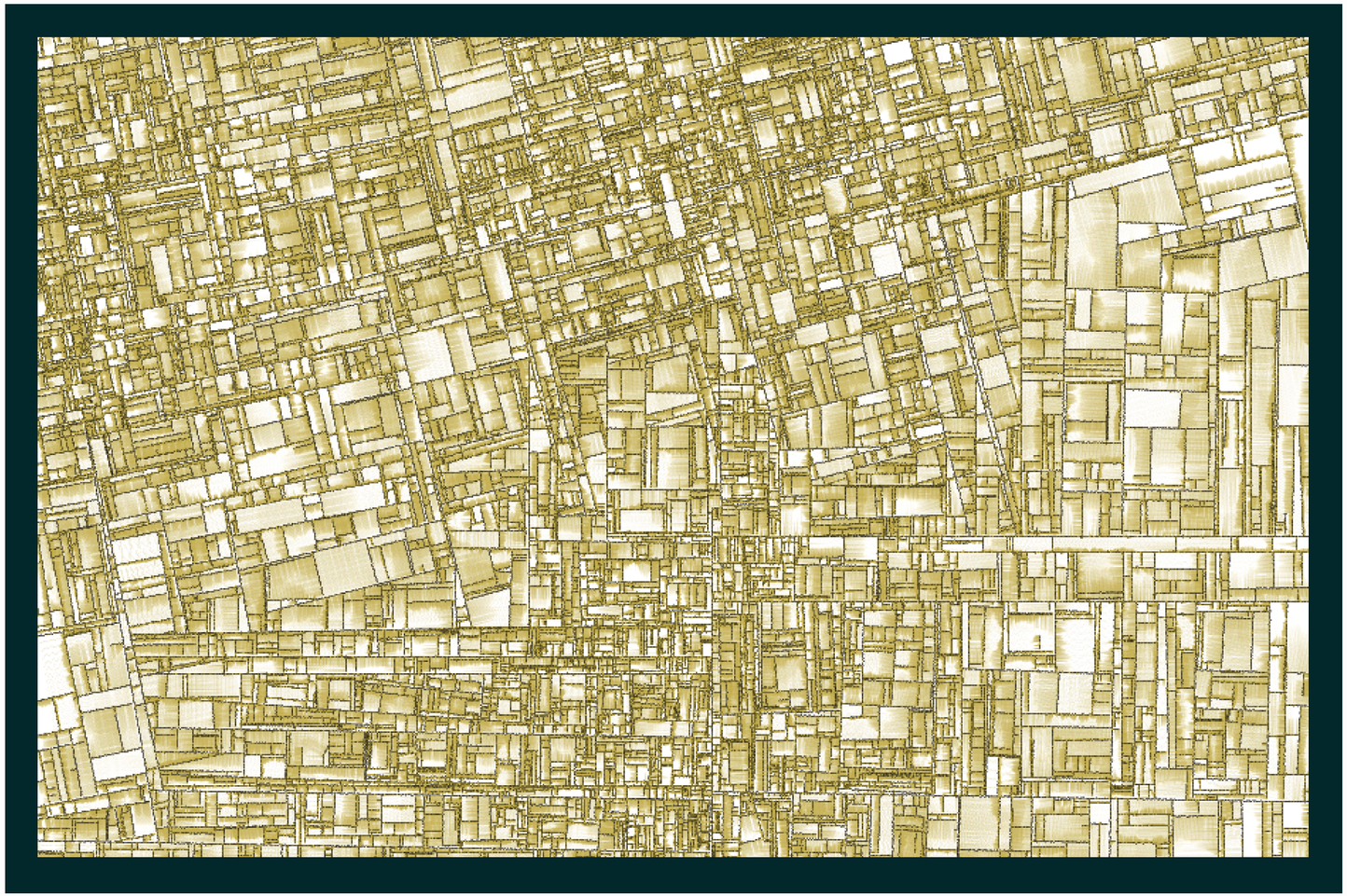
@NEERAJP99

PRINCIPLES AND ELEMENTS

Elements: color, form, line, shape, space, the randomness and the texture.

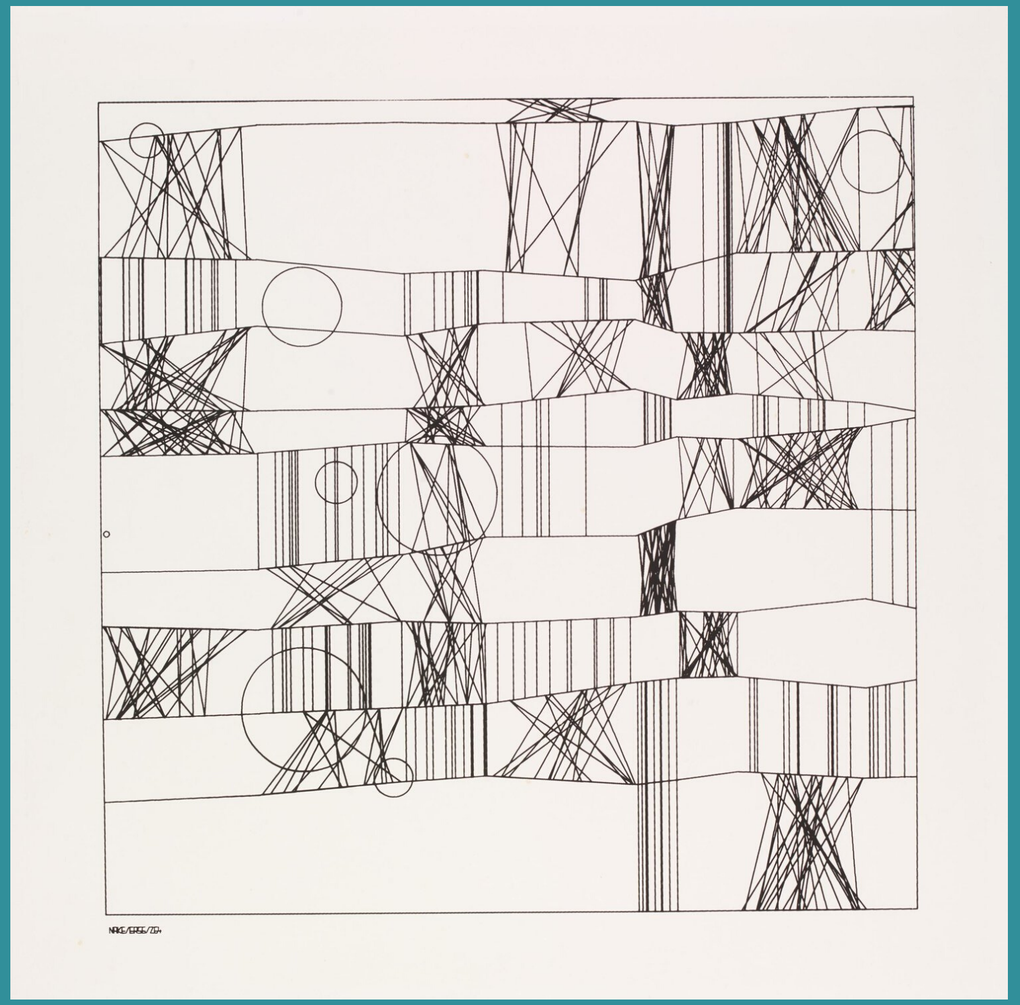
Principles: rhythm, contrast, harmony, balance, movement, proportion

Substrate, Jared Tarbell



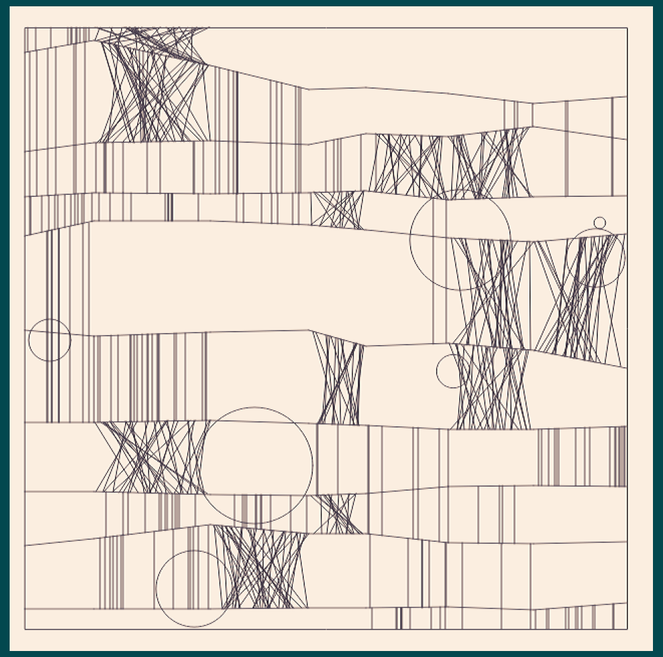
Created using Python mode for Processing

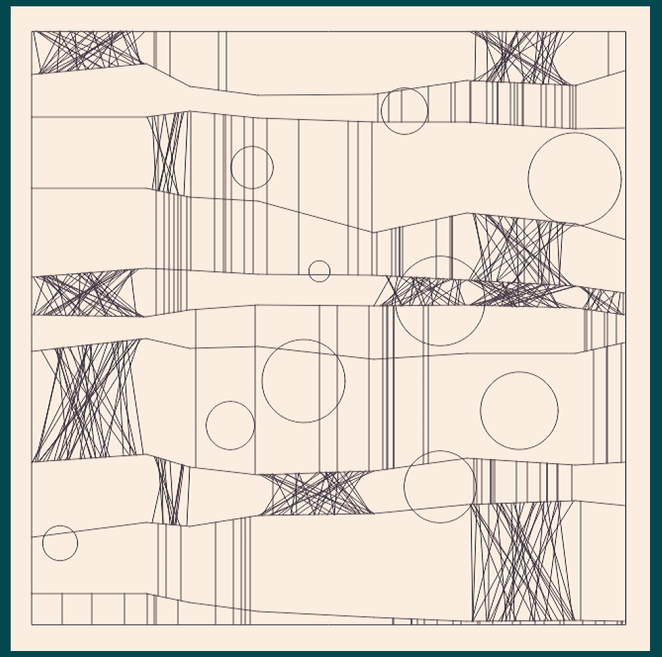
HISTORY OF GENERATIVE ART

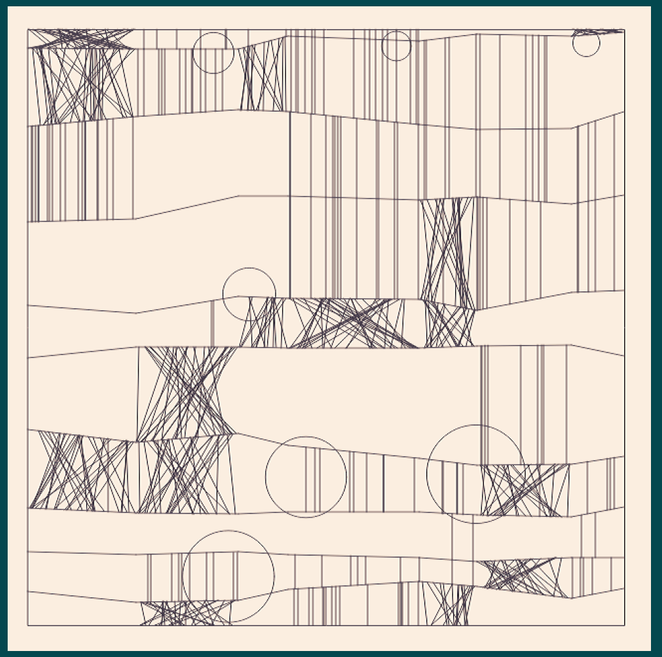


Hommage à Paul Klee - Frieder Nake, 1965

SIMILAR EXAMPLES CREATED USING PROCESSING



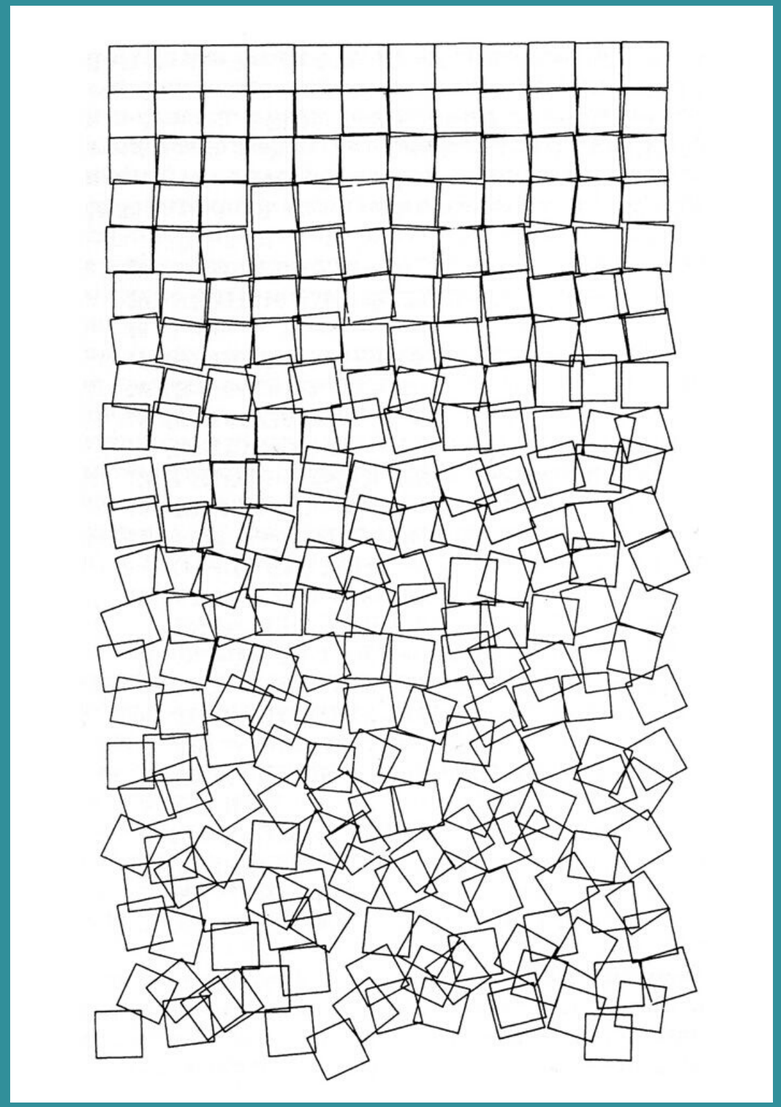


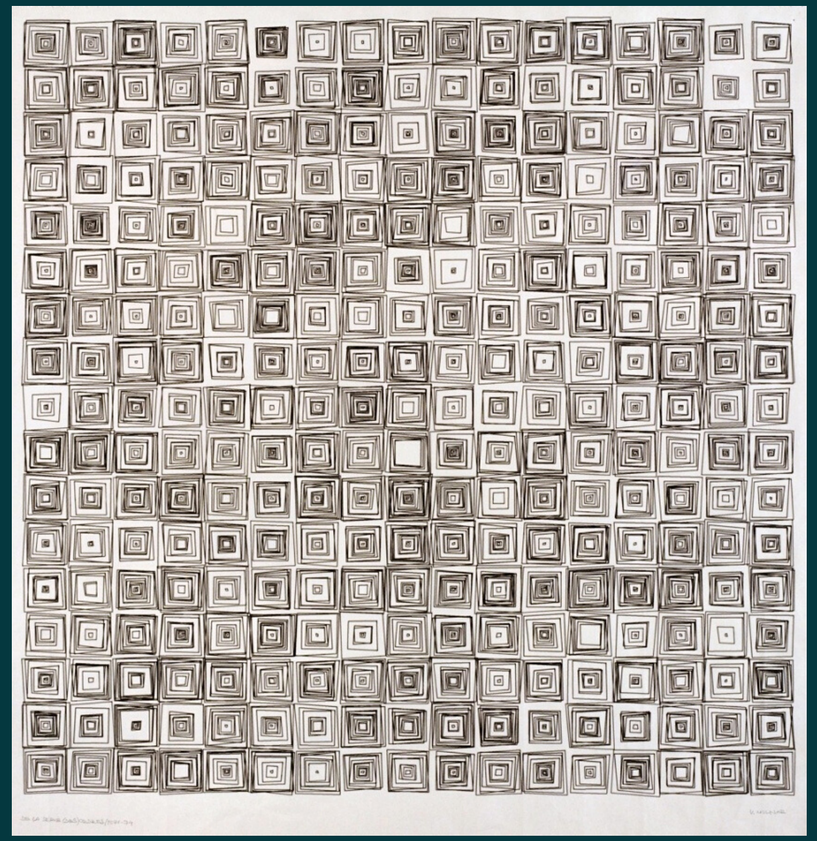


@NEERAJP99

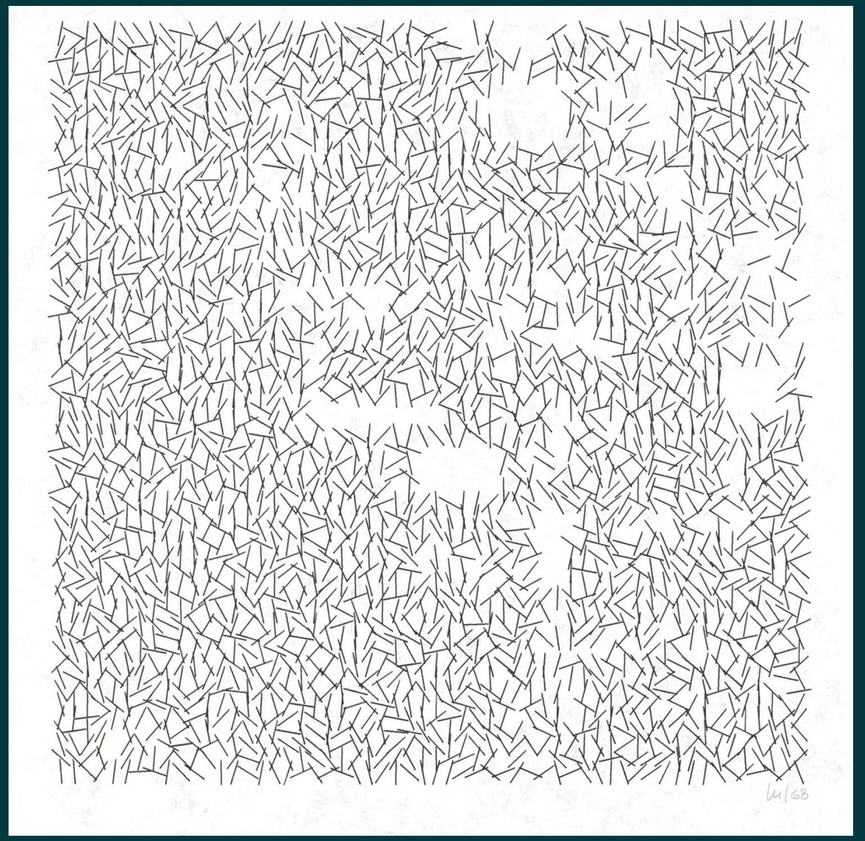
One of the earliest and best-known pieces of generative art.

By Georg Nees, 1968





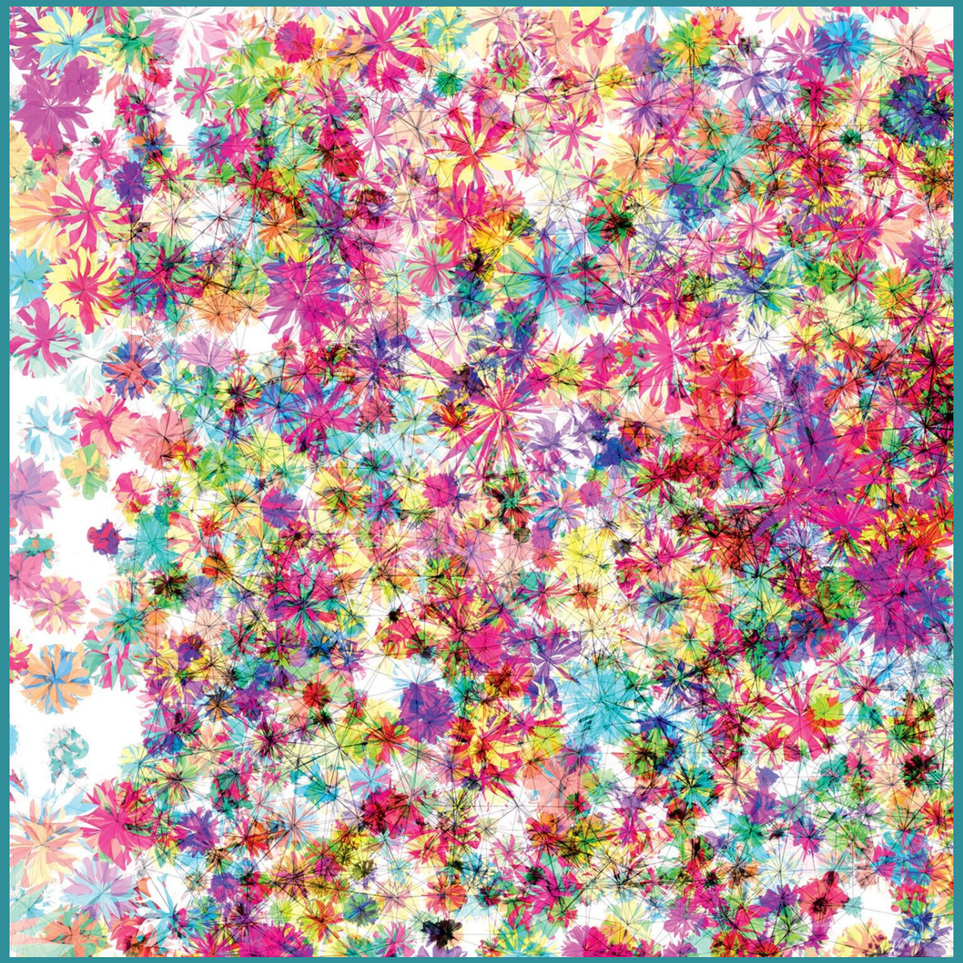
Dés Ordres - Vera Molnár, 1974



Interruptions - Vera Molnár, 1968/69

"When you can do both, you can do things that no one else can do"

Florada - John Maeda, 1990s



PROCESSING FOUNDATION

Processing

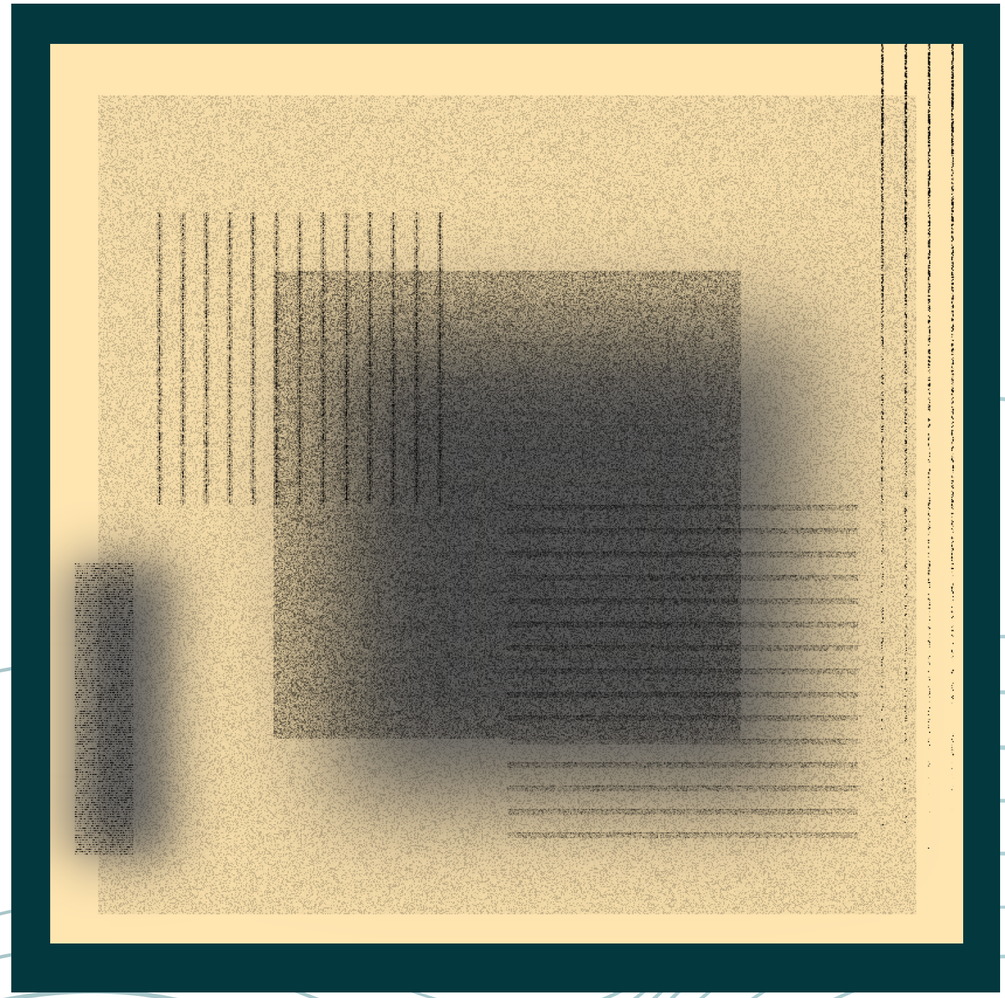
P5.js

Processing.py

Processing for Pi

Processing for Android

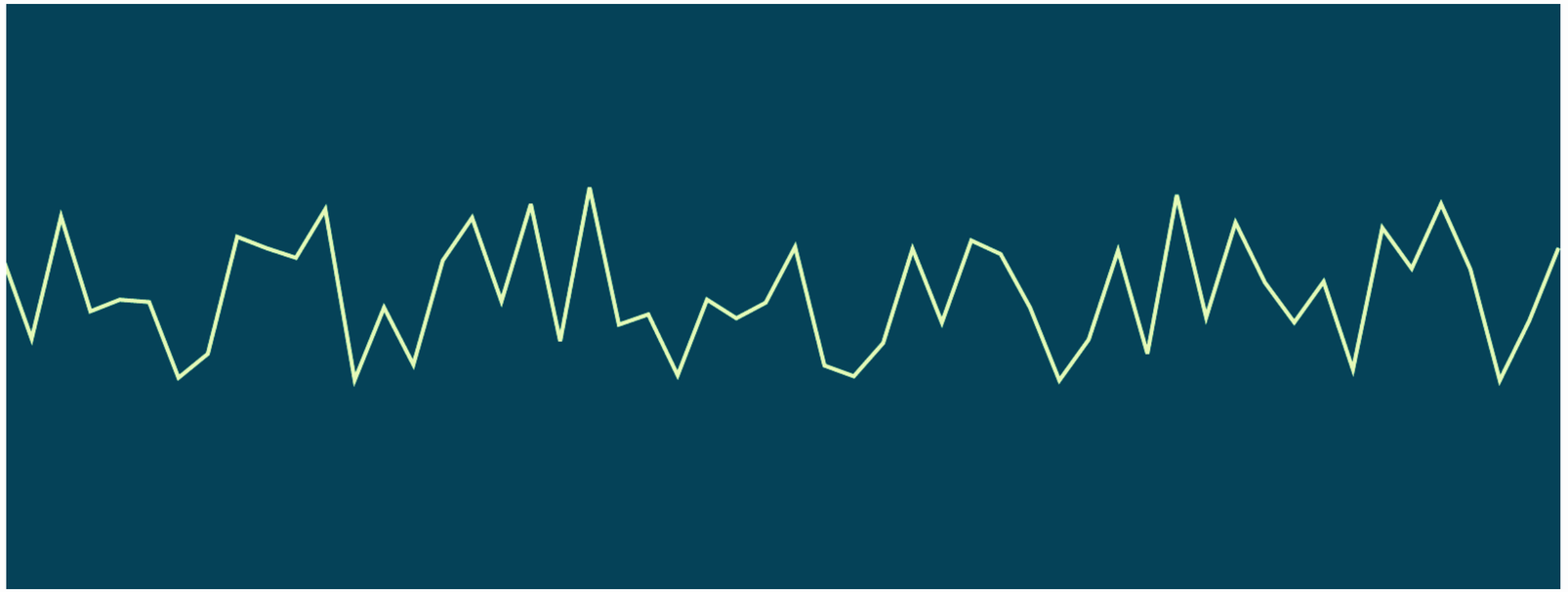
USING MATHEMATICS AND ALGORITHMS



@NEERAJP99

RANDOM

Generates random floating point numbers .



RANDOM

Generates random floating point numbers .

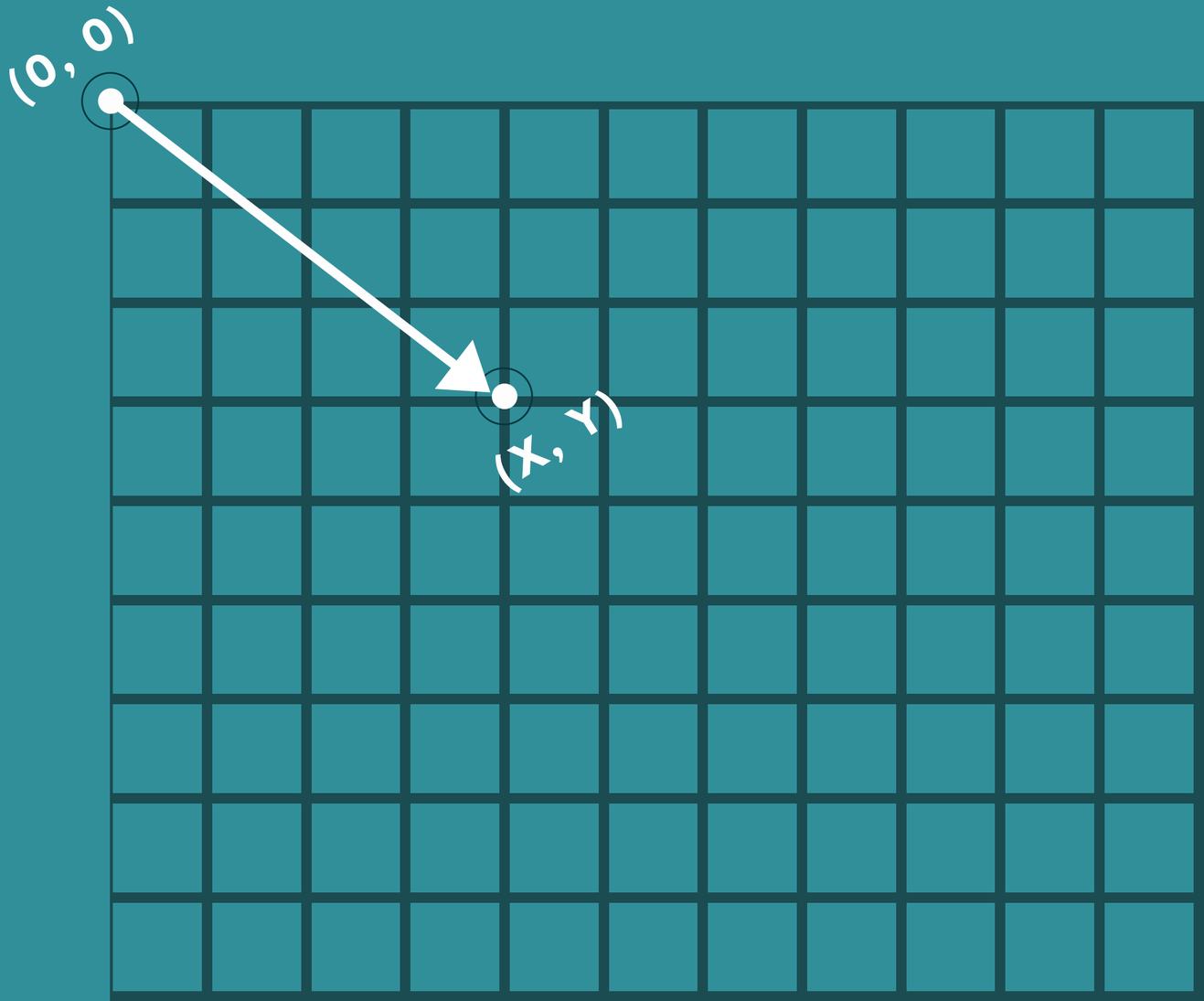
```
1 random()
2 # floating value between [0, 1)
3 random([min], [max])
4 # Random value between [min, max)
5 random(x)
6 # Random value between [0, x)
```

Without processing, use random module in Python.
(https://docs.python.org/3/library/random.html)

ARTIST'S CANVAS

DRAWING A POINT

We consider a 2-D cartesian plane, and each point as a vector.



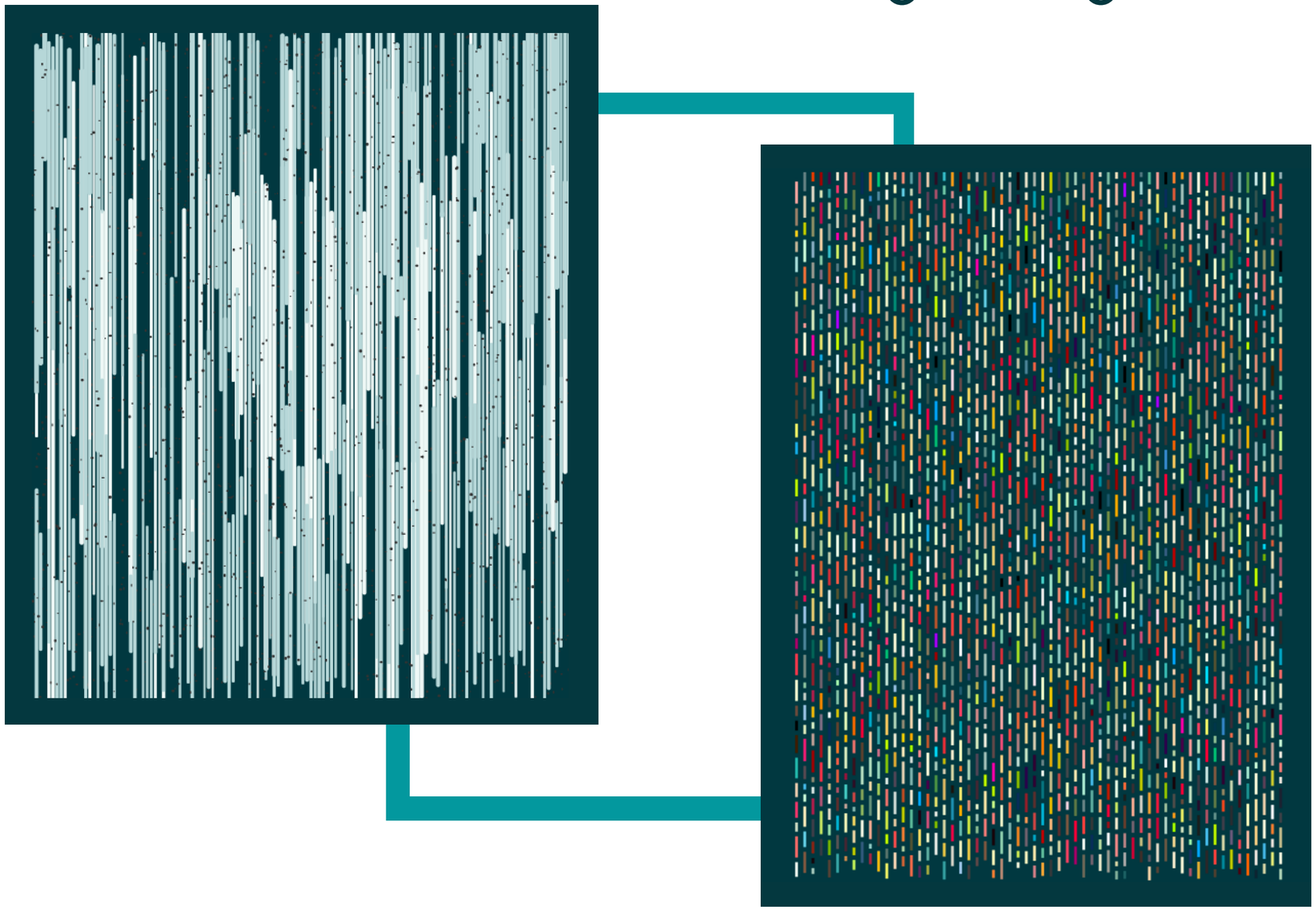
CREATING A POINT AND LINE

```
point(x, y)
# Creates a point at (x, y) from the origin.
point(x, y, z)
# Creates a point at (x, y, x).
dist(x1, y1, x2, y2)
# Calculates the distance between 2 points.
line(x1, y1, x2, y2)
# Create a line by joining 2 points.

stroke(color)
# Sets the color used to draw lines
# and borders around shapes.
strokeWeight(x)
# Sets the thickness of the stroke.
```

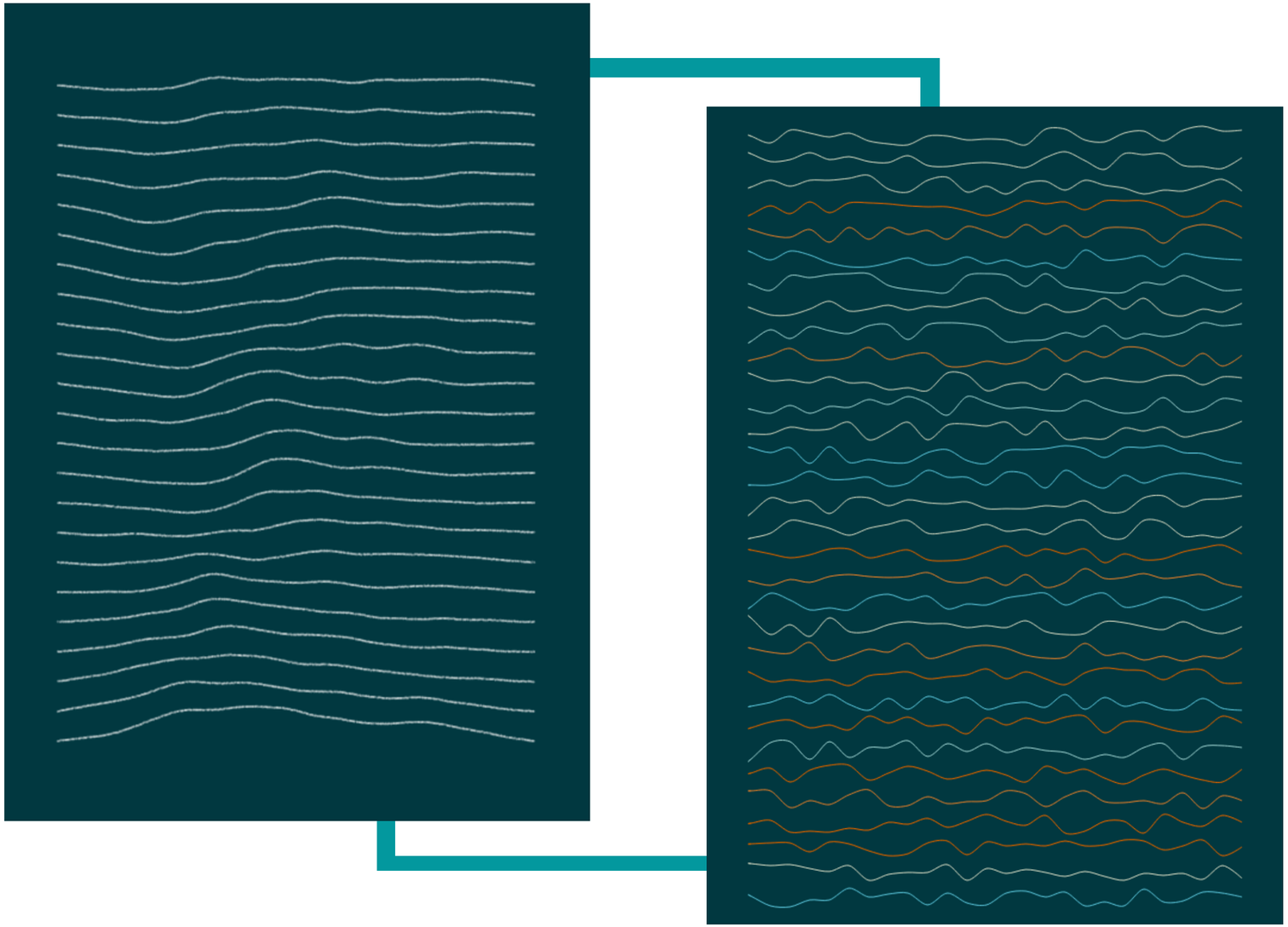
```
context.set_line_width(0.02)
context.move_to(x, y)
context.line_to(x1, y1)
context.stroke()
```

Using straight lines



@NEERAJP99

Using vector operations



CURVE VERTEX & BEZIER CURVES

Curve Vertex: It specifies the vertex coordinates for curves.

Bezier Curves: It is a versatile mathematical curve in vector graphics.

@NEERAJP99

CREATING BASIC SHAPES

```
ellipse(a, b, c, d)
# Creates an ellipse at point a,b
# with "c" width and "d" height
rect(a, b, c, d, tl, tr, br, bl))
# Creates a rectangle at point a,b
# width width "c" and height "d"
square(x, y, c)
# Creates a square
```

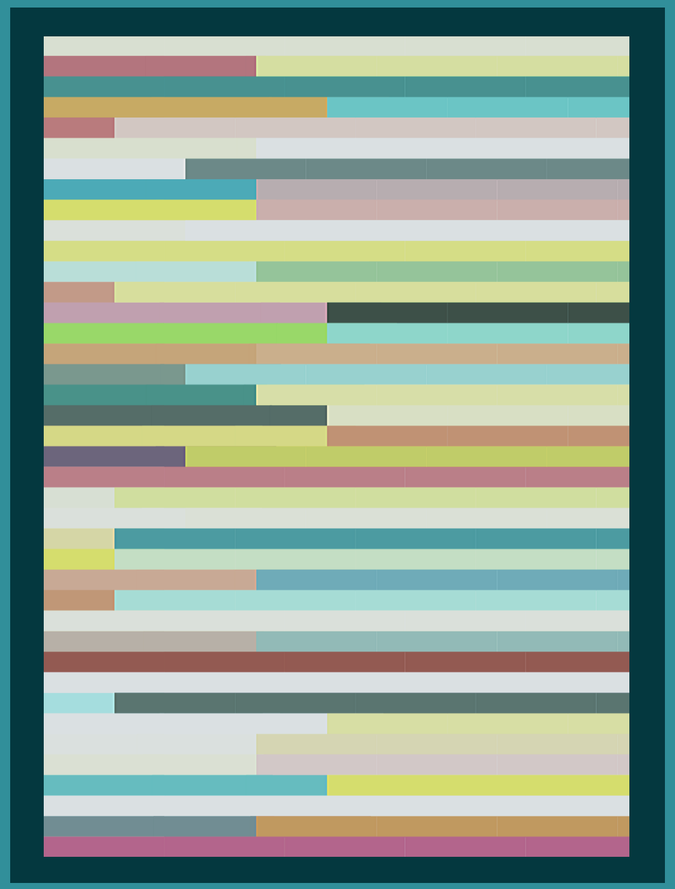
```
context.rectangle(a, b, c, d)
# Creates a rectangle
context.arc(x, y, radius, start_angle, stop_angle)
# Creates an arc, with angles in radians
# with stop angle as math.pi*2 for ellipse
```

LINEAR INTERPOLATION

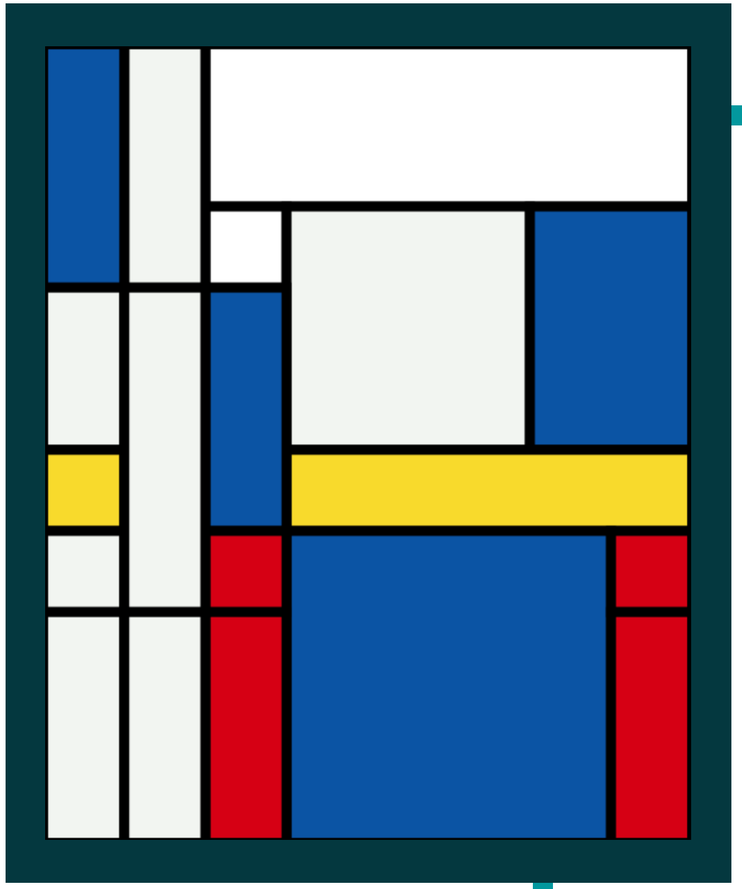
This function interpolates within the range [start..end] based on the amount parameter, where amount parameter is typically within a [0..1] range.

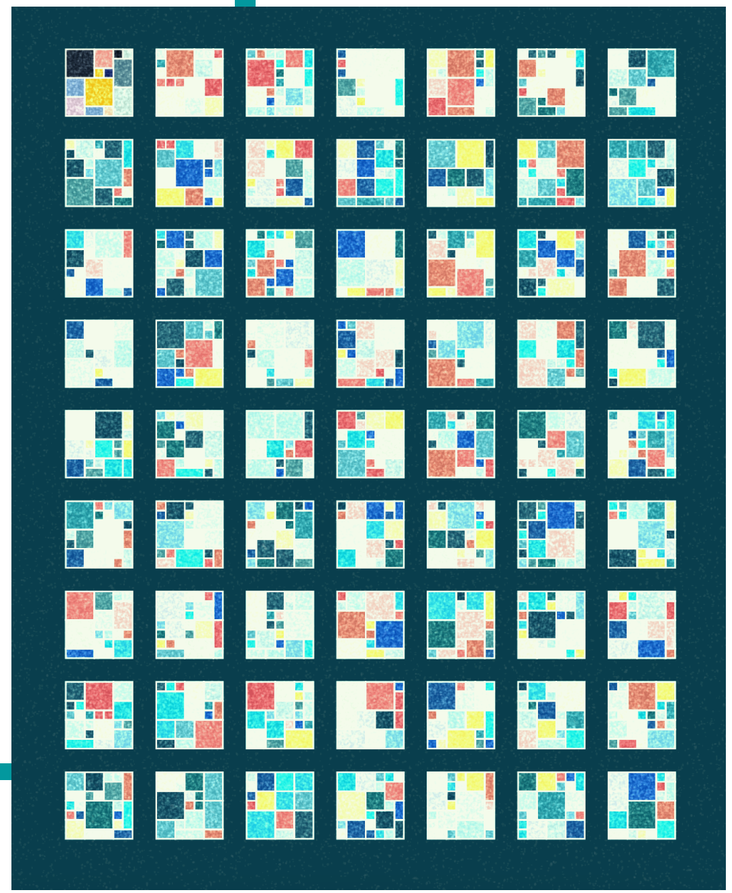
```
lerp(start, stop, amt)
# Calculates a number between two numbers
# at a specific increment.
```

Using Shapes



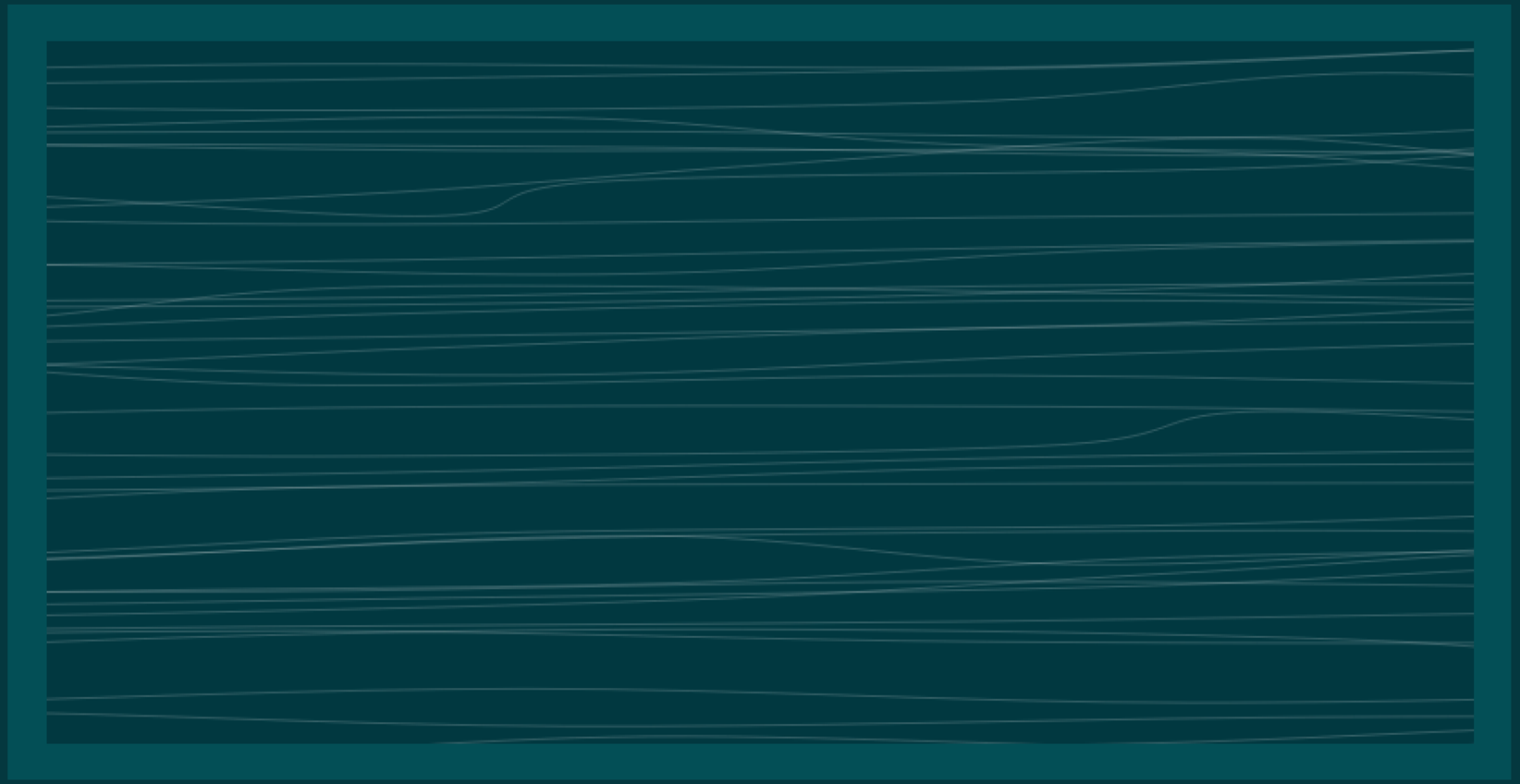
Piet Mondrian Eperiments





@NEERAJP99

PERLIN NOISE / SIMPLEX NOISE



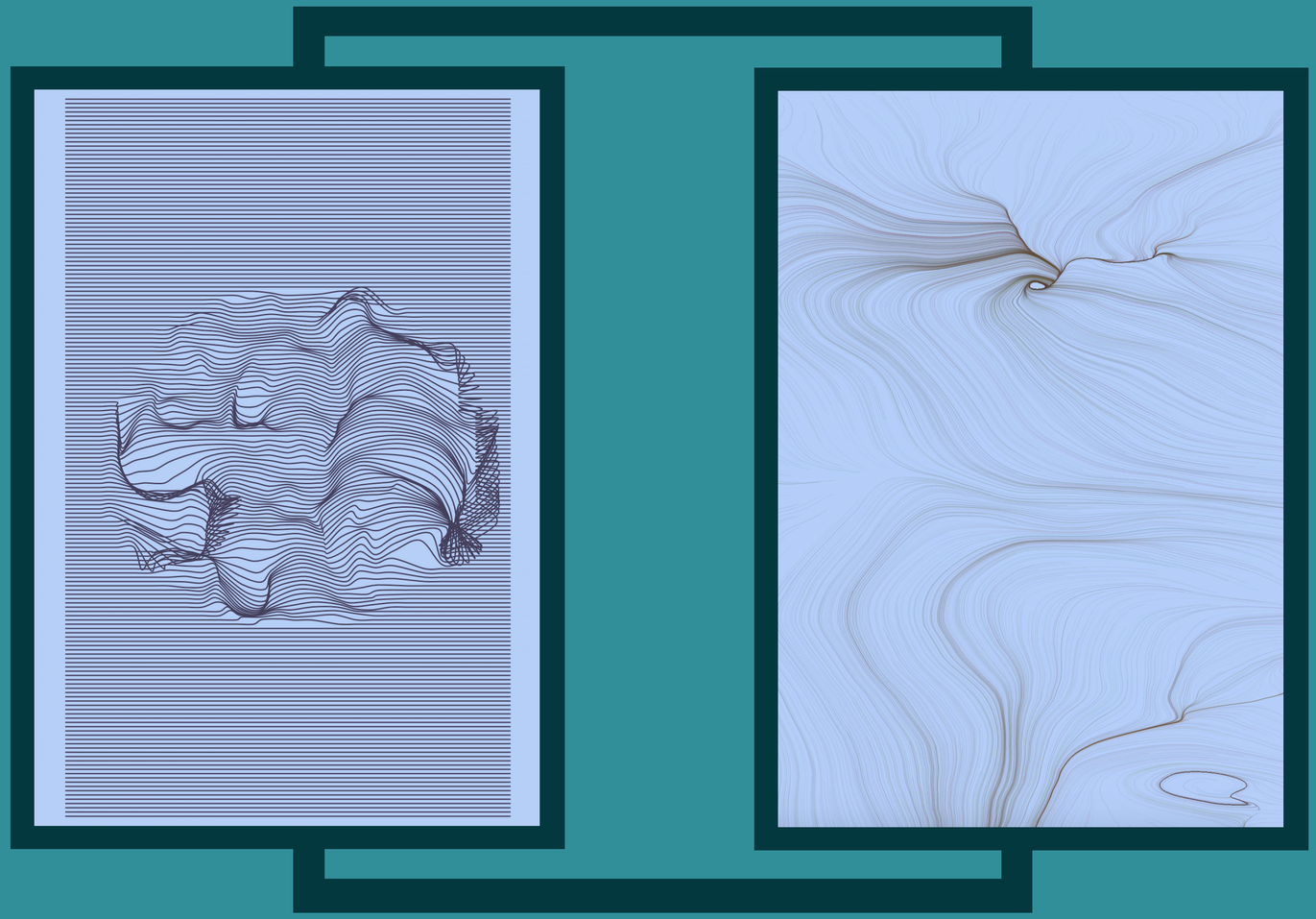
@NEERAJP99

PERLIN NOISE / SIMPLEX NOISE

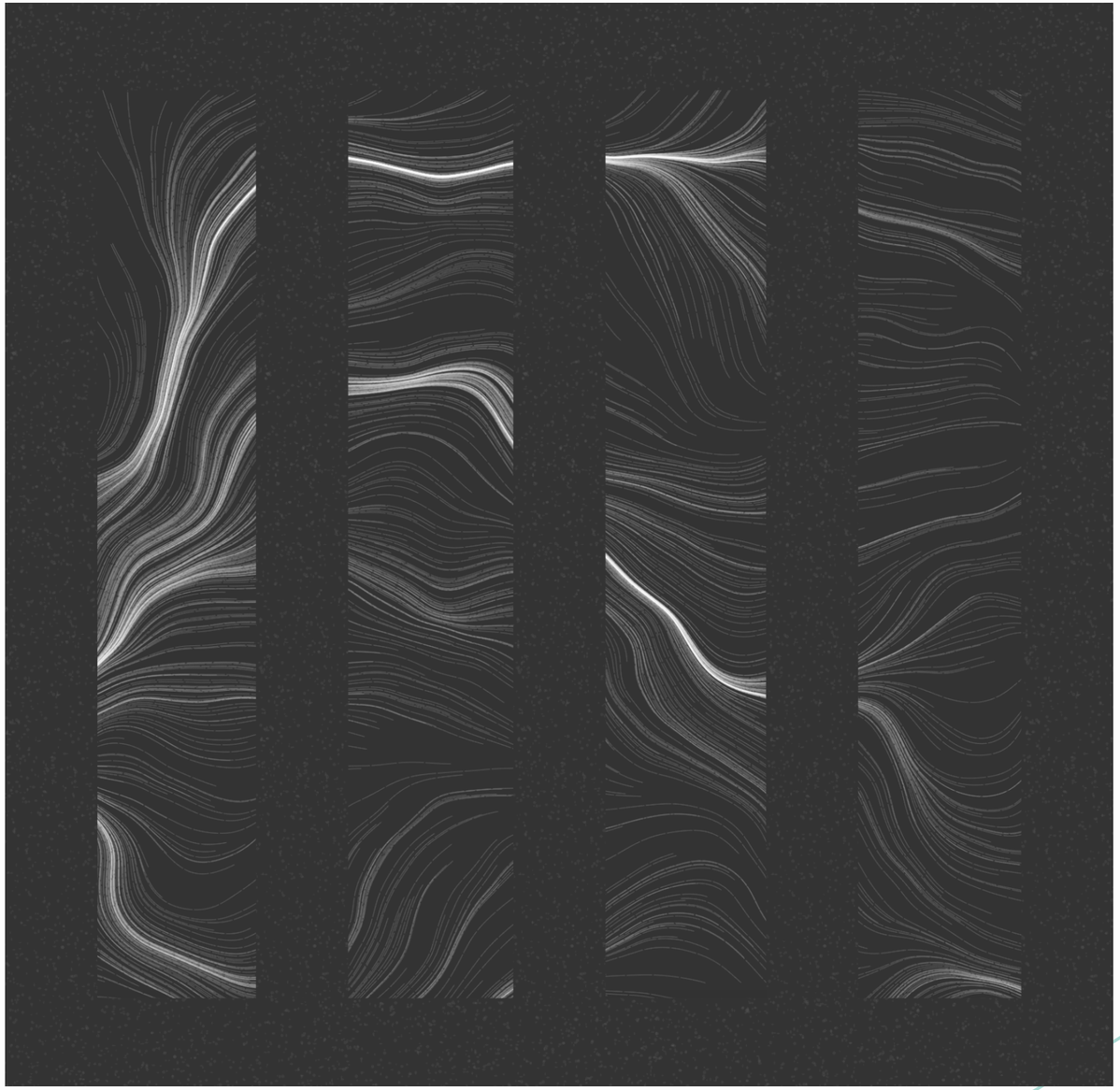
```
noise(x, [y], [z])
#  Returns the Perlin noise value
noiseSeed(seed)
# Sets the sees value for noise()
noiseDetail(lod, falloff)
# Adjusts the level of details produced
# by perlin noise
```

Without processing, use noise module in Python.
(https://pypi.org/project/noise/)

Perlin Noise Field



@NEERAJP99



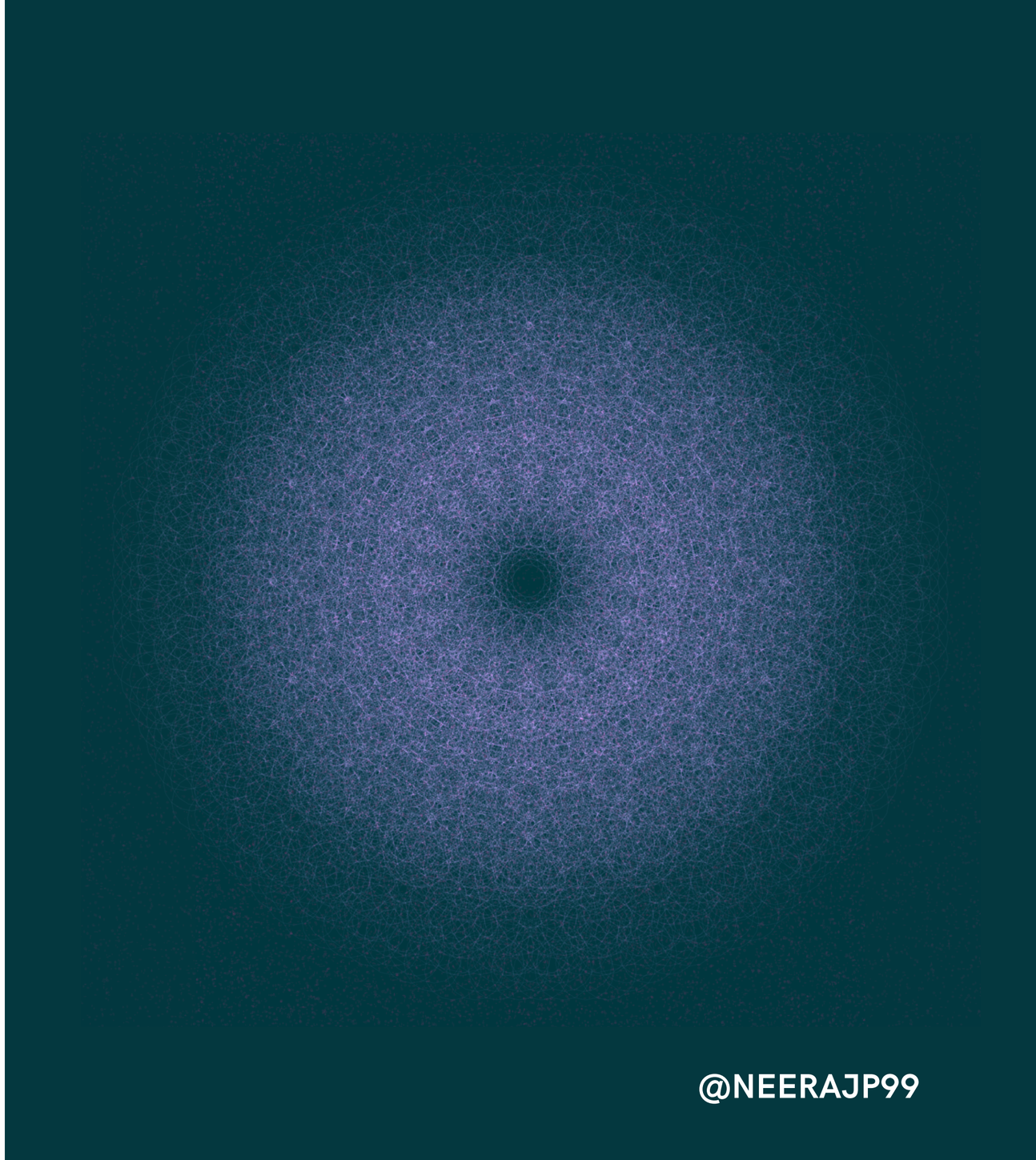
PERLIN NOISE

Using Perlin Noise Field and Perlin Noise generated random noise points(grain like texture).

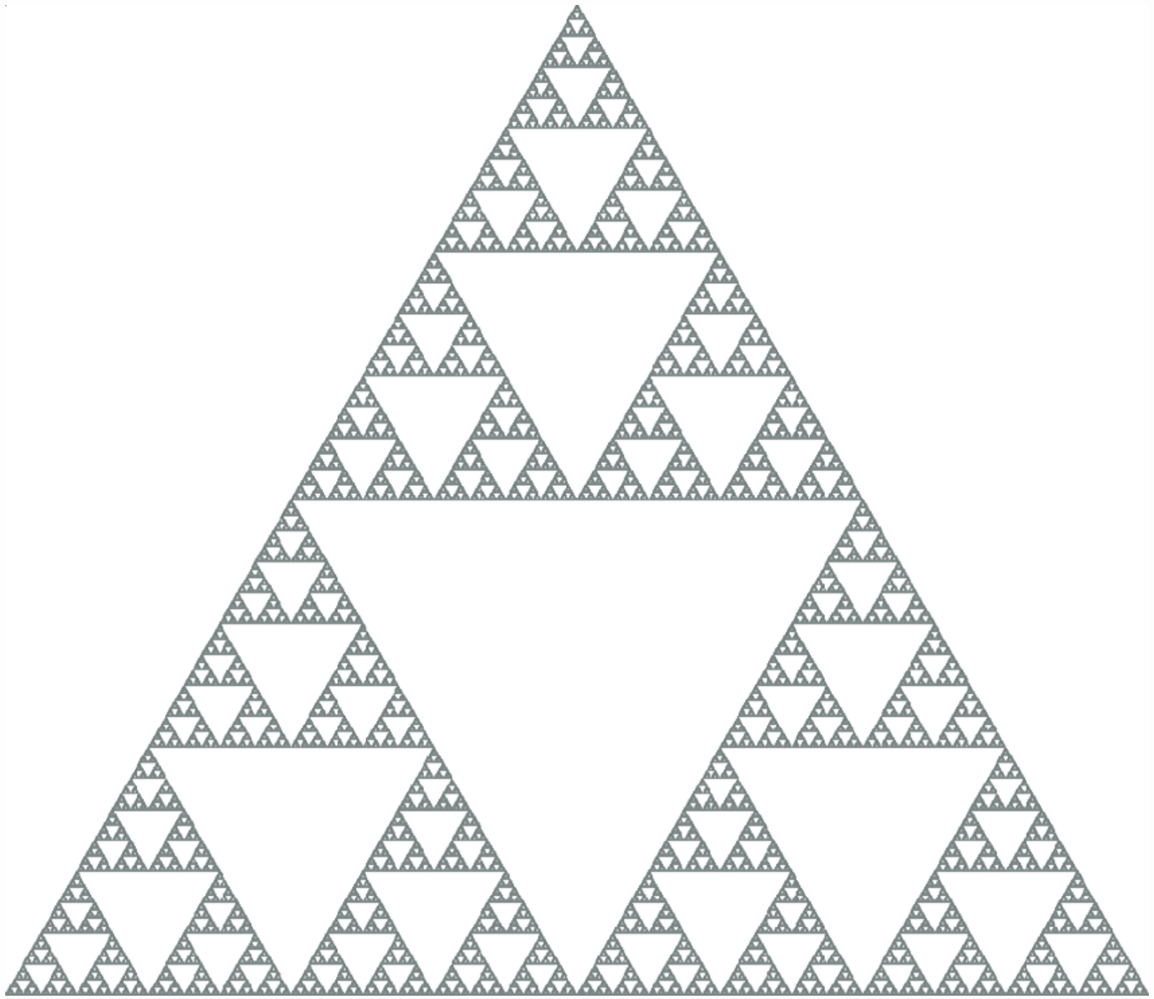
@NEERAJP99

GEOMETRY FRACTALS CHAOS

Using Geometrical patterns, fractals and chaos theory to generate aesthetic art pieces



@NEERAJP99

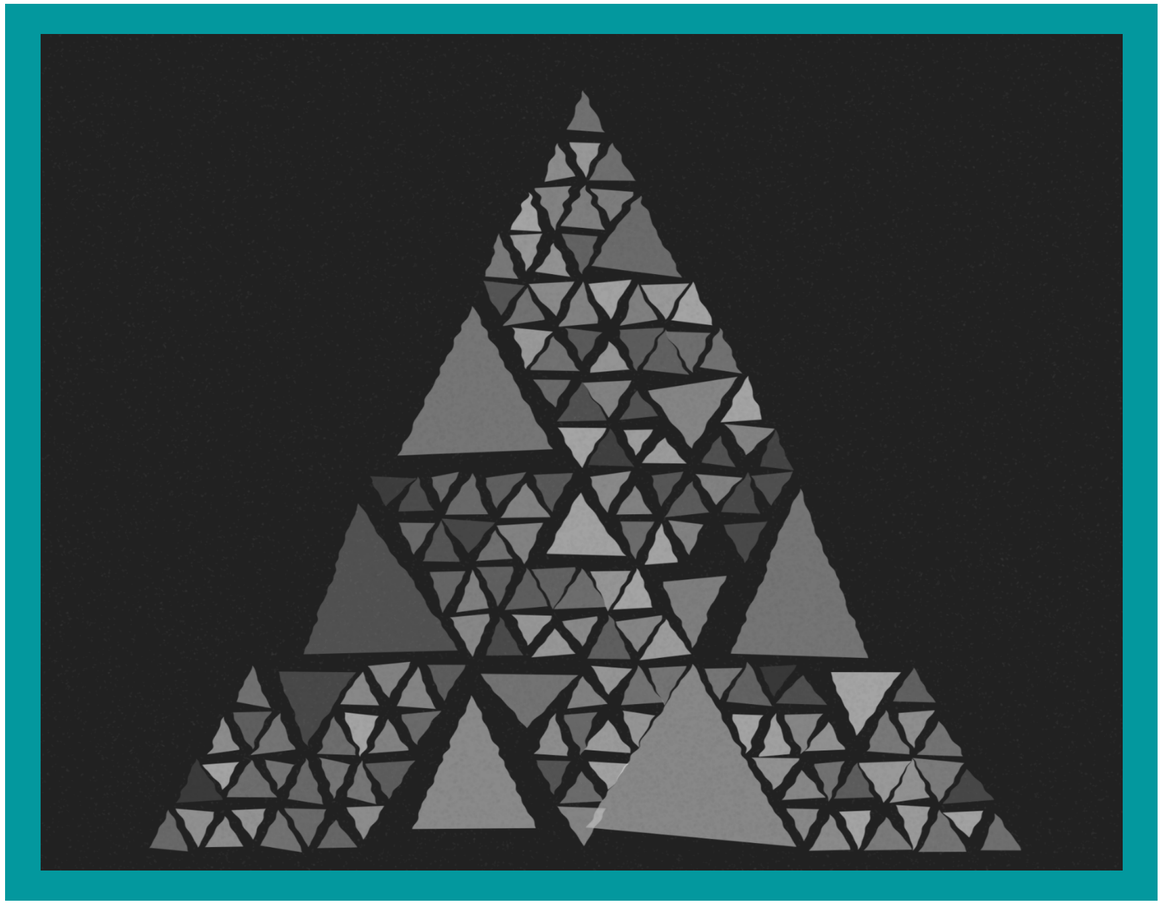


SIERPINSKI TRIANGLE

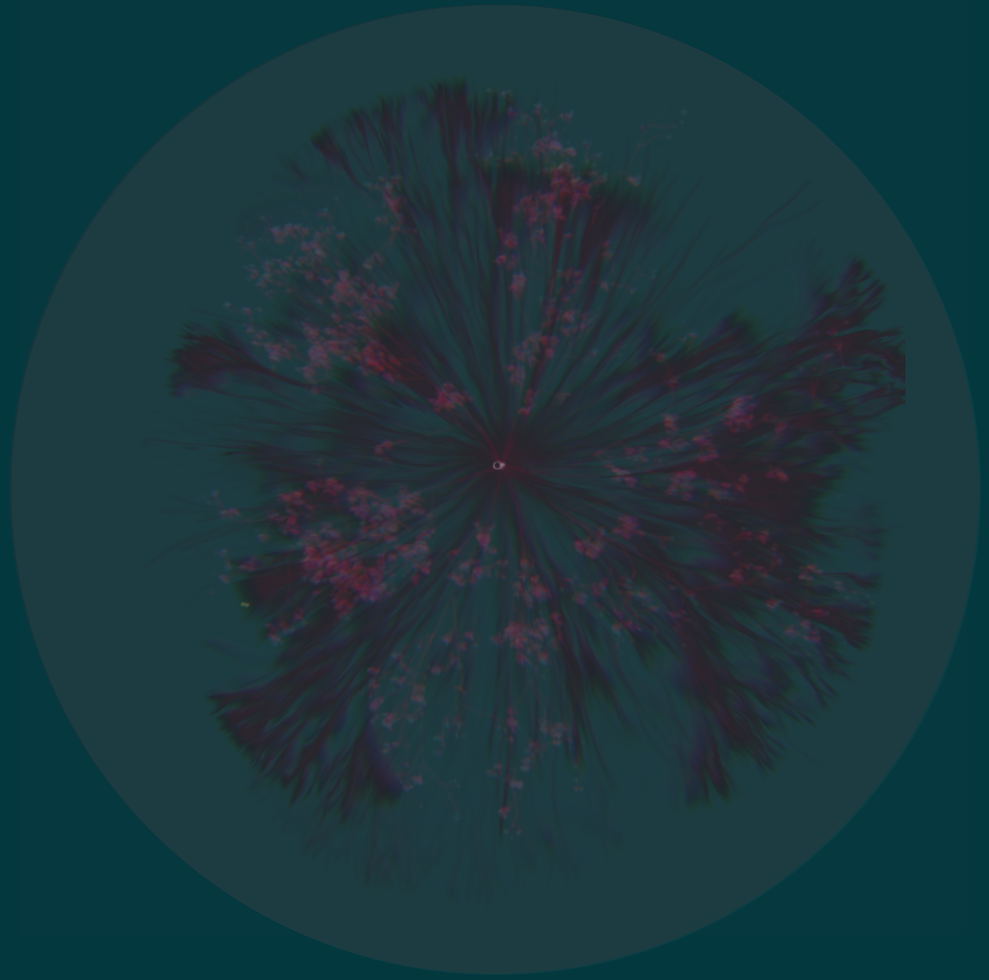
An equilateral triangle, subdivided recursively into smaller equilateral triangles with one recursive call each time.

@NEERAJP99

Modified Sierpinski Triangle

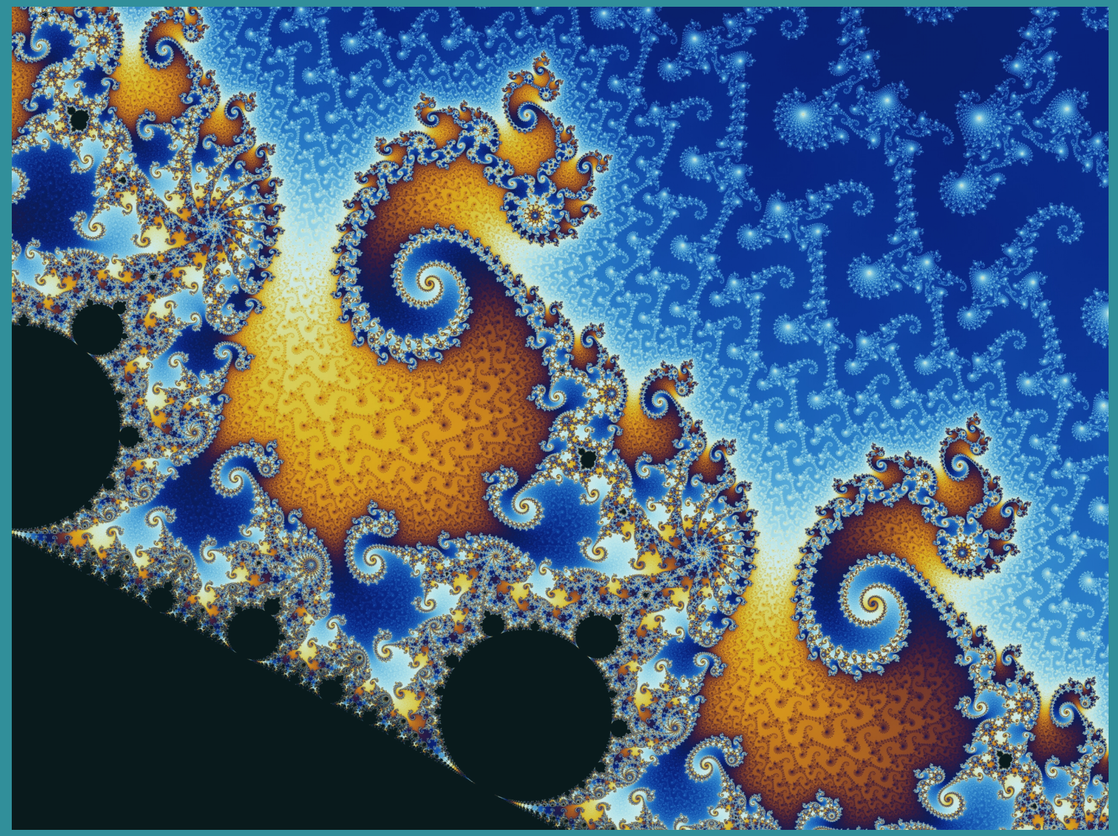


Fractal Flower



@NEERAJP99

Mandelbrot Set



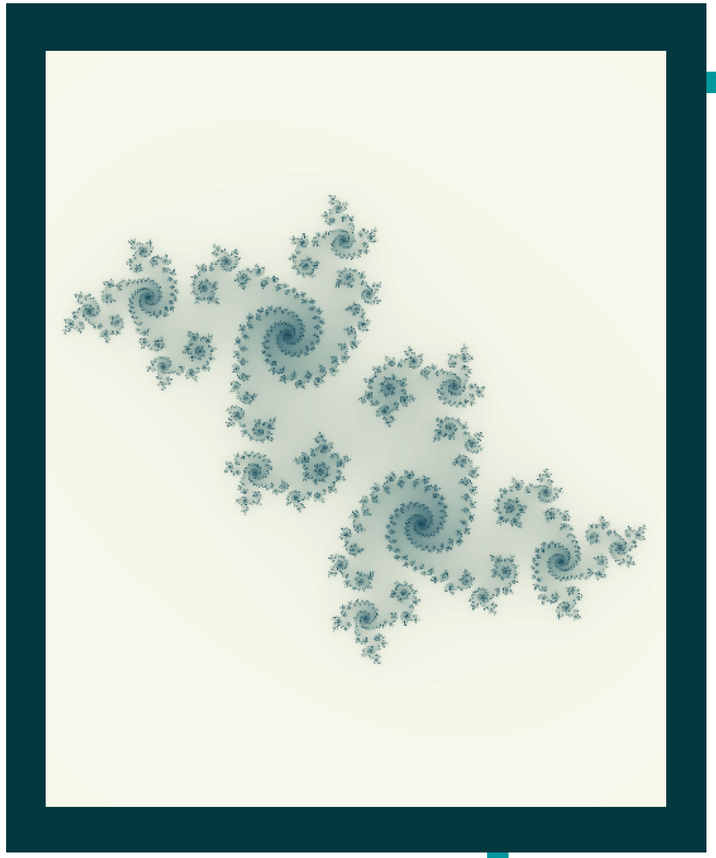
A geometrical figure where each part has the same statistical characters.

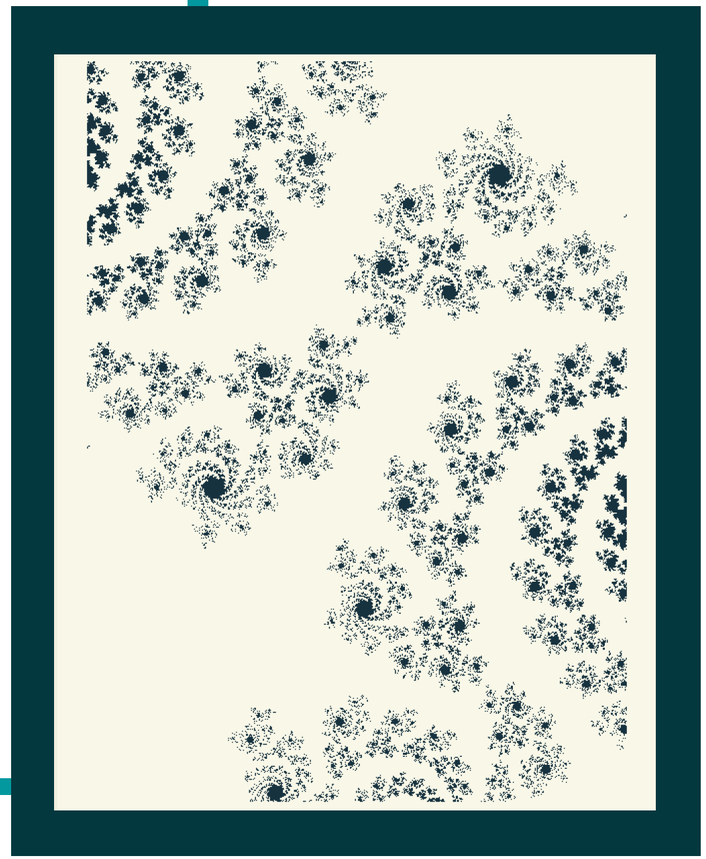
$$z = x + yi$$

$$z_{n+1} = z_n^2 + C$$

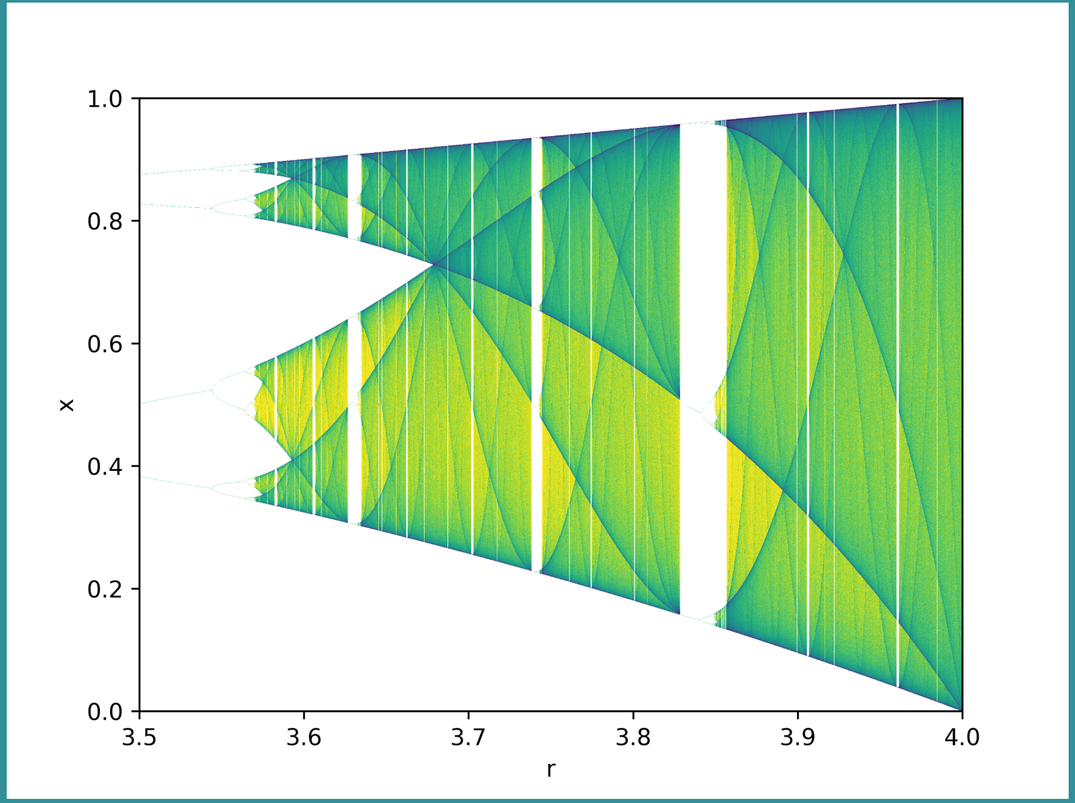
Source: Wikipedia

Julia Set





The Logistic Map

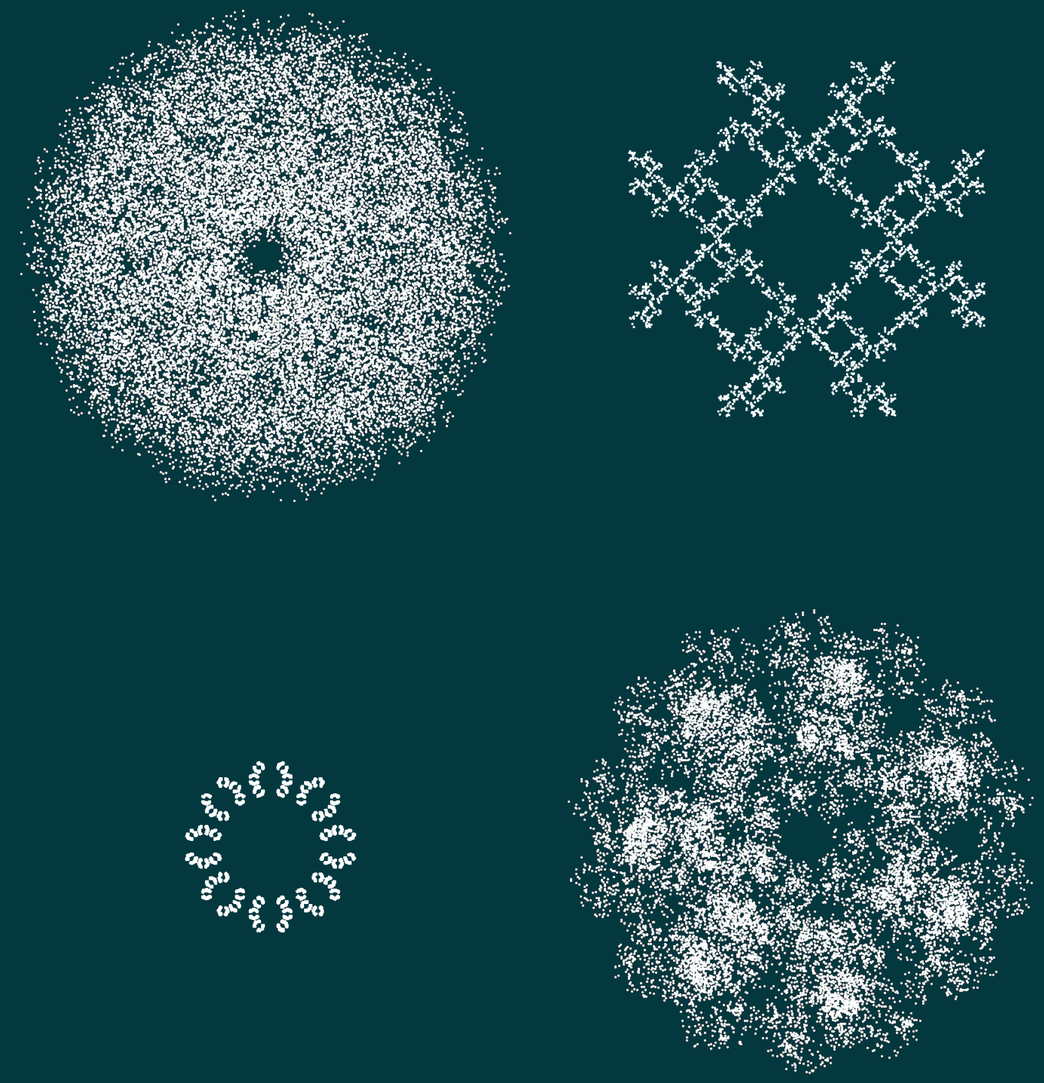


Source: Wikimedia

CHAOS THEORY

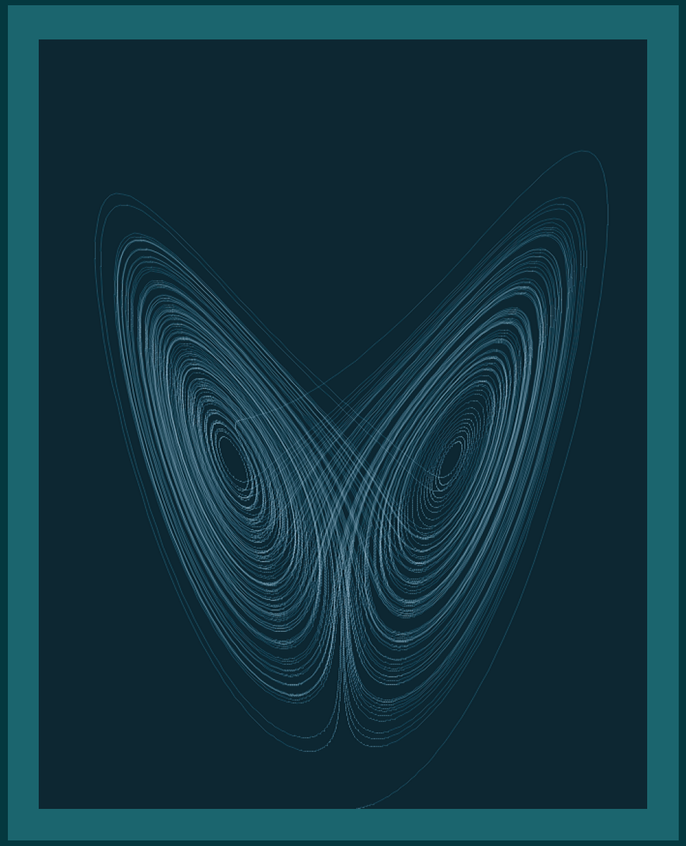
SOMETHING GOES HERE

Deterministic, unpredictable
A small change in the initial state can result
in very large difference in the final outcome

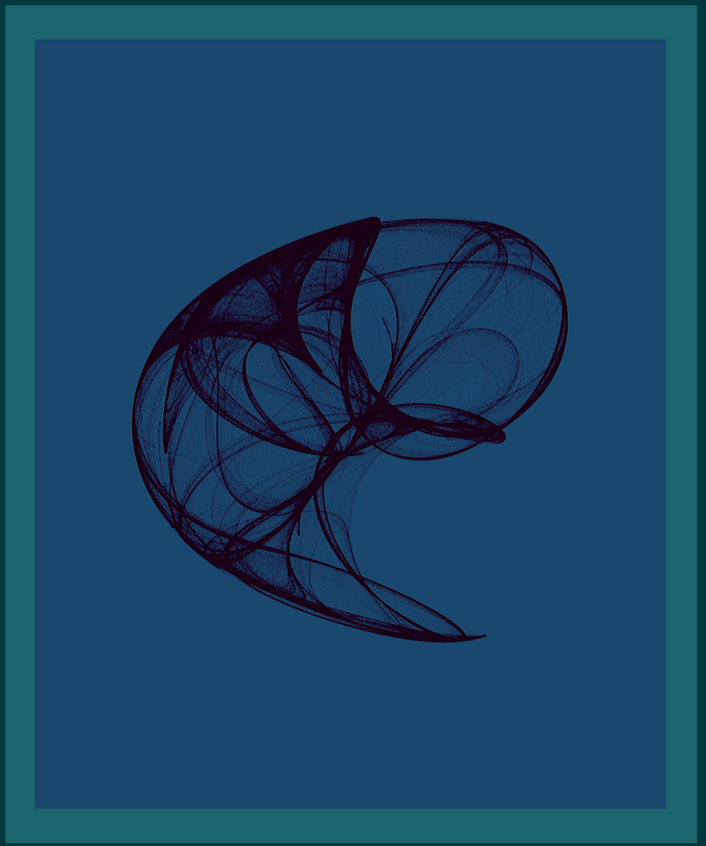


@NEERAJP99

Attractors



Lorenz system



De Jong Attractor

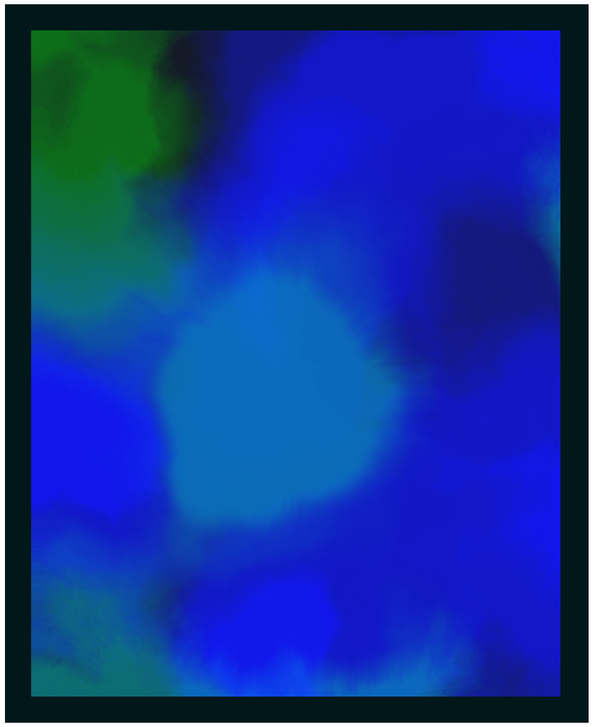
$$x_{t+1} = sin(a * yt) - cos(b * xt)$$
$$y_{t+1} = sin(c * xt) - cos(d * yt)$$

@NEERAJP99

SIMULATING PAINT

Creating oil, water color paint effeccts on our 2D/3D canvas.

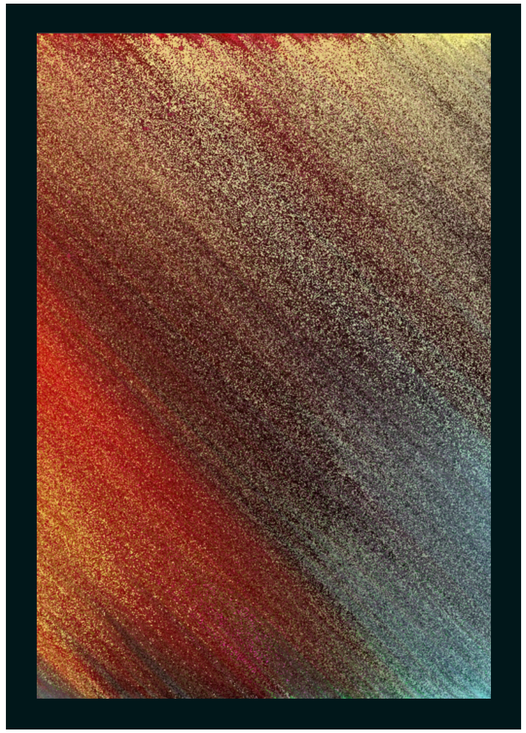
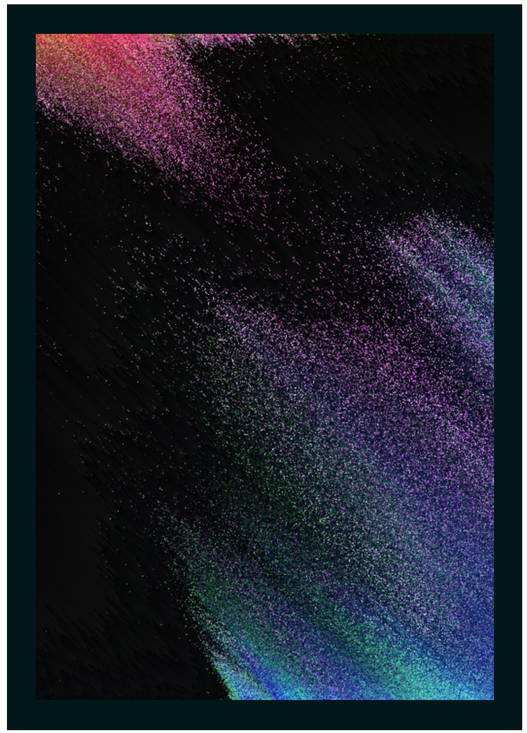
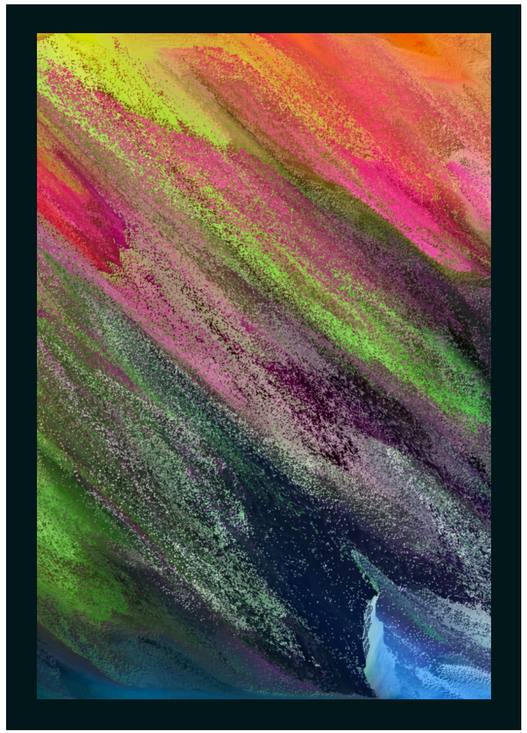
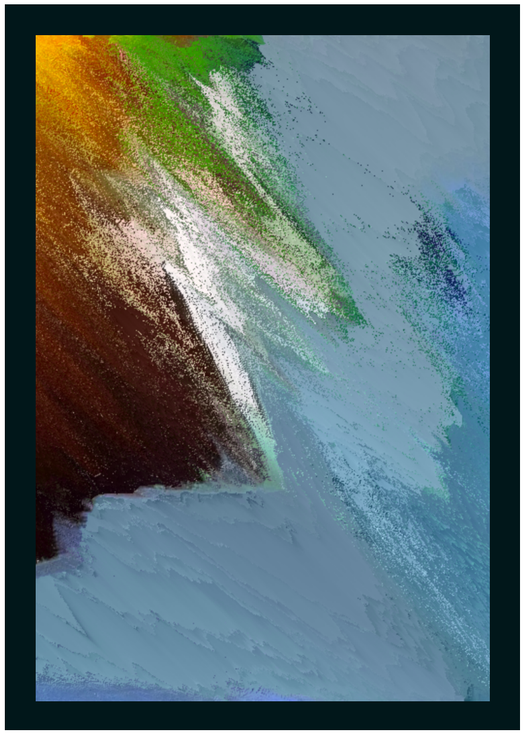
@NEERAJP99



Inspiration: https://tylerxhobbs.com/essays/2017/a-generative-approach-to-simulating-watercolor-paints

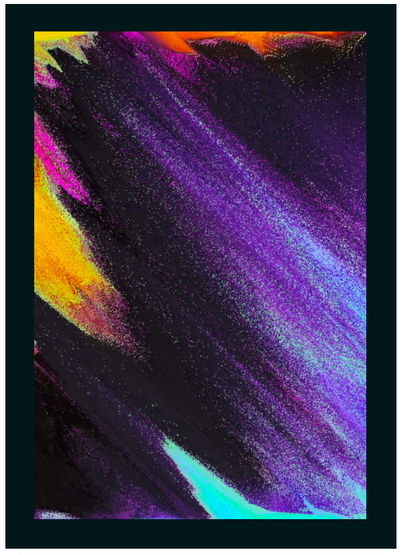
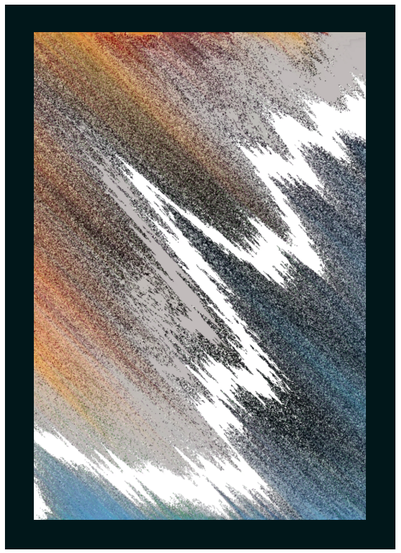
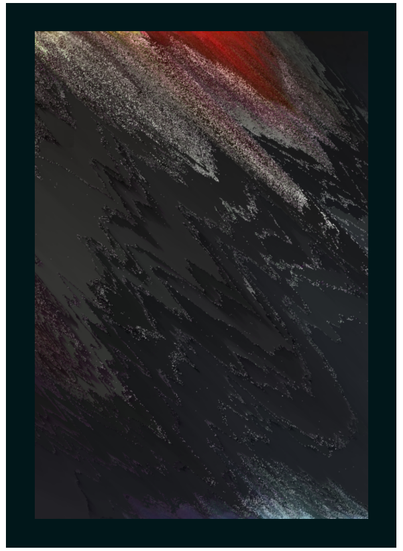
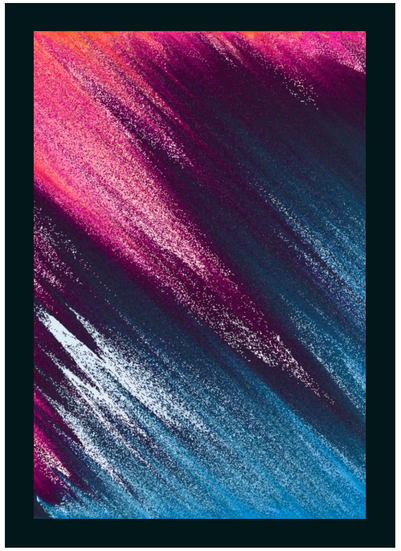
@NEERAJP99

Pixel Sorting Algorithms v1.0

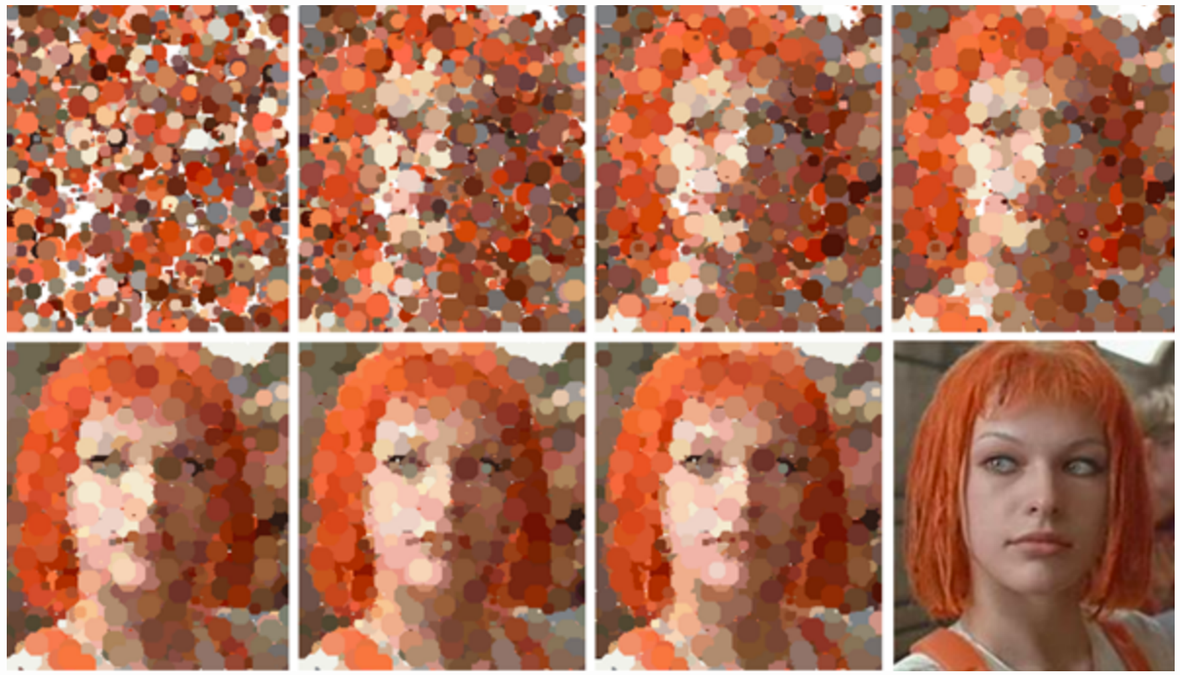





@NEERAJP99

Pixel Sorting Algorithms v2.0

@NEERAJP99



http://www.cosc.brocku.ca/%7Ebross/JNetic/

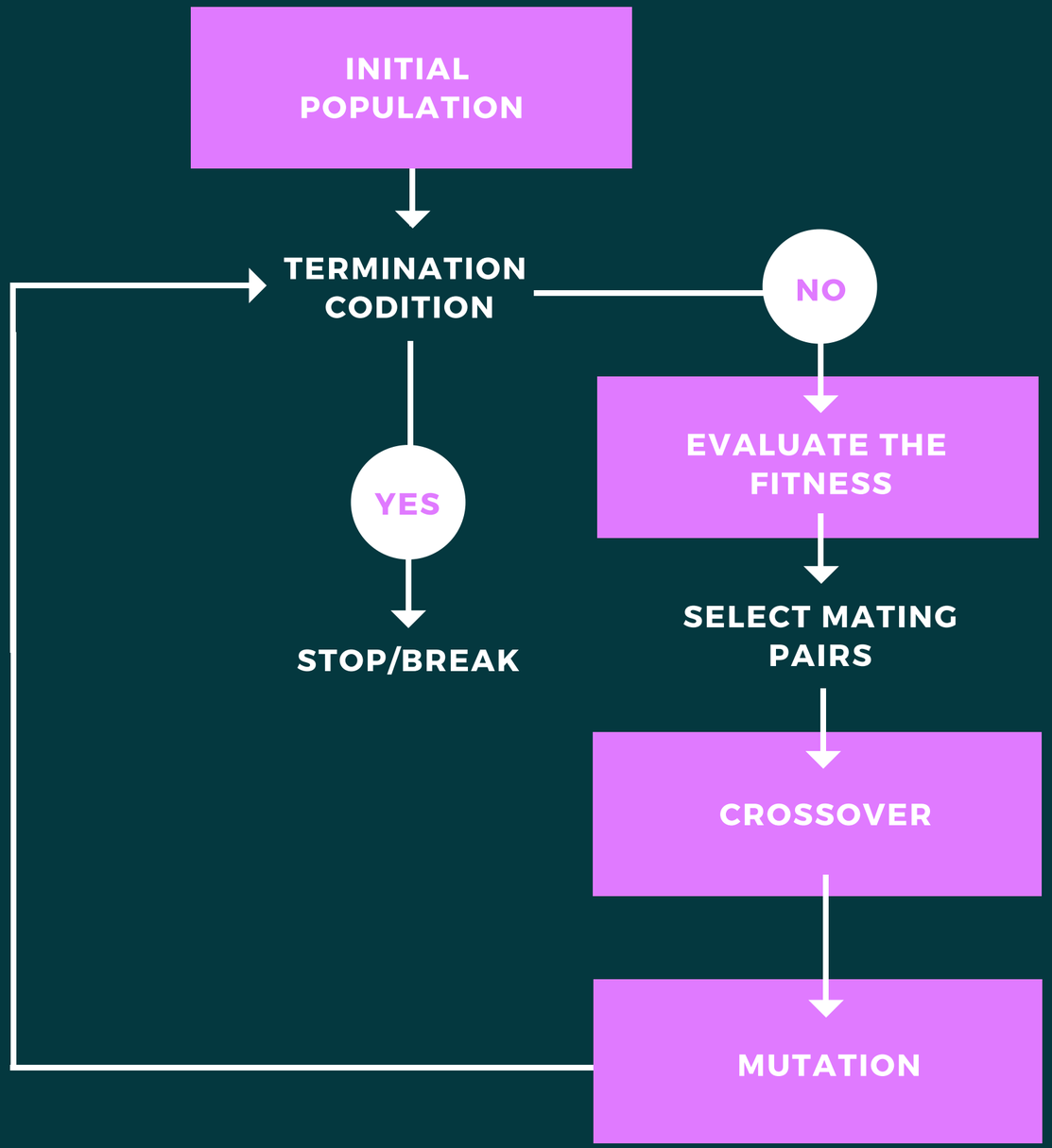
GENETIC ALGORITHMS

Creating evolutionary art that mimics the Darwinian laws of Natural Selection.

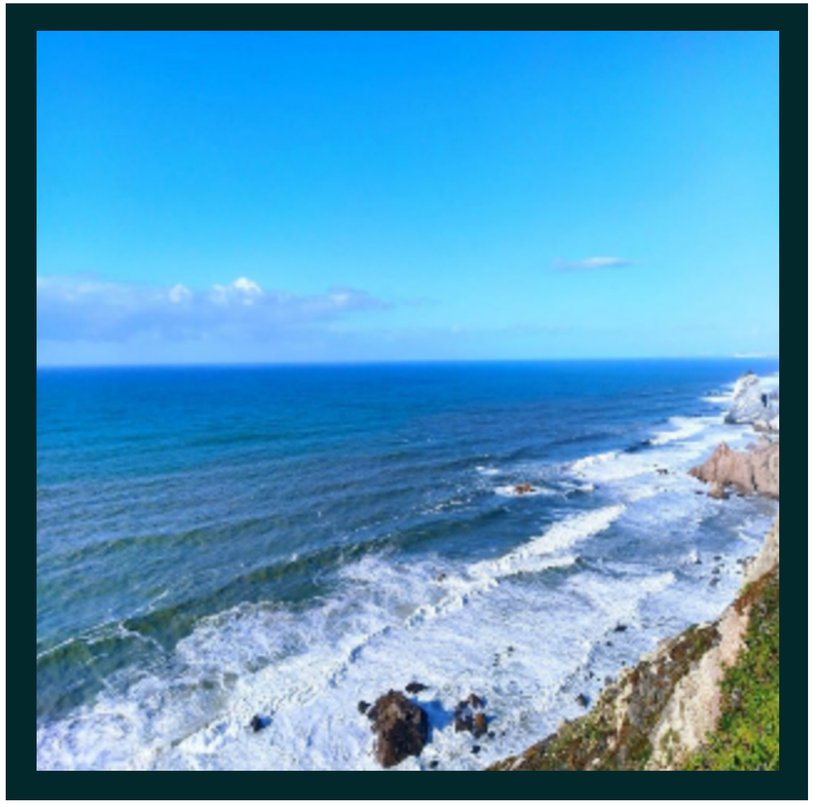
Steps includes:

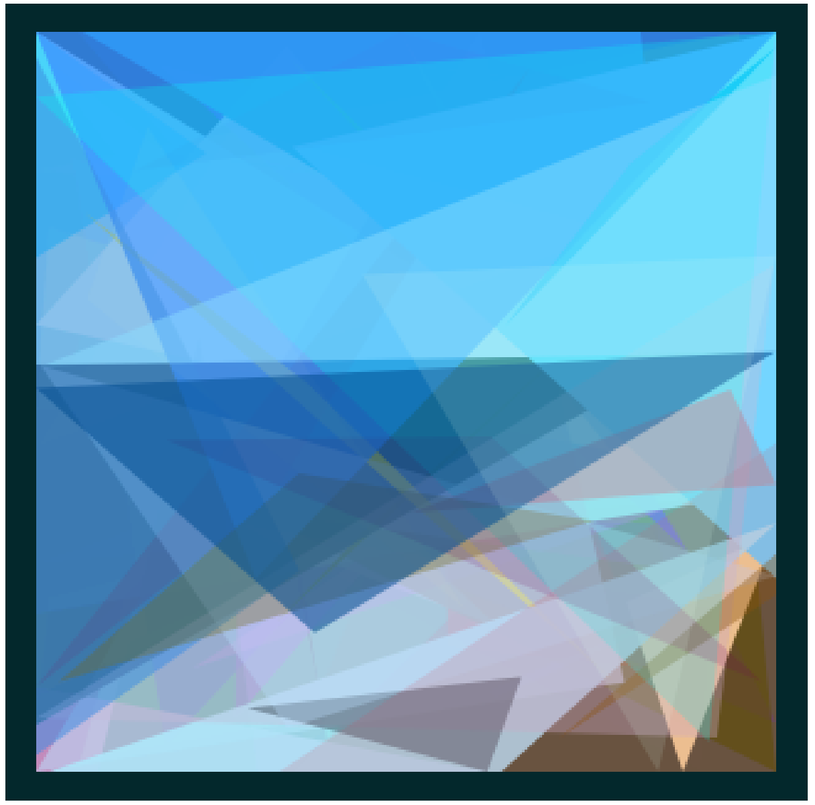
- Initial Population
- Selection (fitness value)
- Crossover
- Mutation

@NEERAJP99


INITIAL POPULATION
TERMINATION CODITION
NO
YES
STOP/BREAK
EVALUATE THE FITNESS
SELECT MATING PAIRS
CROSSOVER
MUTATION

EXAMPLE CREATED USING GENETIC ALGORITHMS





AI Fashion - Robbie Barrat, 2018

GENERATIVE ADVERSIAL NETWORKS

A system of two neural networks which run in competition with one another, the discriminator and generator. GANs are capable to capture and copy variations within a dataset.

@NEERAJP99

THANK YOU

 neerajp99