Computational Creativity

COULD AI BE THE NEXT DADA MOVEMENT?

Neeraj Pandey

NEERAJ PANDEY

@neerajp99

Generative Artist
Senior student at Ashoka University

Computational Arts, Quantitative Finance,
Full Stack Web and Data Science

POINTS FOR DISCUSSION

Dark Arts 2020, Neeraj Pandey

Computational Creativity

AI-Art Movement, The new Dada Movement

Literature

Genetic Algorithms

Generative Techniques

Effective Complexity

Music Intelligence

Generative Design

COMPUTATIONAL CREATIVTY

The use of computer technology to mimic, analyse, stimulate, and develop human creativity.

CREATIVITY?

“I saw the angel in the marble and carved until I set him free”

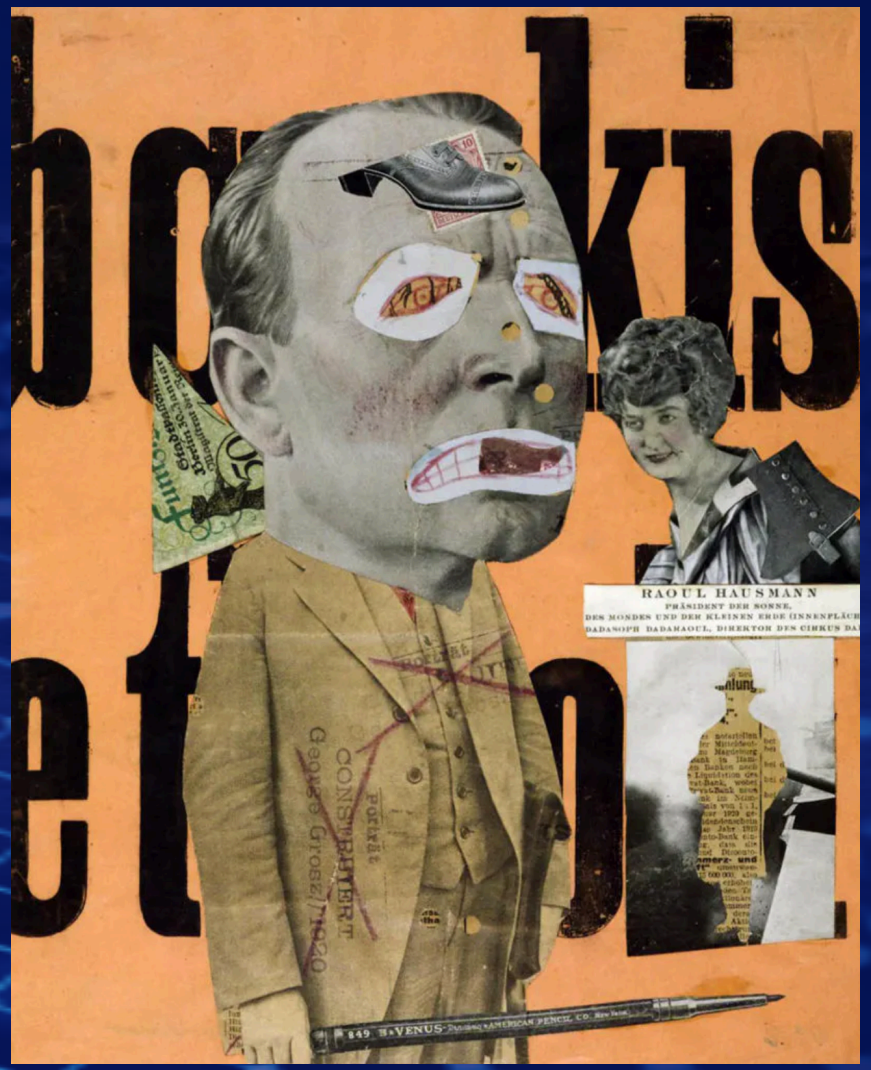
CREATIVITY?

"I saw the angel in the marble and carved until I set him free"

Novelty and Value

DADAISM

Fountain (Duchamp), 1917

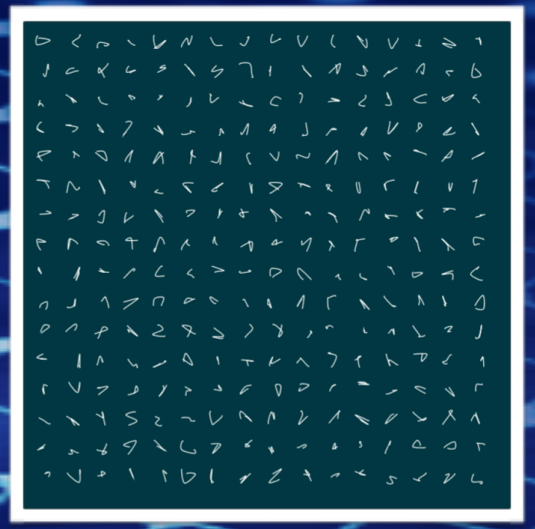


The Art Critic, 1919-20

The Mechanical Head (1920)

The new DADA Movement

COMPUTATIONAL CREATIVITY

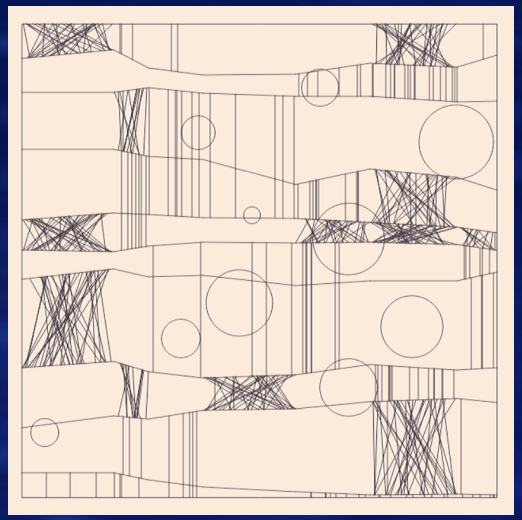


and left him worn them made them not allowed
to rain.' only mean a window glass
him hitched in an impulse was either cloud
i'm asked him to myself the kind to pass

was found the tail and stand perplexed behind
and if it seemed to sit and came to send
provision there a scribble >from the kind
for almost here estelle's to descend

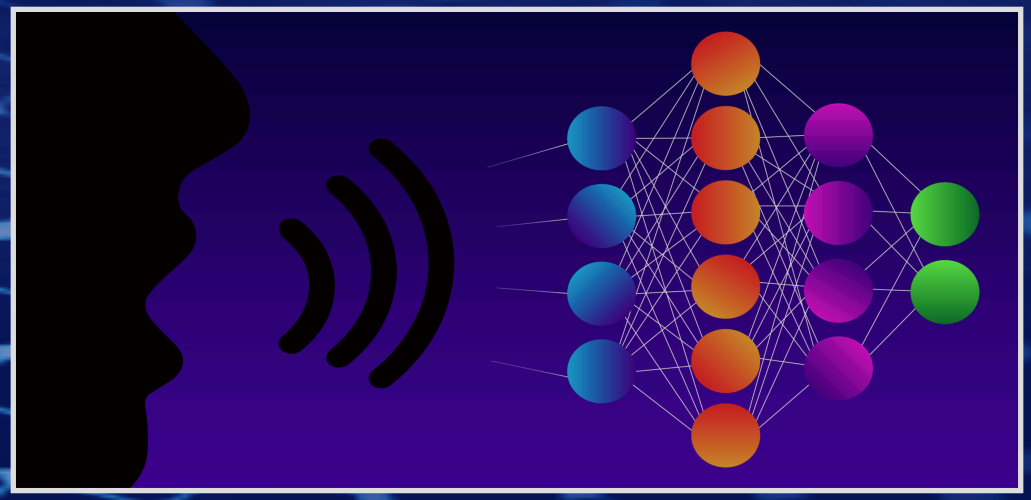
is but estelle to go with him a plane
it then becoming reconciled to yield
mebbe i'm her hardest hue to entertain
a time?' 'it looks surprised to get to field

above a barrel from another street
to give myself upstairs and dangle feet



Literature and Poetry

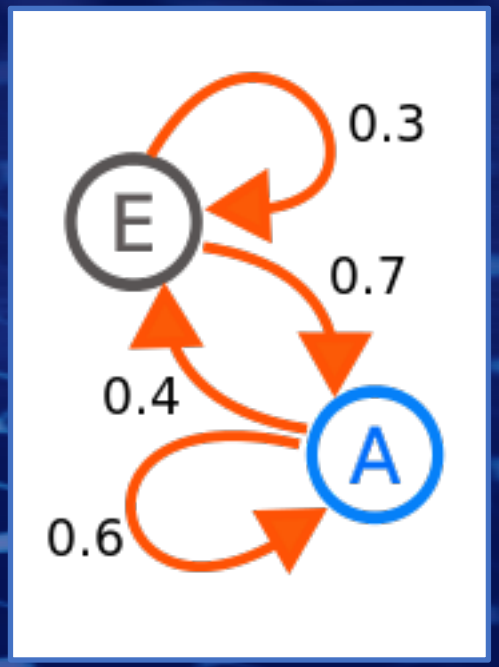
A type of writing in which words are used to create an image, sound, or emotion. Poetry is classified as a type of art since it has its own tone, form, picture, and rhythm.



Markov Chains

A Markov chain or Markov process is a stochastic model describing a sequence of possible events in which the probability of each event depends only on the state attained in the previous event.

~wikipedia



Poetry

When my mother died I was very young,

And my father sold me while yet my tongue

Could scarcely cry " 'weep! 'weep! 'weep! 'weep!"

So your chimneys I sweep & in soot I sleep.

Poetry

When my mother died I was very young,

And my father sold me while yet my tongue

Could scarcely cry " 'weep! 'weep! 'weep! 'weep!"

So your chimneys I sweep & in soot I sleep.

When

Poetry

When my mother died I was very young,

And my father sold me while yet my tongue

Could scarcely cry " 'weep! 'weep! 'weep! 'weep!"

So your chimneys I sweep & in soot I sleep.

When my

Poetry

When my mother died I was very young,

And my father sold me while yet my tongue

Could scarcely cry " 'weep! 'weep! 'weep! 'weep!"

So your chimneys I sweep & in soot I sleep.

When my

Poetry

When my mother died I was very young,

And my father sold me while yet my tongue

Could scarcely cry " 'weep! 'weep! 'weep! 'weep!"

So your chimneys I sweep & in soot I sleep.

When my father

Poetry

When my mother died I was very young,

And my father sold me while yet my tongue

Could scarcely cry " 'weep! 'weep! 'weep! 'weep!"

So your chimneys I sweep & in soot I sleep.

When my father sold

Poetry

When my mother died I was very young,

And my father sold me while yet my tongue

Could scarcely cry " 'weep! 'weep! 'weep! 'weep!"

So your chimneys I sweep & in soot I sleep.

When my father sold me while yet my

Poetry

When my mother died I was very young,

And my father sold me while yet my tongue

Could scarcely cry " 'weep! 'weep! 'weep! 'weep!"

So your chimneys I sweep & in soot I sleep.

When my father sold me while yet my

Poetry

When my mother died I was very young,

And my father sold me while yet my tongue

Could scarcely cry " 'weep! 'weep! 'weep! 'weep!"

So your chimneys I sweep & in soot I sleep.

When my father sold me while yet my tongue

Poetry

When my mother died I was very young,

And my father sold me while yet my tongue

Could scarcely cry " 'weep! 'weep! 'weep! 'weep!"

So your chimneys I sweep & in soot I sleep.

yet my tongue Could scarcely cry " 'weep! 'weep! 'weep!" So your chimneys I sleep.

Poetry

When my mother died I was very young,

And my father sold me while yet my tongue

Could scarcely cry " 'weep! 'weep! 'weep! 'weep!"

So your chimneys I sweep & in soot I sleep.

while yet my mother died I was very young, And my tongue Could scarcely cry " 'weep! 'weep! 'weep! 'weep! ‘weep!"

Creating a Markov chain from text file

```
def markov_chain(file):
    """Takes a file and returns a dictionary that maps
    words to a list of possible suffixes, which is also a dictionary.

    Args:
        file (str): name of file to read

    Returns:
        dict: dictionary of words to list of suffixes
    """
    # open file
    with open(file, 'r') as f:
        # read file
        text = f.read()
```

```python
def markov_chain(file):
    """Takes a file and returns a dictionary that maps
    words to a list of possible suffixes, which is also a dictionary.

    Args:
        file (str): name of file to read

    Returns:
        dict: dictionary of words to list of suffixes
    """
    # open file
    with open(file, 'r') as f:
        # read file
        text = f.read()
        # split text into words
        words = text.split()
        # create dictionary
        output = {}
```

```python
def markov_chain(file):
    """Takes a file and returns a dictionary that maps
    words to a list of possible suffixes, which is also a dictionary.

    Args:
        file (str): name of file to read

    Returns:
        dict: dictionary of words to list of suffixes
    """
    # open file
    with open(file, 'r') as f:
        # read file
        text = f.read()
        # split text into words
        words = text.split()
        # create dictionary
        output = {}
        # loop through words
        for i in range(len(words) - 1):
            # if word is not in dictionary
            if words[i] not in output:
                # add word to dictionary
                output[words[i]] = [words[i + 1]]
            # if word is in dictionary
            else:
                # add word to dictionary
                output[words[i]].append(words[i + 1])
        # return dictionary
        return output
```

```
{
  'When': ['my'],
  'my': ['mother', 'father', 'tongue'],
  'mother': ['died'],
  'died': ['I'],

  .
  .
  .
  .

  'in': ['soot'],
  'soot': ['I']
}
```

Generating text from the Markov chain

```
def random_text(language, length):
    """ Generates random text from a markov chain.

    Args:
        language (dict): dictionary of words to list of suffixes
        length (int): length of text to generate

    Returns:
        str: random text
    """
    keys = list(language.keys())  # create list of keys
    option = random.choice(keys) # get random key, set as the first word
    text = [] # create empty list
    for i in range(length): # loop through length of text
        if option not in language: # if word is not in dictionary
            break
        else: # if word is in dictionary
            text.append(option)
            option = random.choice(language[option]) # set random word
    return " ".join(text) # return text
```

Output

sweep very young, And my mother sold me while yet my mother Could scarcely cry " 'weep! 'weep! ‘weep!”

'weep! 'weep! 'weep! 'weep! 'weep! 'weep! 'weep! 'weep! 'weep!" 'weep!" 'weep! So your chimneys I sweep

scarcely cry " 'weep! 'weep!" So your chimneys I sleep. & in soot I sweep very young, And my father

my tongue died I sleep. & in soot I was

Poem generated from William Shakespeare’s sonnets

will if the death-bed whereon the perfumed
a gilded tomb of brass or being fond
is famish'd for it must expire consumed
bear'st love are the bett'ring of skill that bond

did give profitless usurer that vexed
be it to witness duty strongly knit
and ruined love was summer's welcome next
to woe the winter's ragged hand can sit

possession of fore-bemoaned moan the soil
and found or on the hours that besiege
in war is a dial-hand steal his spoil
that cannot know for cure the wrackful siege

and thou usurer why are you in dark
physic to grace a crow or eyes are mark

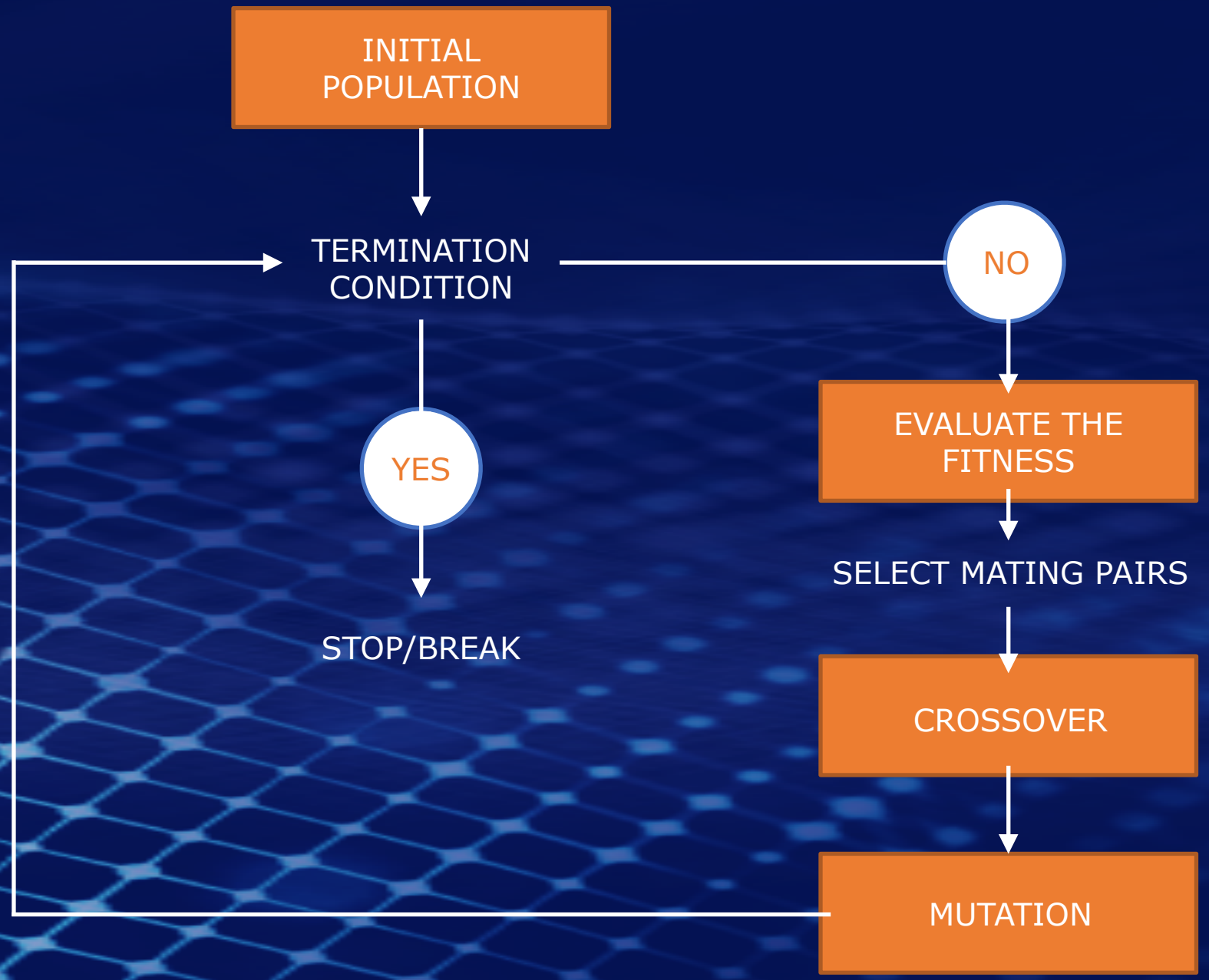
Genetic Algorithms

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

A method for solving both constrained and unconstrained optimisation problems based on a natural selection process that mimics biological evolution.

Five phases of GA(s):

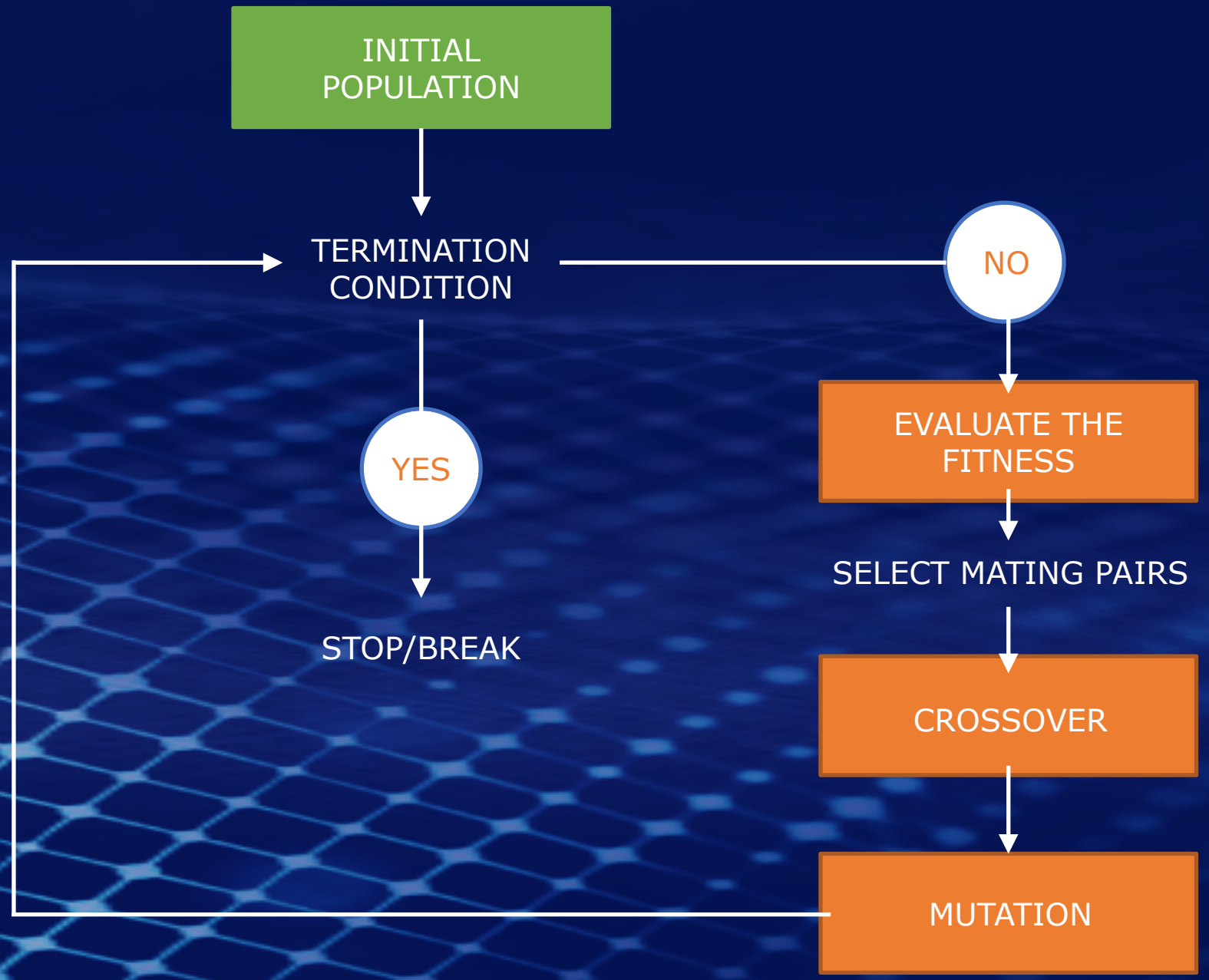
1. Initial Population
2. Fitness Function
3. Selection
4. Crossover
5. Mutation


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

OUTPUT USING GENETIC ALGORITHMS

“ab”

“jareenyednap”

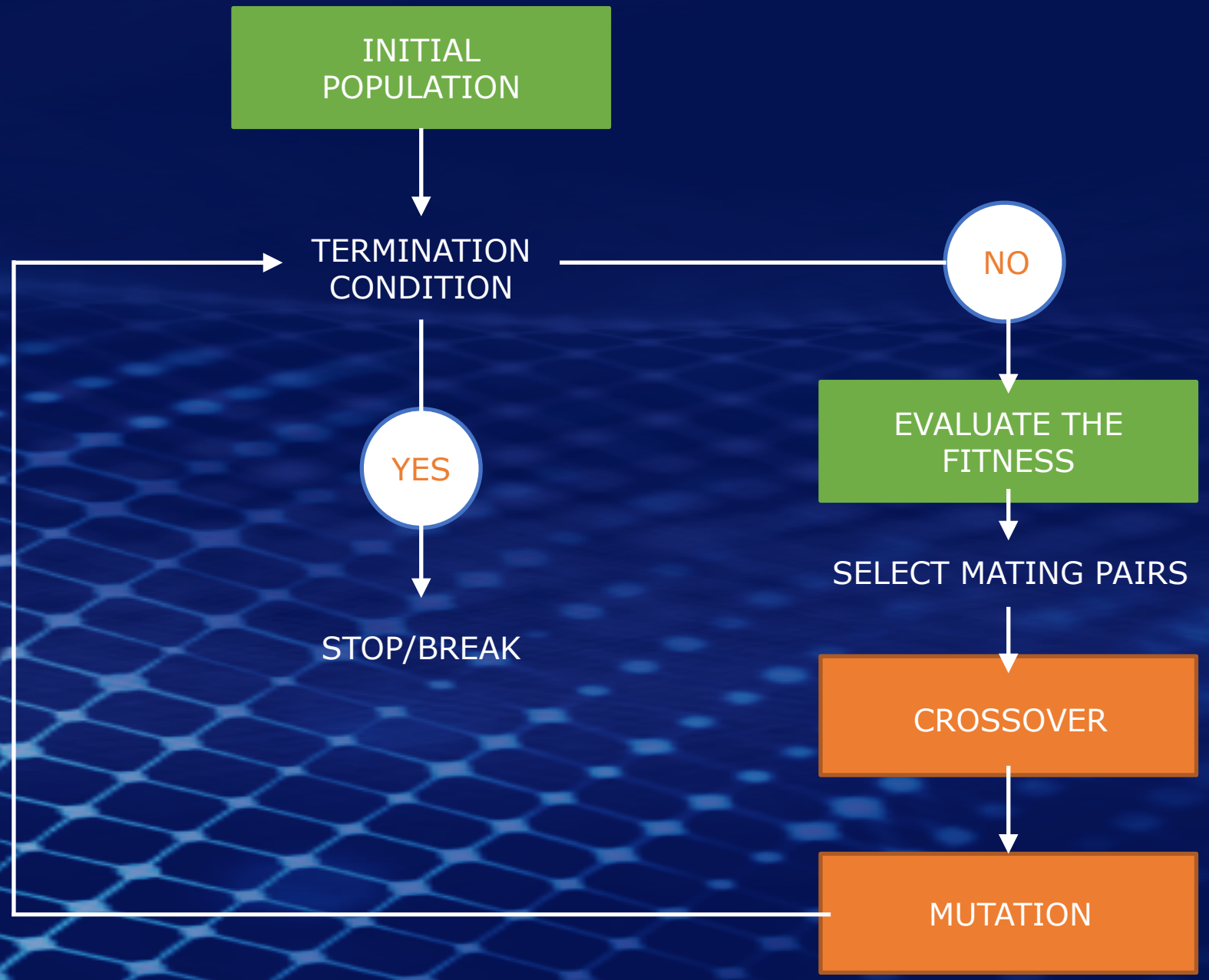

INITIAL POPULATION
TERMINATION CONDITION
NO
YES
EVALUATE THE FITNESS
SELECT MATING PAIRS
STOP/BREAK
CROSSOVER
MUTATION

Generate a random population for the pool

```
def random_population(size, gene_size):
    """Random population of `size` number of individuals of `gene_size` length
    Args:
        size (int): Number of individuals in population
        gene_size (int): The length of each individual
    Returns:
        List[str]: Population of individuals
    """
    population = []
    for i in range(size):
        individual = []
        for j in range(gene_size):
            individual.append(chr(int(random.randrange(32, 126, 1))))
        population.append("".join(individual))
    return population
```

Generate a random population for the pool

```
population = random_population(20, len('jareenyednap'))

[
 'OO6/DtUxJ=G&', '/xjkj>}F#xGr', '4.6"qd#F$tgH', 'XVb5lUx`k):u', '`2Nr03!@cP^f', 'b T`xpE)GAS*',
 '}]U1b_SY>57(', 'Px!2ZKJ>rKyt', '[5V$(kCf4,z7', 'gRQIu(*T7L|[', 'VL@yP.J=PghF', 'q@B6:(s ]10J',
 '^X`_\\_Kc2;BO', '>Eh9$fQYqCi`', ',rxXu;cT("km', "o'a1Q1zn|}Z>", '4ike{RR`l@6j', 'aMKY0wr)2atg',
 'kwT.:D8bh#<P', '2&Gy@Fh2|tH$'
]
```


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

Creating a fitness function

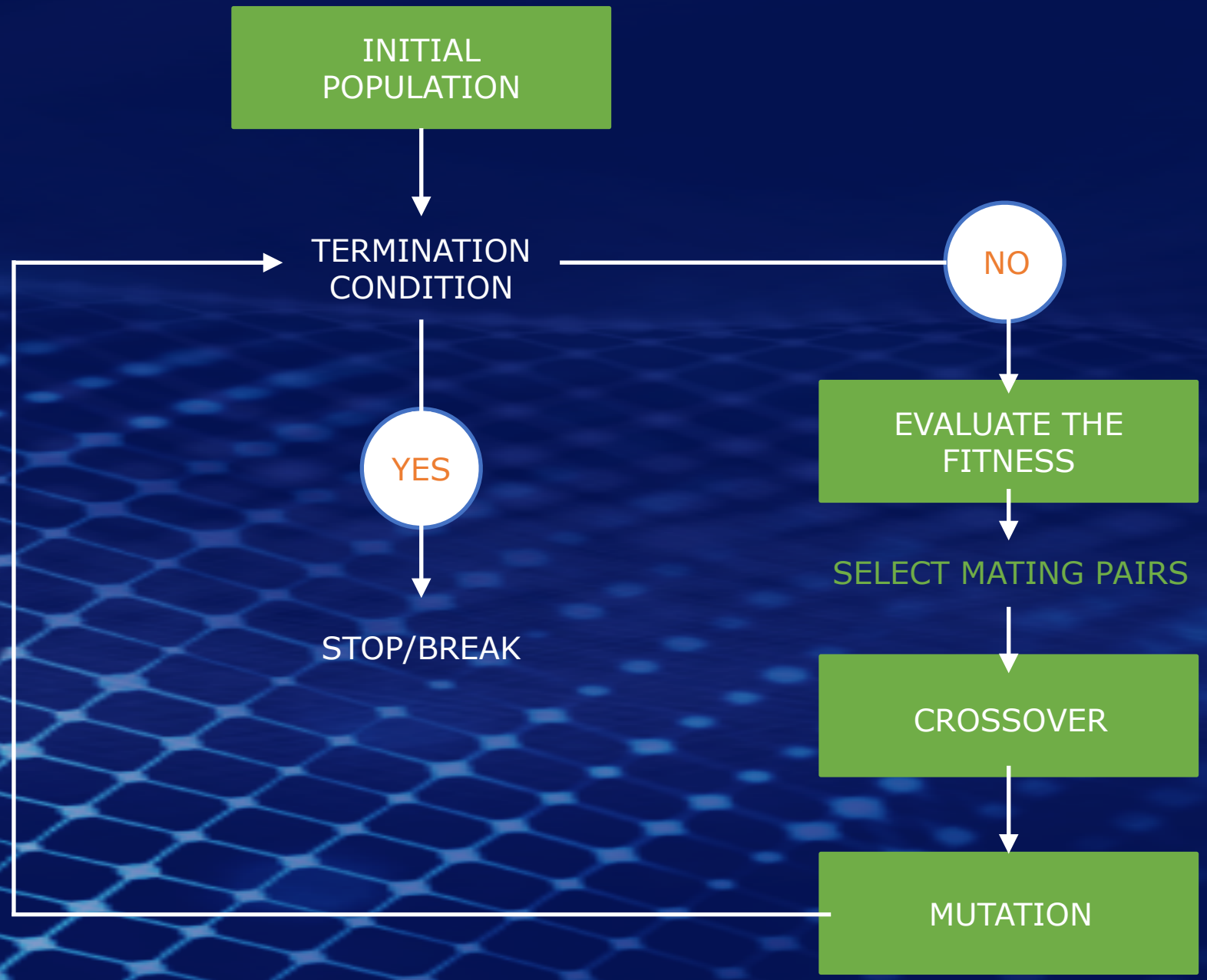
```python
def fitness_func(individual, target):
    """Calculate the fitness of an individual

    Args:
        individual (str): Individual chromosome
        target (str): Target string

    Returns:
        fitness: Fitness value
    """
    fitness = 0
    for index in range(12):
        fitness += abs(ord(individual[index]) - ord(target[index]))
    return fitness
```

Creating a fitness function

```python
def fit_fitness(population, fitness_func, target):
    """Fitness calculation

    Args:
        population (List[str]): Population of individuals
        fitness_func (func): Fitness function
        target (str): Target string
    Returns:
        fitness_list (List[tuple]): List of tuples of fitness and individual
    """
    fitness_list = []
    for individual in population:
        fitness_val = fitness_func(individual, target)
        if fitness_val == 0:
            fitness_list.append((individual, 1.0))
        else:
            fitness_list.append((individual, 1.0 / fitness_val))
    return fitness_list
```

Output of the fitness list

```
fitness_list = fit_fitness(population, fitness_func, 'jareenyednap')

[
  ('C}sSH?v1o@WV', 0.0032258064516129032),
  ('p5>1]\\c\\e=5Y', 0.0030487804878804878),
  ('/Tpw})OS?_Lg', 0.0030581039755351682),
  ('U*:x!C!7u4cH', 0.0019493177387914230),
  ('99!#nxaxS\\Ed', 0.0026809651474530830),

  .
  .
  .
  .

  ('Df^nOO+Lf#]X', 0.0030030030030030030),
  ("j;!UJ$I'].!E", 0.0019083969465648854)
]
```


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

Get a random parent for crossover

```python
def get_random_individual(fitness_list):
    """Get a random individual from fitness list

    Args:
        fitness_list (List[tuple]): List of tuples of fitness and individual
    Returns:
        individual (str): Random individual
    """
    fitness_sum = sum(fitness for individual, fitness in fitness_list)
    n = random.uniform(0, fitness_sum)
    for value, fitness in fitness_list:
        if n < fitness:
            return value
        else:
            n -= fitness
    return value
```

Crossover between the two parent individual

```python
def crossover(parent1, parent2):
    """Crossover between two parent individuals

    Args:
        parent1 (str): First parent
        parent2 (str): Second parent
    Returns:
        child1 (str): First child
        child2 (str): Second child
    """
    # Get random crossover point
    crossover_point = random.randrange(1, len(parent1))

    # Crossover parents
    child1 = parent1[:crossover_point] + parent2[crossover_point:]
    child2 = parent2[:crossover_point] + parent1[crossover_point:]

    return child1, child2
```

Add mutation to an individual

```
def mutation(individual, gene_size, mutation_rate):
    """Add mutation to individual

    Args:
        individual (str): Individual chromosome
        gene_size (int): The length of each individual
        mutation_rate (float): Mutation rate

    Returns:
        individual (str): Mutated individual
    """
    for index in range(len(individual)):
        if random.random() <  mutation_rate:
            individual = individual[:index] + chr(random.randint(32, 126)) +
individual[index+1:]
    return individual
```

Applying selection, crossover and mutation

```python
def selection_mutation(population, fitness_func, size, fitness_list):
    """Selection and mutation

    Args:
        population (List[str]): Population of individuals
        fitness_func (func): Fitness function
        size (int): Size of population
        fitness_list (List[tuple]): List of tuples of fitness and individual

    Returns:
        population (List[str]): New population
    """
    # Select individuals
    population = []
    for index in range(size):
        parent1 = get_random_individual(fitness_list)
        parent2 = get_random_individual(fitness_list)

        # Crossover parents
        child1, child2 = crossover(parent1, parent2)

        # Apply mutation
        child1 = mutation(child1, 12, 0.05)
        child2 = mutation(child2, 12, 0.05)
        population.extend([child1, child2])
    return population
```

ALL TOGETHER

```
if __name__ == "__main__":
    target_string = 'jareenyednap'
    population = random_population(20, len(target_string))

    for i in range(10000):
        fitness_list = fit_fitness(population, fitness_func, target_string)
        population = selection_mutation(population, fitness_func, int(len(population) / 2),
fitness_list)

    threshold_fitness = fitness_func(population[0], target_string)

    for individual in population:
        individual_fitness = fitness_func(individual, target_string)
        if individual_fitness <= threshold_fitness:
            fittest_string = individual
            threshold_fitness = individual_fitness
    print('Final string: ', fittest_string)
```

TARGET: “jareenyednap”

```
[
 'OO6/DtUxJ=G&', '/xjkj>}F#xGr', '4.6"qd#F$tgH', 'XVb5lUx`k):u', '`2Nr03!@cP^f', 'b T`xpE)GAS*',
 '}]U1b_SY>57(', 'Px!2ZKJ>rKyt', '[5V$(kCf4,z7', 'gRQIu(*T7L|[', 'VL@yP.J=PghF', 'q@B6:(s ]10J',
 '^X`_\\_Kc2;BO', '>Eh9$fQYqCi`', ',rxXu;cT("km', "o'a1Q1zn|}Z>", '4ike{RR`l@6j', 'aMKY0wr)2atg',
 'kwT.:D8bh#<P', '2&Gy@Fh2|tH$'
]
```

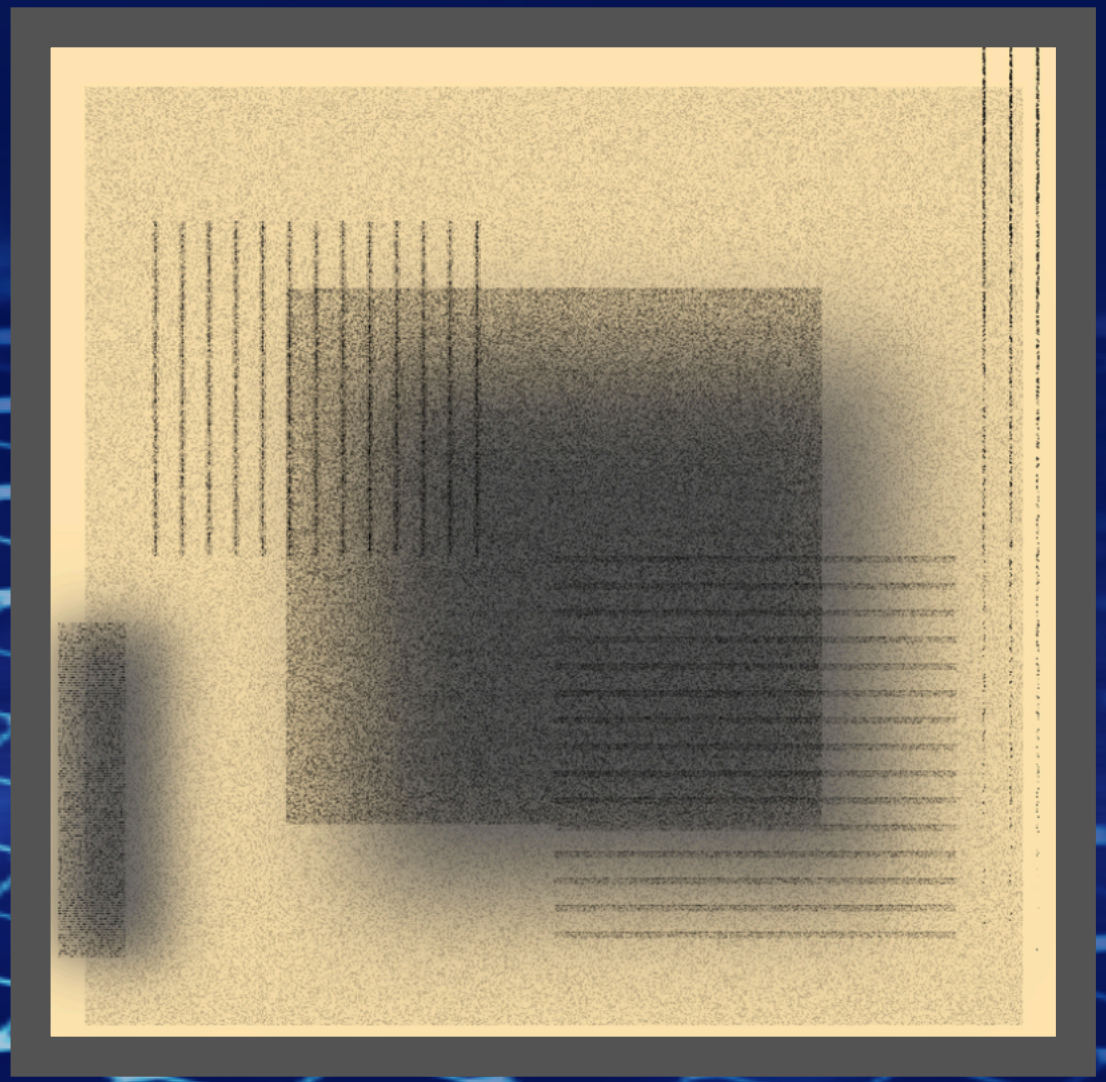
```
Generation 0: fyYEAq2%I'27
Generation 1: U9"cUuu;[[;o
Generation 2: <bdm=fEUl[;o

.
.
.

Generation 3959: jareenxednap
Generation 3960: jtreenxednap
Generation 3961: jareenxed8ap
Generation 3962: jareenxednap

.
.
.

Generation 9994: jareenyfdnap
Generation 9995: jareenyednap
Generation 9996: jareenyfdnap
Generation 9997: jareenyfdnaI
Generation 9998: jareenyfdnap
Generation 9999: jareenyednap
Generation 10000: jareenyednap
```

GENERATIVE ART

Art created through the use of an autonomous system.

ALGORITHMS, MATHEMATICS, GENETIC SEQUENCES

[Chroma, Neeraj Pandey]



[Cityscapes, Neeraj Pandey]

Building Blocks

- **Randomness**: This helps to add uniqueness to art pieces.
- **Algorithms**: Implementing an algorithm visually often generate aesthetic art pieces.
- **Geometry**: Most of the generative art incorporates shapes, and mathematics, even the high school geometry.

RANDOM

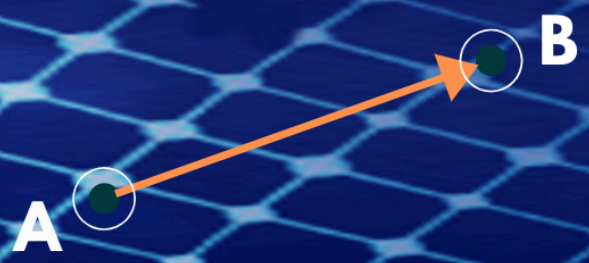
Generates random floating point number

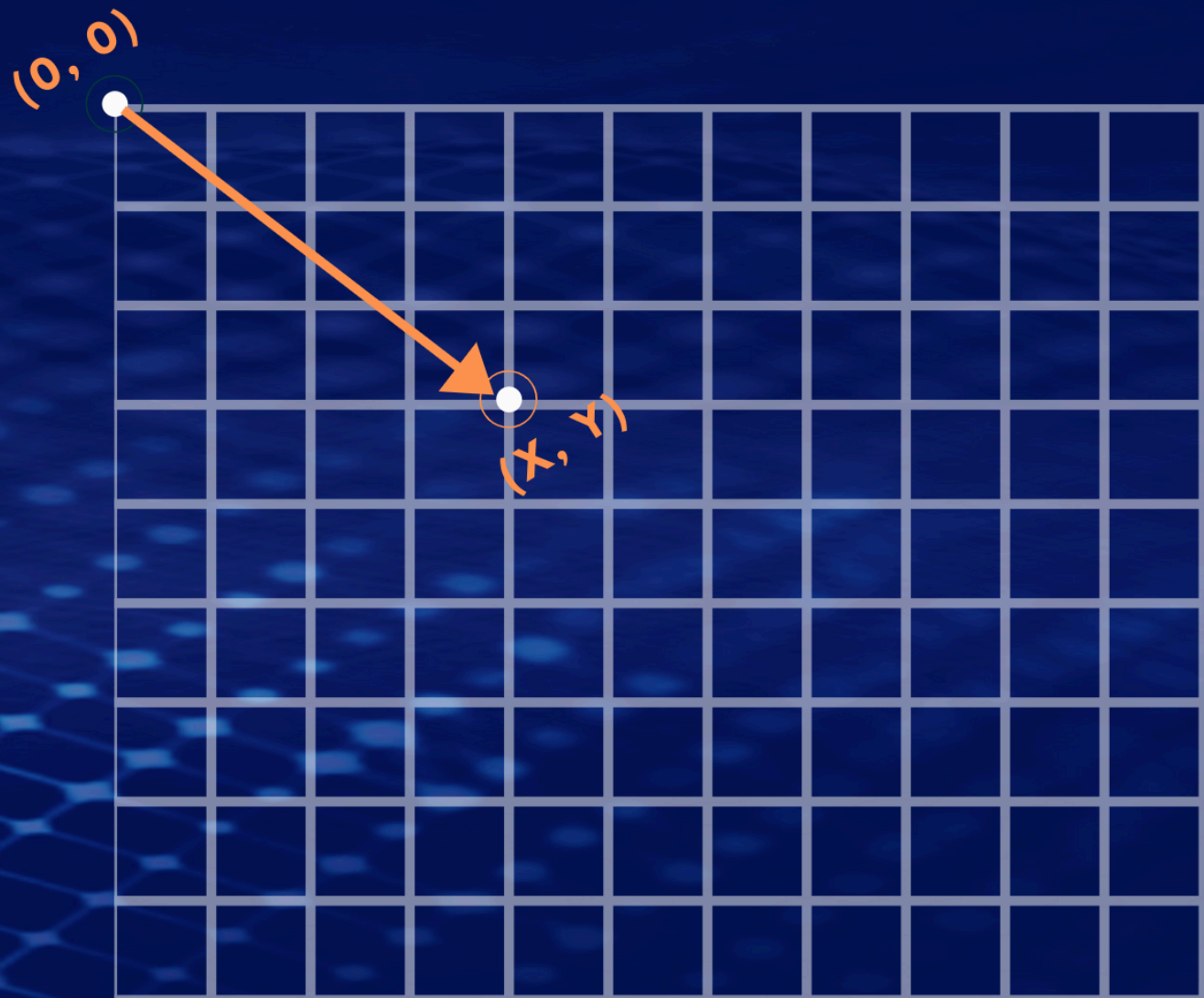
```
random() # floating values between 0 and 1

random([min], [max]) # random value between [min, max]

random(x) # random value between [0, x)
```

ARTIST'S CANVAS

DRAWING A POINT

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

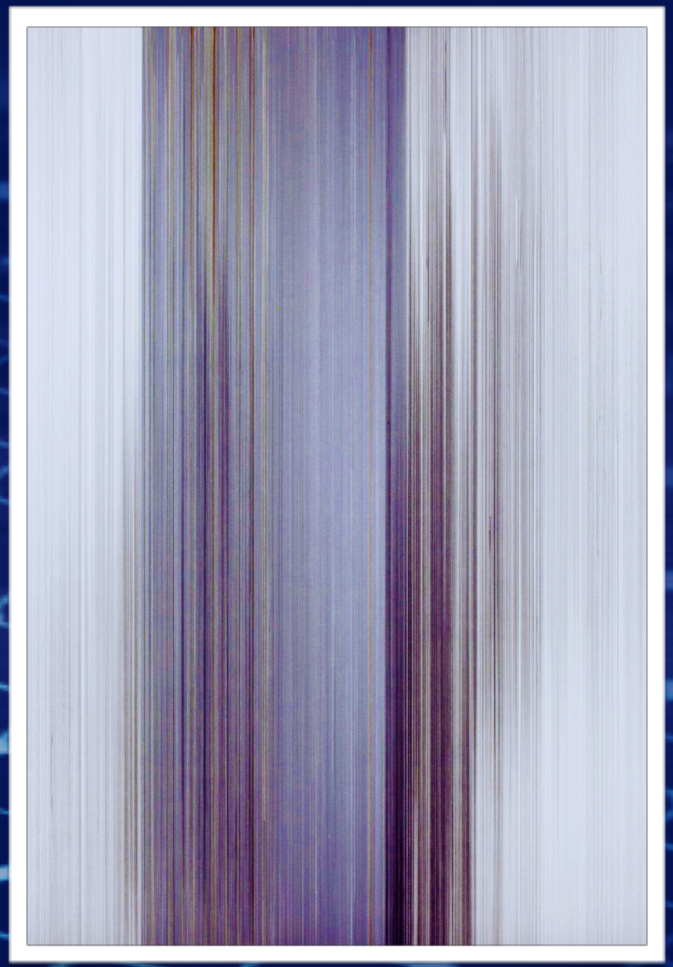


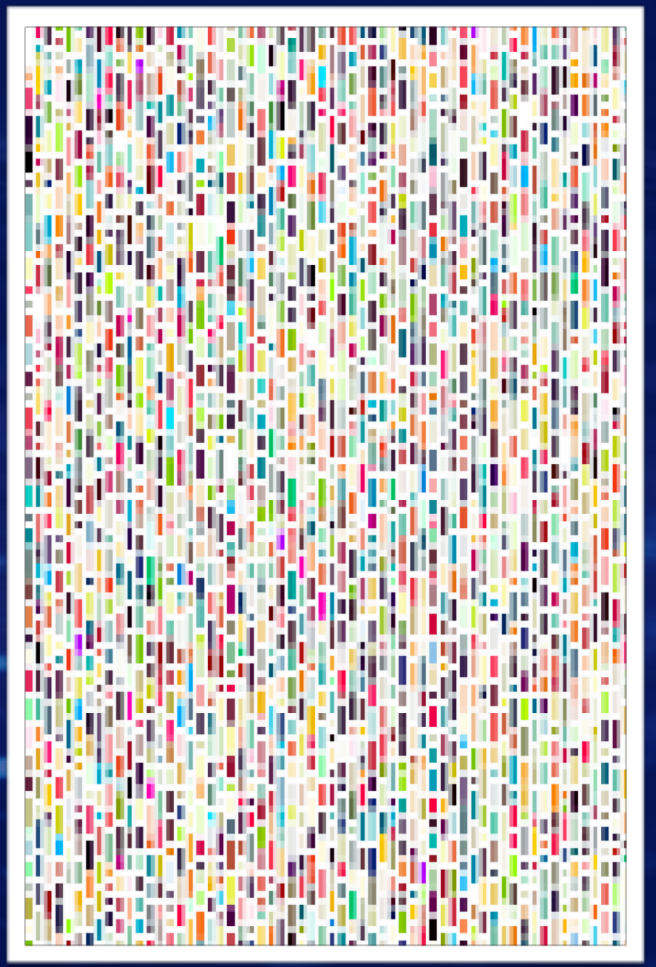


Basic operations in Processing.py

```
point(x, y) # Creates a point at (x, y) from the origin

point(x, y, z) # Creates a point at (x, y, z) from the origin

dist(x1, y1, x2, y2) # Calclulates the distance between two points

line(x1, y1, x2, y2) # Create a line by joining two points

stroke(color) # Sets tht color used to draw lines

strokeWeight(k) # Sets the thickness of the stroke with k
```

USING LINES

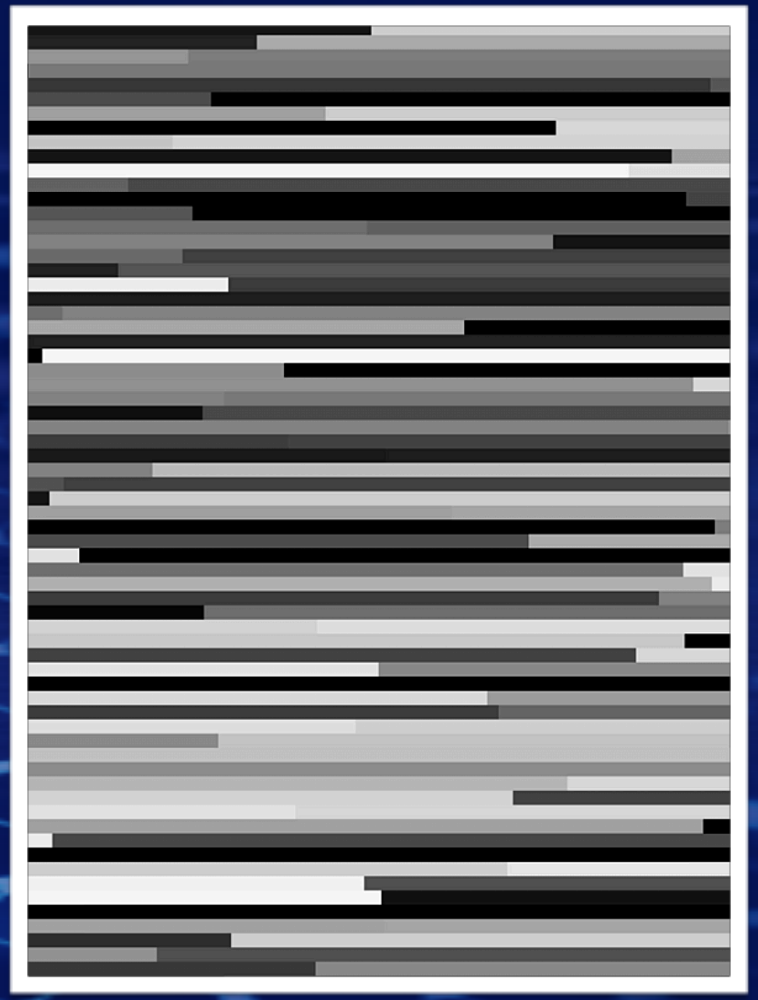




Basic Shapes in Processing

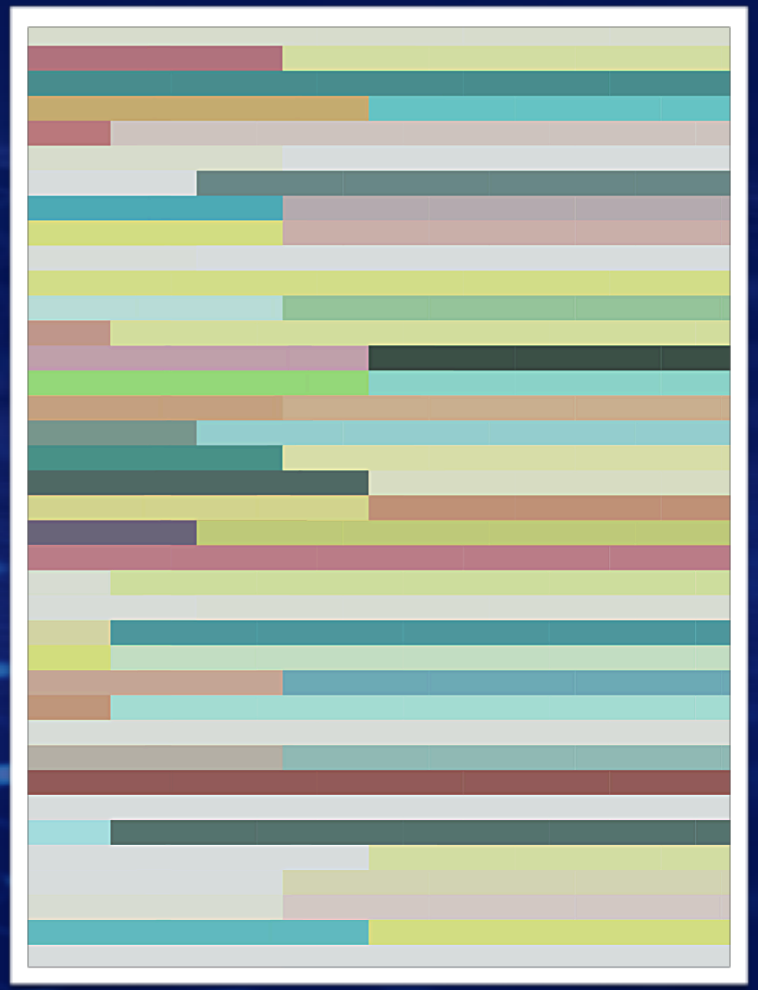
```
# Creates a rectangle at point a, b with "c" width and "d" height
rect(a, b, c, d, t1, t3, br, bl)

# Creates an ellipse at point a, b with "c" width and "d" height
ellipse(a, b, c, d)

# Creates a square
square(x, y, c)
```

USING RECTANGLES

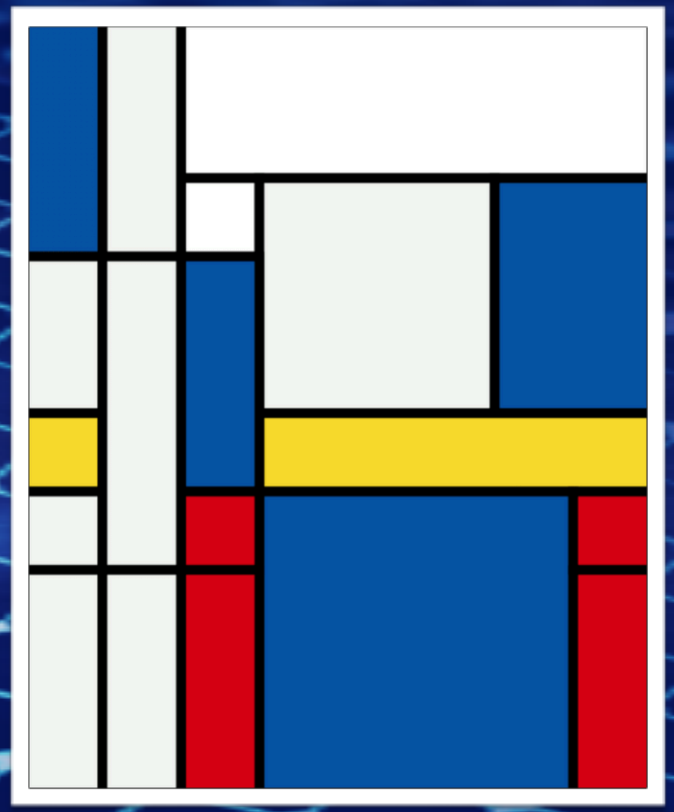


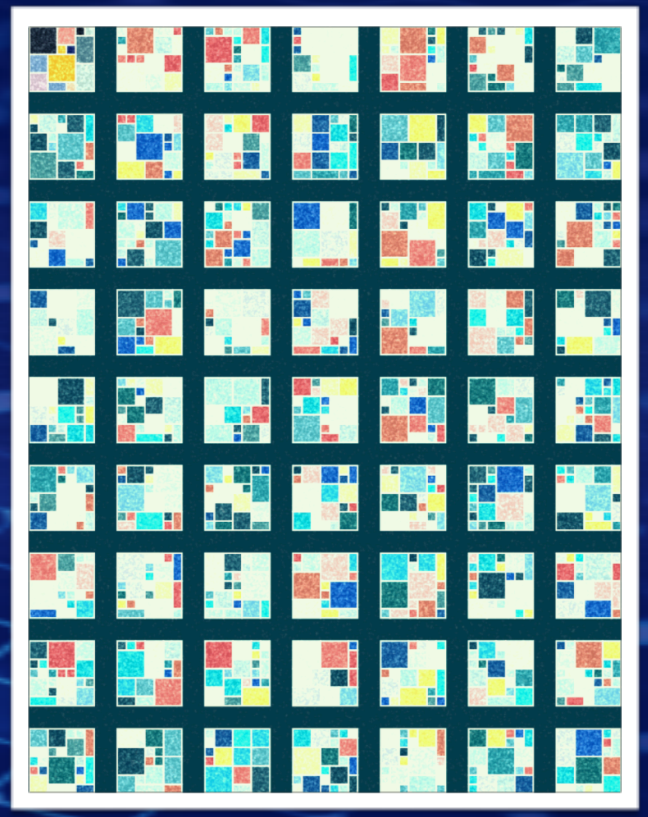
```
def setup():
  size(500, 700)
  background(0)

def draw():
  for y in range(0, 700, 10):
    x1 = random(0, 500)
    x2 = 500 - x1
    noStroke()
    fill(random(255))
    rect(0, y, int(x1), 15)
    fill(random(255))
    rect(int(x1), y, 500, 15)
  noLoop()
```

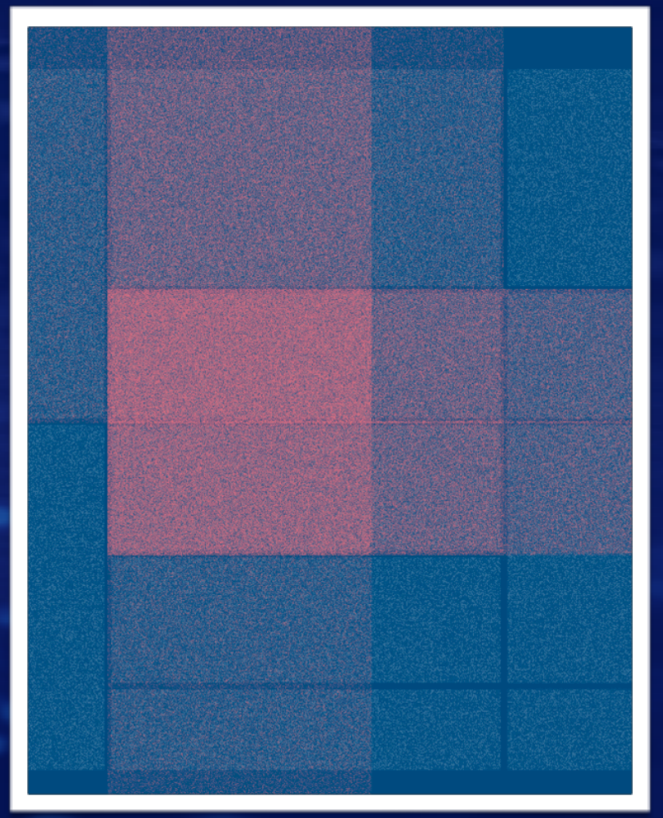
USING RECTANGLES



PIET MONDRIAN EXPERIMENTS

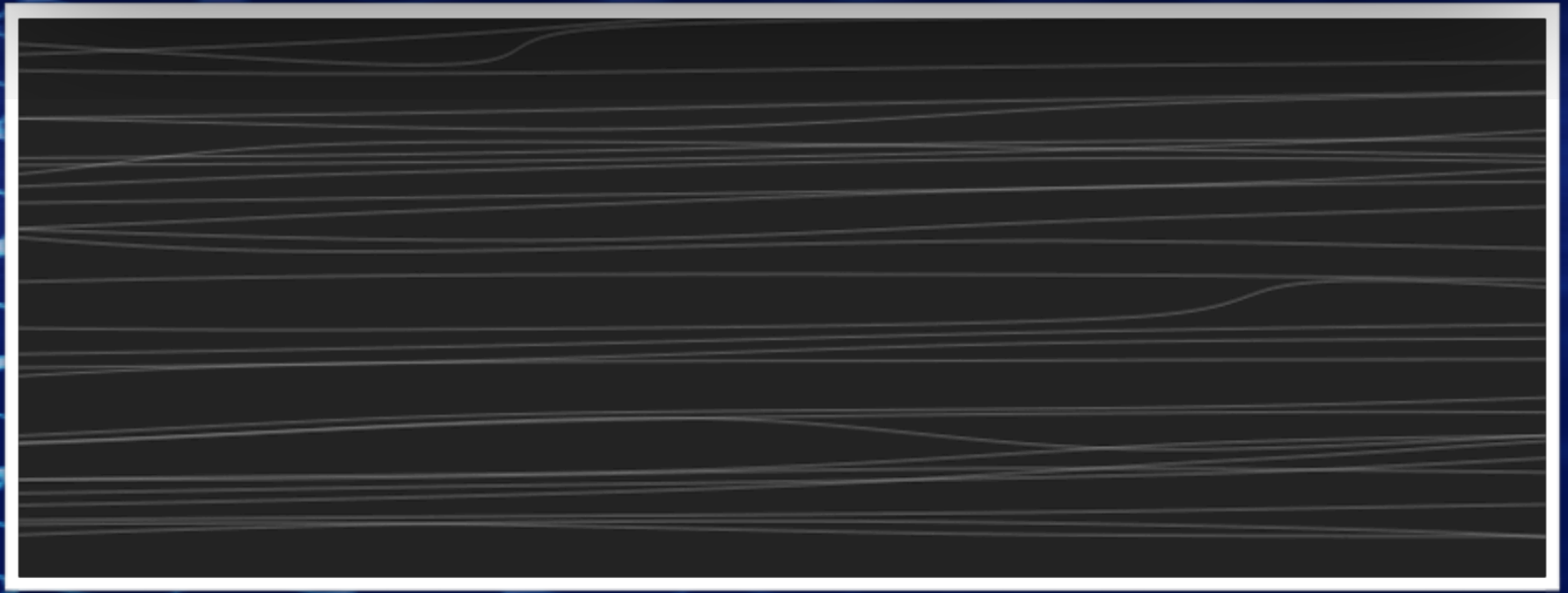




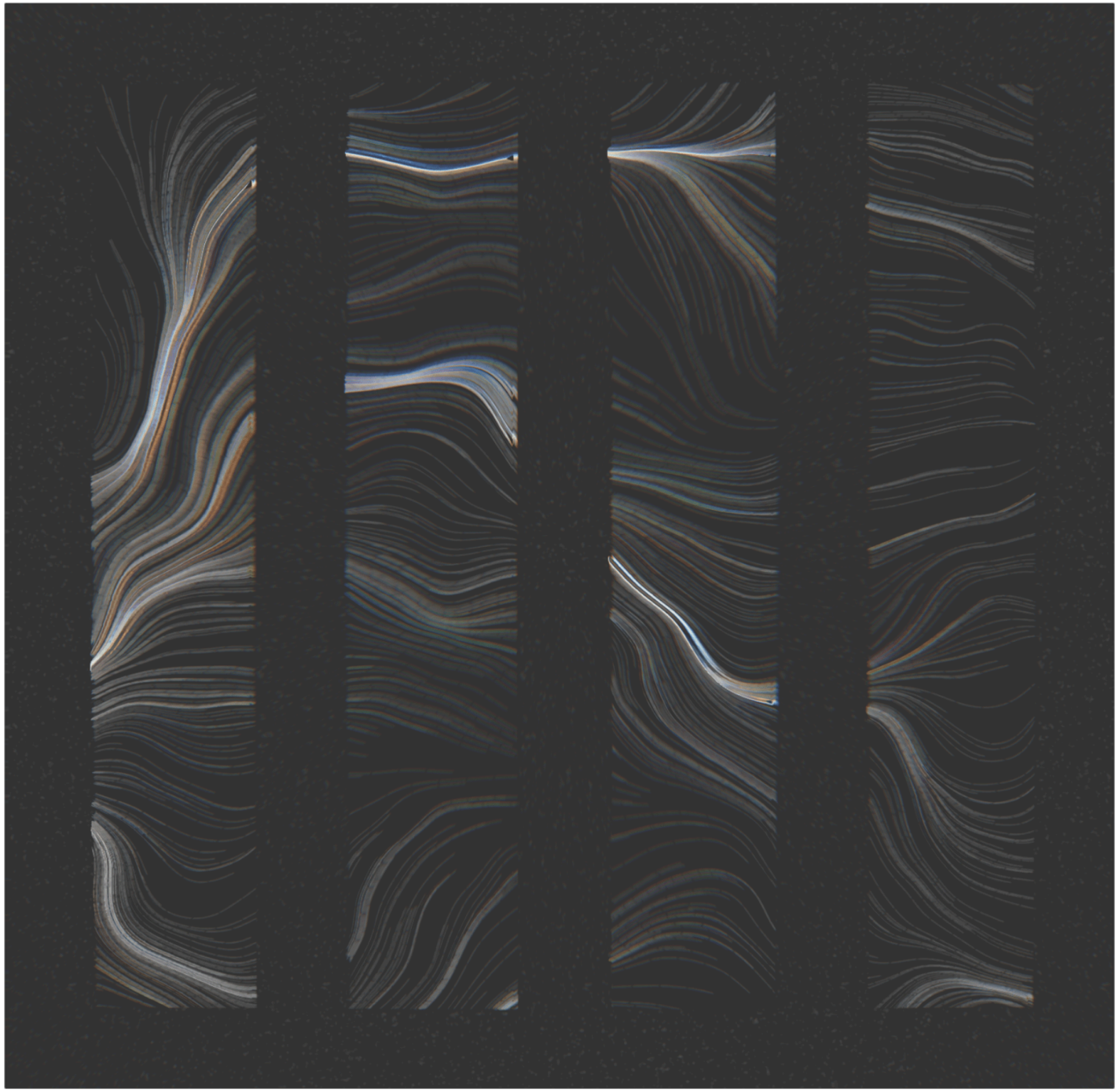


PERLIN NOISE / SIMPLEX NOISE

```
noise(x, [y], [z]) # Returns the perlin noise value

noiseSeed(seed) # Sets the seed valye for the noise()

noiseDetail(lod, falloff) # Adjusts the level of details producted by noise
```



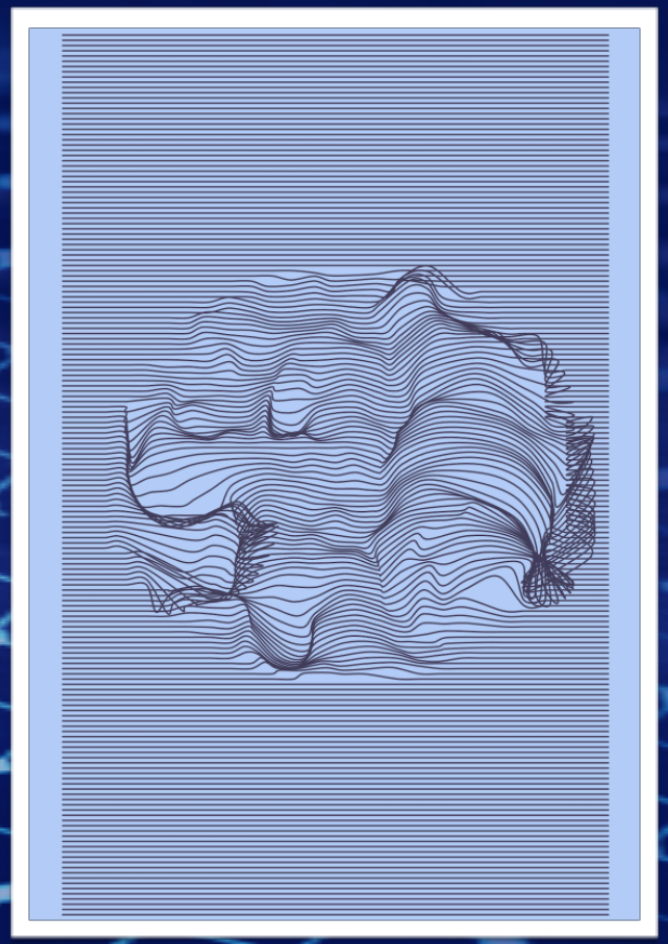
PERLIN NOISE / SIMPLEX NOISE

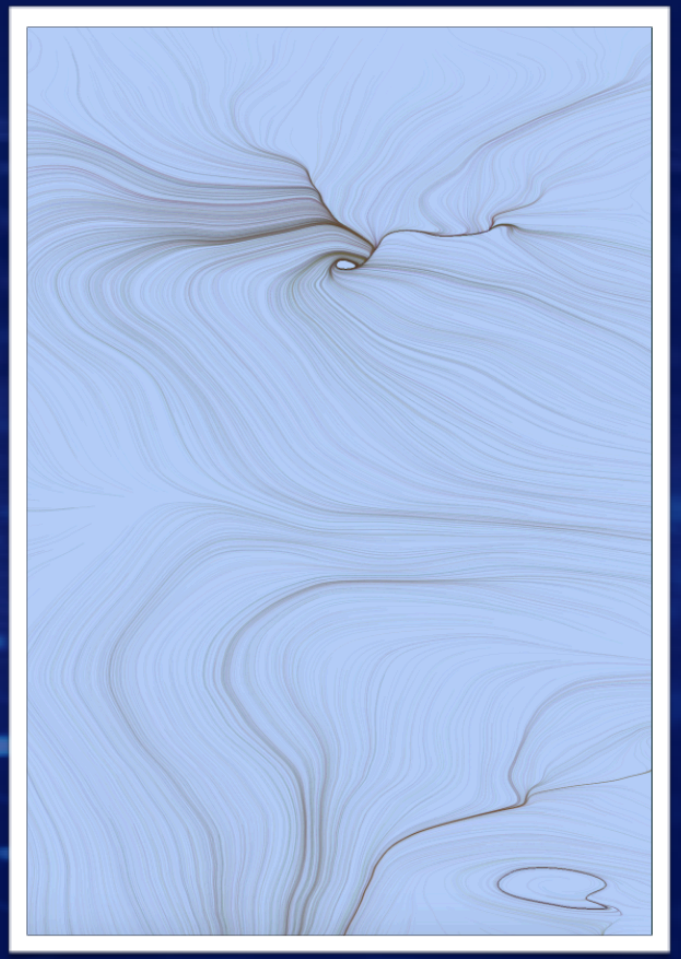


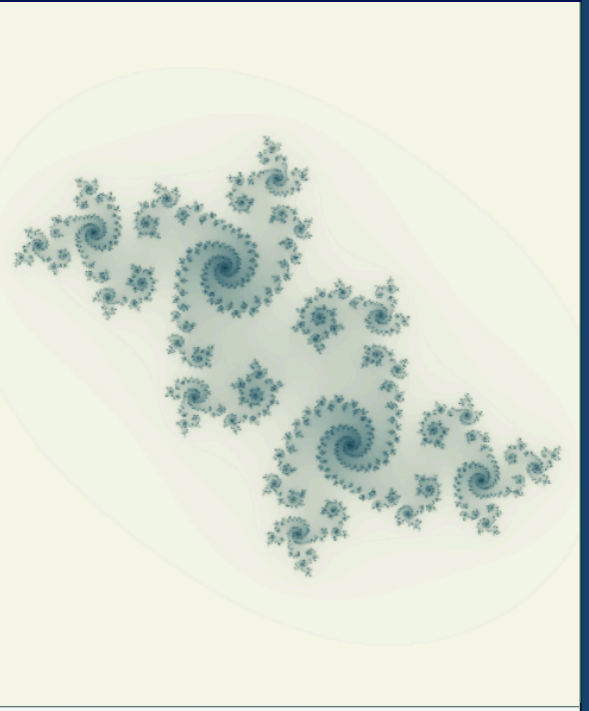
PERLIN NOISE

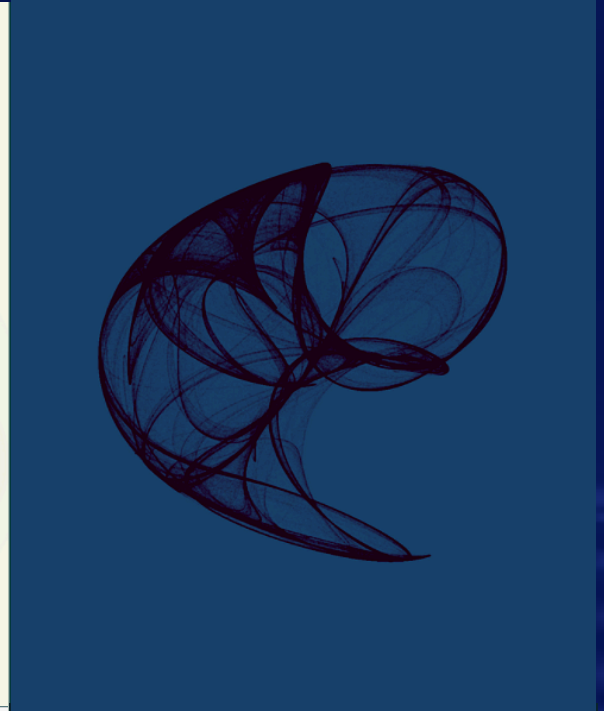
Using Perlin Noise Field and Perlin noise generated random noise points (grain like texture)

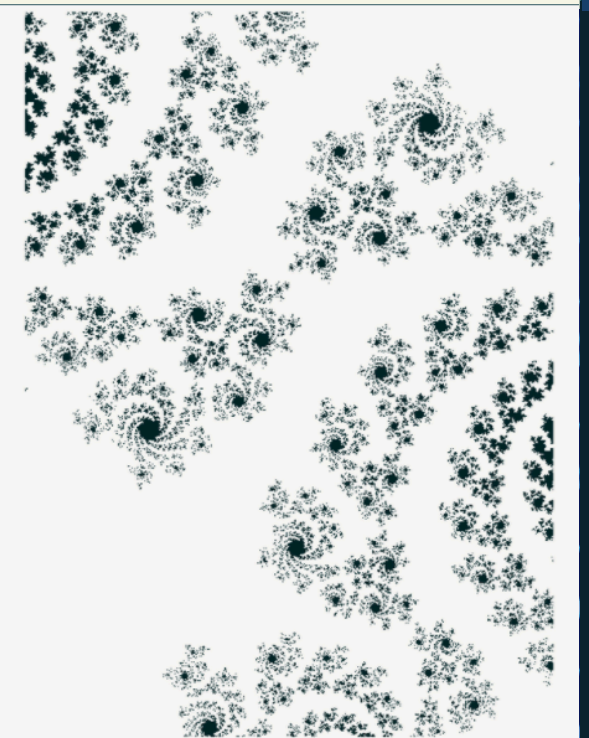
PERLIN NOISE FIELD

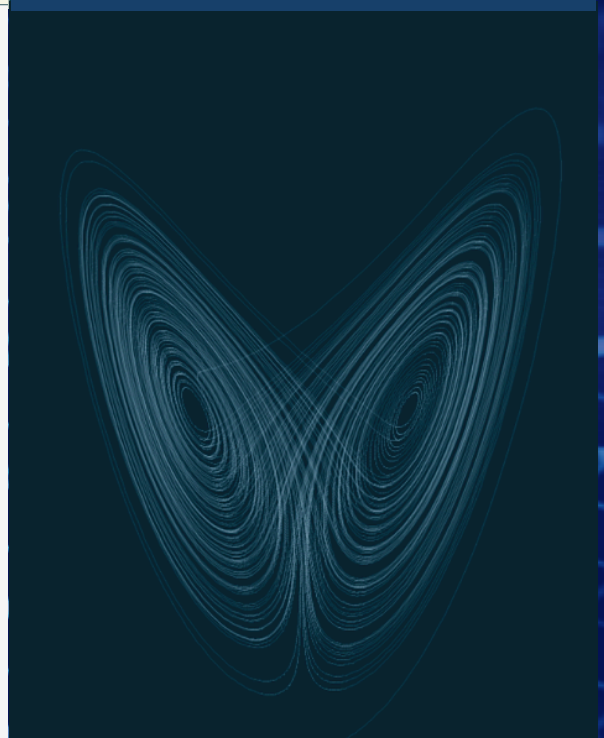








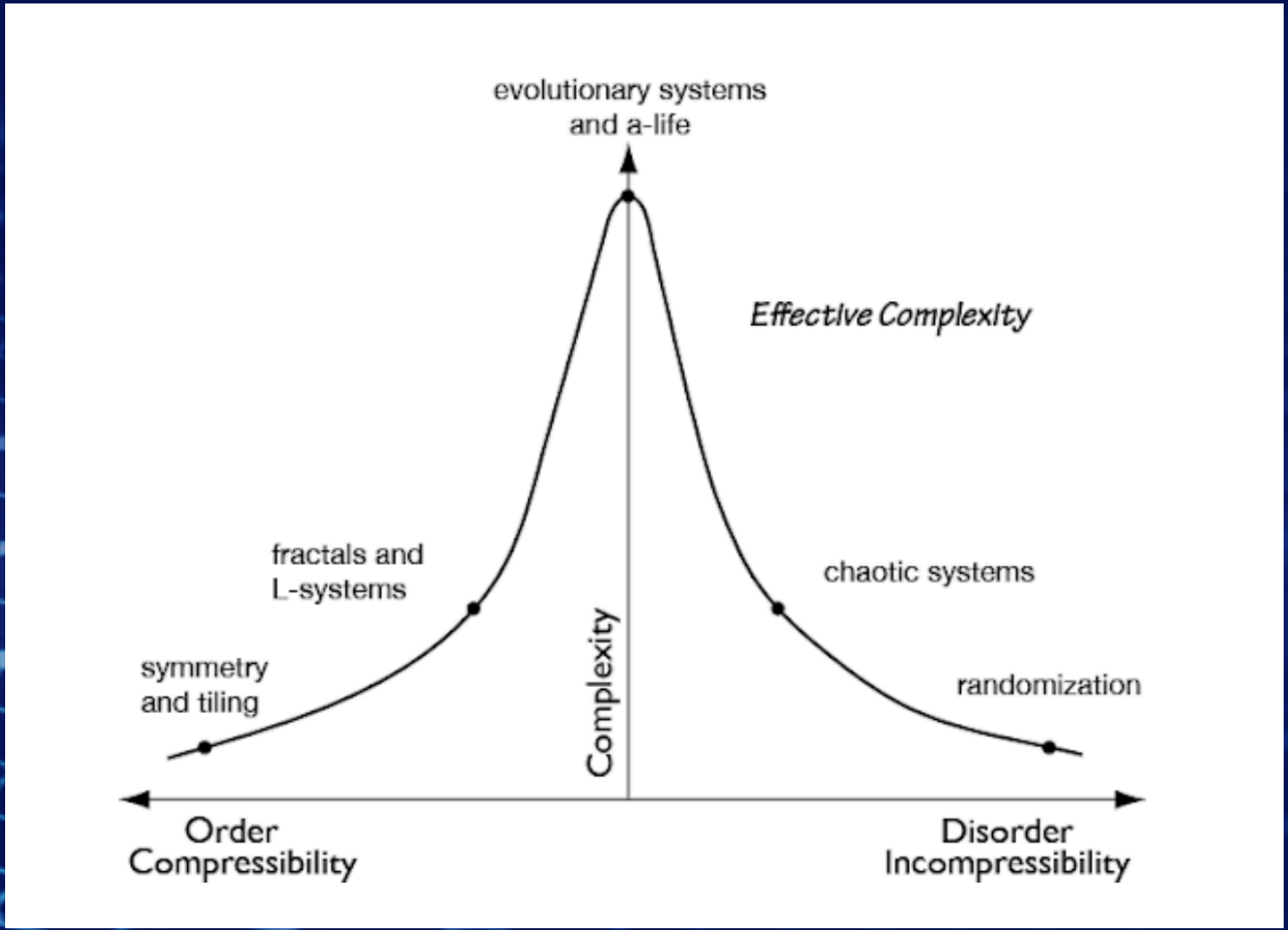




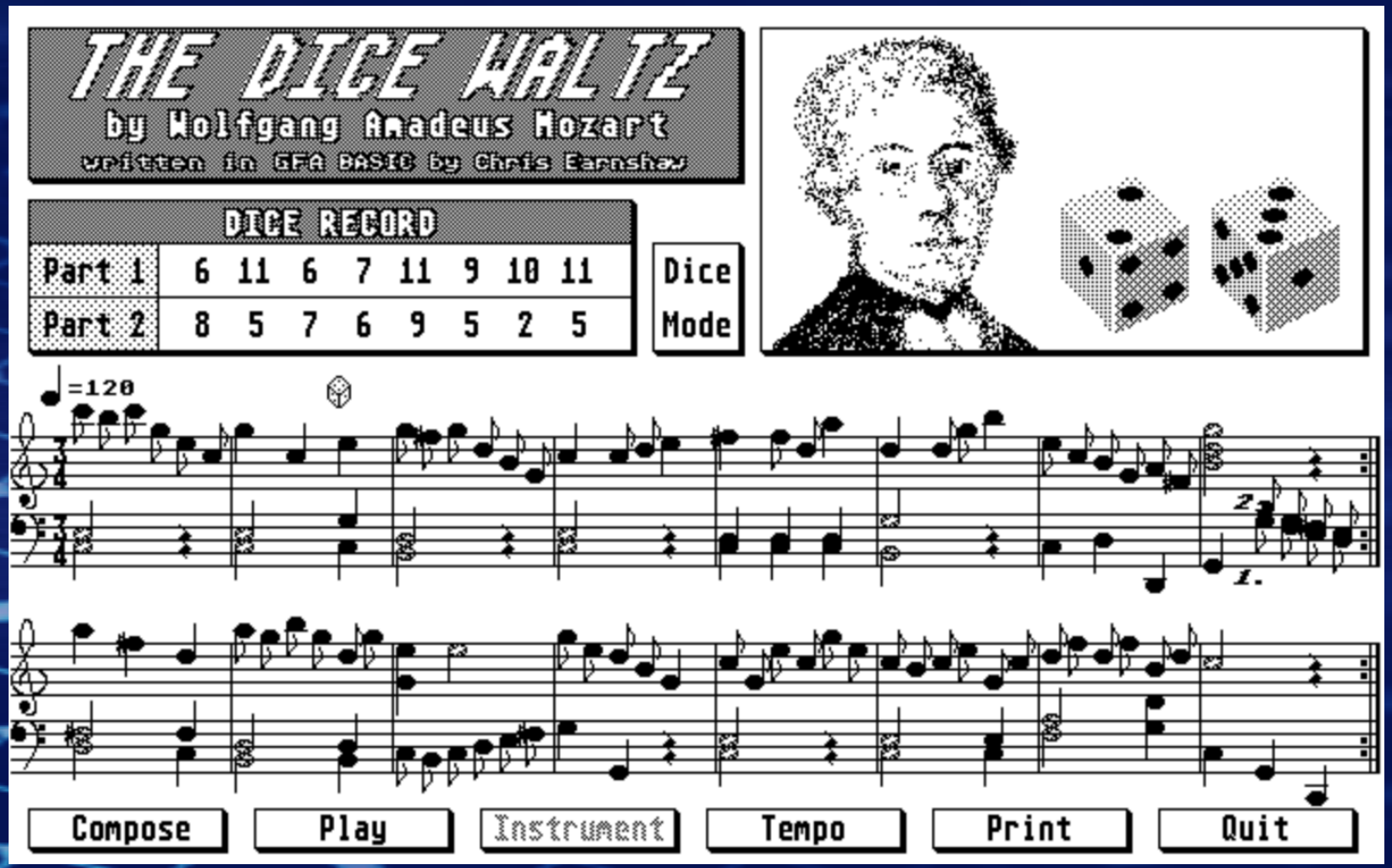
GEOMETRY
FRACTALS
CHAOS

Using geometrical patterns, fractals, and chaos theory to generate aesthetic art forms.

EFFECTIVE COMPLEXITY



GENERATIVE/MUSIC INTELLIGENCE



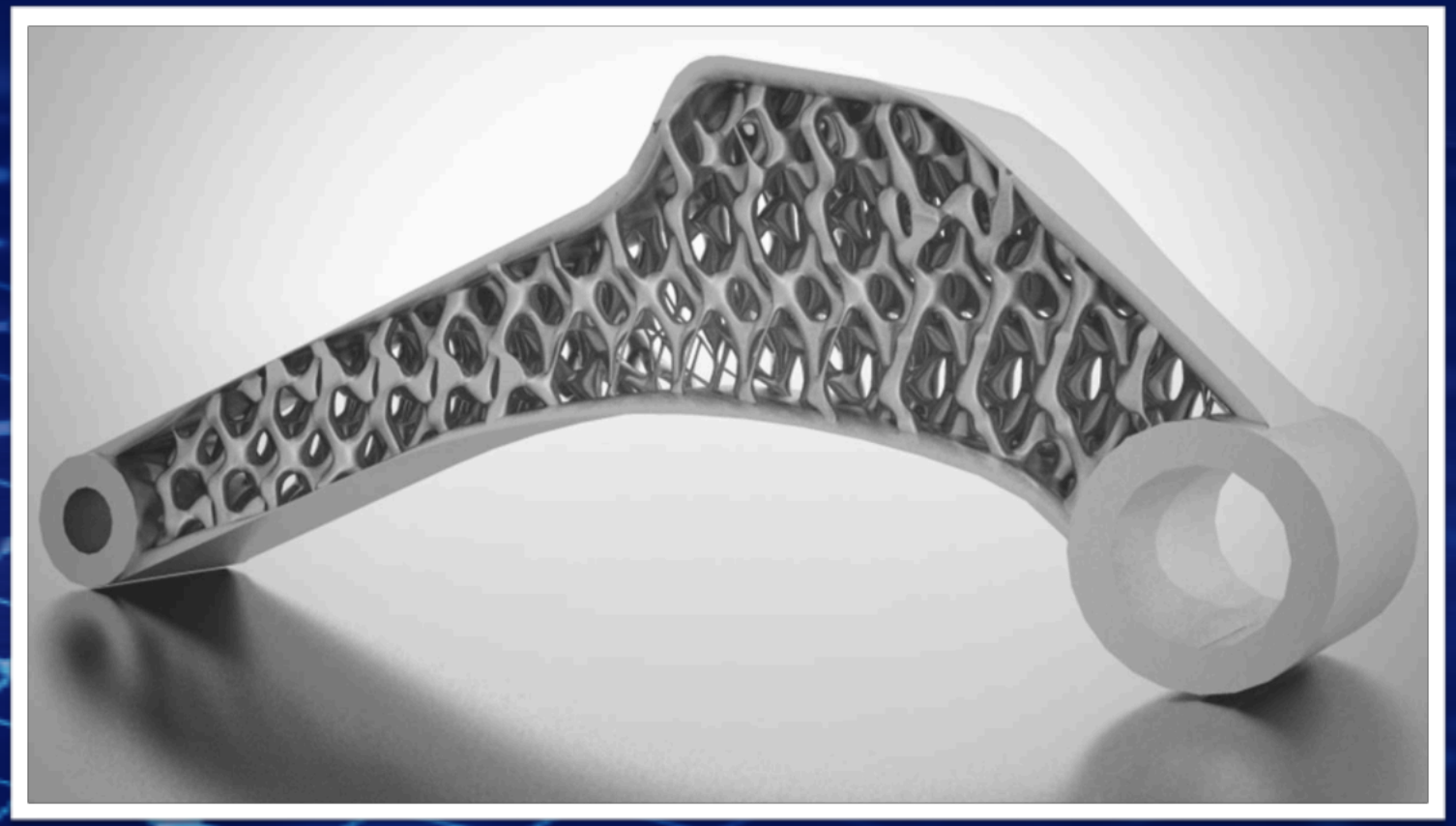
GENERATIVE/MUSIC INTELLIGENCE

Markov Chains

Genetic Algorithms

Deep Learning / Neural Networks

GENERATIVE DESIGN



The above picture represents a part with lattice structure

aidad.studio

One Column Format

Author, Company and/or Logo Information