Aalto University
School of Arts, Design
and Architecture

Creative Coding with Processing
Department of Art and Media
2025/2026

Programming for Visual Artists

Welcome

Algorithmic Patterns

Nested Loops

BREAK

HANDS-ON

Discussion & Q&A

For today...

Before We Start Coding

- Keep the Processing reference open
- If something doesn't work → check the reference
- Debugging is normal
- Errors are part of the process

- Processing reference: https://processing.org/reference

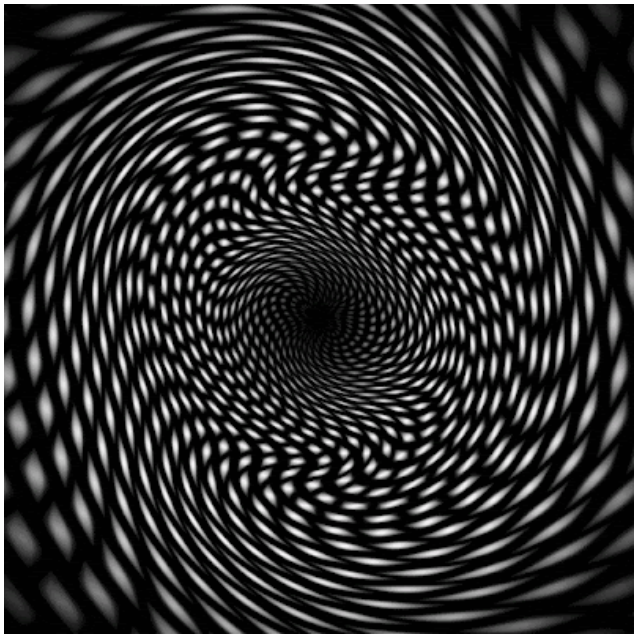
Recap

You now know:

- if → makes decisions
- for → repeats with structure
- draw() → creates motion

Today we combine them to generate systems.

Algorithmic Patterns



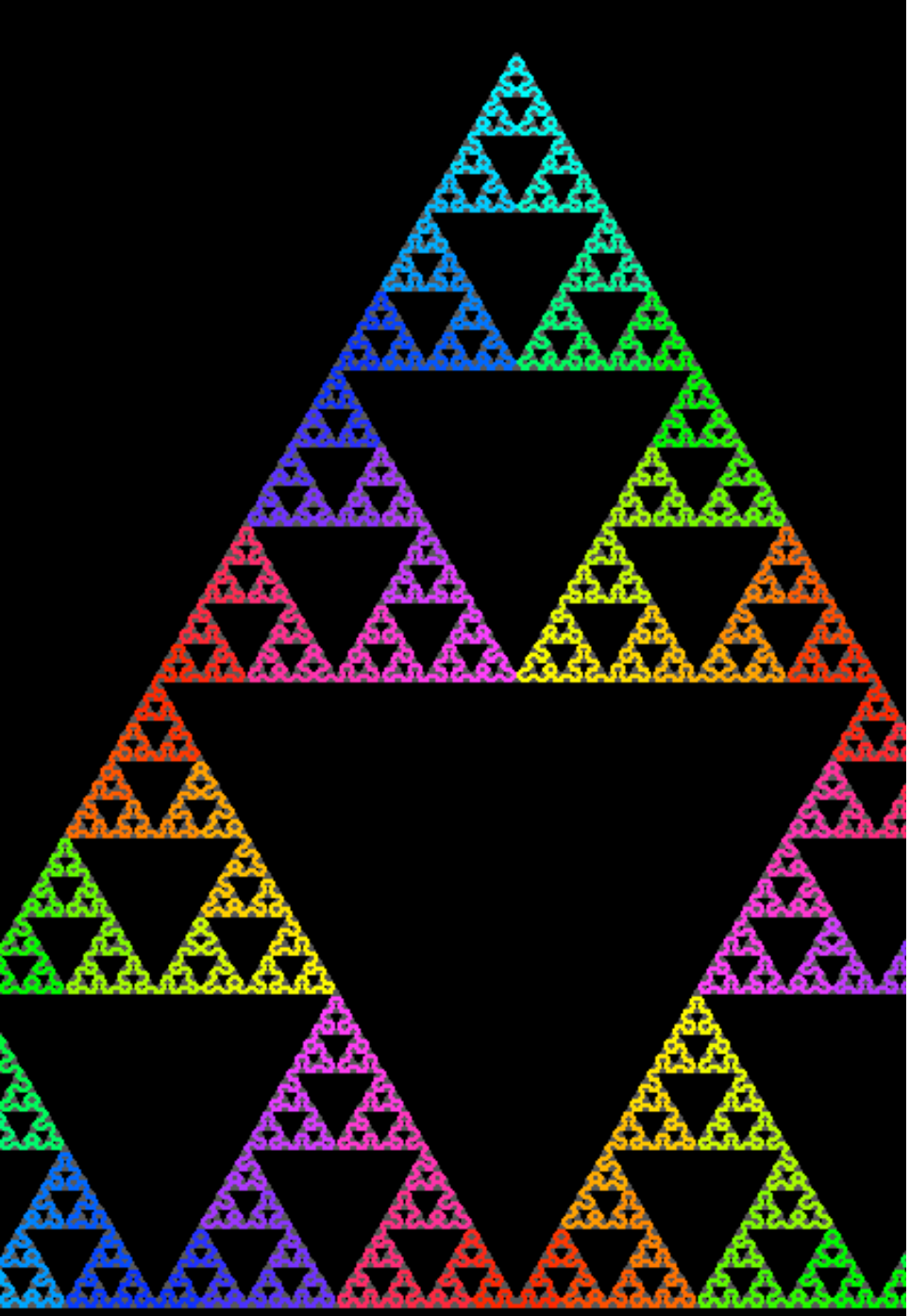
An algorithmic pattern is:

- A visual system generated from rules.
- Instead of placing shapes manually,
  we define rules and the system draws itself.

Algorithmic patterns often use:

- Loops
- Mathematical relationships
- Randomness
- Transformations
- Recursion

Small changes in rules create large changes in visuals.



Algorithmic Patterns - Key Ingredients

Algorithmic systems are built from combinations of:

Structure

Repetition (loops)

Variation

random()
noise()

Mathematics

sin()
cos()
linear relationships

Transformation

translate()
rotate()
scale()

Advanced (optional)

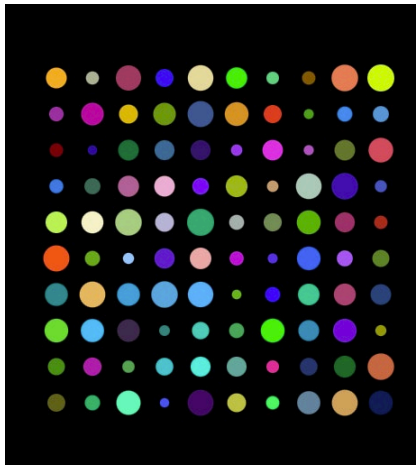
Recursion
Fractals

Aesthetic Control

Color
Transparency

Algorithmic Patterns – examples

Grid-Based Algorithmic Pattern



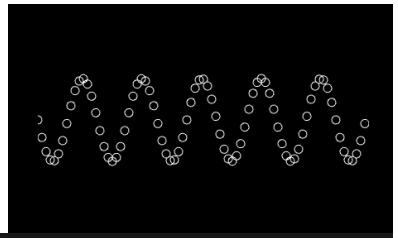
- Nested loops create structure
- random() creates variation
- Small rule → large visual result

The grid is the skeleton.
Randomness is the personality.

```
void setup() {
  size(400, 400);
  background(0);
  noStroke();

  // Outer loop → columns
  for (int x = 0; x < width; x += 40) {

    // Inner loop → rows
    for (int y = 0; y < height; y += 40) {

      // Random size for each circle
      float circleSize = random(10, 30);

      // Random color for each circle
      fill(random(255), random(255), random(255));

      // Draw circle centered in each grid cell
      ellipse(x + 20, y + 20, circleSize, circleSize);
    }
  }
}
```

Algorithmic Patterns – examples

Sinusoidal Wave



```
void setup() {
  size(400, 400);
  background(0);
  stroke(255);
  noFill();

  for (int x = 0; x < width; x += 5) {
    float angle = x * 0.05;
    float y = height/2 + sin(angle) * 50;
    ellipse(x, y, 10, 10);
  }
}
```

Static

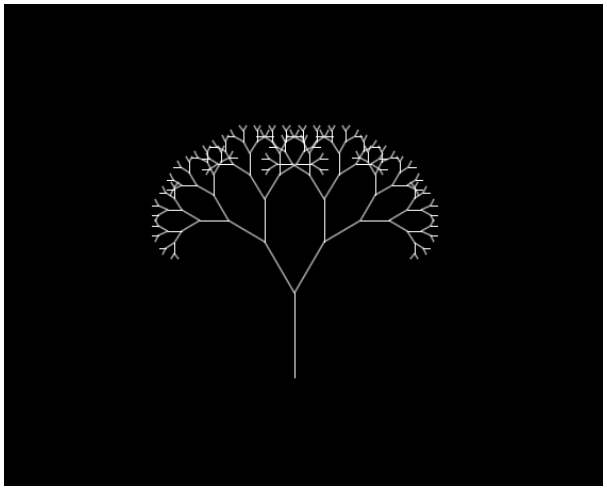
```
float offset = 0;

void draw() {
  background(0);
  stroke(255);
  noFill();

  for (int x = 0; x < width; x += 5) {
    float angle = x * 0.05 + offset;
    float y = height/2 + sin(angle) * 50;
    ellipse(x, y, 10, 10);
  }

  offset += 0.05;
}
```

- Loop controls horizontal movement
- sin() controls vertical variation
- Math can generate form

Animated

Algorithmic Patterns – examples

Recursive Tree (Fractal)



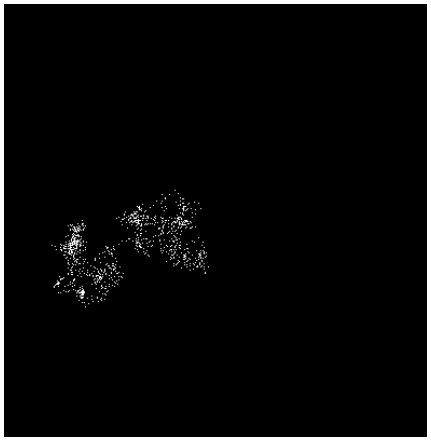
Step 1:
Draw a line.
Step 2:
From its end, draw two smaller lines.
Step 3:
Repeat the rule until the branch length is smaller than 5.

```
void setup() {
  size(400, 400);
  background(0);
  stroke(255);
  drawBranch(width / 2, height, -PI / 2, 80);
}

void drawBranch(float x, float y, float angle, float length) {
  if (length < 5) return;

  float x2 = x + cos(angle) * length;
  float y2 = y + sin(angle) * length;

  line(x, y, x2, y2);

  drawBranch(x2, y2, angle - PI / 6, length * 0.7);
  drawBranch(x2, y2, angle + PI / 6, length * 0.7);
}
```

- A function can call itself
- Each call uses smaller values
- A simple rule creates complex structure

Algorithms can generate organic forms. Not just grids.

Algorithmic Patterns – examples



Random Walk

1.Start in the center.
2.Take a small random step.
3.Repeat many times.

A random walk is controlled randomness.

- Starts at one point
- Moves randomly step by step
- Repetition creates organic form

```
void setup() {
  size(400, 400);
  background(0);
  stroke(255);

  float x = width/2;
  float y = height/2;

  for (int i = 0; i < 2000; i++) {
    point(x, y);
    x += random(-5, 5);
    y += random(-5, 5);
  }
}
```

static

```
float x, y;

void setup() {
  size(400, 400);
  background(0);
  stroke(255);
  x = width/2;
  y = height/2;
}

void draw() {
  point(x, y);
  x += random(-3, 3);
  y += random(-3, 3);
}
```

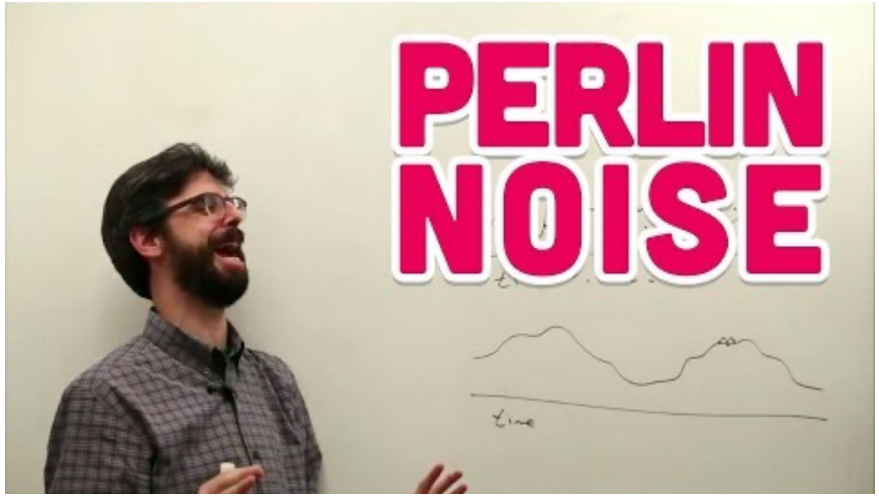
animated

Perlin Noise

The movement was random.
What if randomness could be smooth?

- x * 0.01 → controls smoothness
- noise() returns values between 0 and 1
- Multiply by height to scale to the canvas

```
void setup() {
  size(400, 400);
  background(0);
  stroke(255);

  for (int x = 0; x < width; x++) {
    float y = noise(x * 0.01) * height;
    point(x, y);
  }
}
```



noise() creates continuity between values.

Nested loops

A nested loop is a loop inside another loop.

Outer loop → moves down (rows)
Inner loop → moves across (columns)

```
i = 0, j = 0
i = 0, j = 1
i = 0, j = 2
i = 1, j = 0
i = 1, j = 1
i = 1, j = 2
i = 2, j = 0
i = 2, j = 1
i = 2, j = 2
```

```
for (int i = 0; i < 3; i++) {  // Outer loop
  for (int j = 0; j < 3; j++) {  // Inner loop
    println("i = " + i + ", j = " + j);
  }
}
```

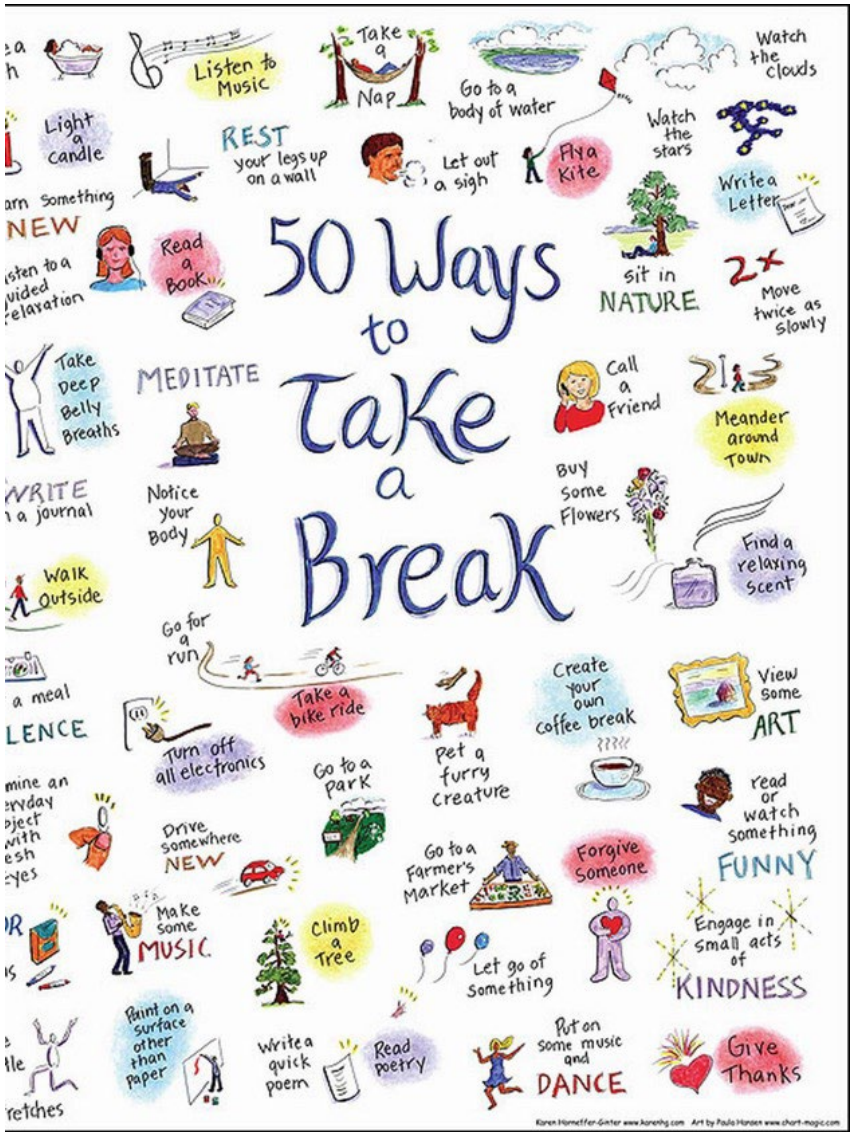
Step 1:
Choose a row.
Step 2:
Fill the row.
Step 3:
Move to the next row.

Break

15 minutes

Stretch. Coffee. Reset.

Back at 10:45



Hands-On Exercise!

Your task

- Create a grid-based visual system.
- Use nested loops to generate structure.
- Use noise() or sin() to generate variation.
- Animate your system over time.

Your goal is not to draw shapes.
Your goal is to design behavior.

Think about:

- What is the rule of your system?
- What variable controls change?
- What changes when you adjust one number?
- How does repetition become pattern?

Break things. Change numbers. Explore.

Template

- Download from: https://codeberg.org/ptiagomp/aalto-programming-visual-artists-25-26/src/branch/main/Session-04_03032026

Optional Challenge

If you finish early:

- Make it react to the mouse
- Combine noise() and distance
- Add transparency
- Use color variation
- Break the grid structure
- Replace noise() with sin()
- Add nested shapes

Discussion and Questions

- What changed your perception of code today?
- When did repetition become pattern?
- When did randomness become structure?
- When did logic become visual?

Today you wrote rules.
The rules wrote the visuals.

Next Week
You built systems.
Now we structure them.
We move from patterns to reusable components.
We introduce functions to organize and scale your generative systems.

Don’t forget the assignments!