Aalto University
School of Arts, Design
and Architecture

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

Programming for Visual Artists

Welcome

What is Creative Computation?

Thinking Algorithmically

Your First Sketch

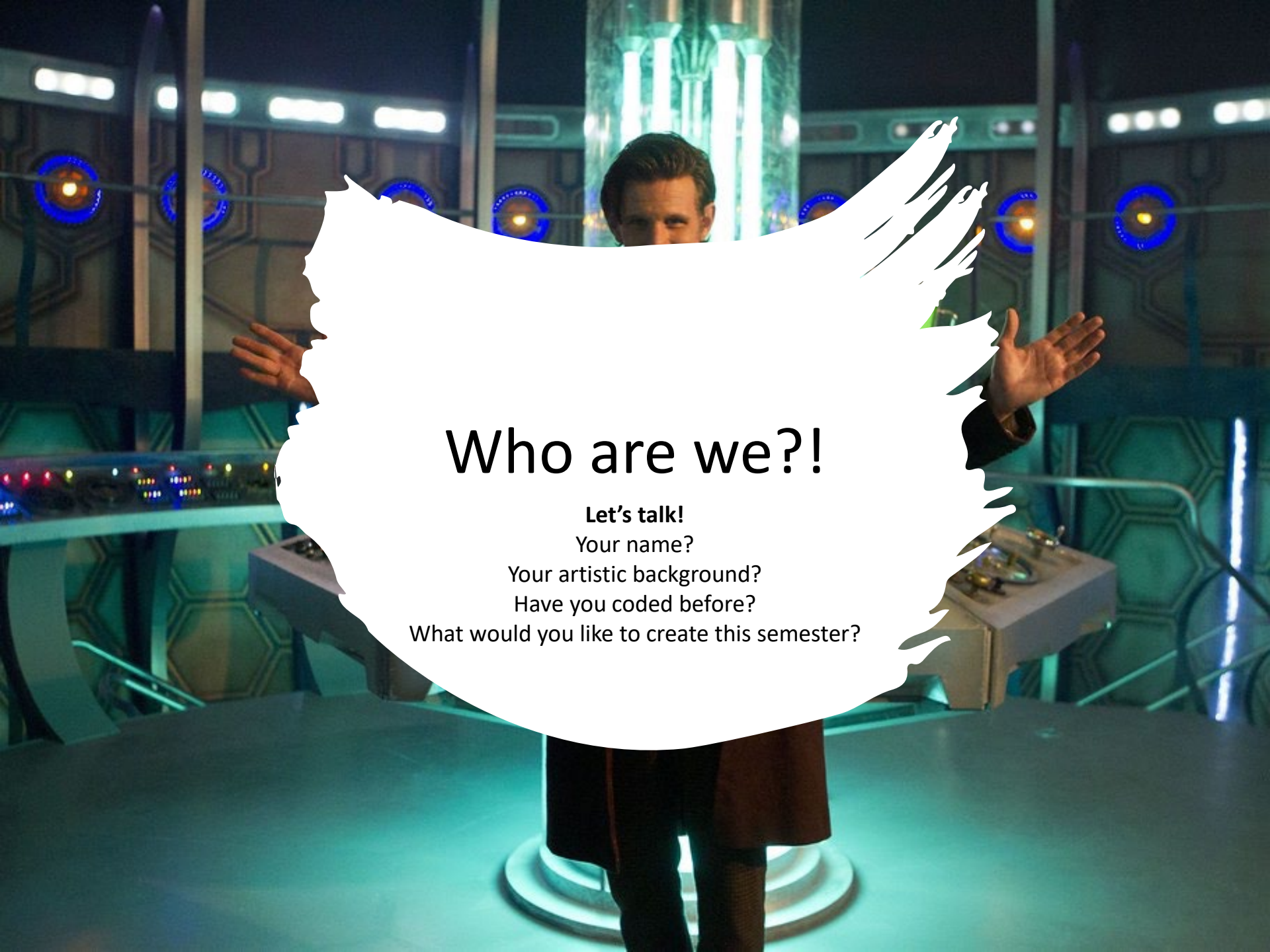
BREAK (10:30–10:45)

Processing Basics

Hands-on Exploration

Wrap-Up Tomorrow

Today...



Who are we?!

Let's talk!

Your name?

Your artistic background?

Have you coded before?

What would you like to create this semester?

```
void setup() {
    size(600, 400);
    background(255);
}

void draw() {
    // Add your creative code here!
}
```

Boilerplate

A boilerplate is a prepared starting structure.

It gives us:

- A ready-made foundation
- A shared structure
- A starting point for experimentation

In this course, the boilerplate sets the "**rules of the game.**"

Syllabus

Schedule

36 hours
Mondays & Tuesdays
9:15 - 12:00
23.02.2026 - 31.03.2026

Learning Goals:

By the end of this course, you will:

- Understand the basics of programming
- Use **Processing** to create visuals
- Develop and present a creative coding project

Week 1
Foundations
Introduction, setup, first sketch
Drawing, color, basic interaction
Week 2
Motion and Generative Systems
Animation and movement
Loops and rule-based patterns

Week 3
Structuring Complexity
Functions and modular code
Arrays and multi element systems
Final project introduction
Week 4
Project Development
Pitch conversations
Prototyping and feedback
Final presentations and critique

Grading & Attendance

Attendance

Minimum 80 percent attendance required.

The first session is mandatory.

If you miss the first session, your place may be given to a student on the waiting list.

Grading Focus

Effort and engagement

Willingness to experiment

Development of ideas

Basic understanding of programming concepts

Technical perfection is not required.

Curiosity and creative exploration are valued.

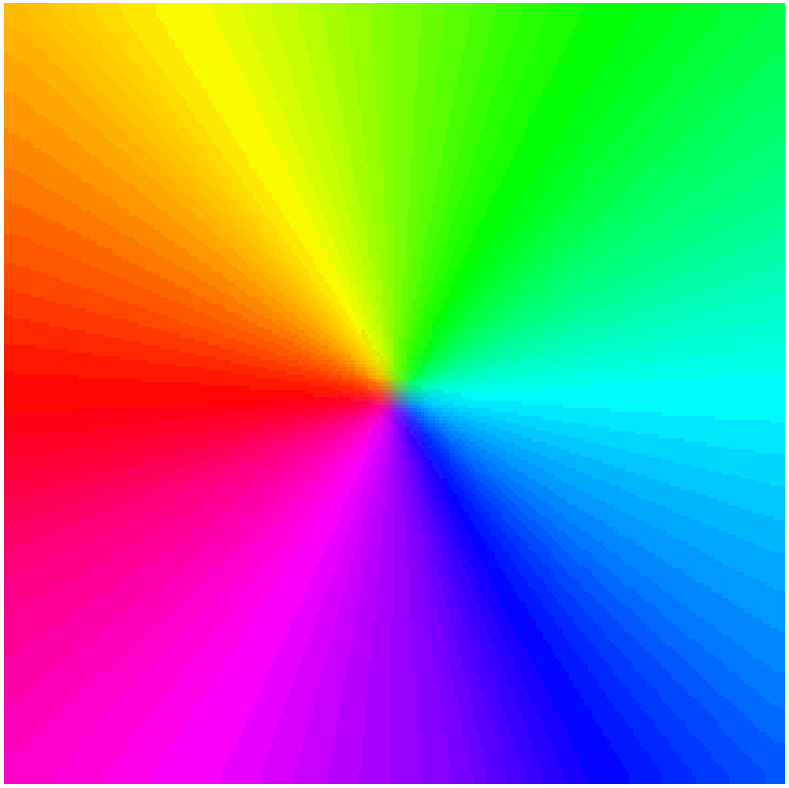
Study 1: Interactive Sketch - Mouse, Color, and Shape

Make a submission | Receive a grade

Opens: Tuesday, 24 February 2026, 12:00 PM
Due: Sunday, 8 March 2026, 11:59 PM

- Cartesian coordinates, color theory, interactivity, mouse input handling.
- Create an interactive visual sketch in Processing that reacts to mouse movement.
- The artwork should change visually as the mouse moves across the screen. Use color, shape, and motion to explore interaction and composition.
- This is an exploratory study. Play, experiment, and see what happens.

Projects / Assigments



Study 2: Dynamic Circle Grid - Loops and Interaction

Make a submission Receive a grade

Opens: Tuesday, 3 March 2026, 10:00 AM
Due: Sunday, 15 March 2026, 11:59 PM

- Create a generative grid of circles that reacts to mouse movement.
- Use repetition and variation to explore pattern, rhythm, and composition.
- The grid should change visually as the mouse moves.
- This is an exploratory study. Try different visual ideas and see what emerges.

Projects / Assigments

Final Project: Interactive Generative Artwork (Processing or p5.js)

Make a submission | Receive a grade | Receive a passing grade

Opens: Tuesday, 10 March 2026, 11:00 AM
Due: Sunday, 29 March 2026, 11:59 PM

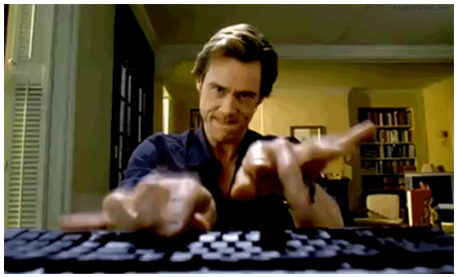
Create an original interactive or generative artwork using Processing or p5.js.

Your project should explore ideas, aesthetics, or experiences through code. It can be playful, poetic, abstract, or conceptual. The goal is to develop a personal direction and apply what you learned in the course.

Your work should include some form of:

- interaction
- generative behavior
- dynamic change over time

This is an open-ended creative project. Think of it as a small digital artwork or prototype rather than a technical demo.



Projects / Assigments

Working Environment

Some background on Creative Computation

From Cybernetics to Systems Thinking

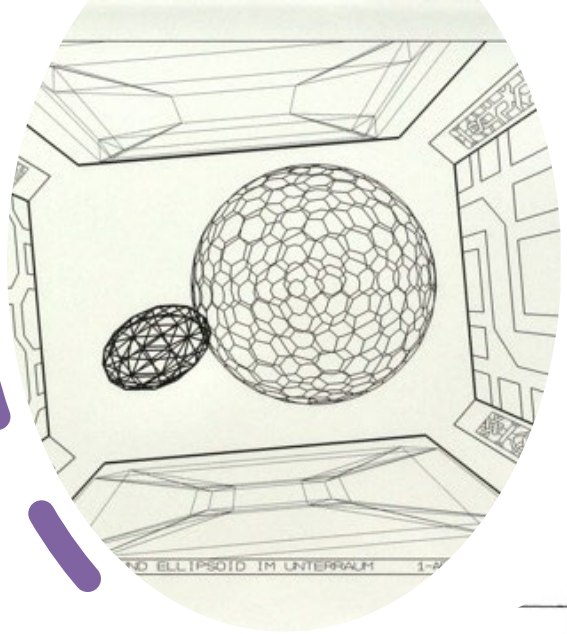
- In 1834, André-Marie Ampère introduced the term ***cybernetics***, referring to it as "**the future science of government**." Derived from the French ***cybernetique*** ("**art of governing**") and the Greek ***kybernetes*** ("**navigator**" or "**steersman**").[1]
- Norbert Wiener popularized ***cybernetics*** in 1949.[2]

Core idea

- Complex behavior can emerge from rule-based systems.
- This systems perspective later influenced artistic practice.

[1] Tsien, Hsue S. (1954). Engineering Cybernetics (pg. vii). McGraw-Hill.
[2] Wiener, Norber. (1948). Cybernetics: or Control and Communication in the Animal and the Machine (pgs. 11-12). Cambridge, Mass.: The MIT Press.

Some background on Creative Computation

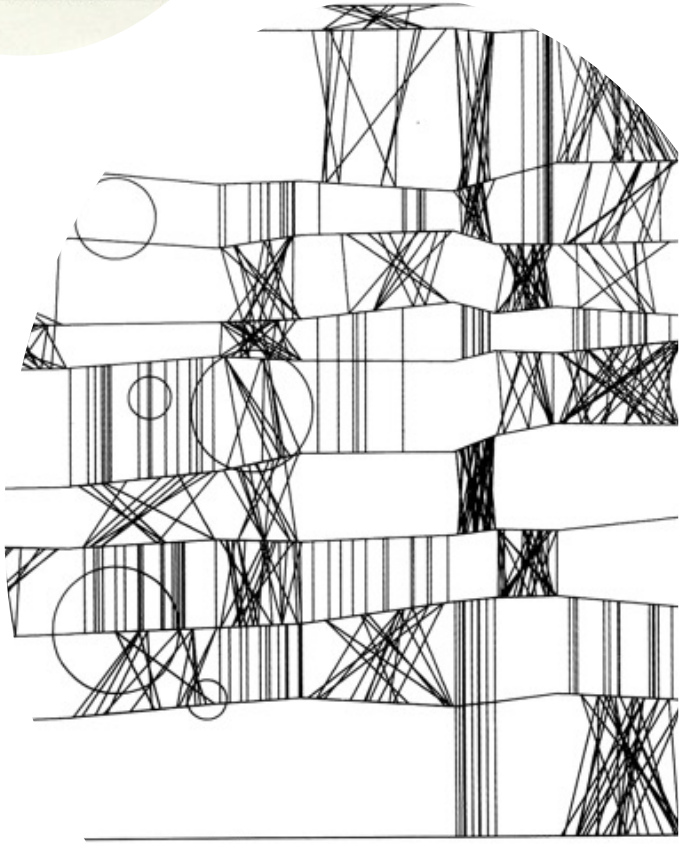


Early Computer Art 1950s and 1960s
Artists began using computers not as tools for drawing, but as machines for executing instructions.
Important shift:
The artist writes rules.
The machine executes them.

Frieder Nake (b. 1938): A mathematician and artist, Nake created some of the earliest algorithmic drawings using a plotter printer and mathematical rules.

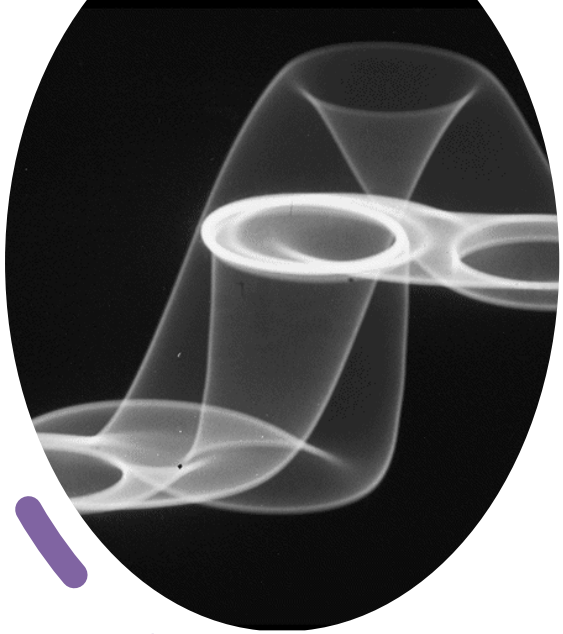
Georg Nees (1926–2016): One of the first to publicly exhibit computer-generated images.[3]

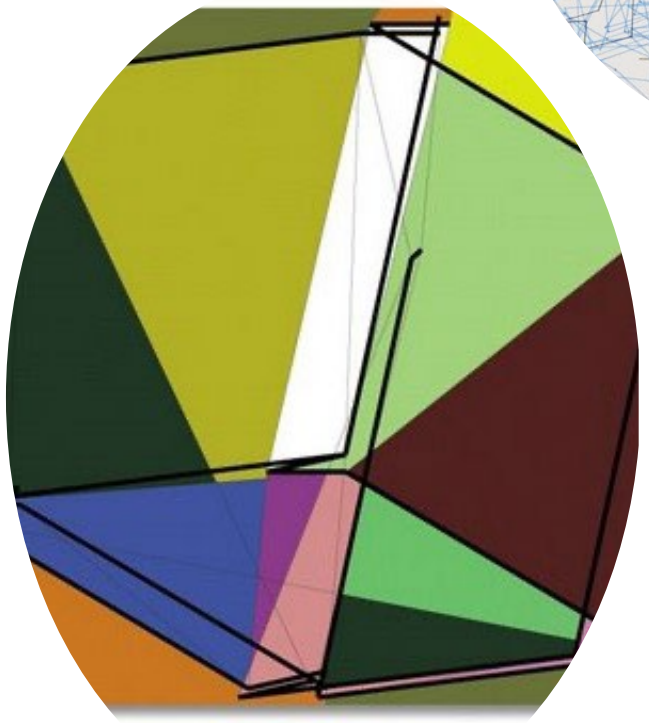
Michael Noll (b. 1939): Conducted research at Bell Labs and explored computational aesthetics.



[3] Nees, Georg (1969). Generative Computergraphik.

Some background on Creative Computation





Generative Art as Conceptual Shift

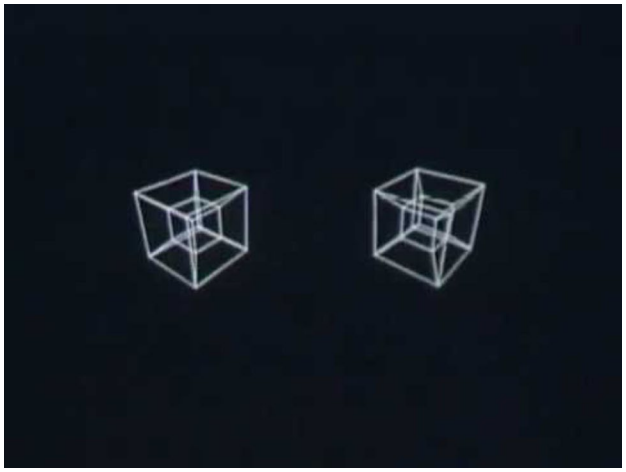
- Algorithmic art redefines authorship.
- Instead of composing a fixed image,the artist designs a system that produces images.
- Key idea:
  - The artwork is a process, not just an object.
  - This keeps it intellectually sharp.

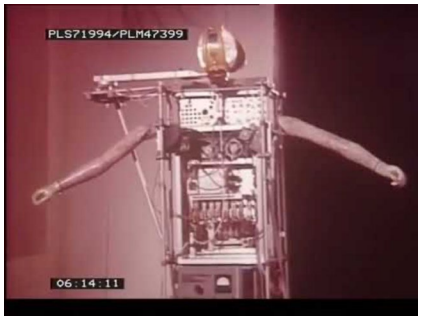
Vera Molnár (1924–2023): Used early computers to explore geometric abstraction and randomness.

Manfred Mohr (b. 1938): Developed visual works based on logical structures and hypercubes.

Herbert W. Franke (1927–2022): Combined scientific knowledge and artistic practice, exploring computational aesthetics.[4]

[4] Franke, H. W. (1971). *Computer Graphics – Computer Art*. Phaidon Press.





Some background on Creative Computation

From Laboratories to the Art World

- Early computer art emerged in research environments.
- Bell Labs (USA) became a key site for experimentation.
- Siemens Research supported generative exhibitions. [5]
- In Britain, Gordon Pask and Roy Ascott explored cybernetic art.

Technological catalyst

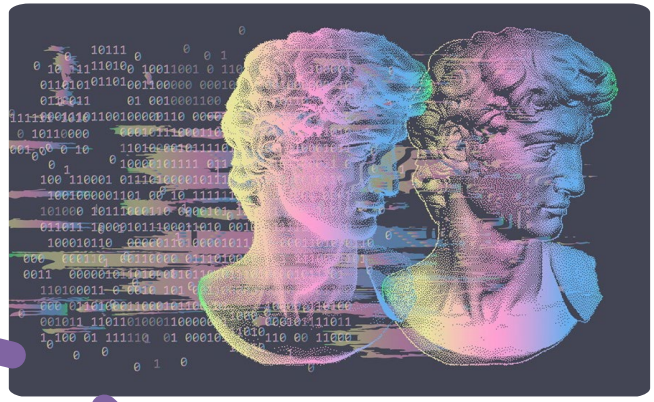
- The plotter printer enabled precise, rule-based drawing.
- Major exhibitions marked the transition:
  - 1965 Generative Computergrafik
  - Cybernetic Serendipity[6]
  - Ars Electronica

Key idea

- Computer art moved from technical research into contemporary art discourse.

[5] Galanter, P. (2003). *What is Generative Art? Complexity Theory as a Context for Art Theory*. GA2003.

[6] Reichardt, J. (1968). Cybernetic Serendipity: The Computer and the Arts. Studio International.



Some background on Creative Computation

Impact and Continuity

By the late 1970s, algorithmic art was established.

Its principles continue through:

- **Creative Coding (Processing, p5.js)**: Algorithmic principles were revived with tools like Processing, co-created by Casey Reas back in 2001.
- **AI-Generated Art (GANs, DeepDream, DALL·E)**: The exploration of rules and randomness continues in modern AI-driven creative systems.
- **NFT and Blockchain Art**: The idea of algorithmically generated uniqueness is now prominent in digital art markets.

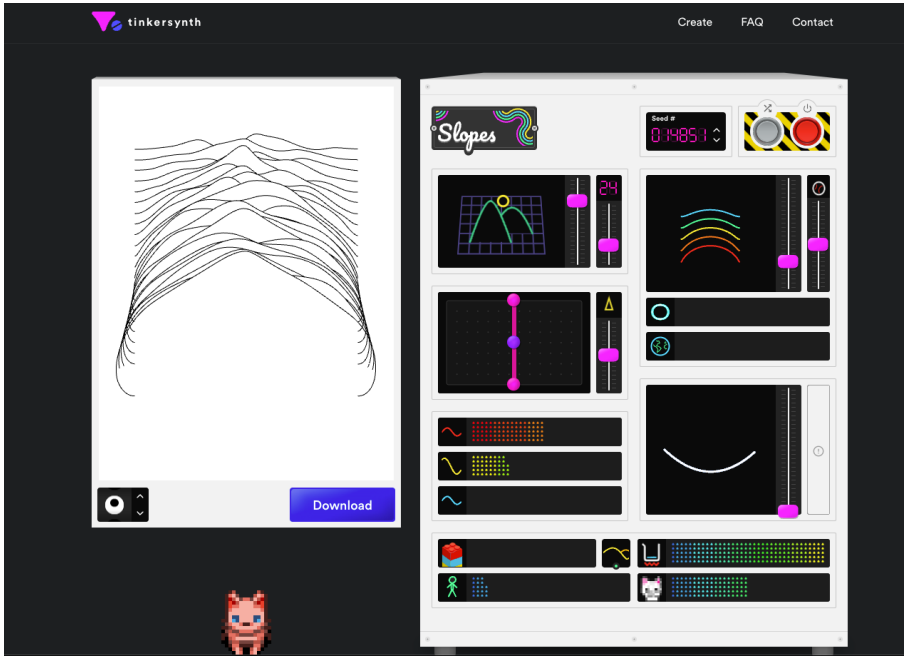
Core continuity

- Design rules.
- Allow systems to produce form.
- Authorship becomes systemic rather than manual.

Grau, O. (2003). Virtual Art: From Illusion to Immersion. MIT Press.

Let’s try out something!

https://tinkersynth.com/



Algorithmic Thinking

Algorithmic thinking means solving problems through precise, ordered steps.

It shapes programming, but also creative processes and everyday actions.

Key Elements of Algorithmic Thinking:

- **Decomposition** - Break a complex problem into smaller parts
- **Pattern recognition** - Identify repetition and structure
- **Abstraction** - Focus on essential elements
- **Sequencing** - Arrange steps logically
- **Optimization** - Refine and improve the solution

Simple Example:
Making a cup of tea can be described algorithmically:

1. Boil water.
2. Add a tea bag to a cup.
3. Pour hot water into the cup.
4. Let the tea steep for 3–5 minutes.
5. Remove the tea bag and enjoy.

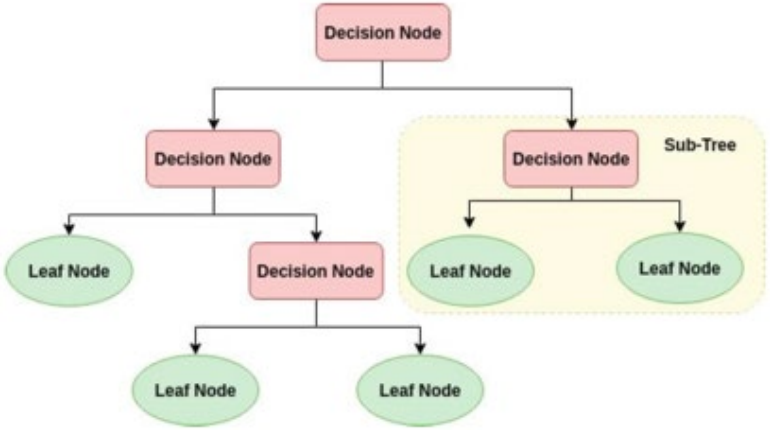
Each step is **precise**, **ordered**, and **repeatable!**
Just like an algorithm in programming.

Algorithmic Thinking

Algorithmic Thinking in Different Fields:

- **Computing** - Writing a program to sort numbers
- **Mathematics** - Solving equations step by step
- **Art and Design** - Creating generative systems with rules
- **Daily Life** - Following a recipe or instructions

$$2 - x^2 = -4 - x^2 + 6x$$
$$2 + 4 = 6x$$
$$6 = 6x$$
$$x = \frac{6}{6} = 1$$



Parts of an Algorithm

- Input - Starting data or conditions
- Process - Steps and operations
- Decision - Conditions and branching
- Iteration - Repetition or loops
- Output – Result
- Termination - When the process stops

Example "Find Max Number from list" Algorithm Steps:

- Input
  - A list of numbers
- Process
  - Assume the first number is the maximum
  - Compare each number to the current maximum
  - If a number is larger, update the maximum
  - Repeat until all numbers are checked
- Output
  - The largest number

Cormen, T. H., Leiserson, C. E., Rivest, R. L., & Stein, C. (2009). *Introduction to Algorithms*. MIT Press.

Programming Languages

Machine Language (Low-Level)

- Binary instructions understood directly by the CPU
- Very fast
- Not human readable
- Hardware specific

```
10110000 01100001
```

This means in Assembly: **MOV AL, 61h**
(*Move hexadecimal value 61 into register AL*).

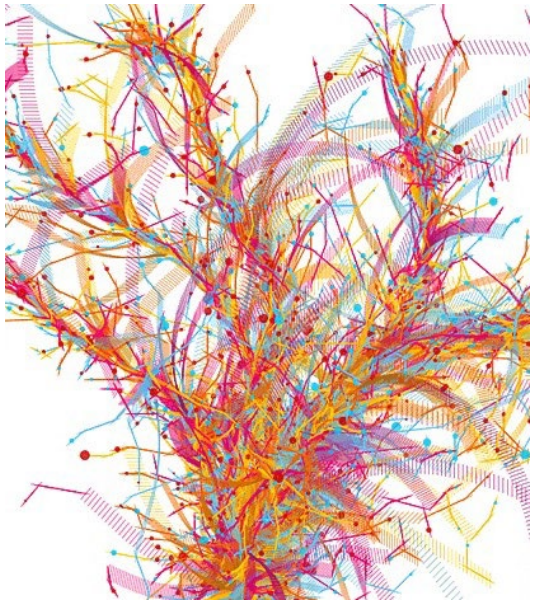
High-Level Language

- Designed for humans to read and write
- Needs translation into machine code
- Portable across systems
- Examples
  - Python
  - Java
  - JavaScript

```python
x = 10
y = 20
sum = x + y
print(sum)   # Output: 30
```

Processing “language”

- Processing is a creative coding environment built on Java.
- It simplifies programming for visual work.
- Designed for
  - Artists
  - Designers
  - Beginners
- You write code, it draws graphics.



Processing Workflow

setup()

- Runs once at the start of the program.
- Used to:
  - Set the canvas size
  - Initialize variables
  - Load assets such as images or fonts

```
void setup() {
    size(400, 400);  // Set canvas size
    background(255);  // Set background color
}
```

```
void draw() {
    ellipse(mouseX, mouseY, 50, 50);  // Draw circle at mouse position
}
```

Processing Workflow

draw()

- Runs continuously after setup().
- Repeats about 60 times per second by default.
- Used for:
  - Animation
  - Interaction
  - Updating visuals

```
void mousePressed() {
    background(random(255));  // Change background color on click
}
```

Processing Workflow

Event Functions

- Respond to user input.
- mousePressed()
  - Runs when the mouse is clicked

```
void mousePressed() {
    background(random(255));  // Change background color on click
}
```

```
void keyPressed() {
    println("Key pressed: " + key);
}
```

Processing Workflow

Event Functions

- Respond to user input.
- mousePressed()
  - Runs when the mouse is clicked
- keyPressed()
  - Runs when a key is pressed

```
void setup() {
    size(400, 400);
    frameRate(30);  // Limit to 30 FPS
}

void draw() {
    line(random(width), random(height), random(width), random(height));
}

void mousePressed() {
    noLoop();  // Stop the draw loop when clicked
}
```

Processing Workflow

Control Functions

- noLoop()
  - Stops draw() from repeating
- loop()
  - Restarts draw()
- frameRate(x)
  - Sets how many times draw() runs per second

```
void drawCircle(float x, float y, float d) {
    ellipse(x, y, d, d);
}

void draw() {
    drawCircle(mouseX, mouseY, 50);
}
```

Processing Workflow

Custom Functions

- You can create your own functions.
- Call them inside setup() or draw().

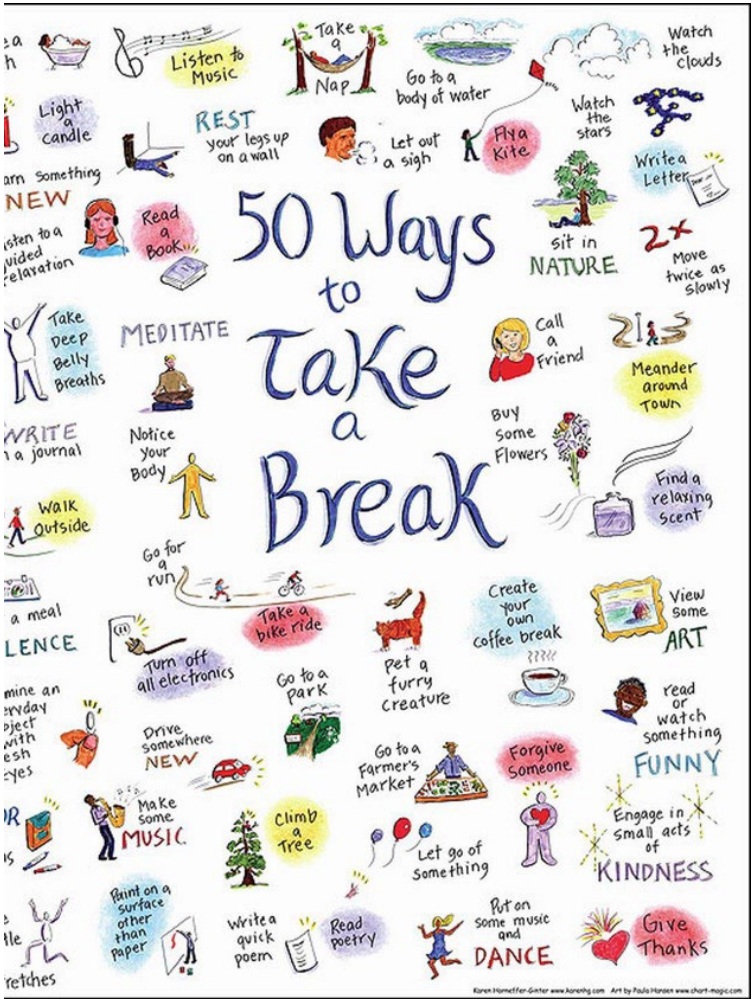
Setup

Break

15 minutes

Stretch. Coffee. Reset.

Back at 10:45



First Sketch explained

- size(400, 200)
  - Sets the window size to 400 by 200 pixels.
- background(0)
  - Sets the background color to black.
  - 0 represents black in grayscale.
- fill(255)
  - Sets the drawing color to white.
  - 255 represents white in grayscale.
- textSize(32)
  - Sets the text size to 32 pixels.
- textAlign(CENTER, CENTER)
  - Aligns the text horizontally and vertically to the center of the canvas.
- text("Hello, World!", width / 2, height / 2)
  - Draws the text in the center of the window.
  - width and height refer to the current canvas dimensions.

```
void setup() {
    size(400, 200);  // Set canvas size
    background(0);   // Set background color to black
    fill(255);       // Set text color to white
    textSize(32);    // Set text size
    textAlign(CENTER, CENTER);  // Center the text
}

void draw() {
    text("Hello, World!", width / 2, height / 2);  // Display text in the center
}
```

Second “First Sketch” explained

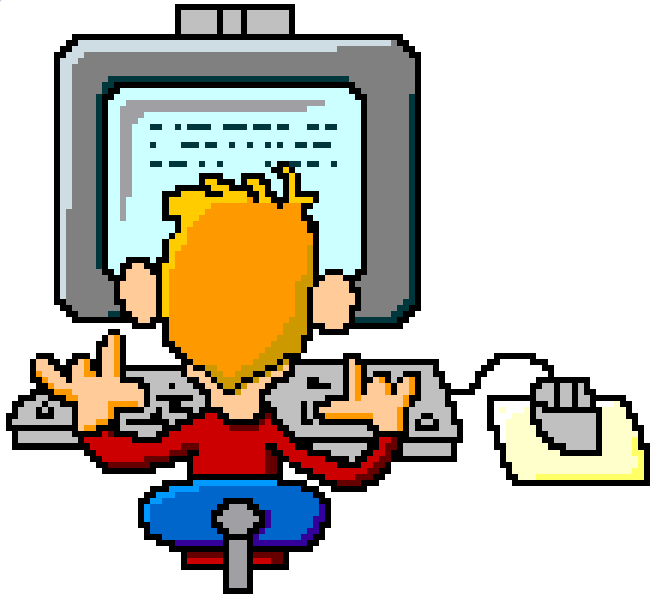
- Window size remains 400 by 200 pixels.
- An ellipse is drawn at the center.
- Text is positioned inside the ellipse.
- background() is placed inside draw(), so the frame refreshes every loop and prevents visual trails.

```
void setup() {
    size(400, 200);   // Set canvas size
    background(0);    // Set background color to black
    textSize(24);     // Set text size
    textAlign(CENTER, CENTER); // Center the text
}

void draw() {
    background(0);    // Clear the screen each frame

    // Draw an ellipse in the center
    fill(255);  // Set ellipse color to white
    ellipse(width / 2, height / 2, 150, 100);

    // Display "Hello, World!" inside the ellipse
    fill(0);   // Set text color to black for contrast
    text("Hello, World!", width / 2, height / 2);
}
```



Hands-On Exercise!

Your task

- Change shape properties
- Modify colors
- Adjust the background
- Experiment. Break things. See what happens.

Template

- Download from:
  - https://codeberg.org/ptiagomp/aalto-programming-visual-artists-25-26/src/branch/main/Session-01_23022026
- Optional
  - If you finish early, try the extra file.

Discussion and Questions

What surprised you today?
What felt difficult?
What felt interesting?

Tomorrow

- We start drawing with code.
- Shapes, color, interaction.