Aalto University
School of Arts, Design
and Architecture

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

Programming for Visual Artists

Quick Recap

Variables & Interaction

Animation Basics

State Variables

Control Structures

BREAK

Hands-On Exercise

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

https://processing.org/reference

What We Already Know

- draw() runs continuously
- Variables store changing values
- mouseX and mouseY update in real time
- Movement = values changing over time

What makes animation happen?

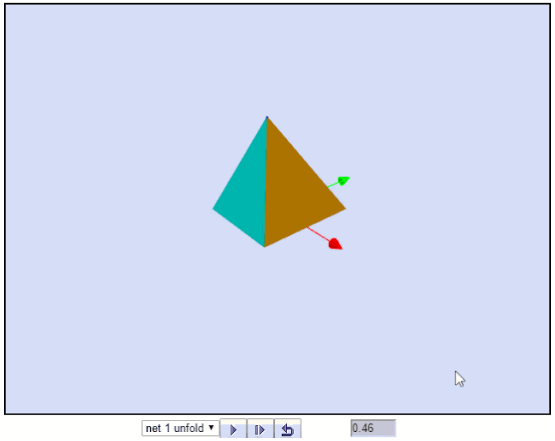
What repeats?

What changes?

Variables We Use Today

Type	What It's For	Example Declaration	Used Like This (Context)
`int`	Counting / loop control	`int i = 0;`	`for (int i = 0; i < 10; i++) { ellipse(50 + i*40, 200, 30, 30); }`
`float`	Smooth values / motion	`float x = 0;`	`x += 0.5;`
`boolean`	State (true / false)	`boolean isDark = false;`	`if (isDark) { background(0); }`
`color`	Store reusable color values	`color c = color(255, 0, 0);`	`fill(c);`

Shapes We Will Use Today



Shape	Purpose in Today's Session	Example
`rect()`	Grid structures / modular systems	`rect(x, y, 40, 40);`
`ellipse()`	Organic grids / interactive points	`ellipse(x, y, 30, 30);`
`line()` *(optional)*	Connecting elements / patterns	`line(x1, y1, x2, y2);`

Interaction = Input → Update → Redraw

- Mouse and keyboard change variables
- Variables affect what draw() displays
- draw() keeps updating

Last session, one object followed the mouse.

Today, many objects will react.

Interaction	What It Changes
`mouseX` / `mouseY`	Position
`mousePressed()`	State (boolean)
`keyPressed()`	Mode / behavior

Logical & Comparison Operators

Operator	What It Means	Example in Code	What Happens Visually
`==`	Equal to	`if (x == 100)`	An event triggers only when an object hits a **precise spot**.
`!=`	Not equal to	`if (state != "paused")`	The animation keeps playing **unless** the game is paused.
`<`	Less than	`if (mouseX < width/2)`	The effect only happens on the **left half** of the screen.
`>`	Greater than	`if (mouseY > height/2)`	The effect only happens on the **bottom half** of the screen.
`&&`	**AND** (Both must be true)	`if (x > 200 && y > 200)`	Limits the "active zone" to a specific **box or corner**.
`||`	**OR** (At least one is true)	`if (x < 50 || x > 450)`	Triggers when something hits **either the left or right edge** (like a boundary).
`!`	**NOT** (Inverts logic)	`if (!isGameOver)`	Logic runs as long as the "Game Over" screen **is not** showing.

Animation in Processing

Frame-Based Movement
(frameRate)

- draw() runs repeatedly
- Each frame updates values
- Changing values = movement

```
float x = 0;

void setup() {
    size(400, 400);
    frameRate(60); // Attempt to run at 60 FPS
}

void draw() {
    background(240);
    ellipse(x, height / 2, 50, 50);
    x += 2; // Moves 2 pixels per frame
}
```

Animation in Processing

Time-Based Animation (millis())

- millis() gives elapsed time
- Movement can depend on time instead of frames
- Speed becomes more consistent

```
float x = 0;
float speed = 100; // pixels per second
float lastTime;

void setup() {
  size(400, 400);
  lastTime = millis();
}

void draw() {
  background(240);
  ellipse(x, height / 2, 50, 50);

  float currentTime = millis();
  float deltaTime = (currentTime - lastTime) / 1000.0; // convert to seconds

  x += speed * deltaTime;

  lastTime = currentTime;
}
```

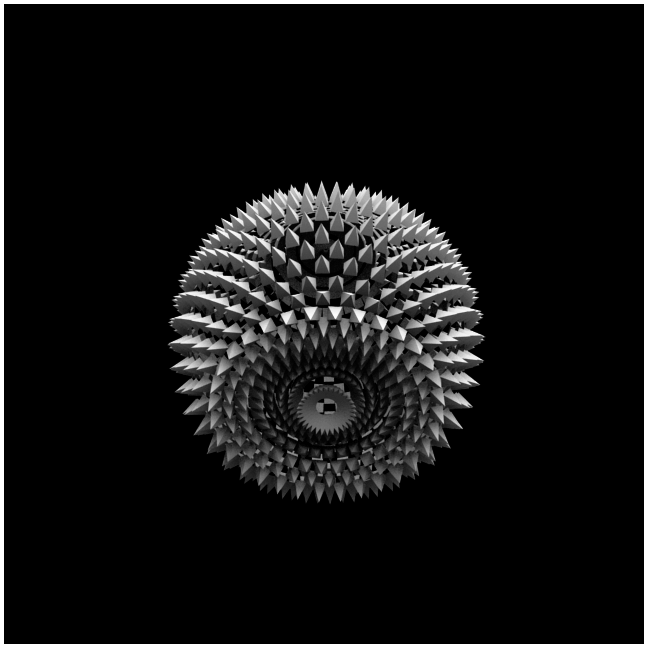
Animation in Processing

When to Use Each?

Frame-based
- Easy
- Good for most sketches

Time-based
- More precise
- Better for complex motion

Most of the time, frame-based animation is enough.
Time-based is useful when motion needs to be precise.



State Variables

A state variable remembers something between frames.

Variable	Type	What It Stores	Example Use
`isDarkMode`	`boolean`	Light or dark mode	`if (isDarkMode) background(0);`
`isMoving`	`boolean`	Whether animation runs	`if (isMoving) x += 2;`
`currentMode`	`int`	Which visual mode is active	`if (currentMode == 1) { ... }`
`hovering`	`boolean`	Mouse over element	`if (hovering) fill(255,0,0);`

State Variables

```
boolean isRed = false; // State variable

void setup() {
    size(400, 400);
}

void draw() {
    if (isRed) {
        background(255, 0, 0); // Red background
    } else {
        background(0, 0, 255); // Blue background
    }
}

void mousePressed() {
    isRed = !isRed; // Toggle state on mouse click
}
```

Control Structures

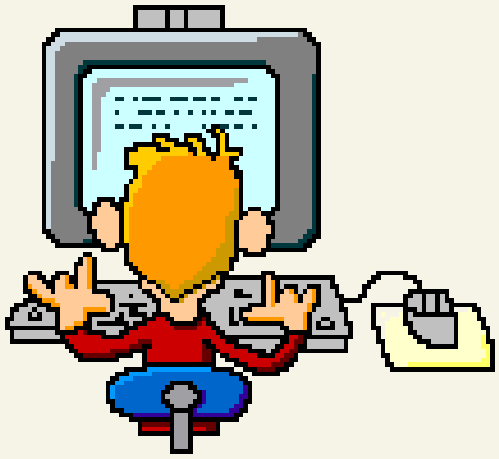
They control:

- When something happens
- How many times something happens

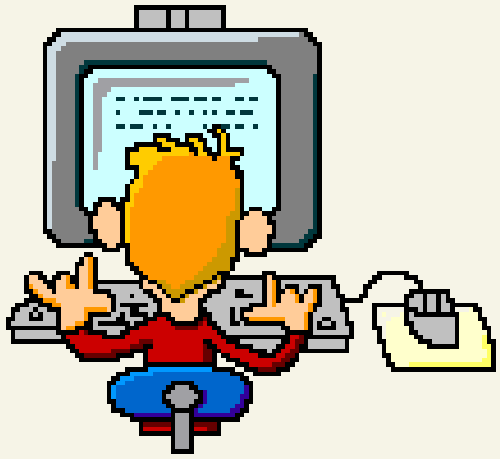
Structure	What It Does
`if`	Makes decisions
`for`	Repeats code

```
void setup() {
  size(400, 400);
}

void draw() {
  if (mouseX > width/2) {
    background(255, 0, 0);
  } else {
    background(0, 0, 255);
  }
}
```



Control Structures - conditionals



```
void setup() {
  size(400, 400);
}

void draw() {
  background(240);

  for (int i = 0; i < 10; i++) {
    ellipse(50 + i * 40, height/2, 30, 30);
  }
}
```

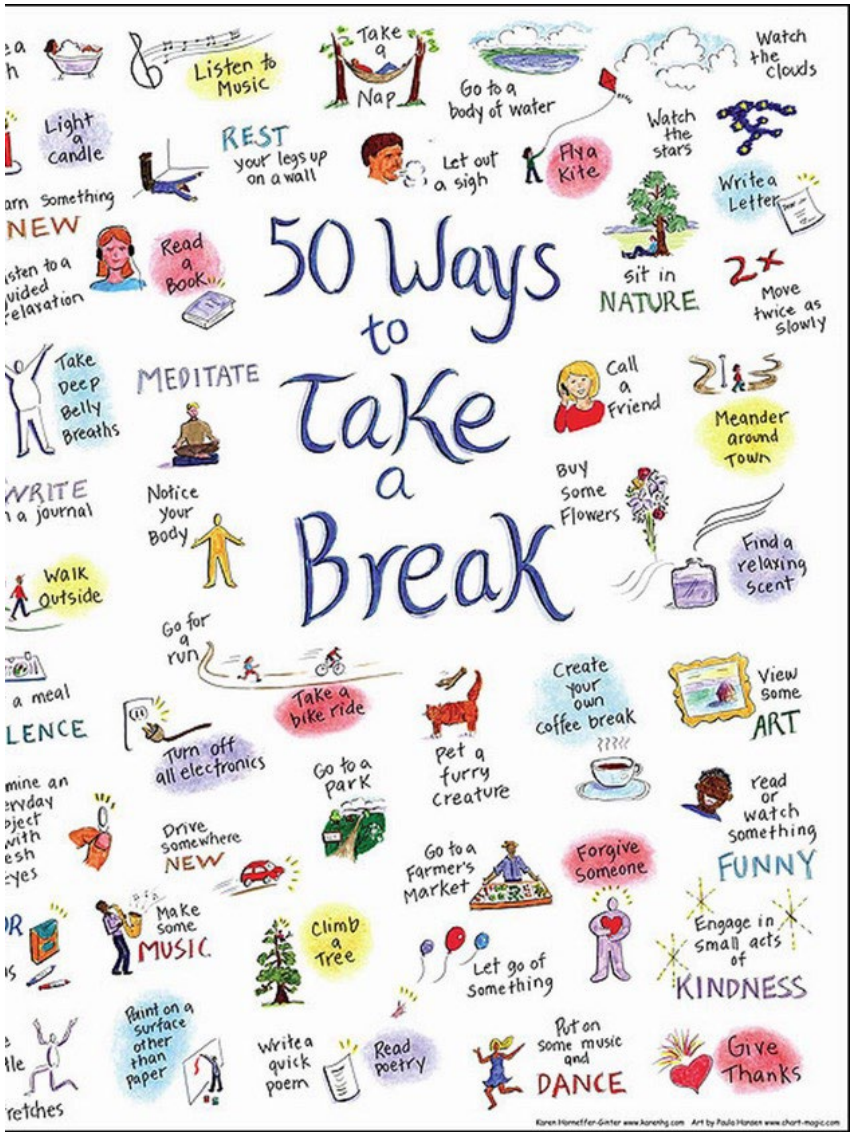
Control Structures - loops

Break

15 minutes

Stretch. Coffee. Reset.

Back at 10:45



Hands-On Exercise!

Your task

- Create a grid using nested for loops
- Calculate each cell's position
- Make each cell react to the mouse
- On hover → change color

Think about:

- How do you compute each cell's x and y?
- How do you check if the mouse is inside a cell?
- Can each cell behave differently?

Break things. Change numbers. Explore.

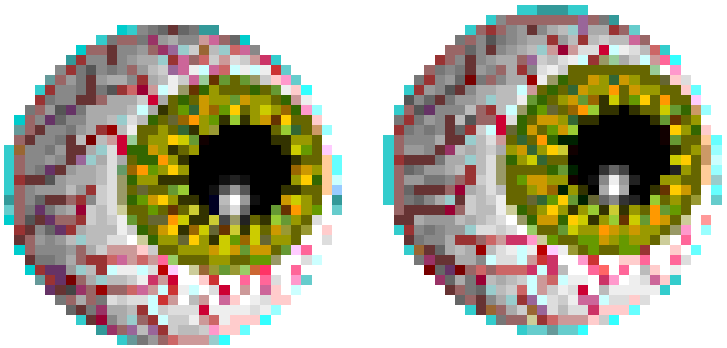
Template

- Download from: https://codeberg.org/ptiagomp/aalto-programming-visual-artists-25-26/src/branch/main/Session-03_02032026

Optional Challenge

If you finish early:

- Add transparency
- Add keyboard interaction
- Animate something inside each cell
- Create a small composition



Discussion and Questions

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

Today you stopped placing shapes.
You started designing behavior.

Tomorrow
We move from grids to algorithmic patterns.
We start shaping behavior into generative art.

Don’t forget the assignments!