



**Aalto University
School of Arts, Design
and Architecture**

Programming for Visual Artists

2024/2025
Department of Art
and Media



Some things to check

Processing reference:

<https://processing.org/reference>



Recap



**IN CASE YOU
MISSED IT**



- Basics of Computation and Maths for Computer Science
- Drawing and Interaction

Recap – variable types

| Type | Example | Description |
|----------------------|---------------------------------|---|
| <code>int</code> | <code>int x = 10;</code> | Whole numbers (e.g., <code>-3</code> , <code>0</code> , <code>42</code>) |
| <code>float</code> | <code>float y = 5.5;</code> | Decimal numbers (e.g., <code>3.14</code> , <code>-0.7</code>) |
| <code>boolean</code> | <code>boolean on = true;</code> | <code>true</code> or <code>false</code> values |
| <code>char</code> | <code>char letter = 'A';</code> | Single characters (e.g., <code>'A'</code> , <code>'3'</code> , <code>'#'</code>) |

| Type | Example | Description |
|--------------------------------------|--|--------------------------------|
| <code>String</code> | <code>String name = "Hello";</code> | A sequence of characters |
| <code>int[]</code> | <code>int[] numbers = {1, 2, 3};</code> | An array of integers |
| <code>float[]</code> | <code>float[] values = new float[10];</code> | An array of floats |
| <code>ArrayList<String></code> | <code>ArrayList<String> list = new ArrayList<String>();</code> | A dynamic list of elements |
| <code>PVector</code> | <code>PVector pos = new PVector(10, 20);</code> | A 2D/3D vector for coordinates |

Processing includes special types for graphics and interaction:

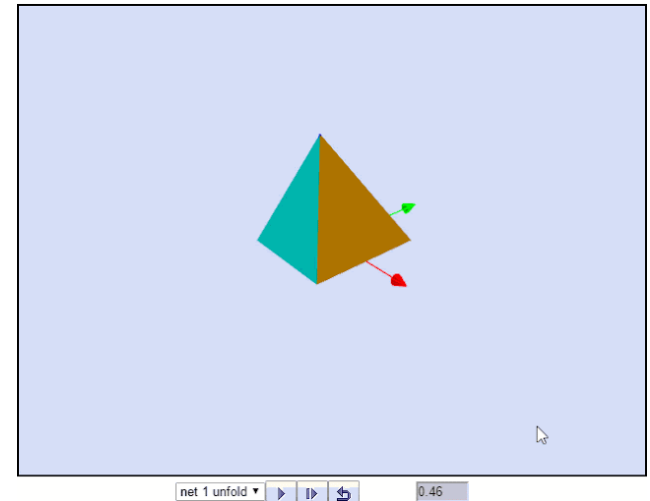
| Type | Example | Description |
|---------------------|---|-------------------------|
| <code>color</code> | <code>color c = color(255, 0, 0);</code> | Stores RGB color values |
| <code>PImage</code> | <code>PImage img = loadImage("pic.jpg");</code> | Stores an image |
| <code>PFont</code> | <code>PFont f = createFont("Arial", 16);</code> | Stores a font |
| <code>PShape</code> | <code>PShape s = loadShape("star.svg");</code> | Stores vector shapes |



Recap - shapes

Common shapes:

- `point(x, y);`
- `line(x1, y1, x2, y2);`
- `rect(x, y, width, height);`
- `ellipse(x, y, width, height);`
- `triangle(x1, y1, x2, y2, x3, y3);`
- `arc(x, y, width, height, start, stop);`



Recap - interaction

// Mouse Interaction Functions

void mousePressed() {}

void mouseReleased() {}

void mouseClicked() {}

void mouseDragged() {}

void mouseMoved() {}

void mouseWheel(MouseEvent event) {}

// Keyboard Interaction Functions

void keyPressed() {}

void keyReleased() {}

void keyTyped() {}

Welcome

Recap

Today's Goals

Animation in Processing

Frame-Based Animation

Time-Based Animation

State Variables

Control Structures

If...Else

While/For

Case/switch

Examples

BREAK (10:30–10:45)

Coding tasks

Grid of Shapes


Animation ball

Q&A

For tomorrow...

For tomorrow...

Both `pushMatrix()` and `popMatrix()` save and restore transformations to avoid full canvas transformation. Otherwise the transformations would affect the full canvas!



```
pushMatrix(); // Save current transformation
translate(100, 100);
rotate(radians(30));
rect(0, 0, 50, 50);
popMatrix(); // Restore original transformation
```

```
float angle = radians(45); // Convert degrees to radians
translate(width/2, height/2); // Move the origin to the center
rotate(angle); // Rotate 45 degrees
rect(-25, -25, 50, 50); // Draw rectangle centered at origin
```

Good to know

```
scale(2); // Doubles the size of all shapes
rect(10, 10, 50, 50); // Now appears twice as large
```

```
translate(50, 50); // Moves the origin (0,0) to (50,50)
rect(0, 0, 50, 50); // Draws the rectangle at the new origin
```


Good to know

Logical Operators

&& (AND)

- Returns true if **both** conditions are true.

|| (OR)

- Returns true if **at least one** condition is true.

^ (XOR - Exclusive OR)

- Returns true if **exactly one** condition is true, but not both.

Comparison Operators

== (Equal)

- Checks if two values are the same.

!= (Different)

- Checks if two values are **not** the same.

Animation in Processing

Frame-Based Animation

(**frameRate**)

- Movement and updates happen based on **frame count**.
- Objects move a fixed number of pixels **per frame**.
- If the frame rate drops (lag, slow computer), the animation **slows down**.
- The **speed is inconsistent** across different devices.

```
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()**)

- Uses **real time** (**millis()**, system time in milliseconds).
- Objects move based on **time elapsed**, not frames.
- **Smooth and consistent speed** even if the frame rate changes.
- Movement is **independent of frame rate**.
- **Consistent speed** across different computers.
- No lag or slowdown even if FPS drops.

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

void setup() {
  size(400, 400);
  lastTime = millis(); // Get initial time
}

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

  int currentTime = millis();
  float deltaTime = (currentTime - lastTime) / 1000.0; // Convert ms to seconds
  x += speed * deltaTime; // Move based on elapsed time
  lastTime = currentTime;
}
```

Animation in Processing

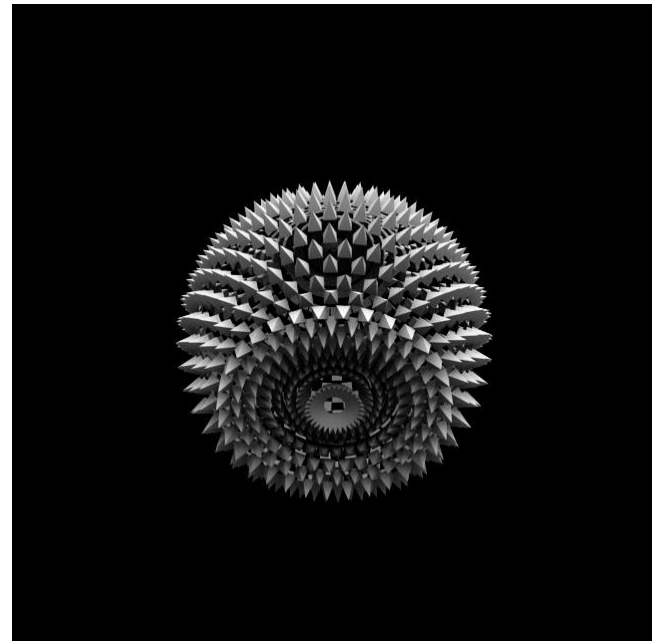
When to Use Each?

Use **Frame-Based Animation** for:

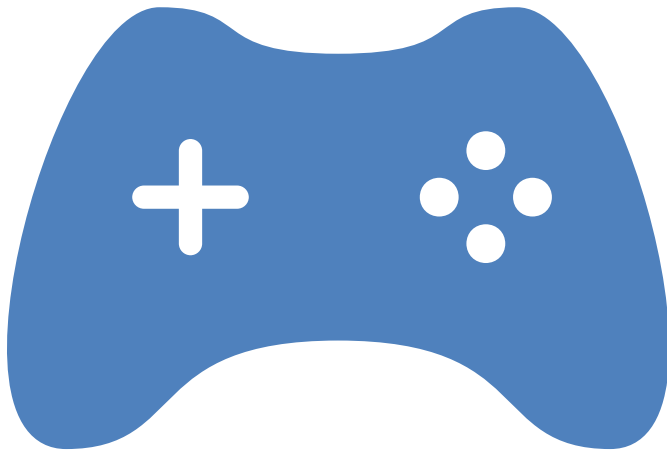
- Simple sketches where performance is stable.
- Projects where FPS is controlled (`frameRate(60);`).

Use **Time-Based Animation** for:

- Physics-based games, real-time applications.
- Smooth, **frame-rate independent** movement.



State Variables



Examples:

- **Game states** (PLAYING, PAUSED, GAME_OVER)
- **Tracking UI elements** (isMenuOpen, isButtonPressed)
- **Animation control** (isMoving, currentFrame)
- **Physics-based simulations** (isJumping, hasCollided)

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
}
```

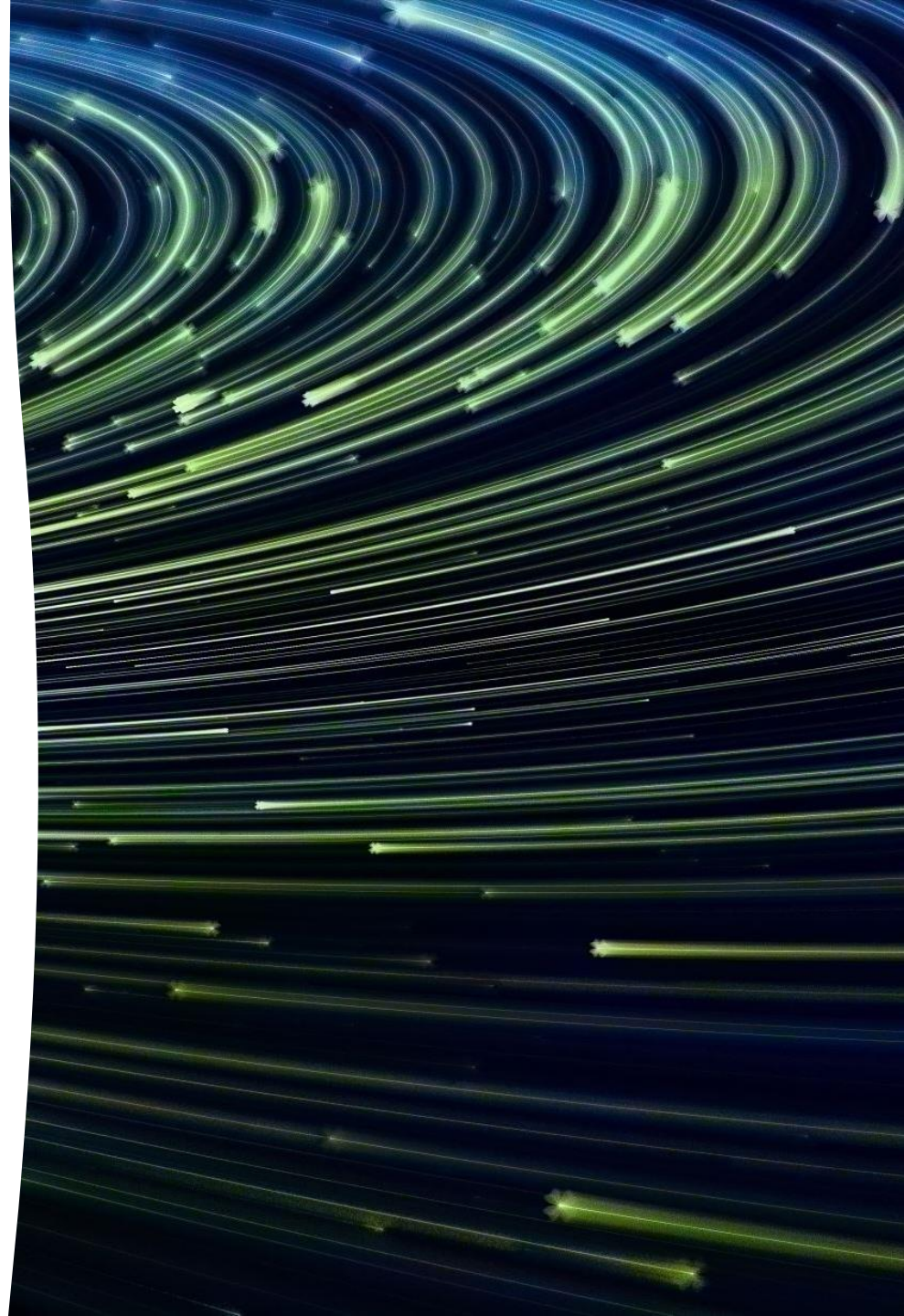
Control Structures

Conditionals

(if, else) for logic

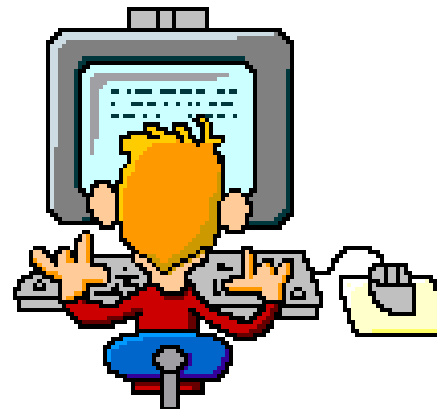
Loops (for, while) for
repetitive patterns or grids.

Switch/Case for options



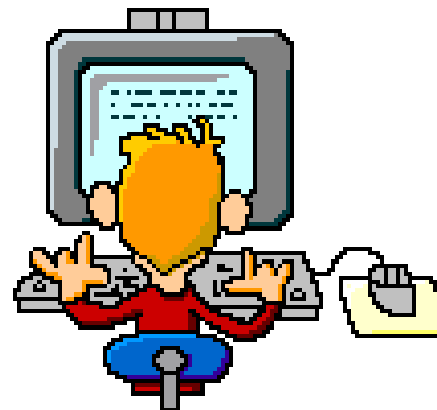
```
int x = 10;  
if (x > 5) {  
    println("x is greater than 5");  
} else {  
    println("x is 5 or less");  
}
```

Control Structures - conditionals

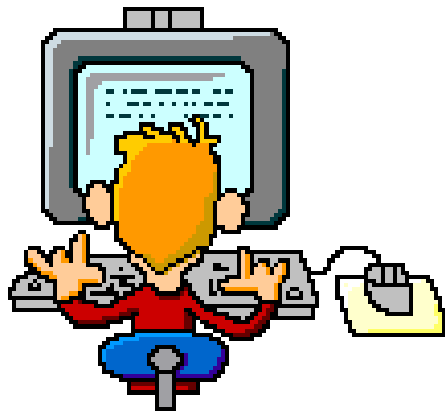



```
int i = 0;  
while (i < 10) {  
    println("Iteration: " + i);  
    i++;  
}
```

Control Structures - loops



Control Structures – switch/case



```
int num = 2;

switch (num) {
    case 1:
        println("One");
        break;
    case 2:
        println("Two");
        break;
    case 3:
        println("Three");
        break;
    default:
        println("Unknown number");
}
```

```
void setup() {  
    size(400, 400); // Canvas size  
}  
  
void draw() {  
    if (mouseX < width / 2) {  
        background(255, 0, 0); // Red  
    } else {  
        background(0, 0, 255); // Blue  
    }  
}
```

Examples -
conditionals

Examples – loops

```
void setup() {
    size(400, 400);
    int cols = 8; // Number of columns
    int rows = 8; // Number of rows
    int spacing = width / cols; // Spacing between shapes

    int y = 0;
    while (y < rows) {
        int x = 0;
        while (x < cols) {
            float posX = x * spacing + spacing / 2;
            float posY = y * spacing + spacing / 2;
            ellipse(posX, posY, spacing * 0.8, spacing * 0.8);
            x++; // Increment column
        }
        y++; // Increment row
    }
}
```

```
void setup() {
    size(400, 400);
    int cols = 8; // Number of columns
    int rows = 8; // Number of rows
    int spacing = width / cols; // Spacing between shapes

    for (int y = 0; y < rows; y++) {
        for (int x = 0; x < cols; x++) {
            float posX = x * spacing + spacing / 2;
            float posY = y * spacing + spacing / 2;
            ellipse(posX, posY, spacing * 0.8, spacing * 0.8);
        }
    }
}
```

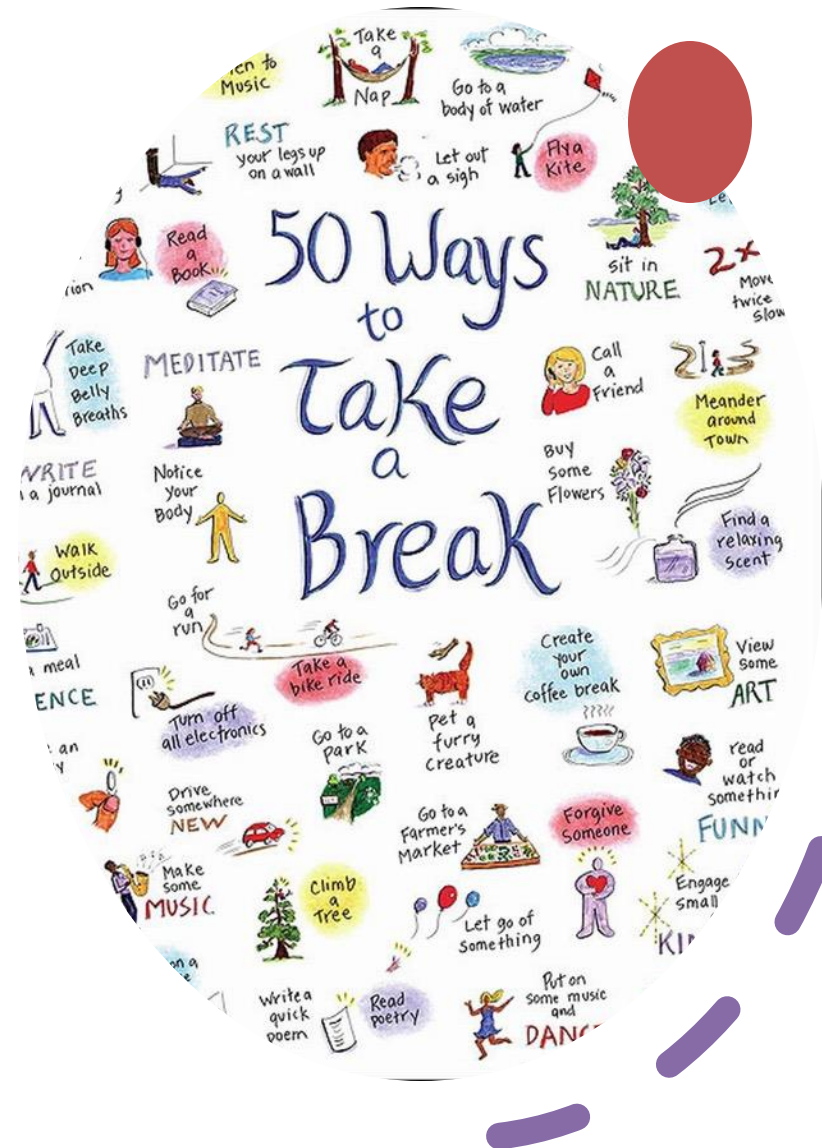
Examples – case/switch

```
void setup() {  
    size(400, 400);  
}  
  
void draw() {  
    // Background color is set in keyPressed(), so nothing needed here  
}  
  
void keyPressed() {  
    switch (key) {  
        case '1':  
            background(255, 0, 0); // Red  
            break;  
        case '2':  
            background(0, 255, 0); // Green  
            break;  
        case '3':  
            background(0, 0, 255); // Blue  
            break;  
        default:  
            background(200); // Gray (for other keys)  
            break;  
    }  
}
```

Break

15 min.!

Please don't be late!



Hands-On Exercise!

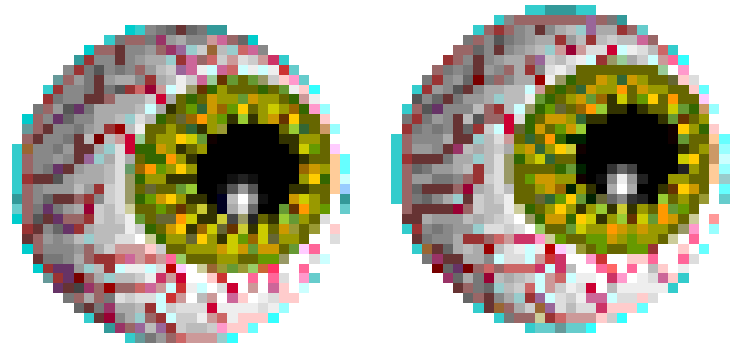
Objectives:

- Make a grid.
- On hover change color.
- Experiment! Try out things!

Get an example from here:

https://github.com/ptiagomp/aalto-programming-visual-artists-24-25/tree/main/Session-03_03032025

(if bored, check for the “extra” files!)





FEEDBACK

Discussion & Q&A

Share your feedback!

Am I going too fast or too slow? Is this too easy or too hard?

Tomorrow's topic:

- Generative patterns and loops

Don't forget the assignments!

