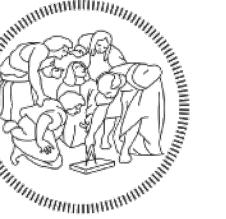


POLITECNICO  
MILANO 1863

# Digital Art

2021-2022

Introduction to processing



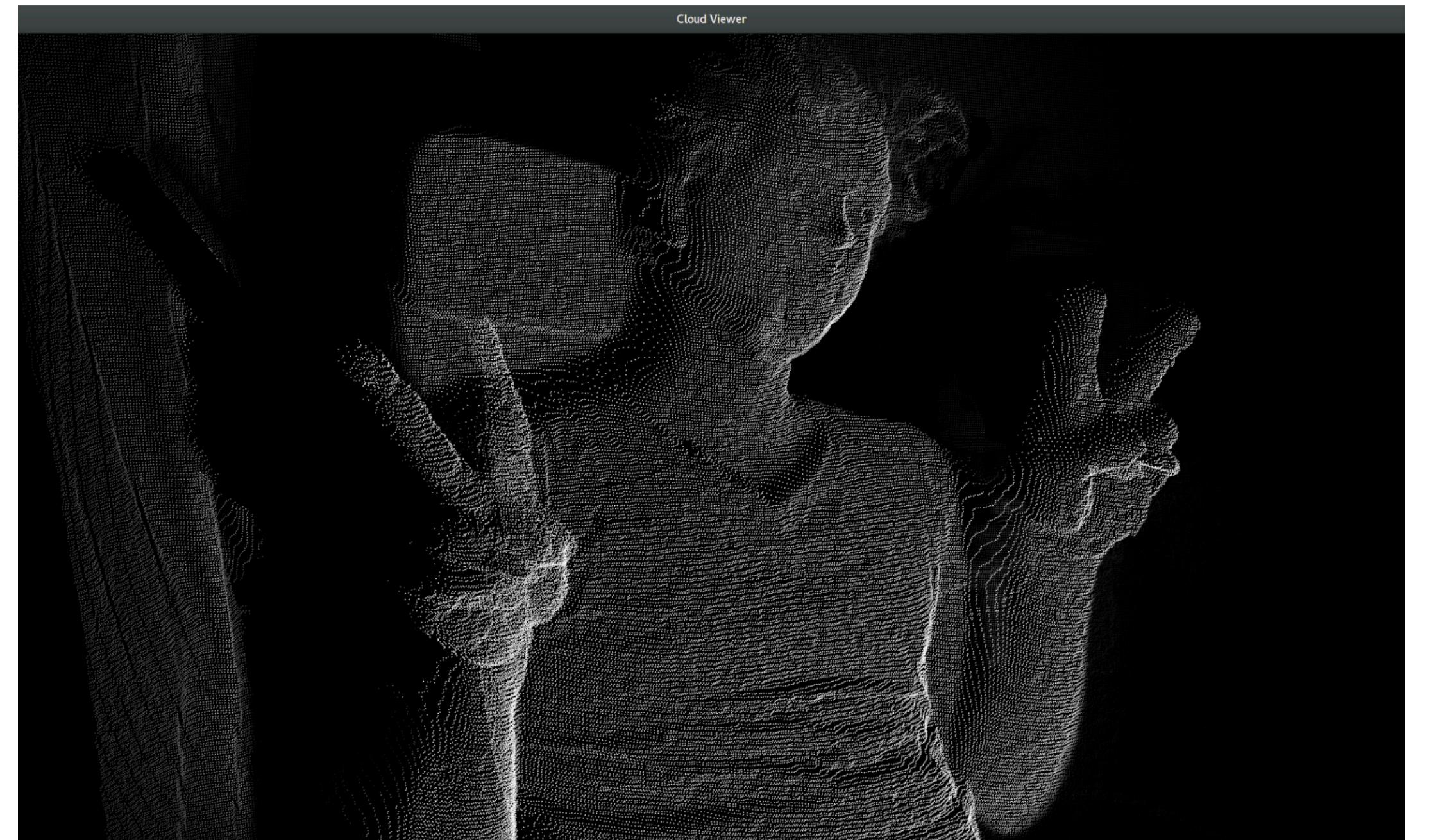
# Processing?

---

When they started Processing in 2001, the goal was to **bring ideas and technologies** out of MIT and **into the larger world**.

They called this **sketching with code**.

Processing emerged directly from the Aesthetics and Computation Group (ACG), a research group started at the Media Lab by John Maeda in 1996.



<https://medium.com/processing-foundation/a-modern-prometheus-59aed94abe85>



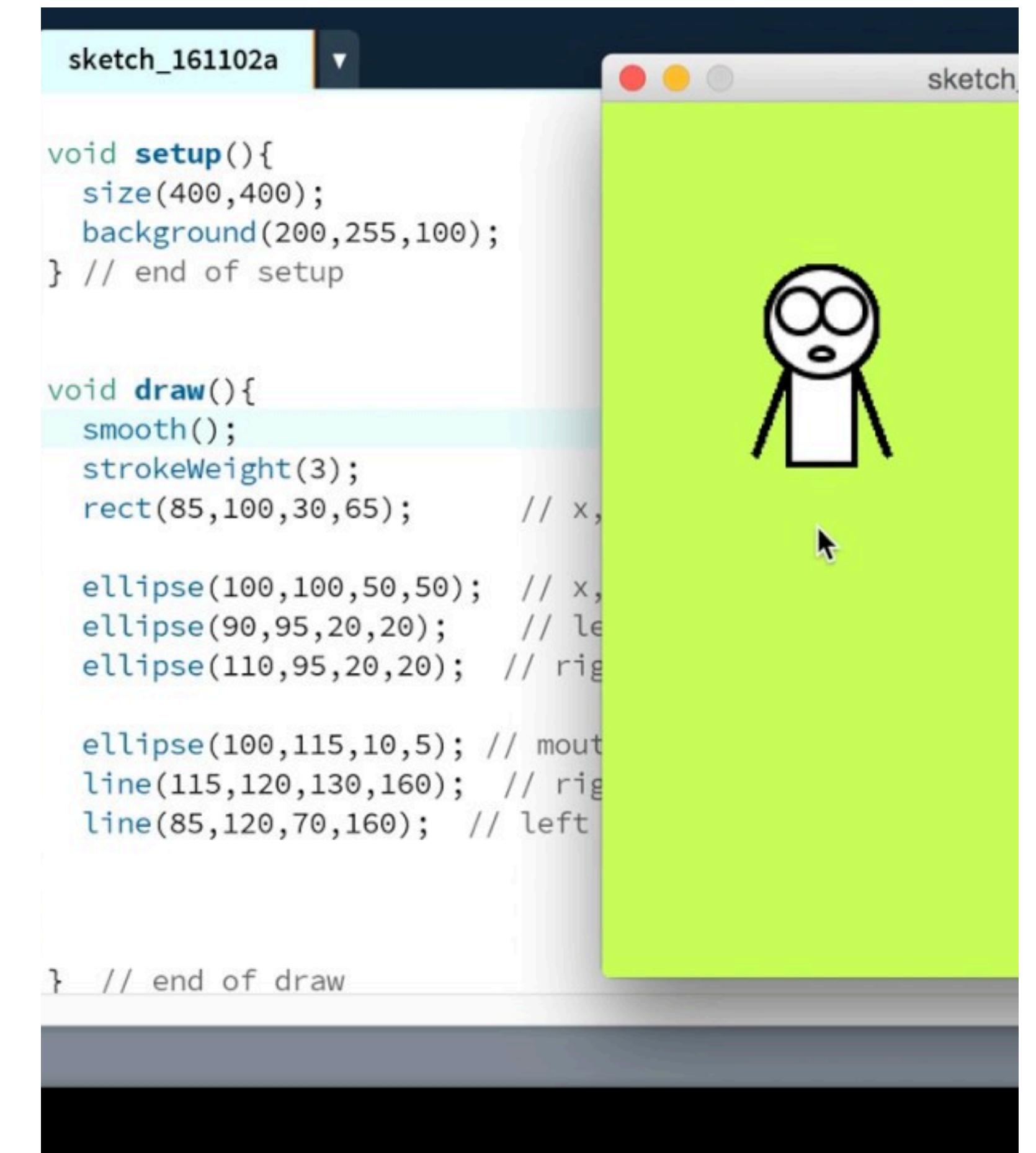
# Sketching?

A Processing program is called a **sketch**. This is more than a change in nomenclature, it's a different approach to coding.

The more traditional method is to resolve the entire plan for the software before the first line of code is written.

This approach can work well for well-defined domains, but when **the goal is exploration and invention**, it prematurely cuts off possible outcomes.

Through sketching with code, unexpected paths are discovered and followed. Unique outcomes often emerge through the process.



The screenshot shows a Processing sketch titled "sketch\_161102a". The code is as follows:

```
void setup(){
    size(400,400);
    background(200,255,100);
} // end of setup

void draw(){
    smooth();
    strokeWeight(3);
    rect(85,100,30,65); // x
    ellipse(100,100,50,50); // x
    ellipse(90,95,20,20); // left eye
    ellipse(110,95,20,20); // right eye
    ellipse(100,115,10,5); // mouth
    line(115,120,130,160); // right arm
    line(85,120,70,160); // left arm
} // end of draw
```

The sketch displays a simple character with a white body, black arms, and a black head featuring large, round, black eyes and a small black mouth.



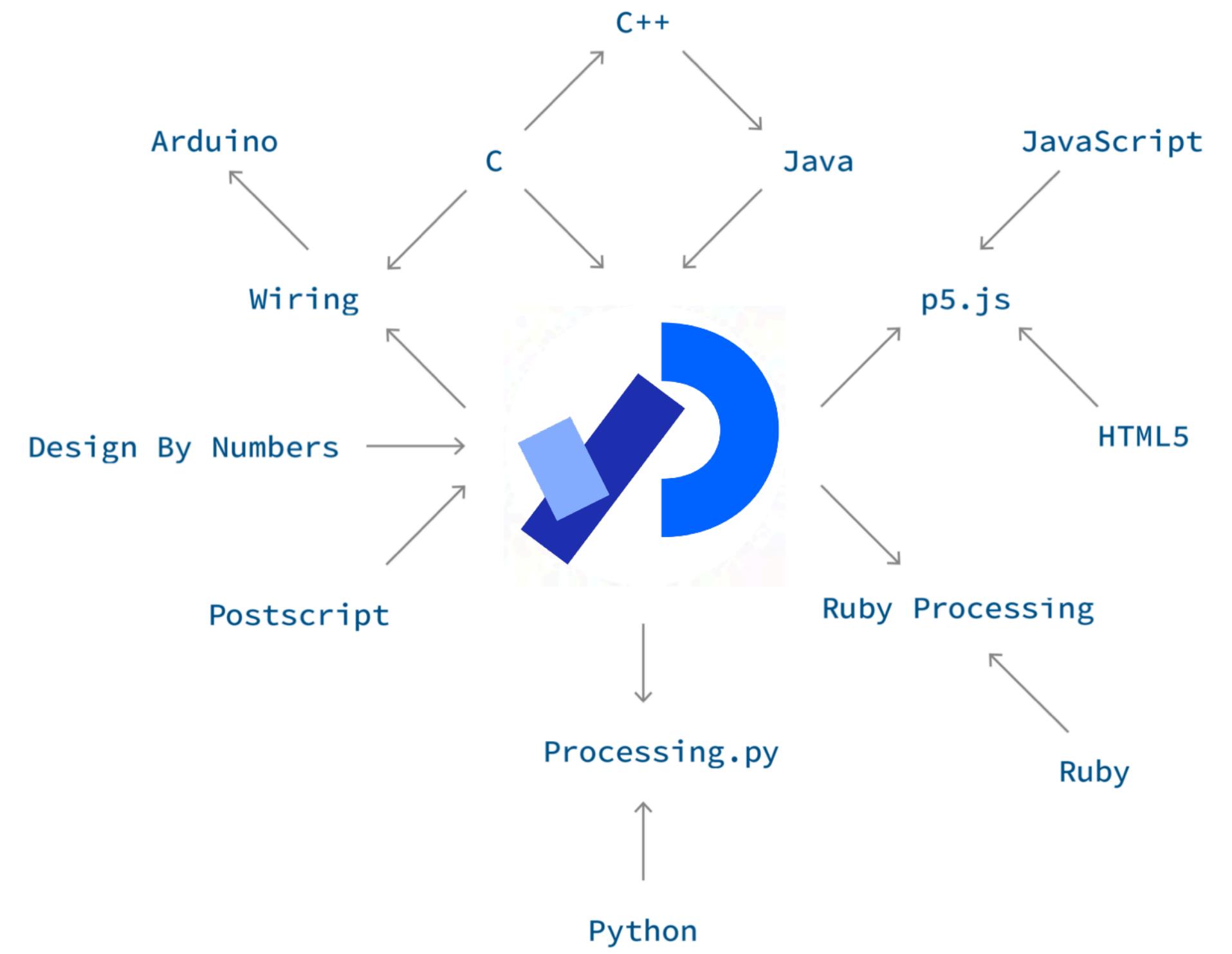
# What's for?

*The original mission of Processing was to create software that made learning to code accessible for visual people (designers, artists, architects) and to help a more technical audience work fluidly with graphics.*

From the original Processing software, the Foundation is now supporting a range of different projects.

The **p5.js** project is a JavaScript reimagining of Processing within the context of contemporary web browsers.

**Processing.py** it's now a Mode for the Processing 3 editor. Additionally, **Processing for Android** as a Mode for Processing 3, Processing 3 running well on **Raspberry Pi** and CHIP hardware, and there is a library to read and write directly to the I/O pins.





# Download

---

\_On **Windows**, you'll have a .zip file.

Double-click it, and drag the folder inside to a location on your hard disk. It could be Program Files or simply the desktop, but the important thing is for the processing folder to be pulled out of that .zip file. Then double-click processing.exe to start.

\_The **Mac OS X** version is also a .zip file.

Double-click it and drag the Processing icon to the Applications folder. If you're using someone else's machine and can't modify the Applications folder, just drag the application to the desktop. Then double-click the Processing icon to start.



# Download

---

**Download**

This Git with slides and examples



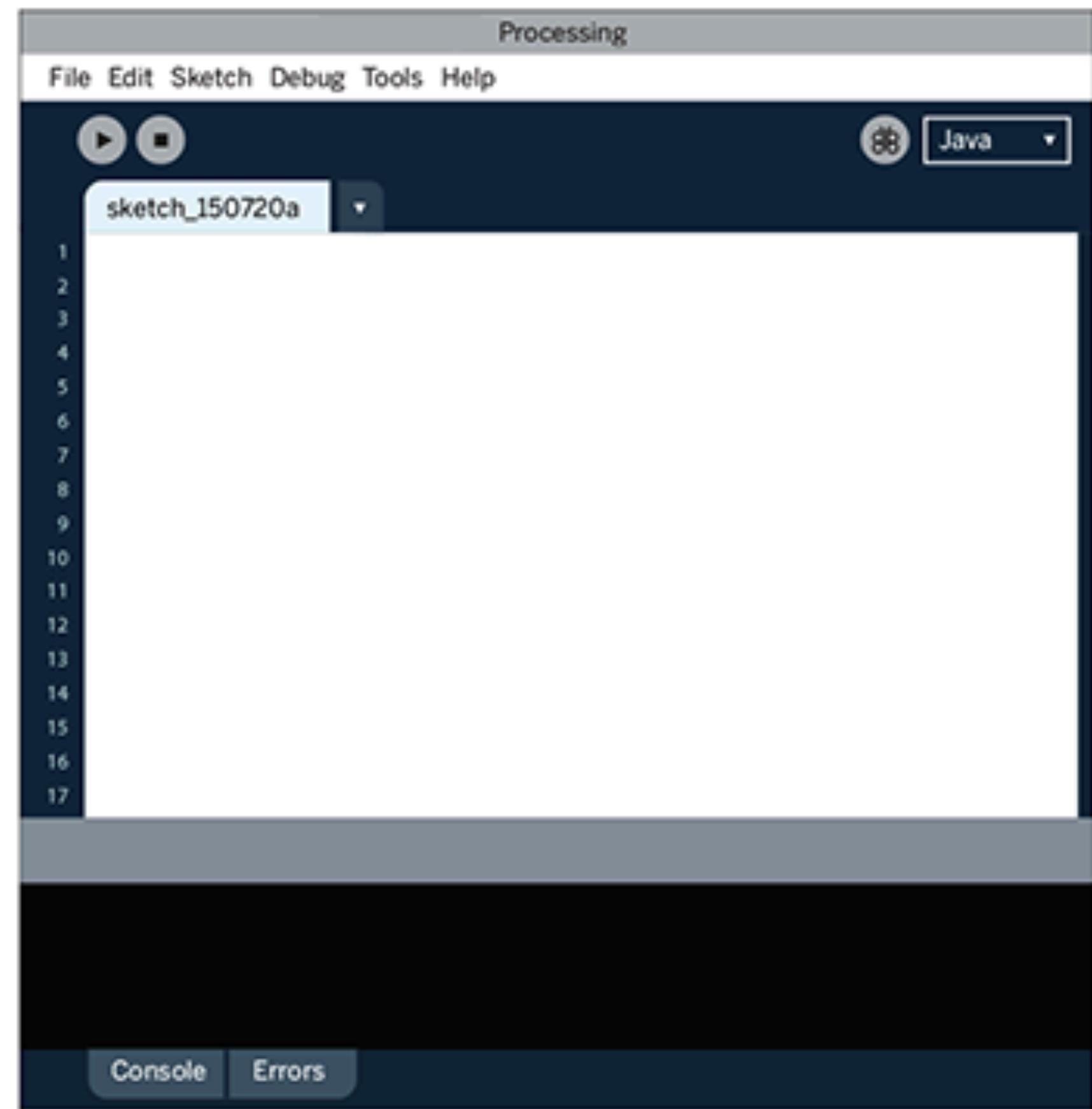
<https://github.com/giulioriot/digitalArt>



# Hello, I'm Processing



Display Window



Menu

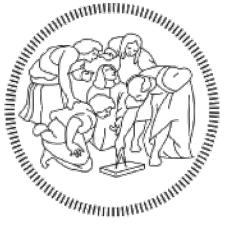
Toolbar

Tabs

Text Editor

Message Area

Console



# How it works?

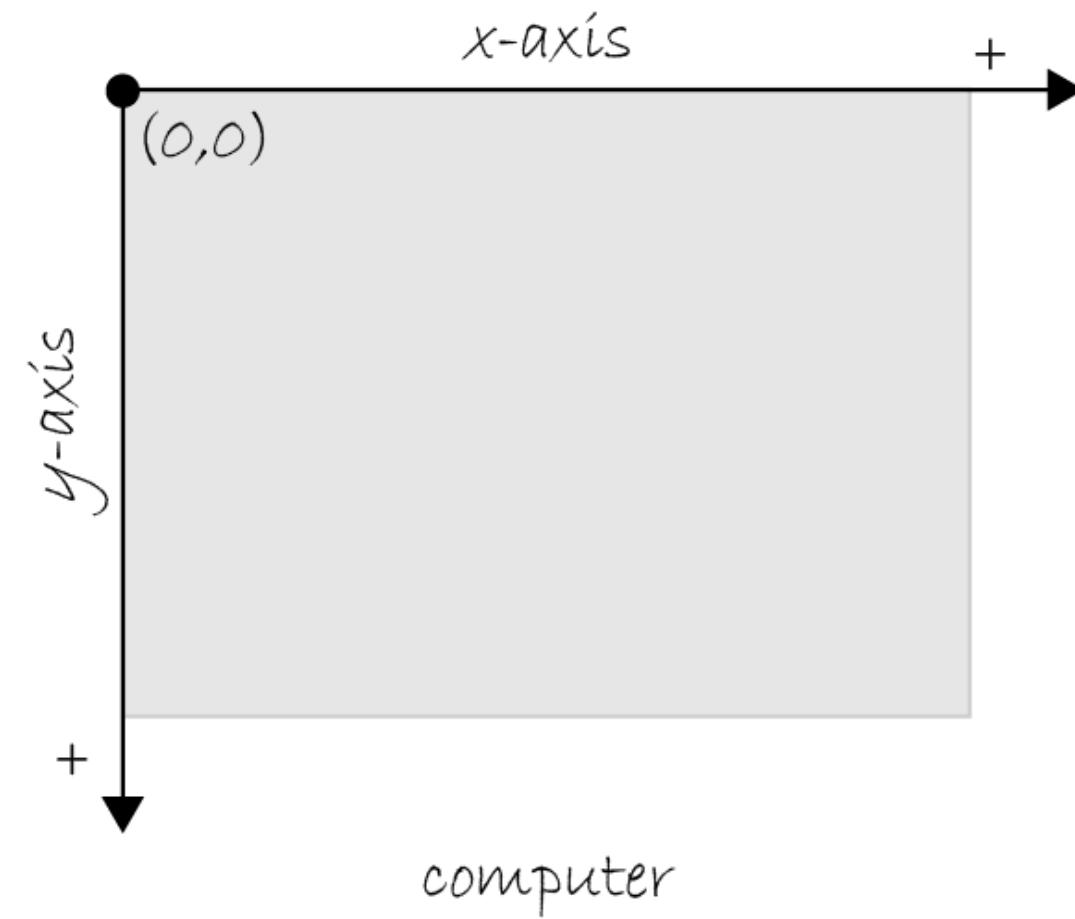
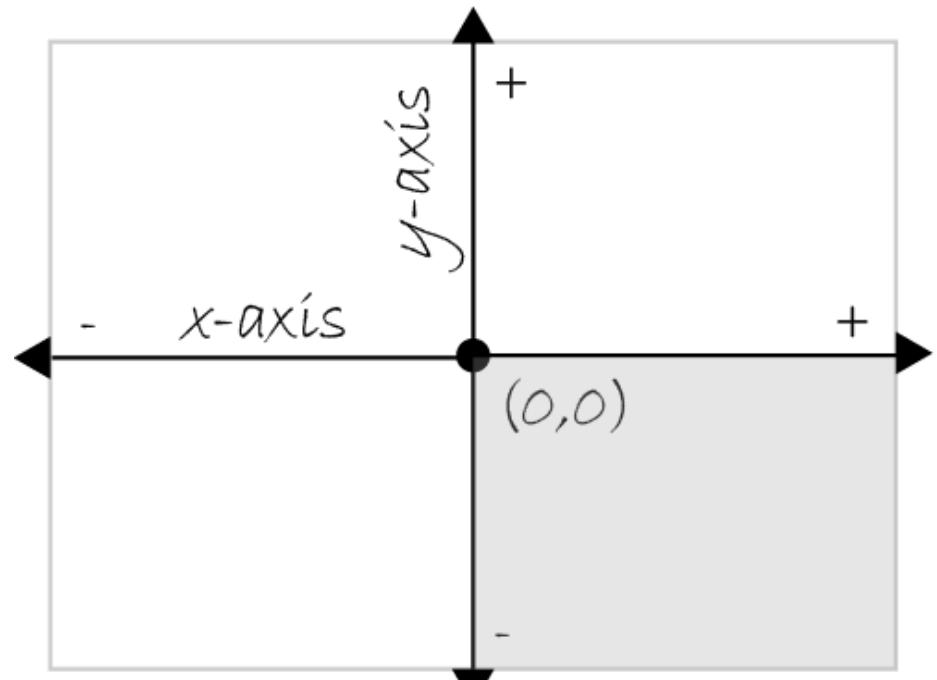
*A journey of a thousand miles begins with a single step.*

—Lao-tzu

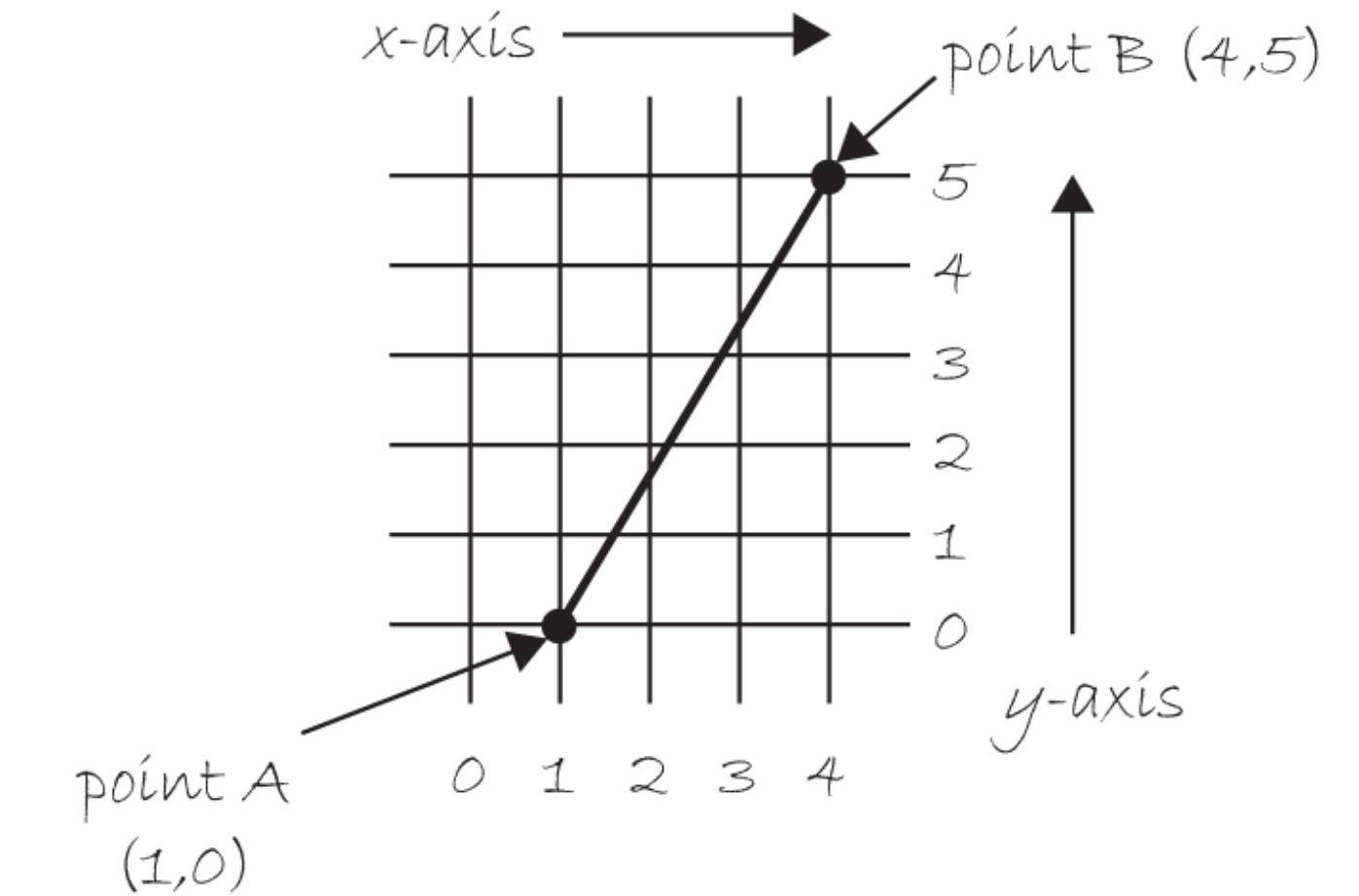
## PIXELS

Digital is made by pixels, when you have to create something on a display you have to specify where you want it...

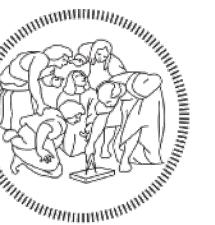
Computers thinks by pixels



graph paper



This figure shows a line between point A (1,0) and point B (4,5). If you wanted to direct a friend of yours to draw that same line, you would say “draw a line from the point one-zero to the point four-five, please.”



# How it works?

*'I try to apply colors like words that shape poems, like notes that shape music'*

—Joan Miró

## COLORS

Color is defined with a range of numbers.

The simplest case: *black and white or grayscale*.

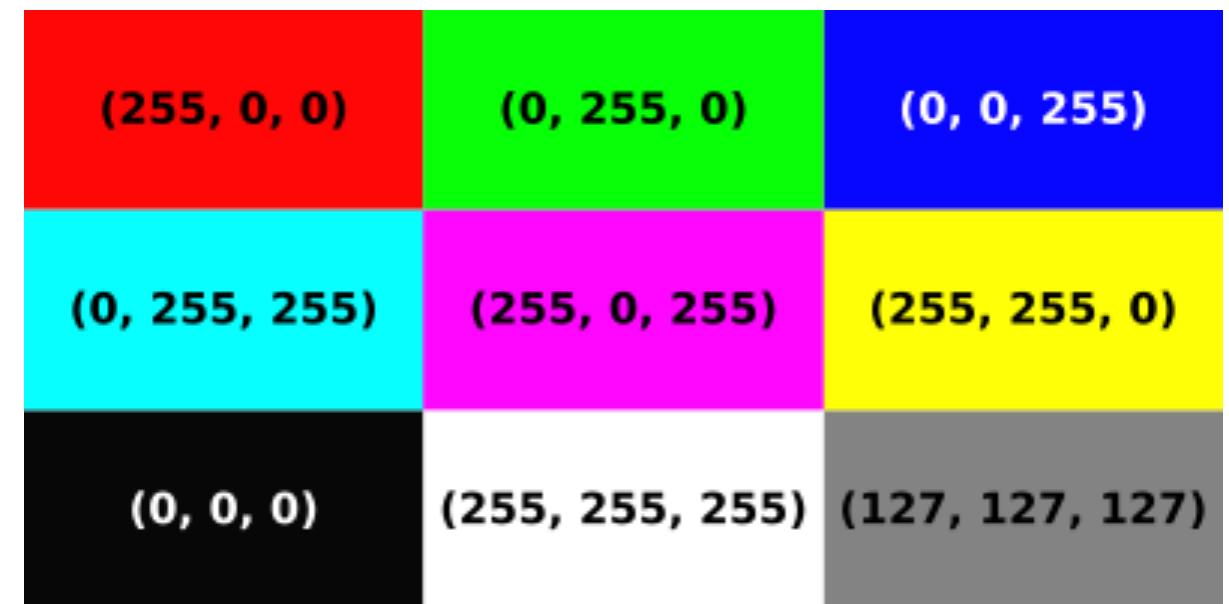
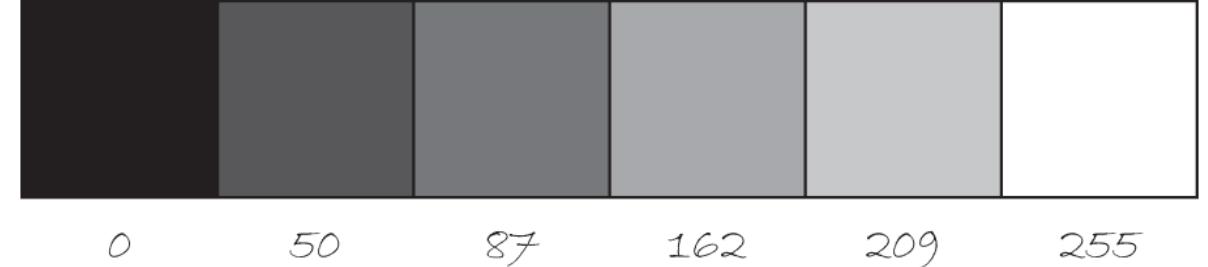
To specify a value for grayscale, use the following:

0 means black, 255 means white.

In between, every other number – 50, 87, 162, 209, and so on – is a shade of gray ranging from black to white.

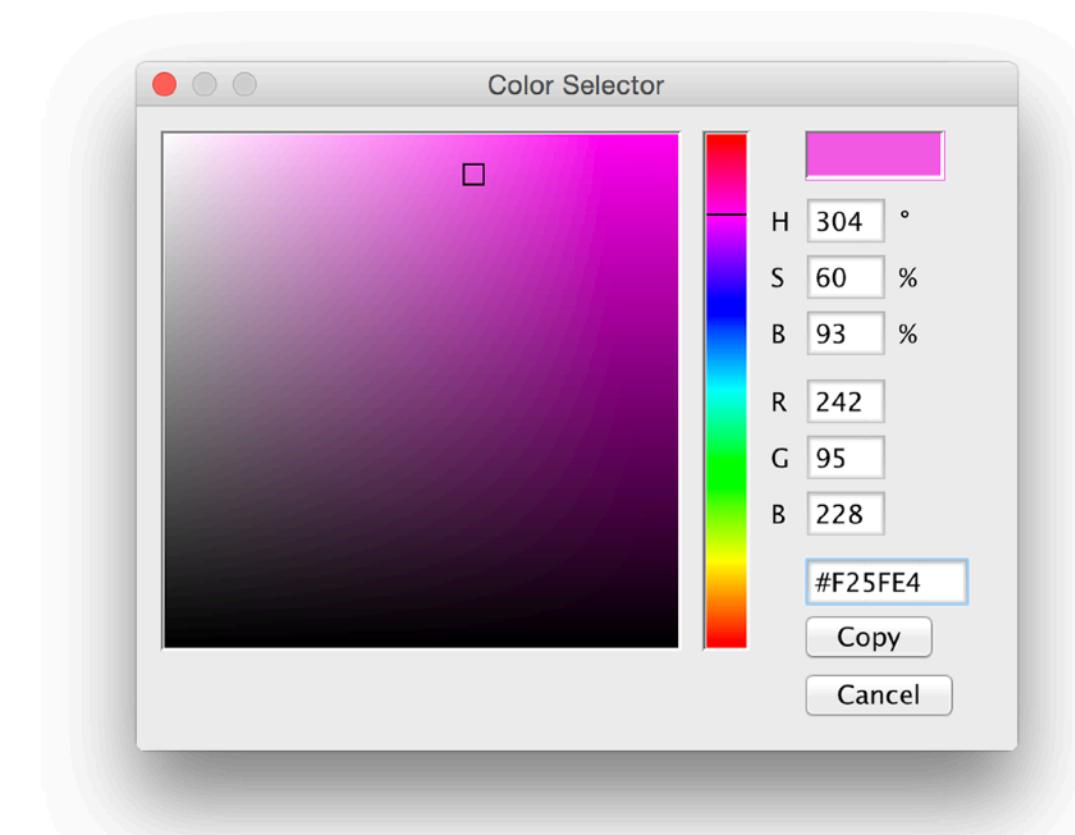
```
fill(0);  
fill(255,0,0);  
fill(0,255,0);  
fill(0,0,255);
```

Syntax    fill(rgb)  
          fill(rgb, alpha)  
          fill(gray)  
          fill(gray, alpha)  
          fill(v1, v2, v3)  
          fill(v1, v2, v3, alpha)



Processing also has a color selector to aid in choosing colors.

Access this via “Tools” (from the menu bar) → “Color Selector.”





# Always;

The console in the lower part shows you errors

the most frequent error is a missing semicolon one, so

remember :

Line (0,0,200,200); ← Ends with semi-colon

Function name

Arguments in parentheses

```
// I am a comment, you can use double slash (//) to create a
void setup(){
    size(800, 600); //size is measured in pixel, I choose 800x600
    background(0); //background is the color of the background, :
}

void draw(){
}
```

Syntax error, maybe a missing semicolon?  
expecting SEMI, found '}'  
Syntax error, maybe a missing semicolon?

everything you need is here:

<https://processing.org/reference/>

Cover Reference. Processing was designed to be a flexible software sketchbook.

Download Download

Exhibition Structure Shape Color

Reference 0 (parentheses) createShape() Setting
Libraries , (comma) loadShape() background()
Tools . (dot) PShape clear()
Environment /\* \*/ (multiline comment) colorMode()
Tutorials // (comment) 2D Primitives fill()
Examples ; (semicolon) arc() noFill()
Books = (assign) circle() noStroke()
Overview [] (array access) ellipse()
People Overview curly braces line()
People point()
Tutorials class quad()
Examples draw()
Exhibitions extends rect()
Reference exit()
Structure false square()
Libraries Forum draw()
Tools Issues triangle()
Environment Environment extends
Tutorials People false
Examples Forum draw()
Exhibitions GitHub exit()
Reference Libraries Issues implements
Libraries Tools Wiki import
Tools Environment FAQ loop()
Environment Tutorials Medium new
Exhibitions Reference new
People Forum noLoop()
People GitHub Issues null
People Medium pop()
People Medium popStyle()
People Medium private
People Medium public
People Medium 3D Primitives
People Medium Loading & Displaying



# Let's create a sketch

The keyword **void** indicates a function with no value.

If you don't know what a function is...

**void setup(){**

is the space in which I setup my sketch

Everything you write in the SETUP will run just once

}

**void draw(){**

is the space in which I draw in my sketch

Everything you write in the DRAW, will run in an infinite loop

}

The screenshot shows the Processing 3.5.3 IDE interface. The title bar reads "BASICS | Processing 3.5.3". The code editor window displays the following Java code:

```
1 void setup(){
2 }
3
4 void draw(){
5 }
```

The code editor has line numbers from 1 to 22 on the left. Below the code editor is a large gray workspace area. At the bottom of the interface, there is a toolbar with icons for file operations, and tabs for "Console" and "Errors". A status bar at the very bottom right shows "Updates 5".



# Basics

---

```
// open the folder of examples
```

```
sketch_1BASICS
```

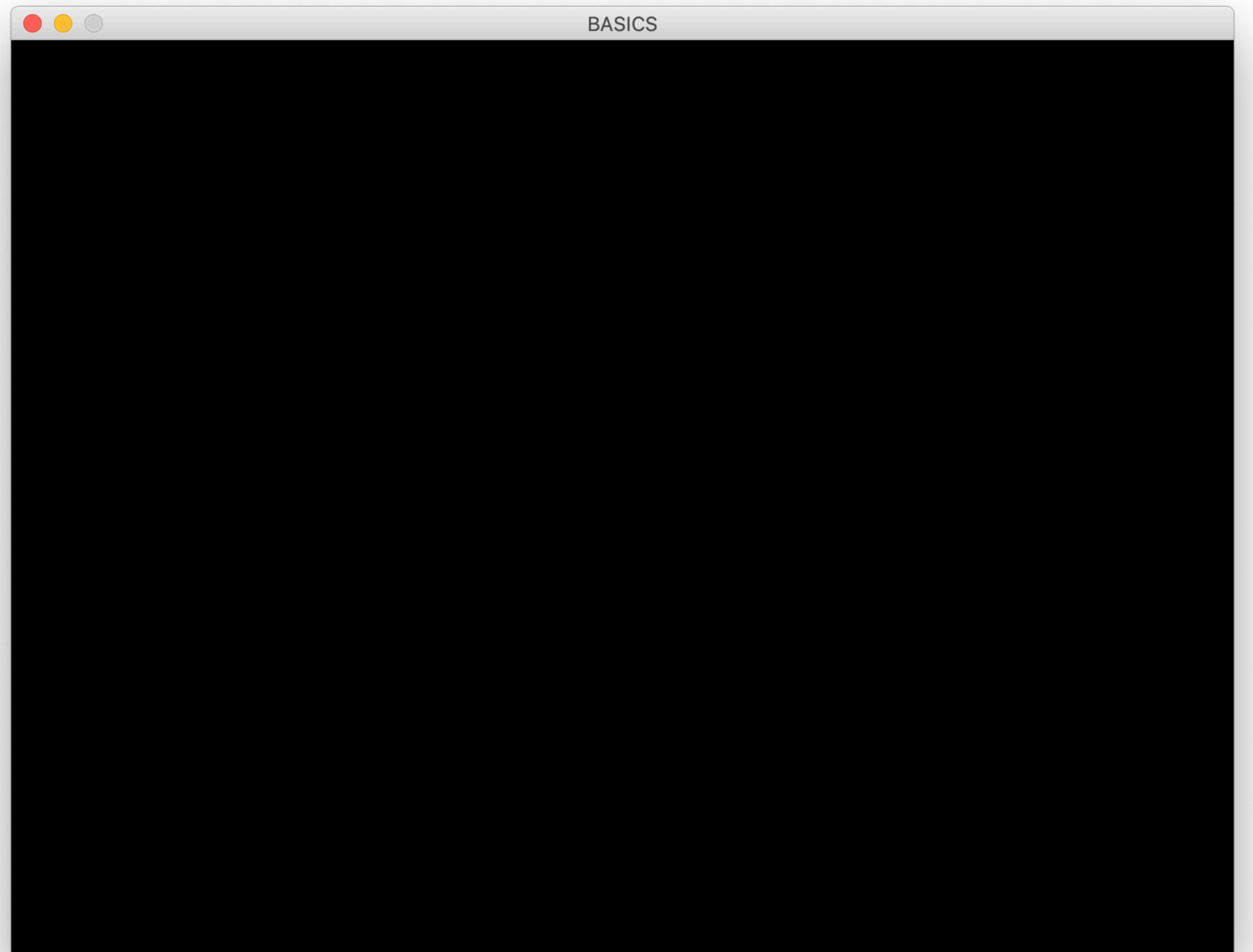
here you find

**A black blank page**

our hello world

**tips:**

try to change the size of the sketch and the background color at line 10 and 12





# Shapes

---

```
// open the folder of examples  
  
sketch_2BASIC_SHAPES
```

here you find

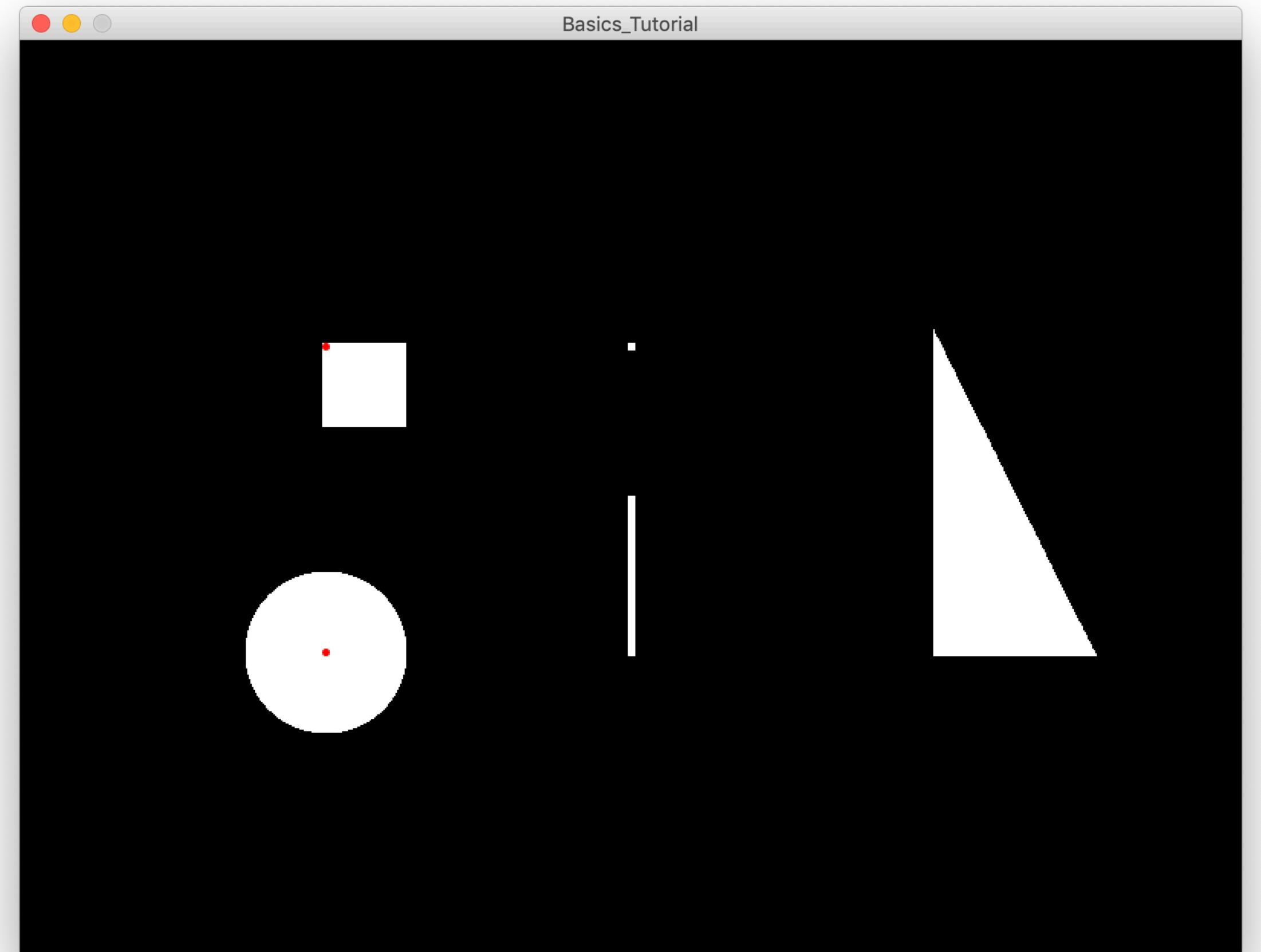
## Primitive shapes

to understand how to draw by code

### tips:

change from a grayscale color to rgb at lines 8 and 14

try to modify the triangle shape at line 24





# Stickman

```
// open the folder of examples  
  
sketch_STICK_MAN
```

here you find

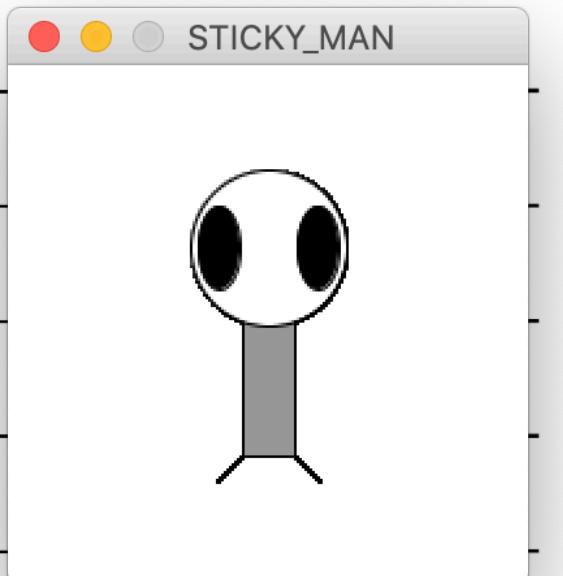
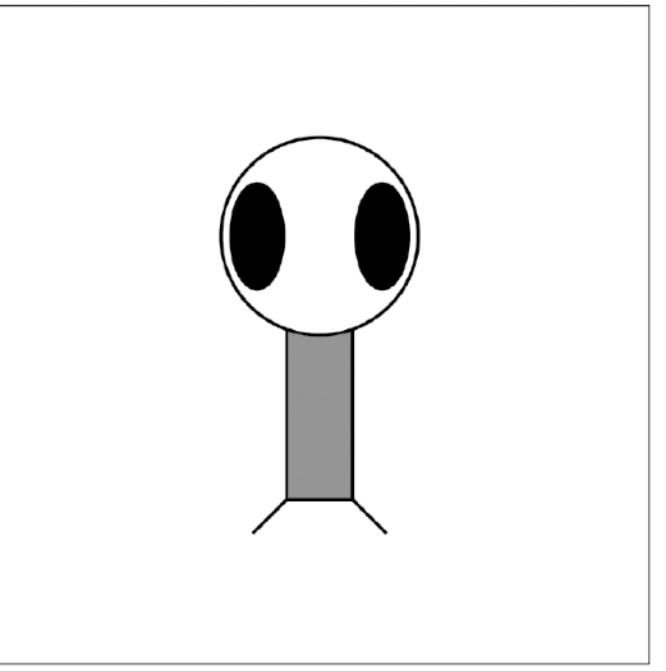
## A simple stickman

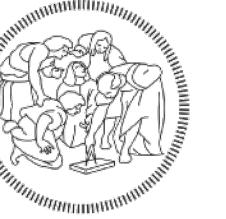
hack and modify code to obtain a personal output

### tips:

try to add arms and details, be creative!

```
background(255);  
ellipseMode(CENTER);  
rectMode(CENTER);  
stroke(0);  
fill(150);  
rect(100, 100, 20, 100);  
fill(255);  
ellipse(100, 70, 60, 60);  
fill(0);  
ellipse(81, 70, 16, 32);  
ellipse(119, 70, 16, 32);  
stroke(0);  
line(90, 150, 80, 160);  
line(110, 150, 120, 160);
```

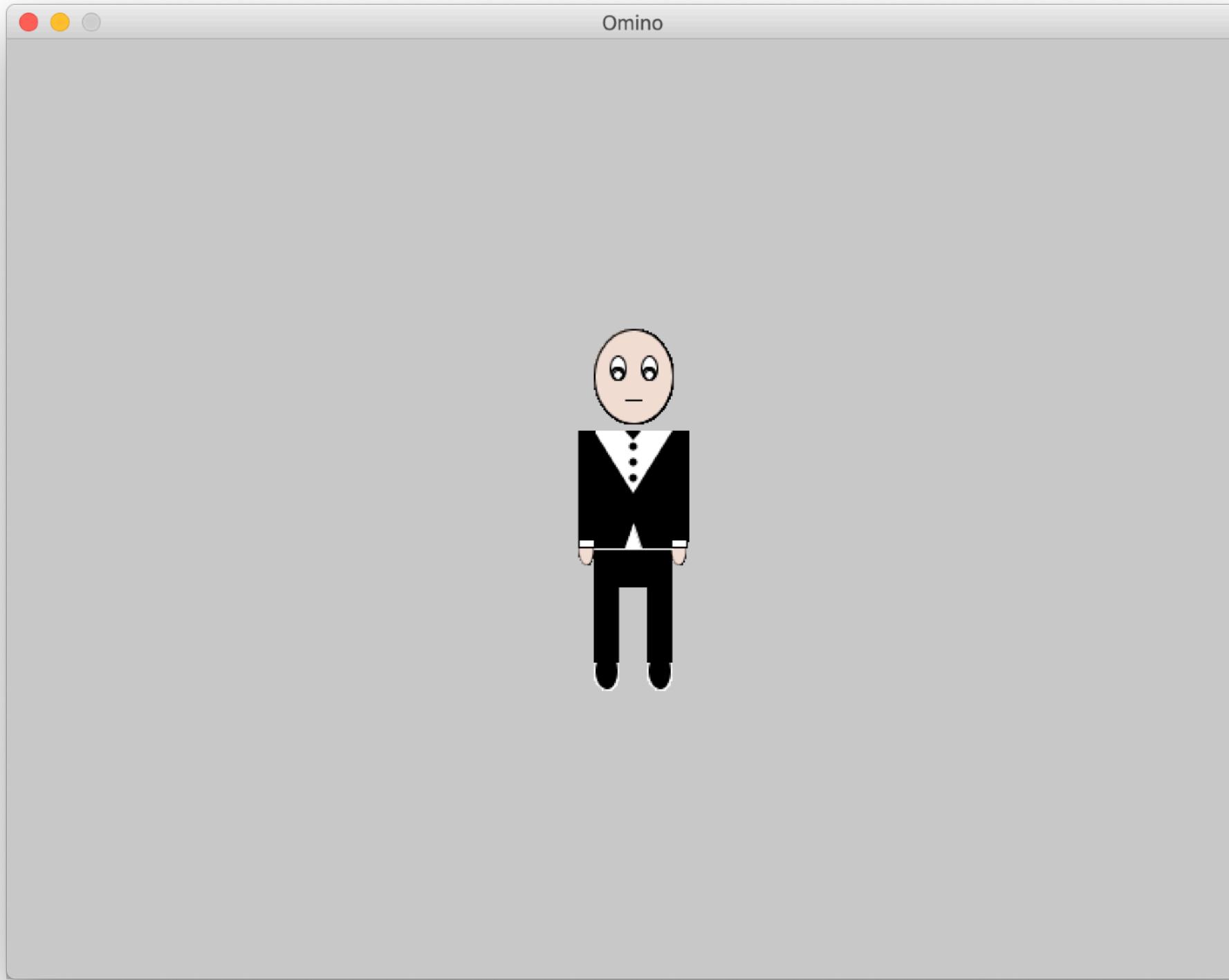




# Be creative!

Let's create your own

that was my exercise some years ago



```
//C O M P U T E R - V I S I O N
//Accademia delle belle arti di Catania
//Docente: Giovanni Maria Farinella
//Studente: Giulio Interlandi
//Esercizio per esame di CV
// 04/07/2014

void setup() {
  size(800,600);
}
void draw() {
  noStroke();
  //body
  fill(0);
  rectMode(CENTER);
  rect(400,300, 50,100);
  fill(255);
  triangle(375,250,400,290,425,250);
  fill(0);
  ellipseMode(CENTER);
  ellipse(400,270,5,5);
  ellipse(400,280,5,5);
  ellipse(400,260,5,5);
  triangle(395,250,400,256,405,250);
  //arms
  strokeWeight(0.1);
  stroke(0);
  fill(240,220,210);
  ellipse(370,325,9,21);
```

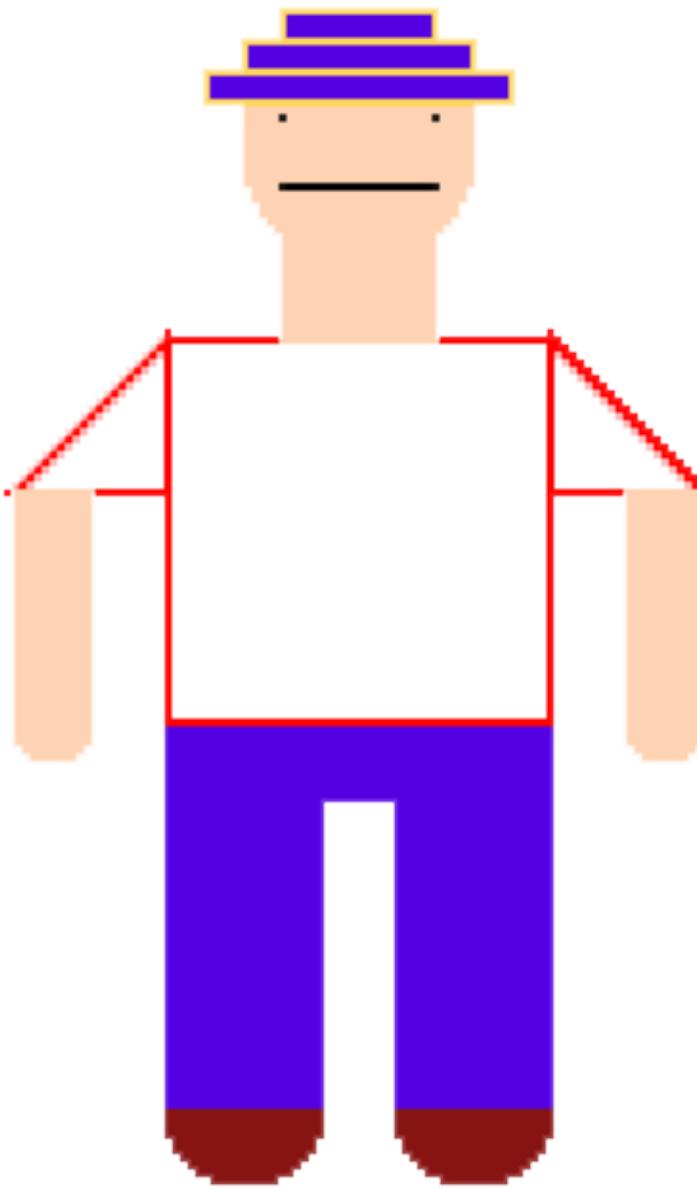
The screenshot shows the Processing IDE interface with the title bar "Omino | Processing 3.5.3". The code editor contains the Java code for the "Omino" sketch. The code defines a setup function that sets the size to 800x600 pixels. The draw function starts with noStroke(). It then creates a body using a rectangle centered at (400, 300) with a width of 50 and a height of 100. It fills the rectangle with white. It also draws a triangle at (375, 250), (400, 290), and (425, 250). After that, it creates three small ellipses at (400, 270), (400, 280), and (400, 260), each with a diameter of 5 pixels. Finally, it draws another triangle at (395, 250), (400, 256), and (405, 250). The code uses various Processing functions like rect, triangle, and ellipse, along with Java-specific functions like strokeWeight and stroke. The code is well-commented with descriptive strings starting with double slashes (//).



# Be creative!

Let's create your own

this is a funny sketch done by a student last year.



```
// feet
stroke(133, 21, 24);
fill(133, 21, 24);
ellipse(385, 475, 20, 20); // left foot (center X, center Y, width, height)
ellipse(415, 475, 20, 20); // right foot (center X, center Y, width, height)
// trousers
stroke(84, 21, 222);
fill(84, 21, 222);
rect(375, 425, 50, 10); // pacco (starting X, starting Y, width, height)
rect(375, 435, 20, 40); // left leg (starting X, starting Y, width, height)
rect(405, 435, 20, 40); // right leg (starting X, starting Y, width, height)

// face
stroke(252, 211, 181);
fill(252, 211, 181);
rect(390, 355, 20, 20); // neck (starting X, starting Y, width, height)
ellipse(400, 350, 30, 30); // face (center X, center Y, width, height)
stroke(0);
fill(255);
point(390, 346); // left eye
point(410, 346); // right eye
line(390, 355, 410, 355); // mouth (starting X, starting Y, ending X, ending Y)
// cap
stroke(252, 211, 102);
fill(84, 21, 222);
rect(380, 340, 40, 4); // cap down (starting X, starting Y, width, height)
rect(385, 336, 30, 4); // cap middle (starting X, starting Y, width, height)
rect(390, 332, 20, 4); // cap middle (starting X, starting Y, width, height)
```

Credits: Luca Ghezzi



# Let's create a sketch

```
// open the folder of examples  
  
sketch_4DRAWING TOOL
```

here you find

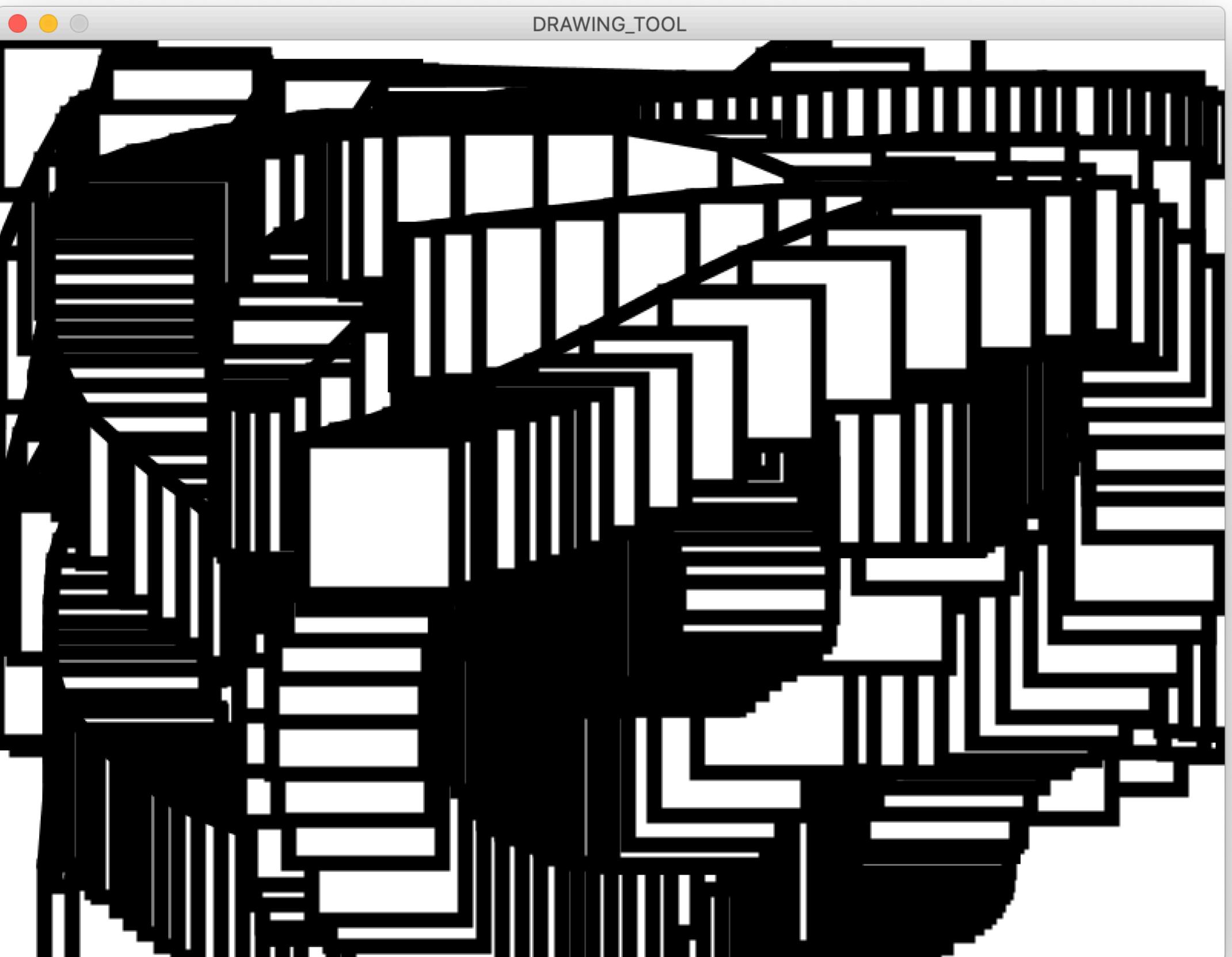
## Drawing principles

to understand:

- how to connect and draw by mouse input
- if (key pressed) case

## tips:

delete or comment the background at line 16 and the rectangle at line 17, uncomment line 19 and the “if” at line 22 to 24

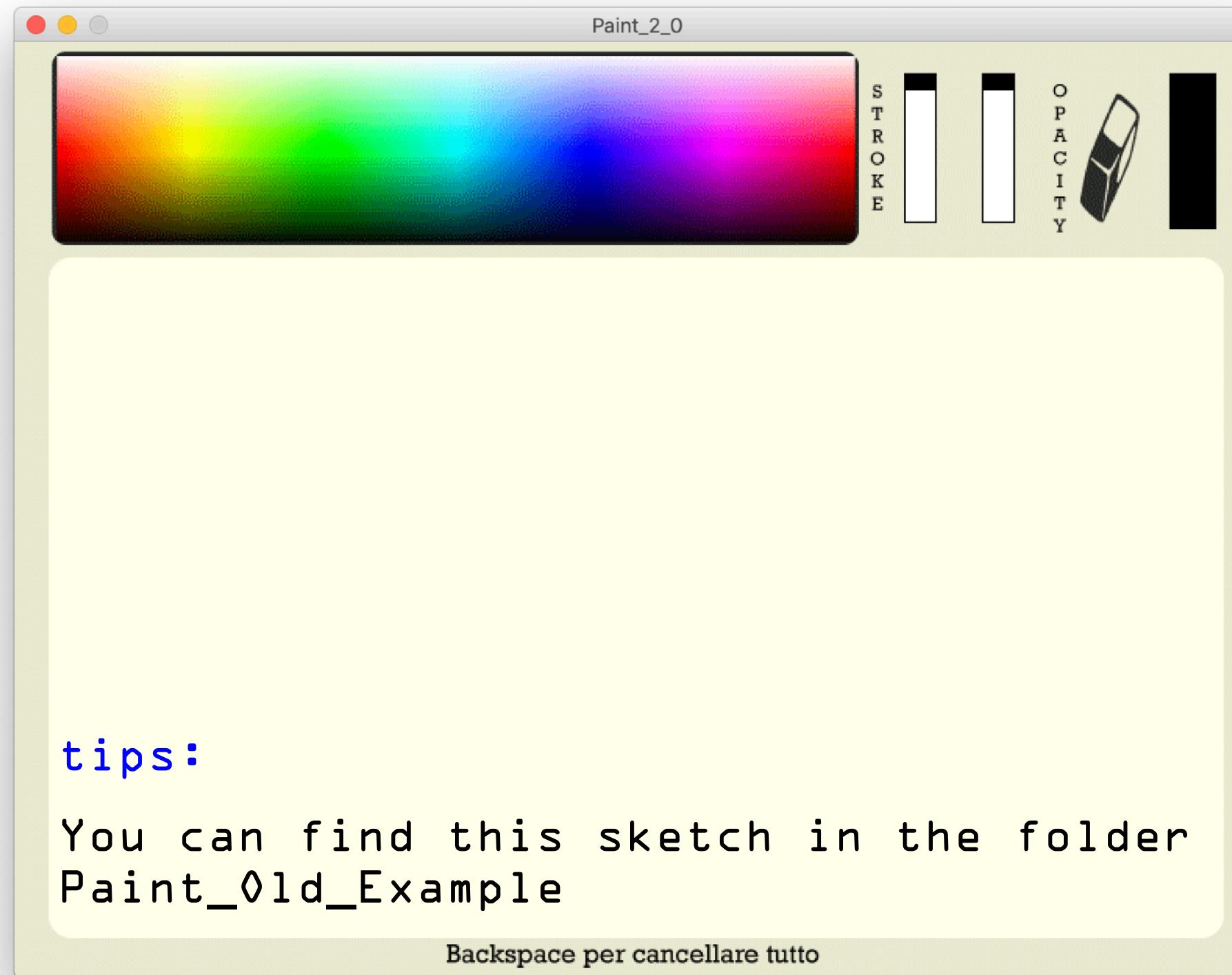




# Be creative!

Let's create your own

that was my exercise some years ago



## tips:

You can find this sketch in the folder  
Paint\_Old\_Example

The screenshot shows the Processing IDE interface with a sketch titled "Paint". The code in the editor is as follows:

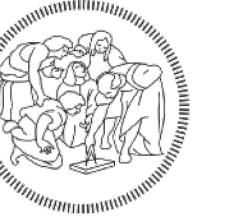
```
//COMPUTER-VISION
//Accademia delle belle arti di Catania
//Docente: Giovanni Maria Farinella
//Studente: Giulio Interlandi
//Esercizio per esame di CV
// 04/07/2014

PImage layout;

float move = 20;
float move2 = 20;
float spessore;
float opacita2;
color c;
color gomma;

void setup() {
  size (800, 600);
  background(255);
```

At the bottom, a message says "Auto Format finished." There are tabs for "Console" and "Errors".



# Let's create a sketch

---

```
// open the folder of examples
```

```
sketch_RANDOM_RELATIVE
```

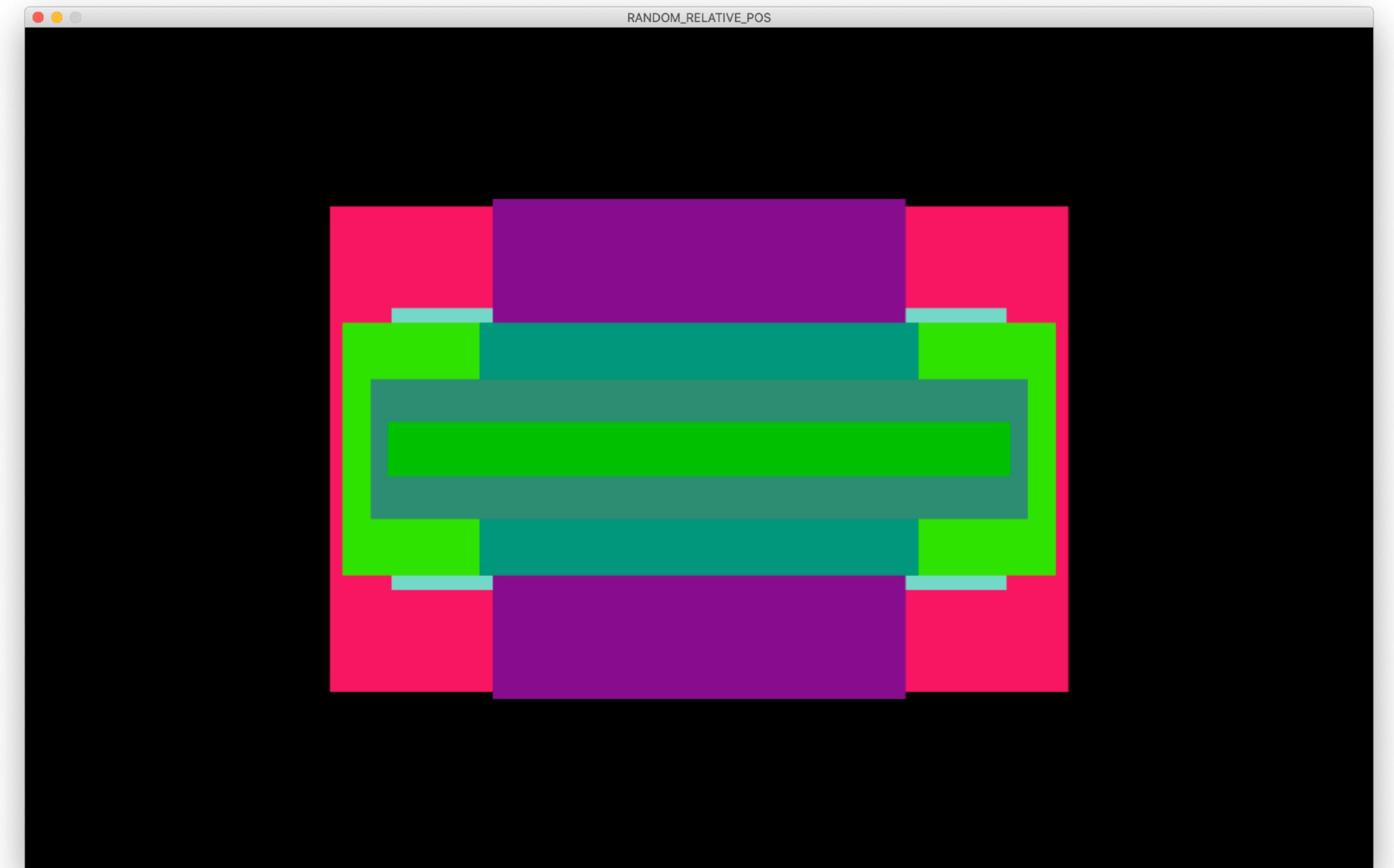
here you find

**Random principles and coordinates**

to understand randomness and positioning shapes  
without writing pixel

**tips:**

try to comment the filter at line 15 and delay at line 16





# Frieder Nake's remixes

---

```
// open the folder of examples
```

```
NAKE_REMIX
```

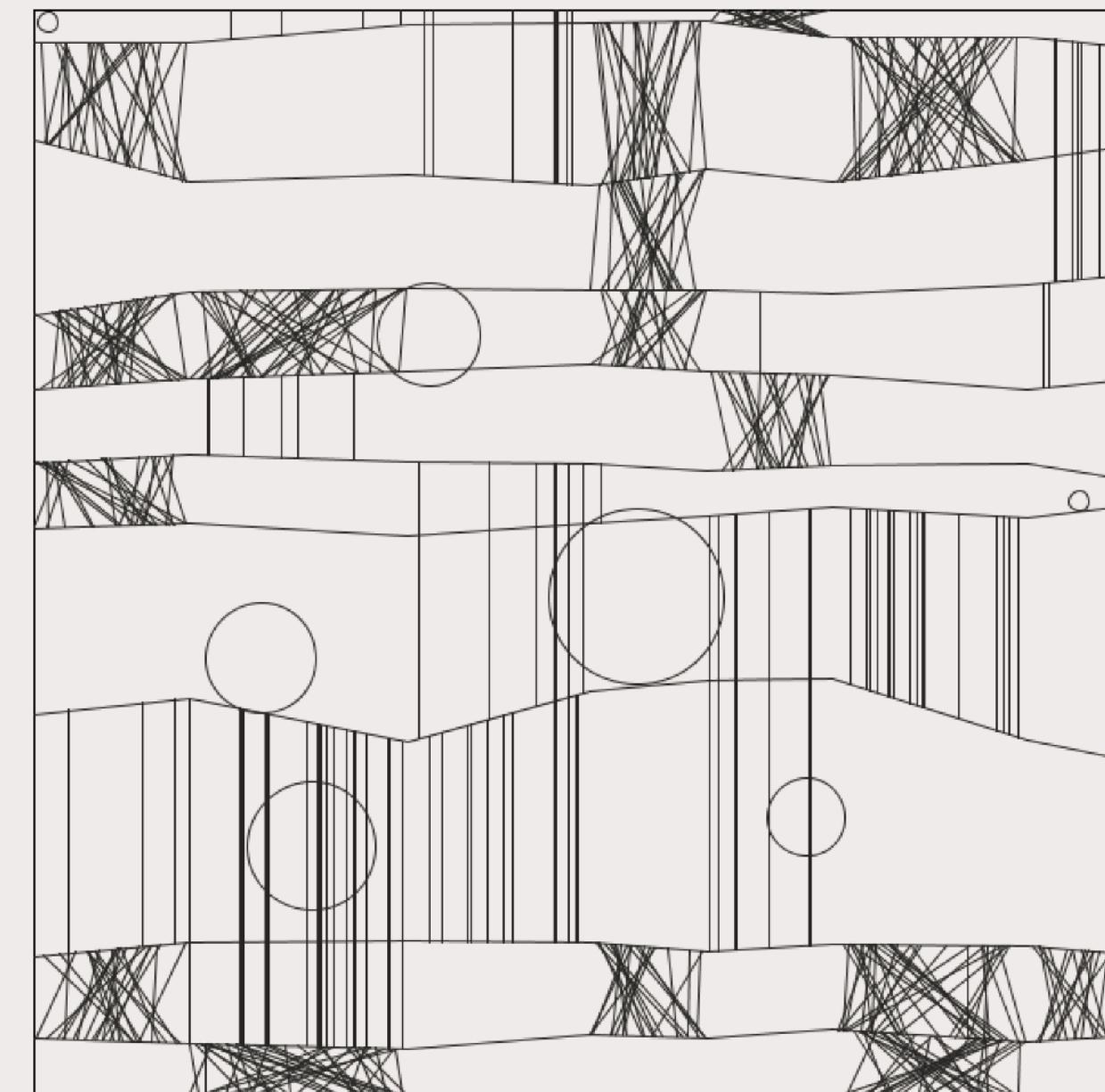
here you find

**Remixes of Nake's works and open processing platform**

to understand how to manage more complex sketches  
and discover the open processing platform

**tips:**

try to understand how these sketches work, and make  
your own remix





# Sound Reactive

In order to work with sound information in Processing, we first need to **install an extension library** since Processing doesn't natively support this feature.

It can be simply installed by opening:

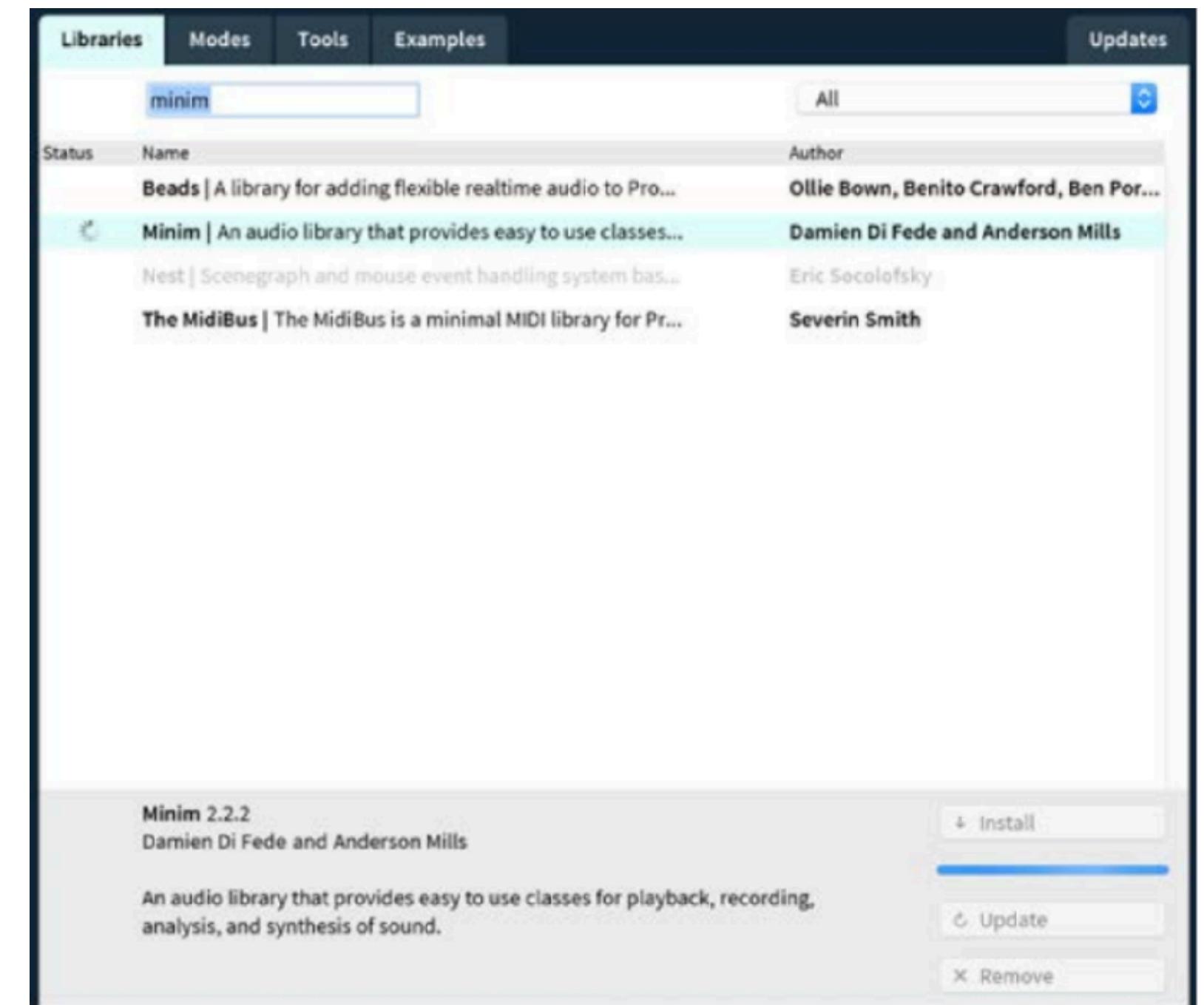
**Sketch**

→

**Import Library**

→

**Add Library**, typing ‘minim’ into the search field and then clicking on ‘Install’.



You can find more in the document:

[Processing-Generative\\_Design\\_Tutorial\\_soundmapping](#)



# Sound Reactive

Use this code before void setup:

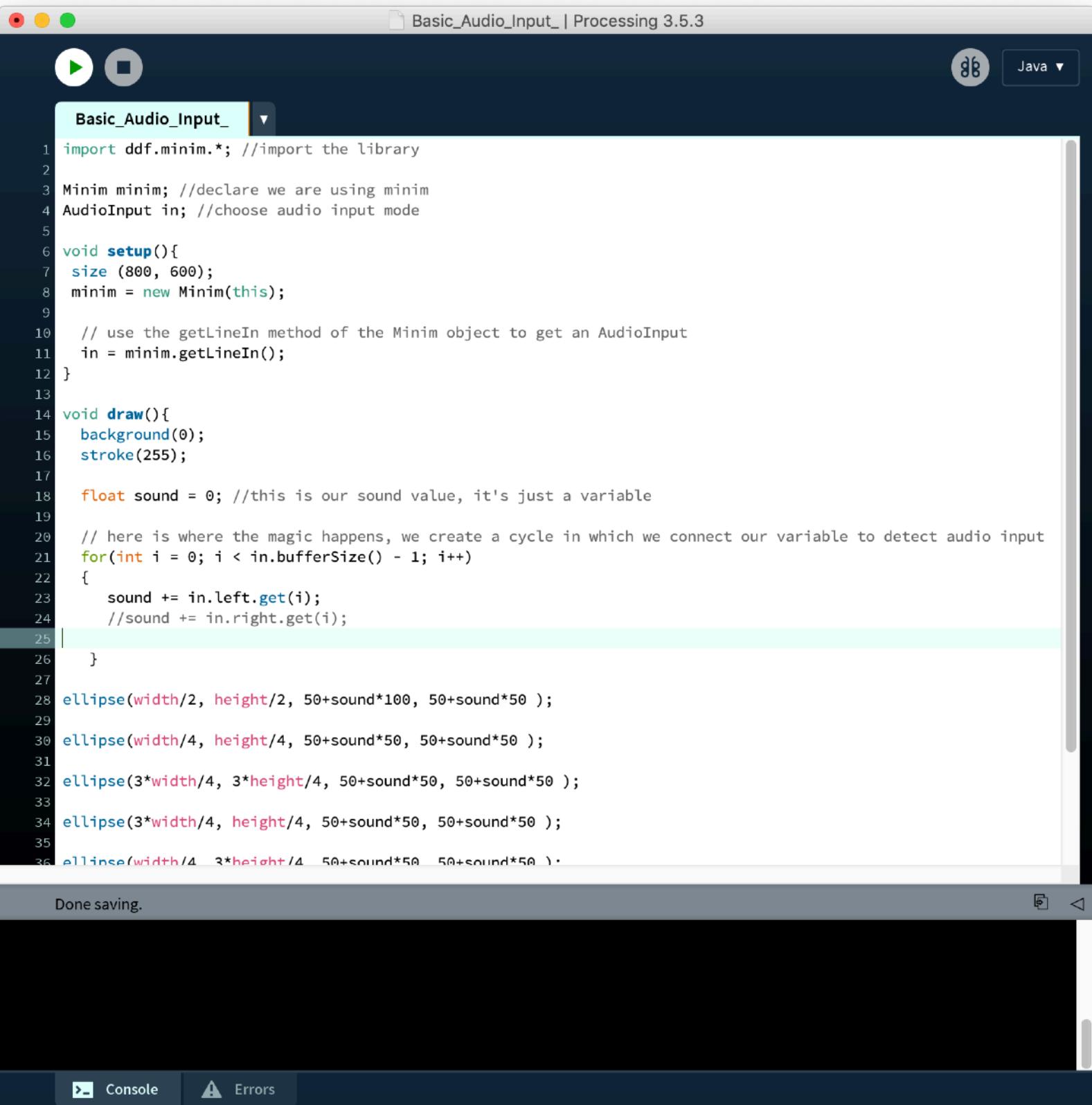
```
import ddf.minim.*; //import the library  
  
Minim minim; //declare we are using minim  
  
AudioInput in; //choose audio input mode
```

Use this code inside void setup:

```
minim = new Minim(this);  
  
in = minim.getLineIn();
```

Use this code inside void draw:

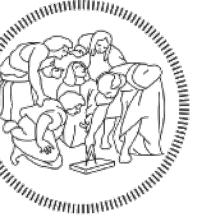
```
float sound = 0;  
  
for(int i = 0; i < in.bufferSize() - 1; i++) {  
  
    sound += in.left.get(i);  
  
}
```



```
Basic_Audio_Input_.java  
import ddf.minim.*; //import the library  
  
Minim minim; //declare we are using minim  
AudioInput in; //choose audio input mode  
  
void setup(){  
    size (800, 600);  
    minim = new Minim(this);  
  
    // use the getLineIn method of the Minim object to get an AudioInput  
    in = minim.getLineIn();  
}  
  
void draw(){  
    background(0);  
    stroke(255);  
  
    float sound = 0; //this is our sound value, it's just a variable  
  
    // here is where the magic happens, we create a cycle in which we connect our variable to detect audio input  
    for(int i = 0; i < in.bufferSize() - 1; i++)  
    {  
        sound += in.left.get(i);  
        //sound += in.right.get(i);  
    }  
  
    ellipse(width/2, height/2, 50+sound*100, 50+sound*50 );  
    ellipse(width/4, height/4, 50+sound*50, 50+sound*50 );  
    ellipse(3*width/4, 3*height/4, 50+sound*50, 50+sound*50 );  
    ellipse(3*width/4, height/4, 50+sound*50, 50+sound*50 );  
    ellipse(width/4, 3*height/4, 50+sound*50, 50+sound*50 );  
}
```

Connect audio input

now you can connect  
the variable sound you  
just created to your  
shapes



# Equalizer

```
// open the folder of examples  
  
sketch_BASIC_AUDIO_INPUT
```

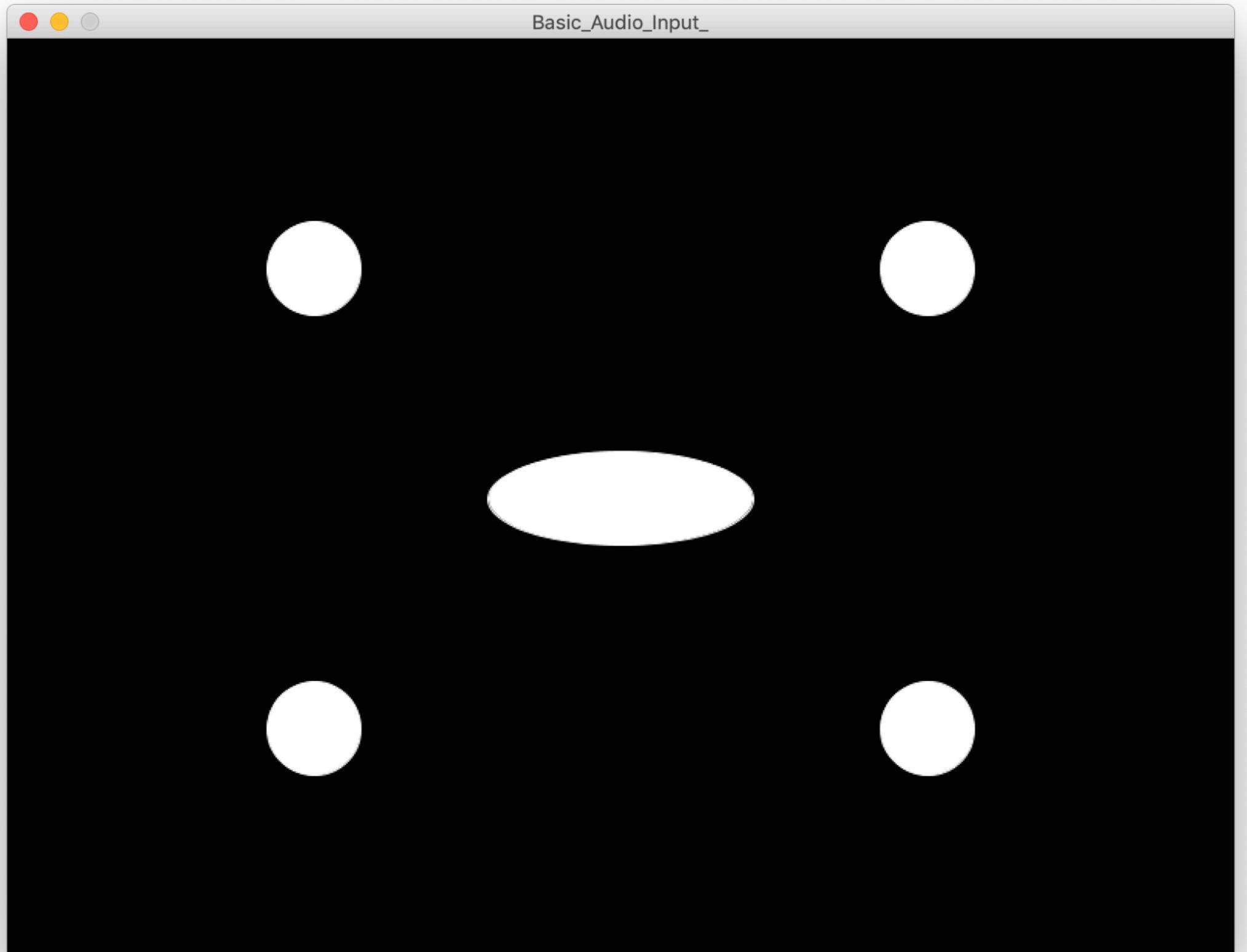
here you can find

## **Basic principles of sound reactivity**

to understand how to import a library and how to implement sound reaction in shapes

### **tips:**

try to have different reactions for different shapes, using math to adjust sensibility

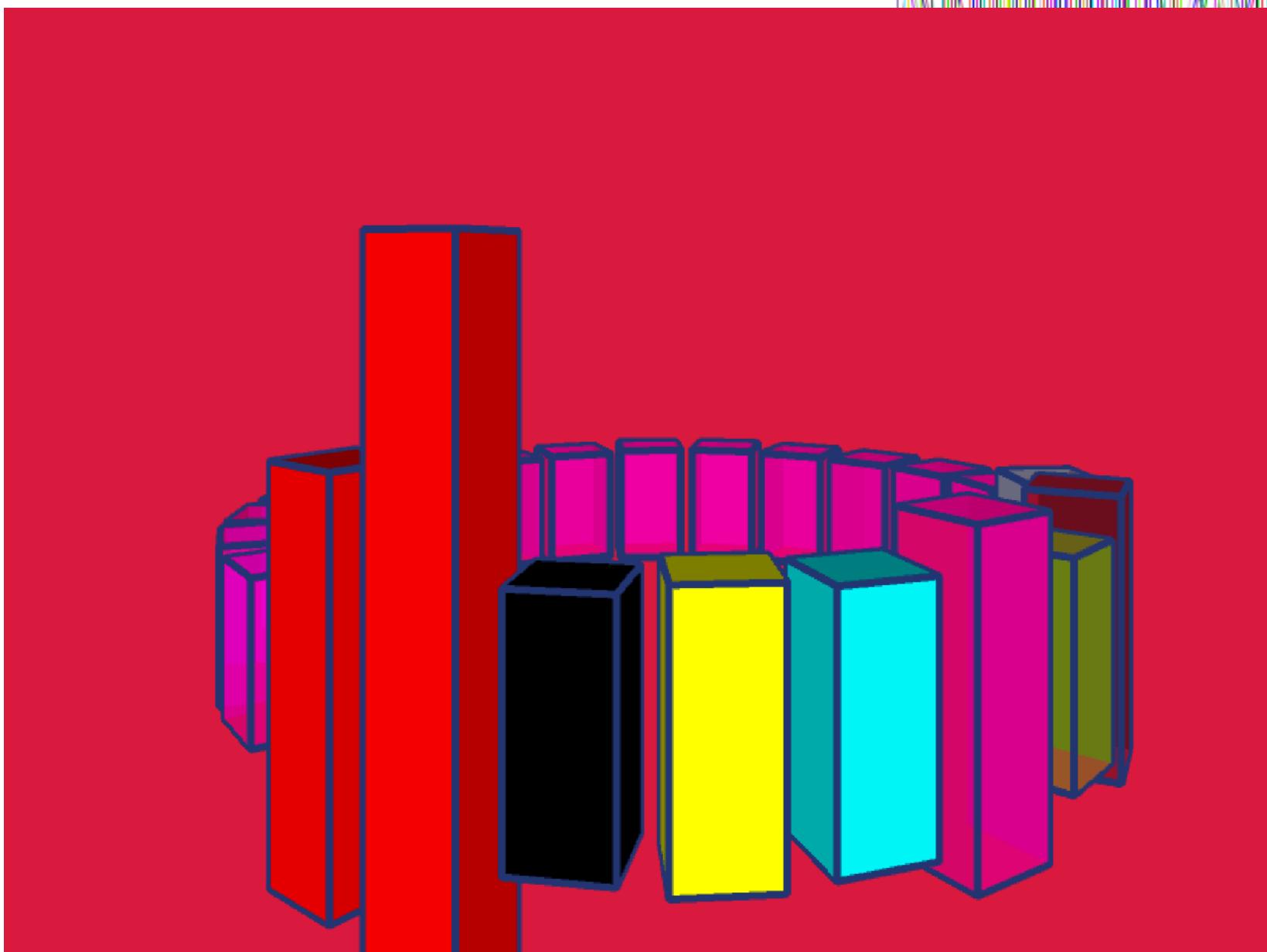




# Be creative!

Let's create your own!

here some sketches done by  
student from past years



Here a link to a p5 game made by a student  
<https://editor.p5js.org/Cuchavira/sketches/diEpF0D7Z>

// open the folder past years' examples

Credits: Luca Ghezzi, Andrea Zito, Hongni Ye



# Kinect

Kinect® is a **motion sensing input device** produced by Microsoft. It was initially developed as a **gaming accessory**, but artists, third-party developers and researchers found several after-market uses because of its **low-cost and advanced features**. There are 2 versions of Kinect commercially available, and a [Kinect Azure Development Kit](#) recently became available on the market.

[OpenKinect](#) is one of the libraries for interfacing the Kinect to Processing. It can be simply installed by opening:

**Sketch**  
→  
**Import Library**  
→  
**Add Library**, typing ‘kinect’ into the search field and then clicking on ‘Install’.



You can find out more on:

<https://shiffman.net/p5/kinect/>



# Kinect

## How it works:

Kinect sensors can detect a three dimensional area thanks to its **infrared** technology. It works by projecting infrared dots on the scene, and calculating their distortion on the projected surface. The Kinect has 1 RGB camera, 1 infrared camera, one infrared emitter, and a microphone array.

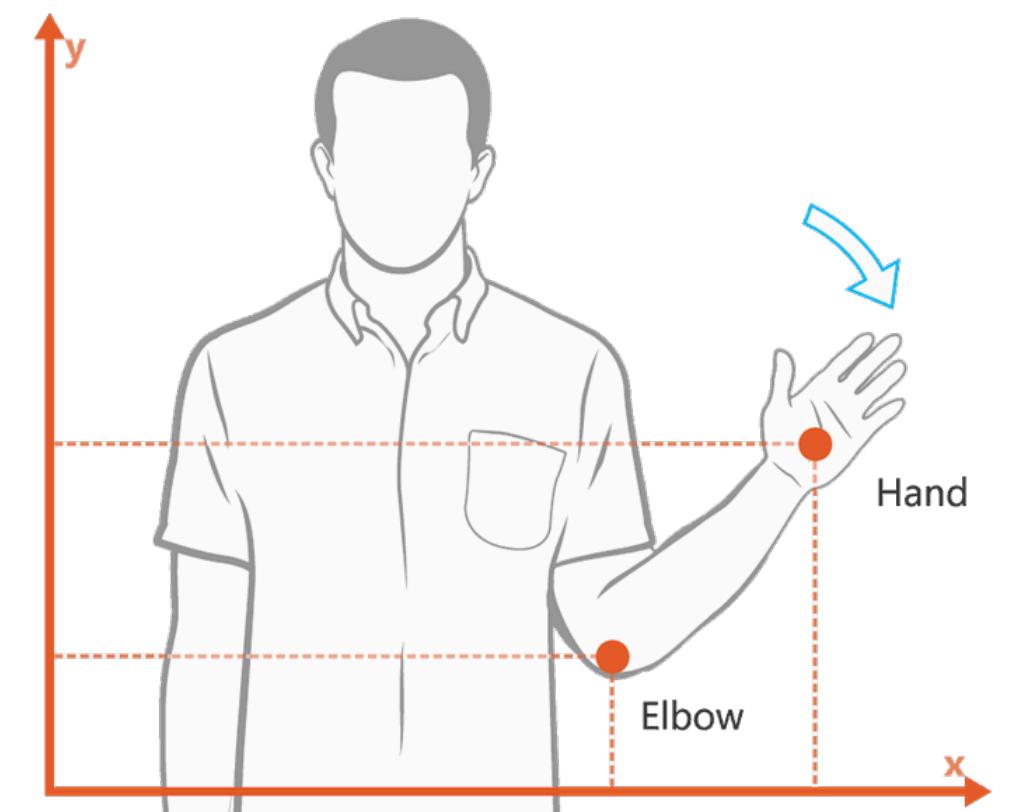
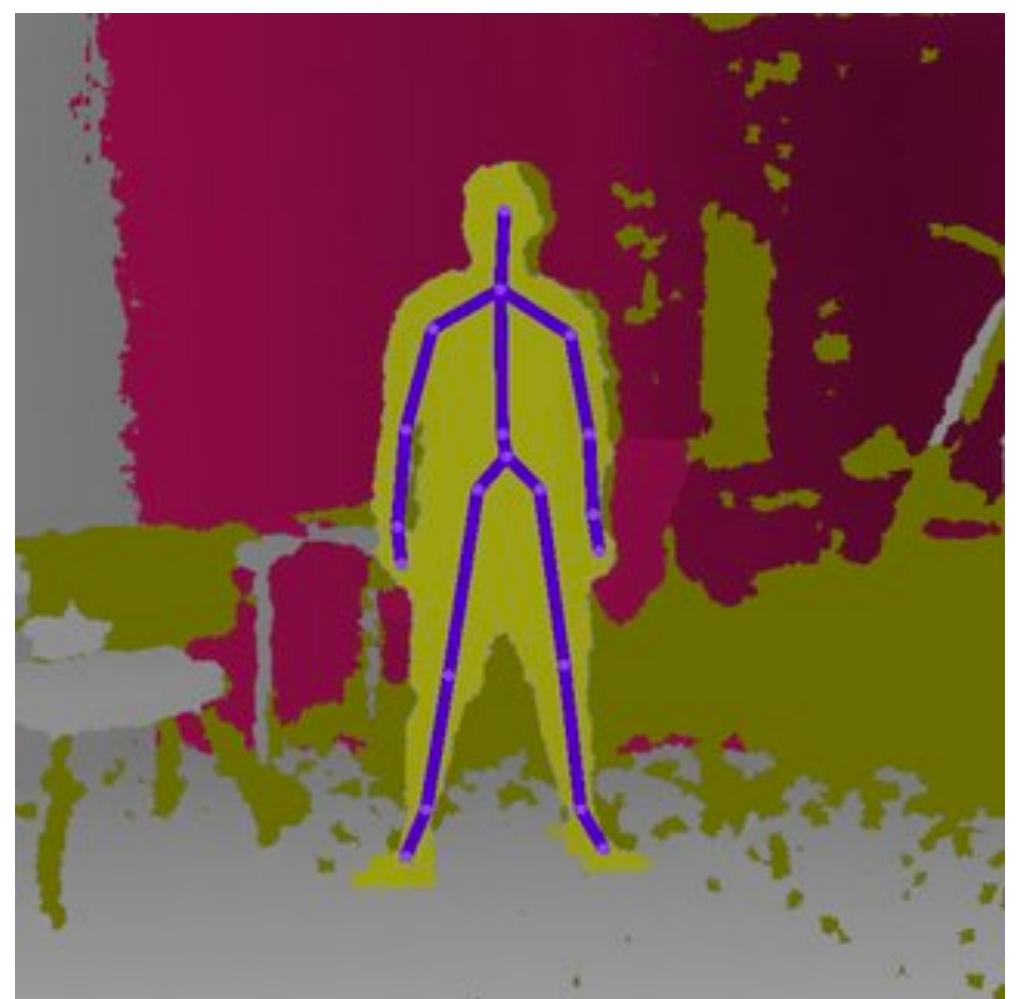


## Features:

**Depth map**, it's possible to detect the depth of a scene in order to visualise silhouettes at a given distance.

**Skeleton** feature, is available for up to six users depending on the Kinect version.

**Hands and Gestures** recognition.





# Kinect to Syphon

---

```
// extra example for kinect v1
```

**KINECT\_SYPHON\_2020**

here you can find

## **Kinect depth feed to Syphon**

to understand how to use the Kinect Depth Map feature and how to output it through a video sharing framework, in our case Syphon, and use it in a media server or mapping software (MadMapper, Resolume Arena).

### **tips :**

try and change the minimum and maximum tilt and threshold values; these are very useful in the calibration process.





POLITECNICO  
MILANO 1863

# Thank You

[giulio.interlandi@mail.polimi.it](mailto:giulio.interlandi@mail.polimi.it)