Processing
P01

1- intro
2- pixel

Cédric MASSART

Intro

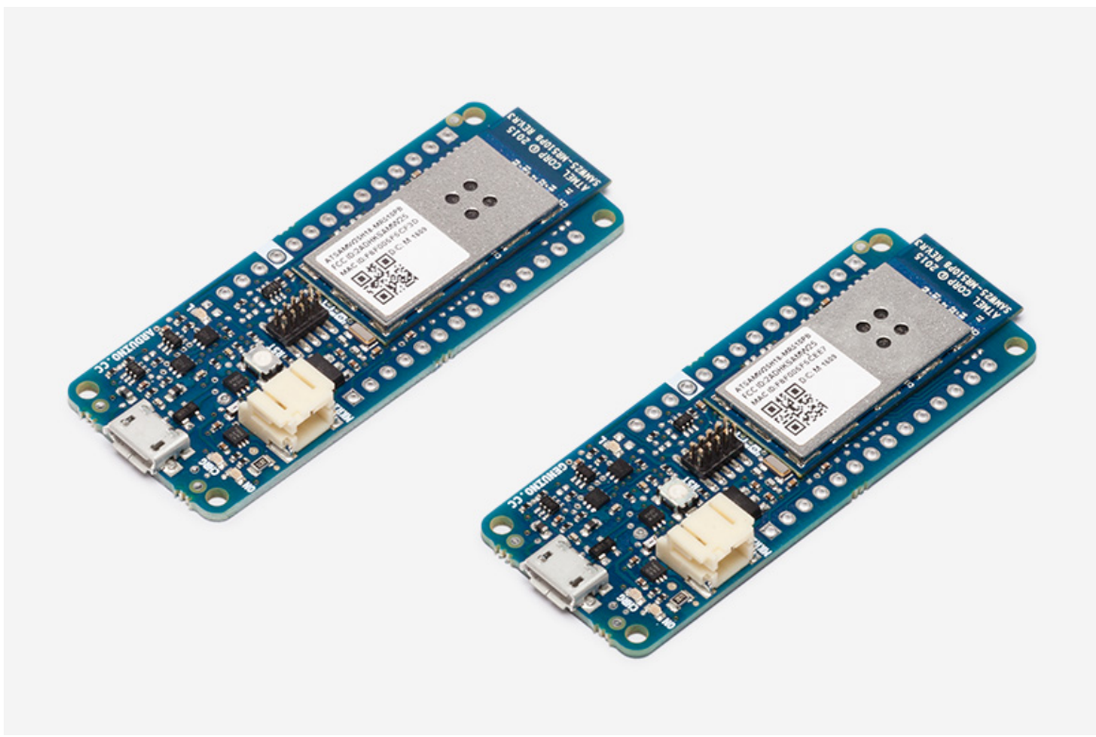
Processing

- **Processing** est un langage de programmation dédié à la production artistique, notamment à la productions d'images et de design graphique et interactif,
- **Processing** peut communiquer avec des dispositifs électroniques de type Arduino,
- C'est un langage à la fois simple, puissant appartenant à la famille de Java,
- Le logiciel **Processing** (qui sert à rédiger et à exécuter des programmes) est gratuit et disponible sur trois plate-formes : Mac OS, Window et Linux.

https://processing.org/

Processing

- **Processing** peut communiquer avec des dispositifs électroniques de type **Arduino**



https://arduino.cc/

Processing

ENTRY LEVEL	ARDUINO UNO · ARDUINO 101 · ARDUINO PRO · ARDUINO PRO MINI · ARDUINO MICRO · ARDUINO STARTER KIT · ARDUINO BASIC KIT · MKR1000 BUNDLE
ENHANCED FEATURES	ARDUINO MEGA · ARDUINO ZERO · ARDUINO PROTO SHIELD
INTERNET OF THINGS	ARDUINO MKR1000 · ARDUINO WIFI SHIELD 101 · ARDUINO YÚN SHIELD
WEARABLE	ARDUINO GEMMA · LILYPAD ARDUINO USB · LILYPAD ARDUINO MAIN BOARD · LILYPAD ARDUINO SIMPLE · LILYPAD ARDUINO SIMPLE SNAP
3D PRINTING	MATERIA 101

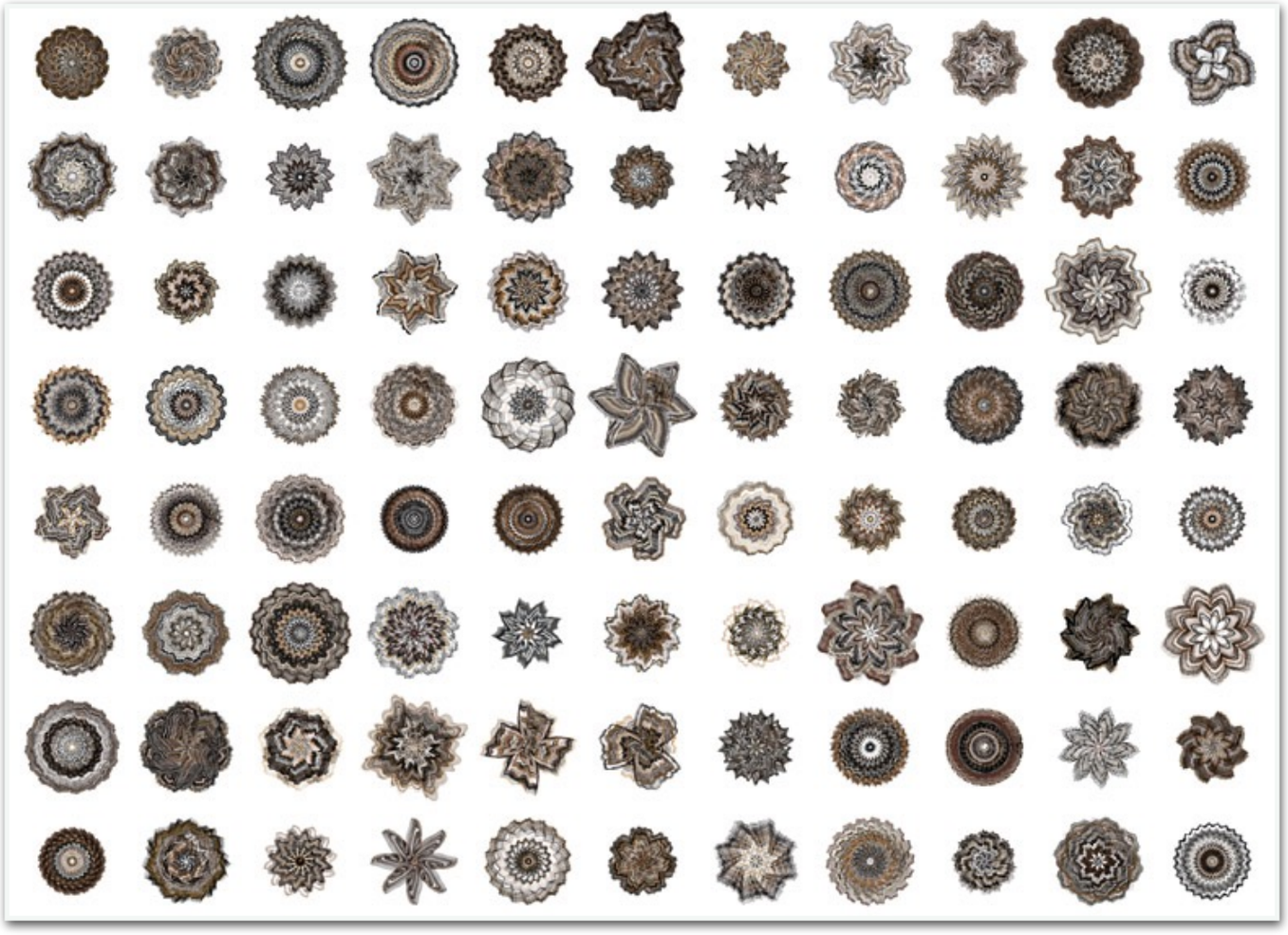
BOARDS · MODULES · SHIELDS · KITS · ACCESSORIES · COMING NEXT

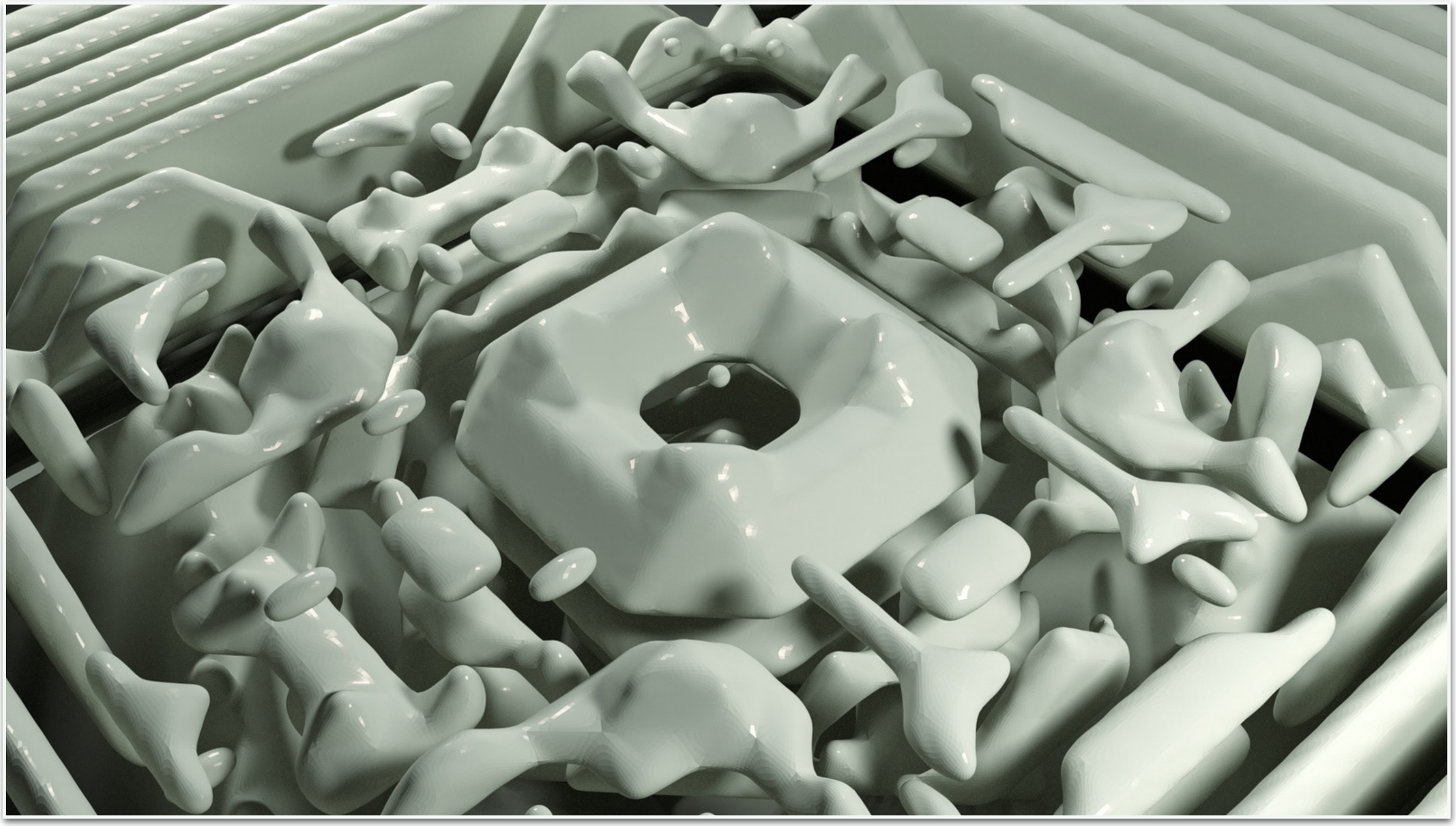
Processing

- On peut produire avec **Processing** des installation interactives avec des périphériques tels que :
  - la souris,
  - le clavier,
  - la caméra,
  - Kinect (capteur de mouvements)
  - une carte de prototype **Arduino** avec des capteurs :
    - de distance,
    - de mouvements,
    - de températures,
    - de localisation GPS

Processing

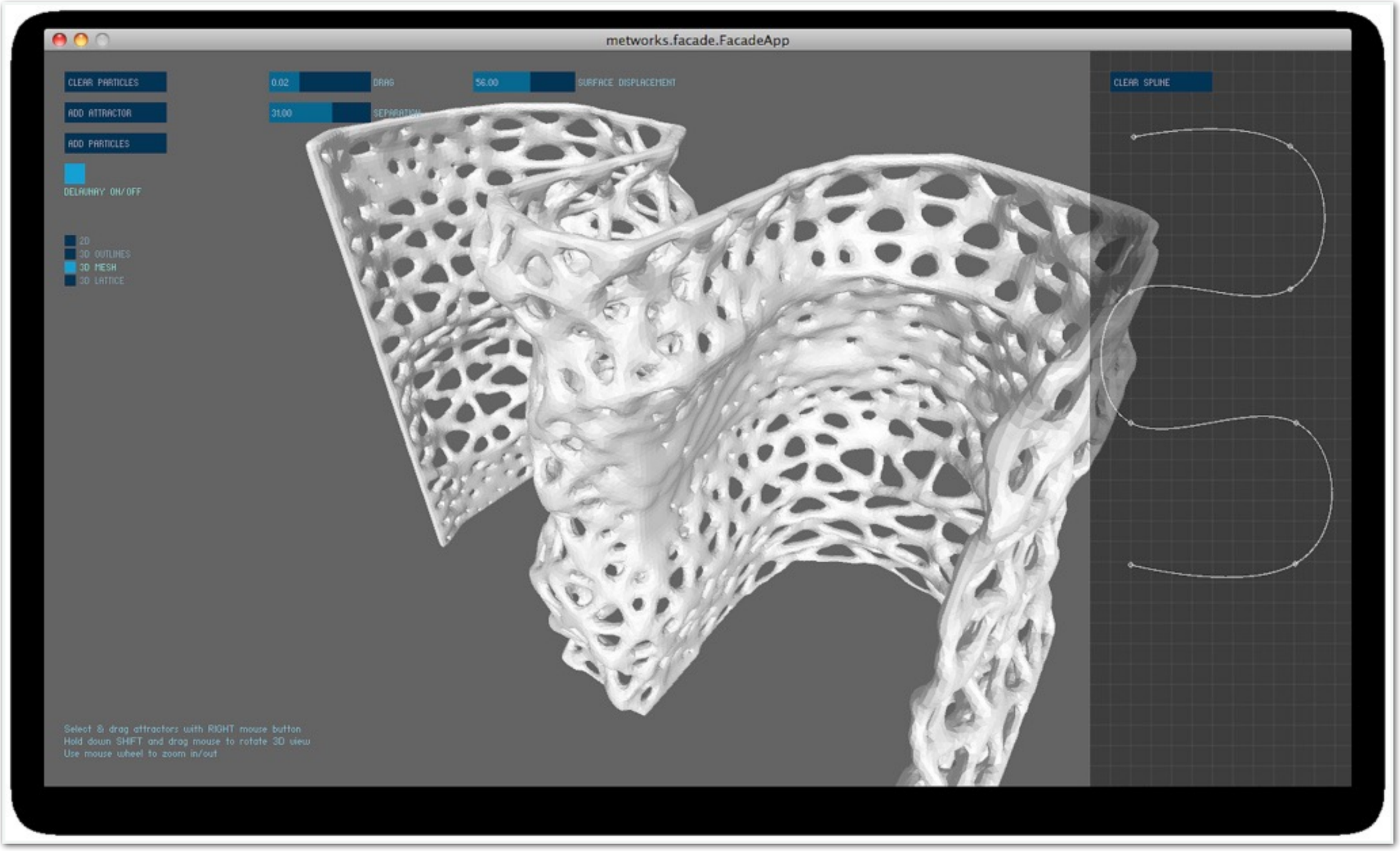
- **Processing** permet aussi au graphistes de générer des images ou des motifs à partir de données (data visualisation)





3D extruded timeline of 2D cellular automata simulation

Rendered with Luxrender

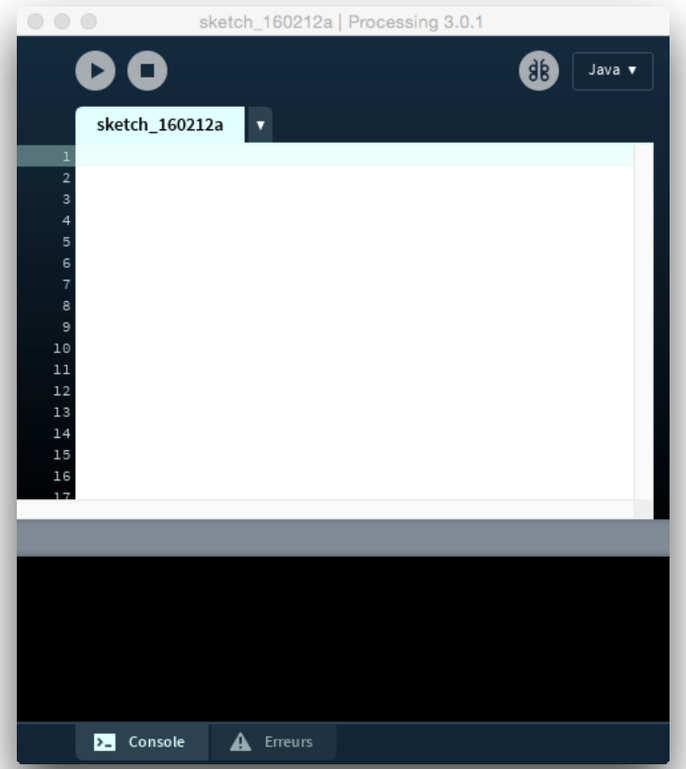


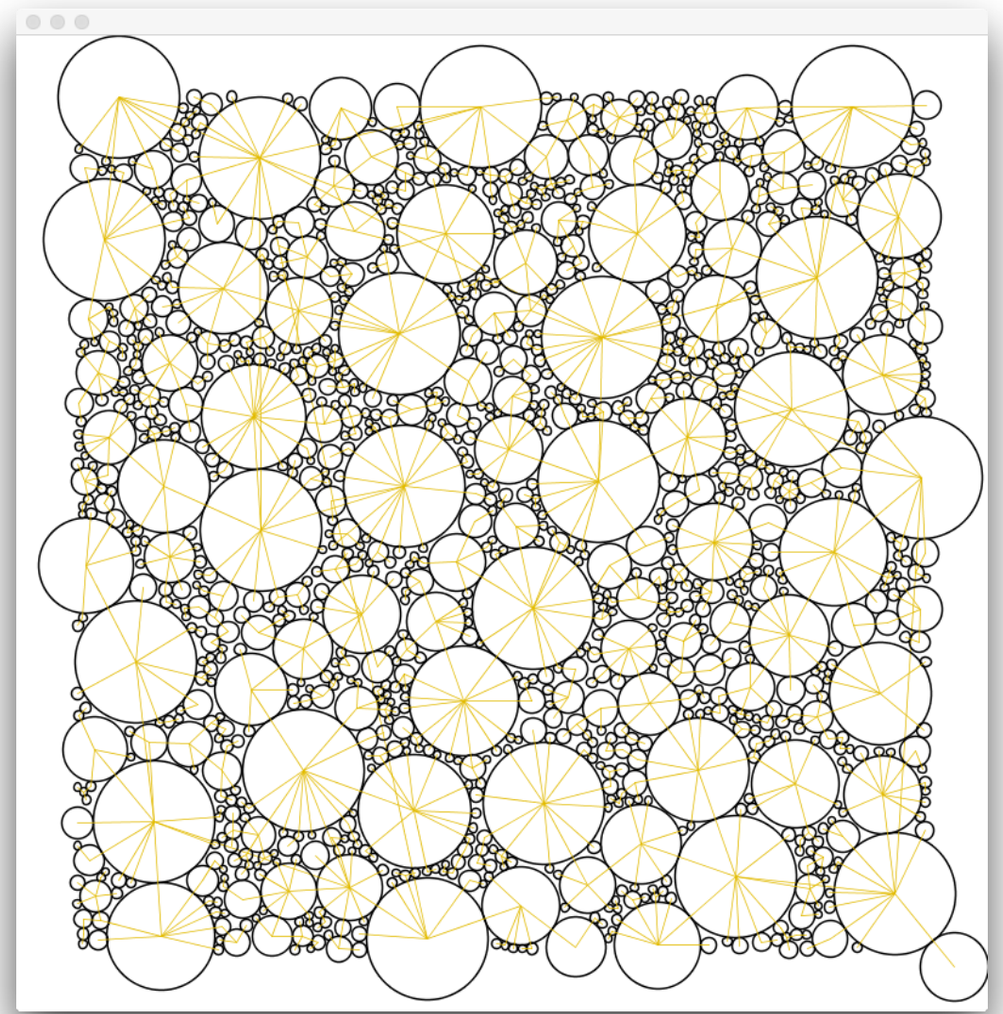
http://toxiclibs.org/2011/12/metworks-workshop-facade/

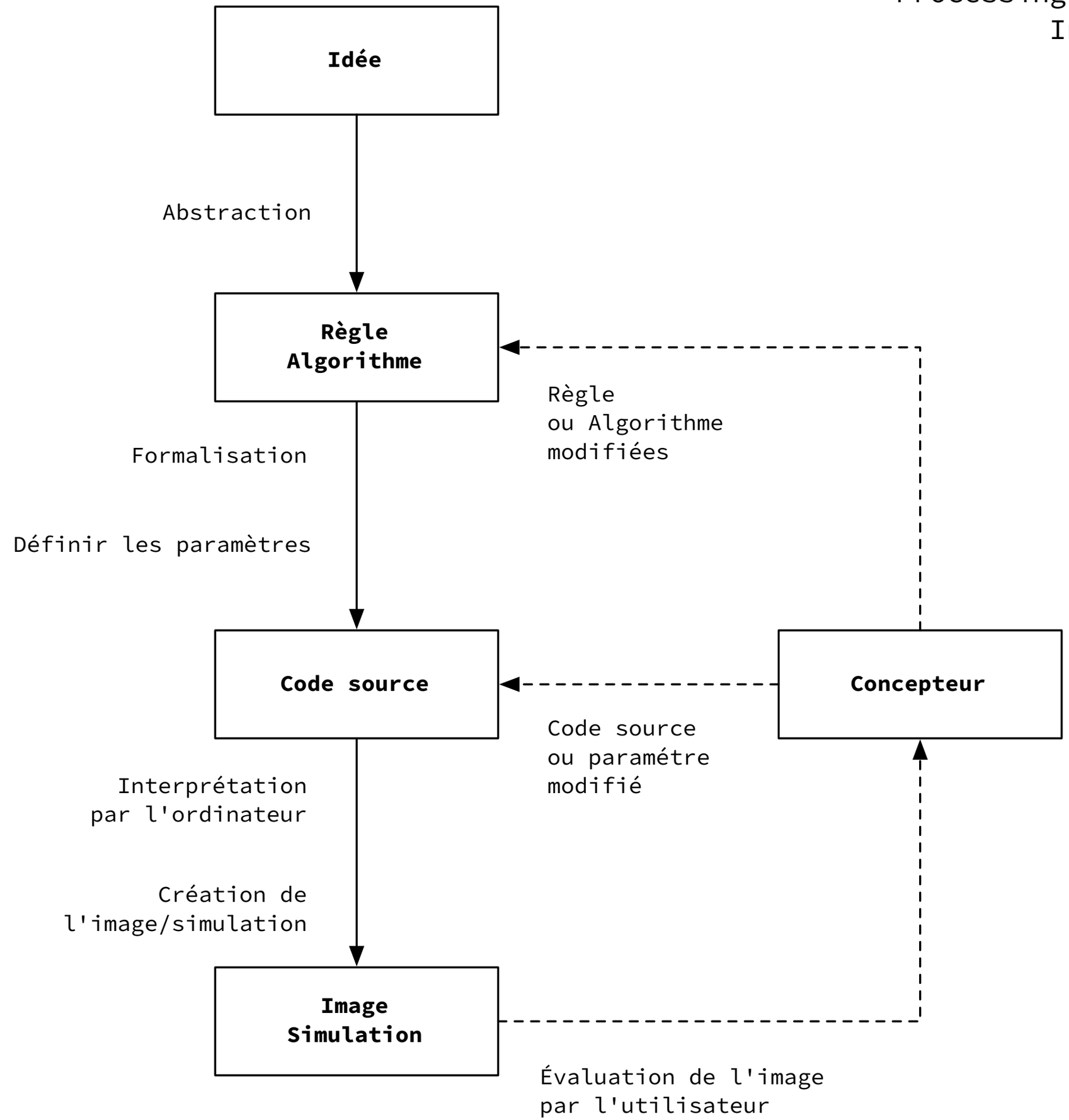
Origine

- 2001 : initié par Ben Fry et Casey Reas (étudiants de John Maeda au MediaLab du MIT)
- 2005 : prix au festival Art Electronica
- 2008 :
  - passage en production avec la version 1.0
  - démarrage de Processing.js par John Resig (auteur de jQuery)
- 2012 : Création de la **Fondation Processing**
- 2015 : version 3.0

L’environnement de travail

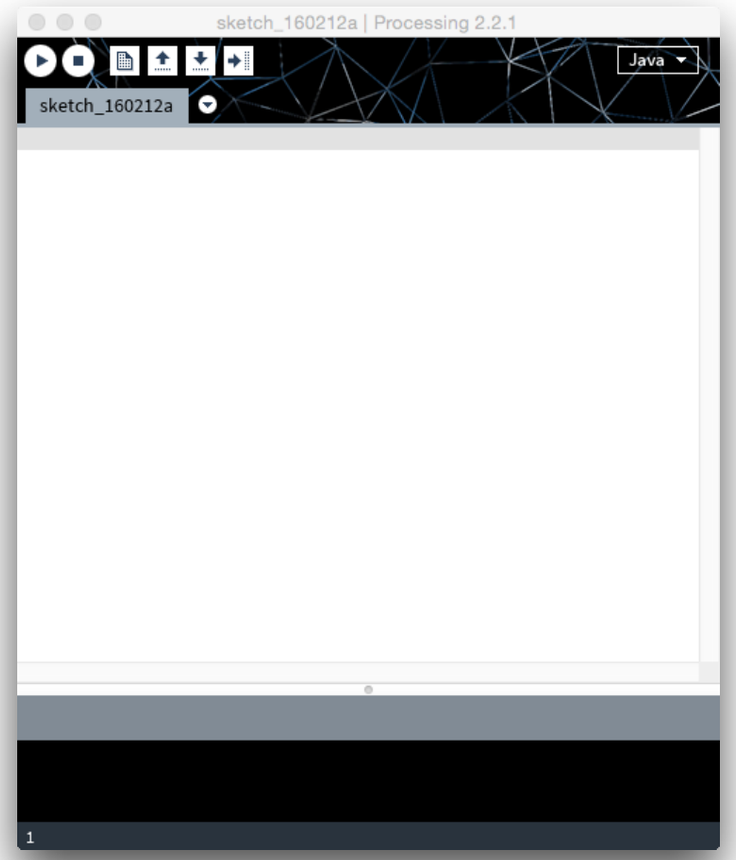


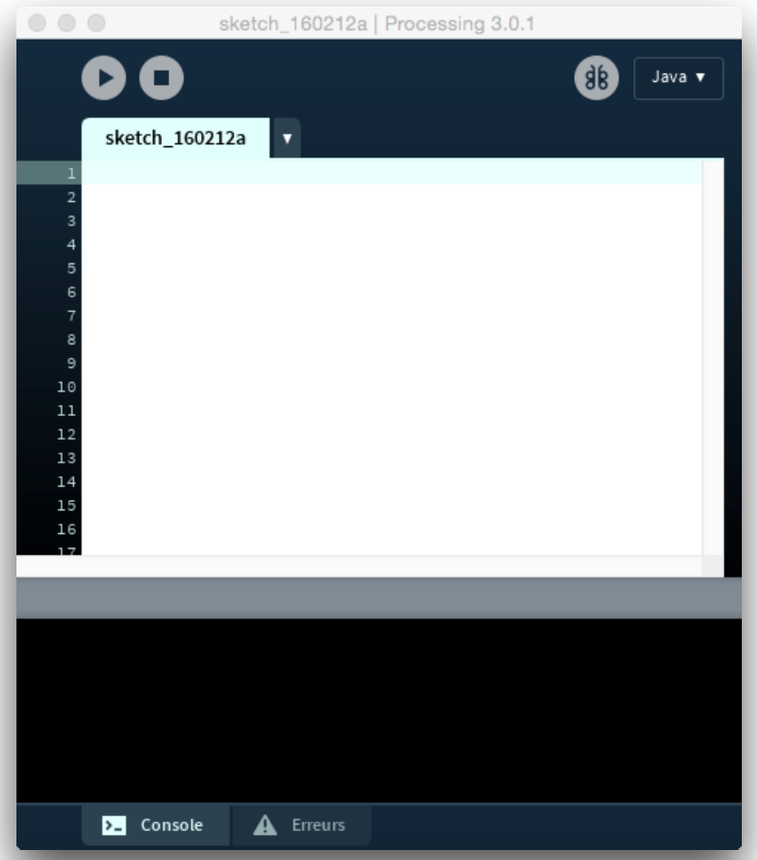



Idée
Abstraction
Règle
Algorithme
Formalisation
Définir les paramètres
Règle
ou Algorithme
modifiées
Code source
Concepteur
Code source
ou paramétre
modifié
Interprétation
par l'ordinateur
Création de
l'image/simulation
Image
Simulation
Évaluation de l'image
par l'utilisateur

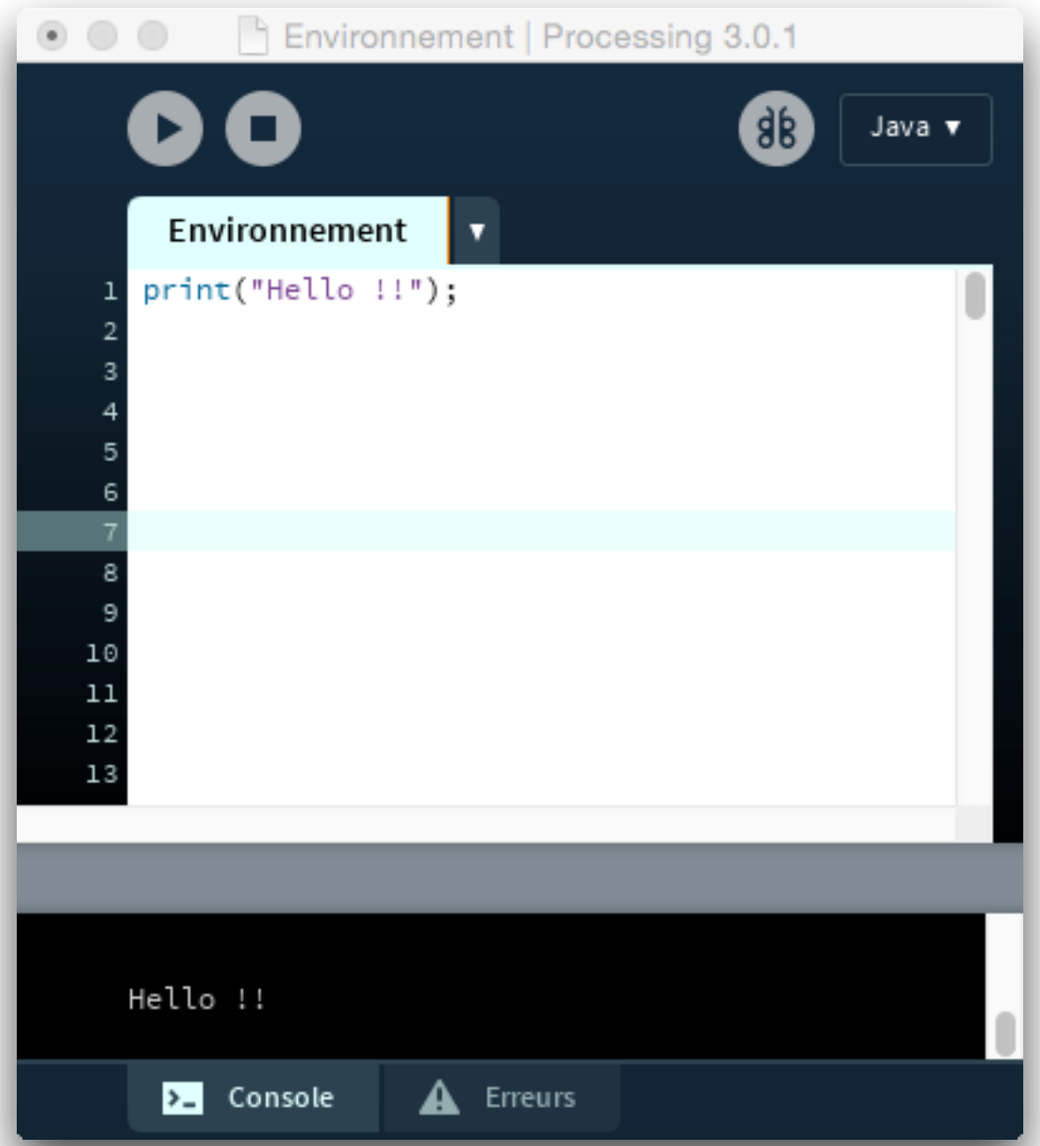
Processing

L'application



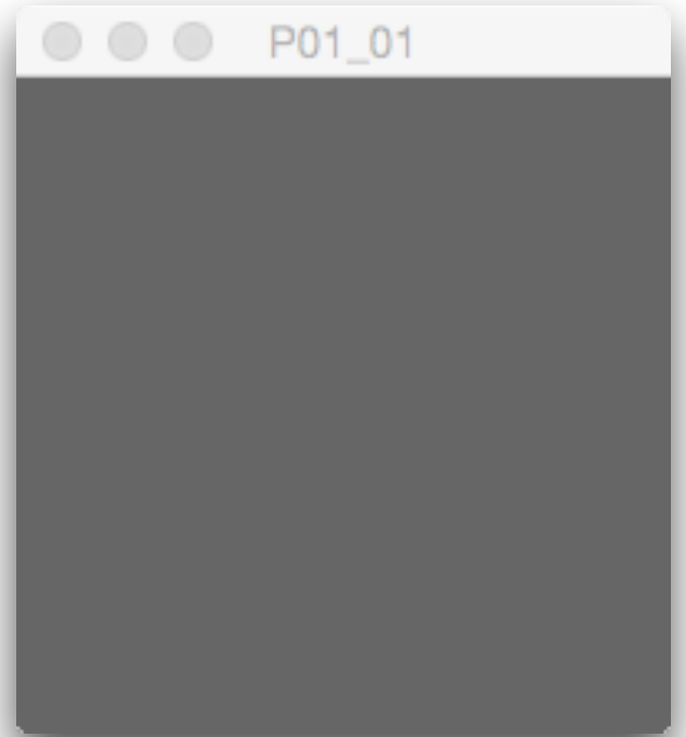


Premier « Sketch »



Premier « Sketch »

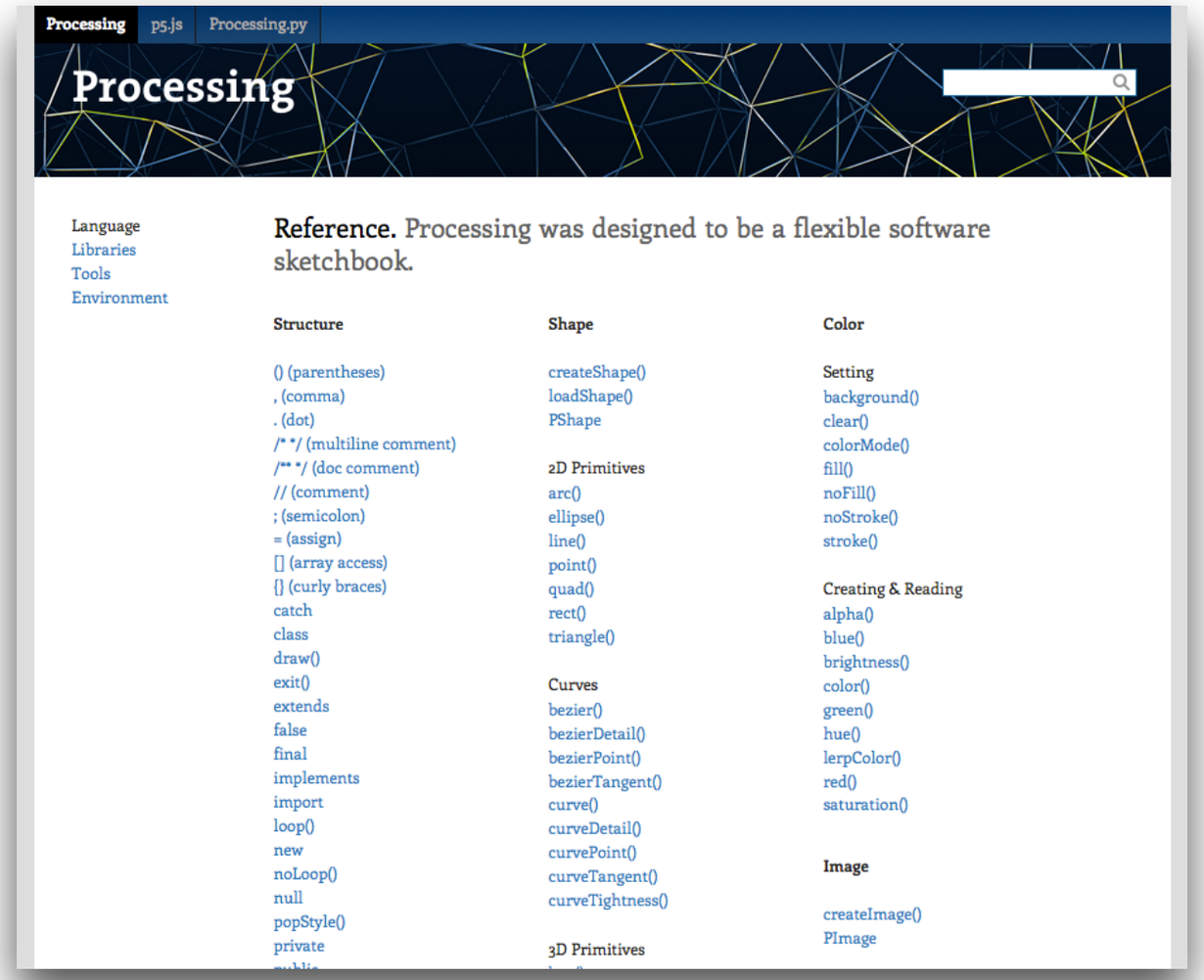
```
size(200, 200);
background(102);
```



Règles de mise en forme

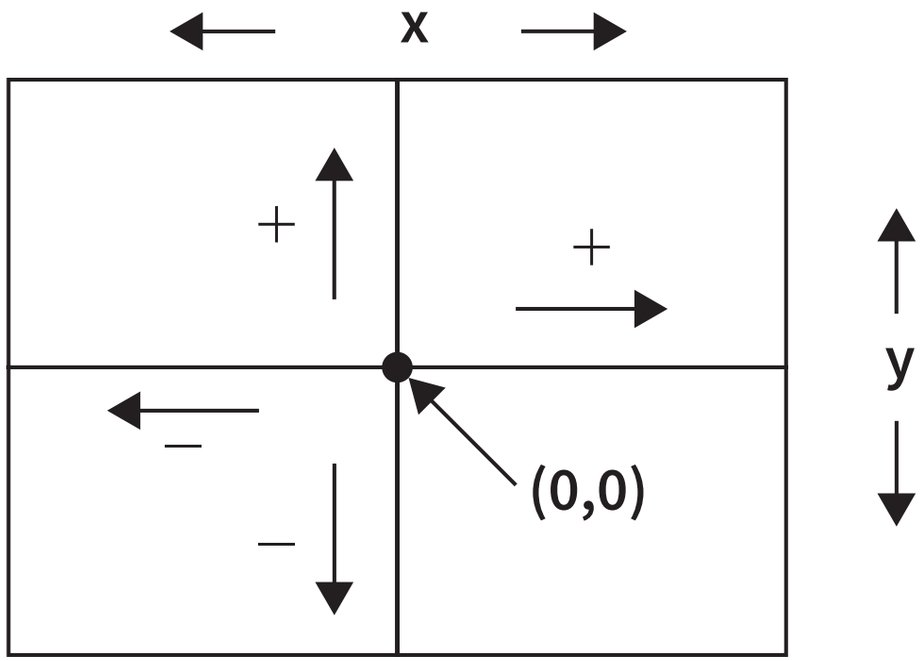
- Processing est sensible à la casse
- Processing n’est pas troublé lorsque l’on place plusieurs espaces
- Dans Processing les sauts de ligne sont cosmétiques
- Ce qui est ouvert doit être fermé :
  - les doubles guillemets et les simples guillemets,
  - parenthèses (), accolades {} et cochet []
- Les commentaires /* */ et //

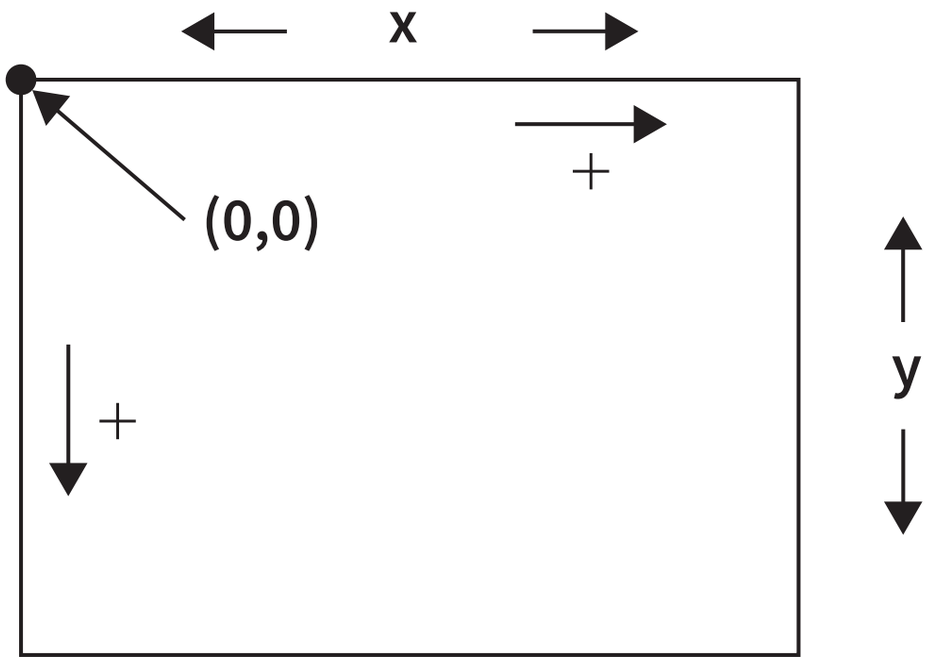
La documentation



Pixels

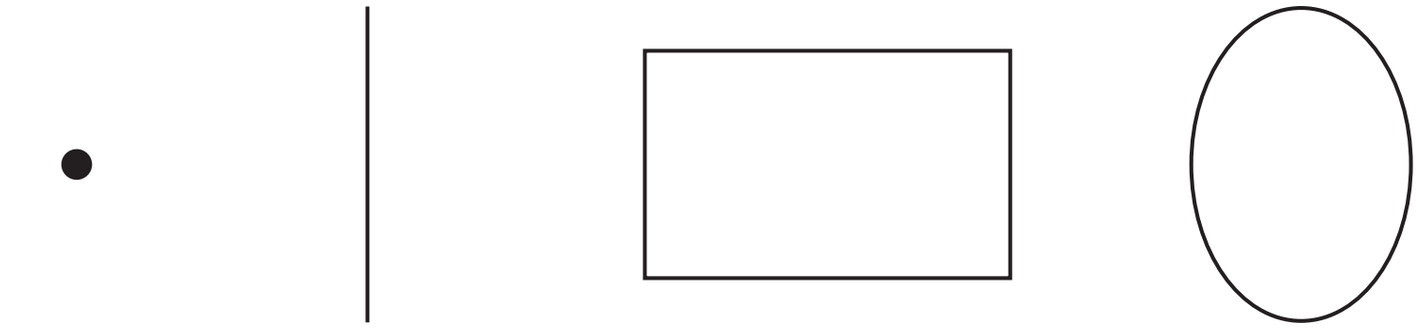
Système de coordonnées





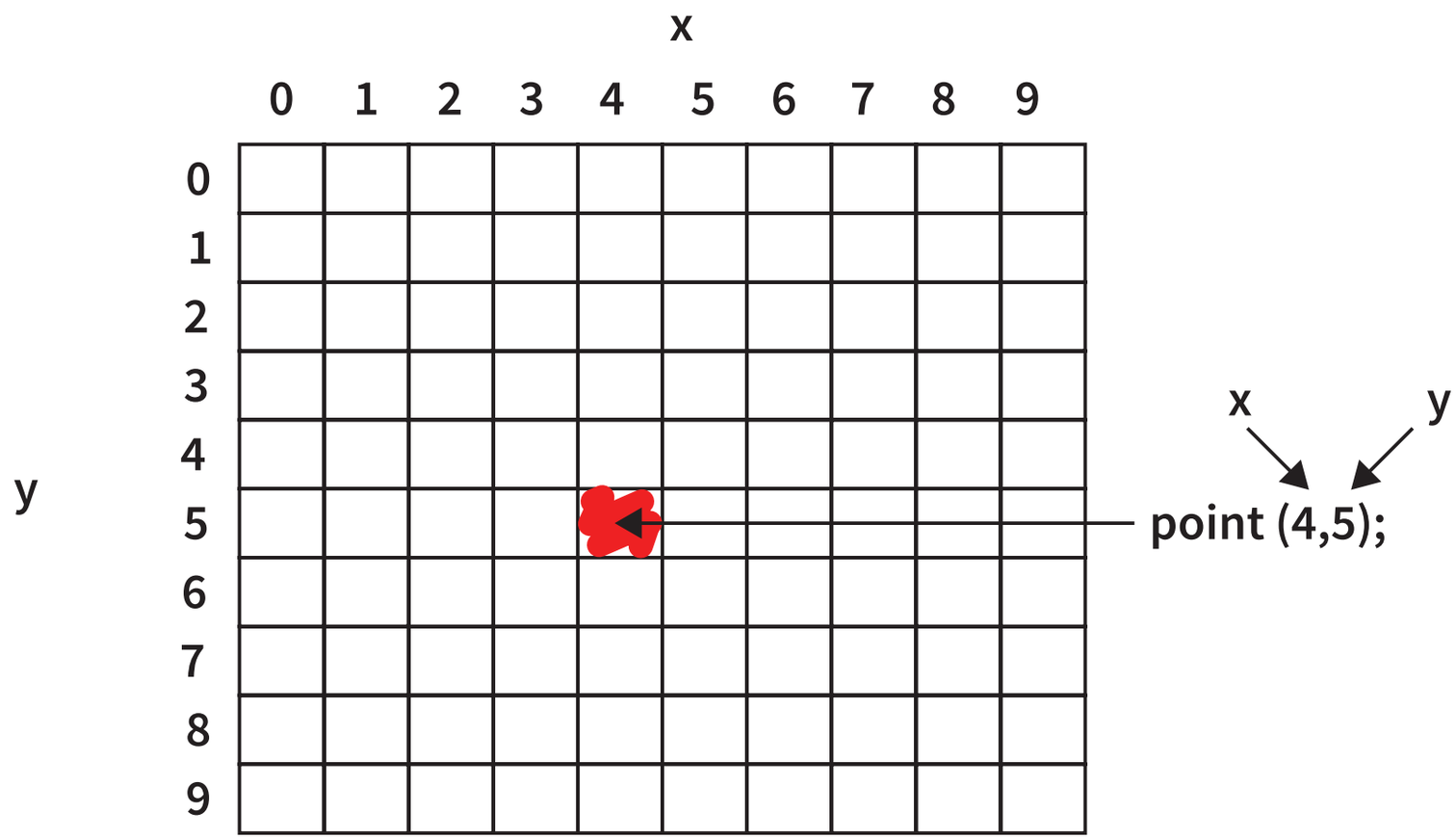
Ordinateur

Les formes



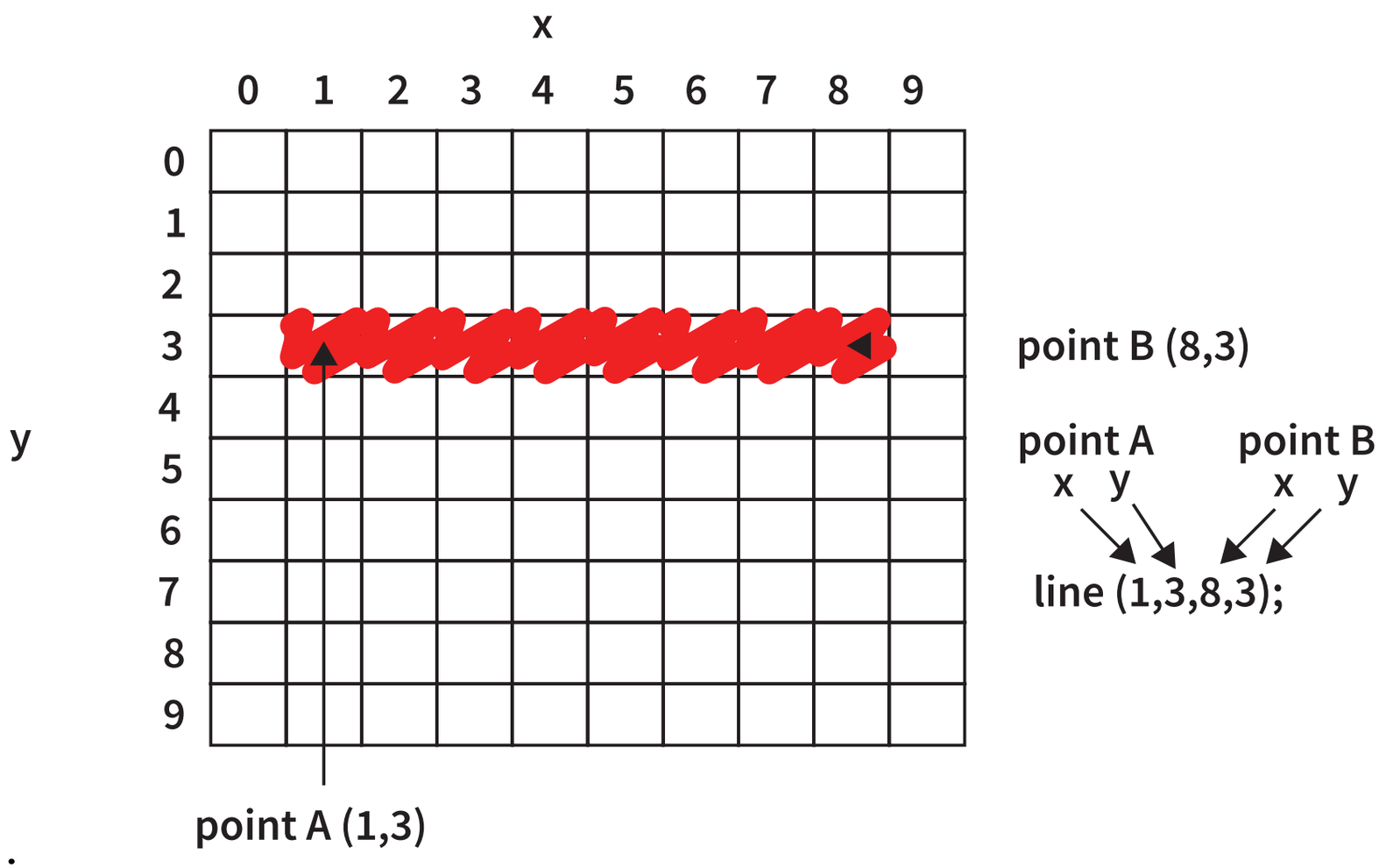
Point Line Rectangle Ellipse

Afficher un point



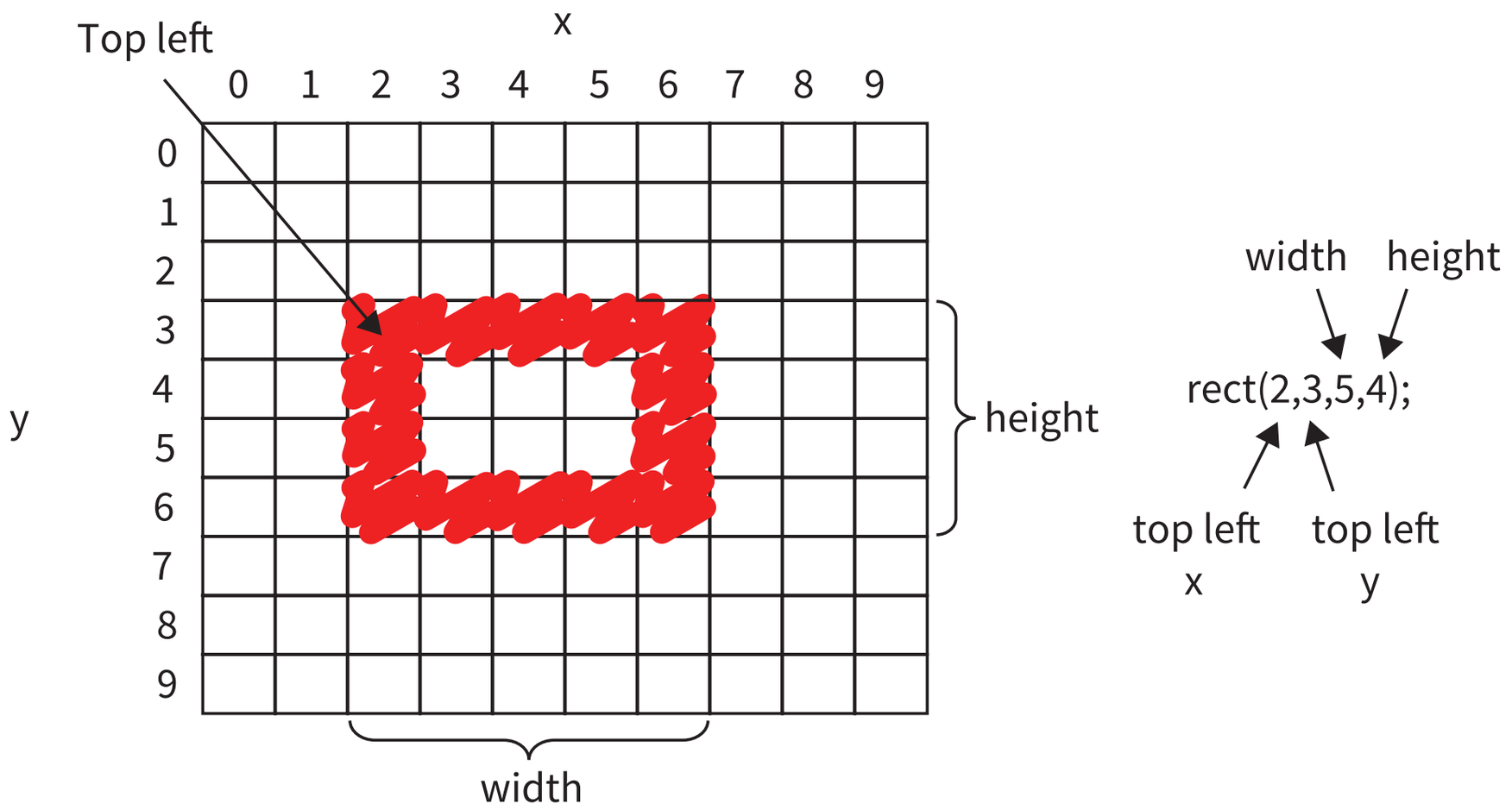
- en Processing:
  - `point(x, y);`

Afficher une ligne



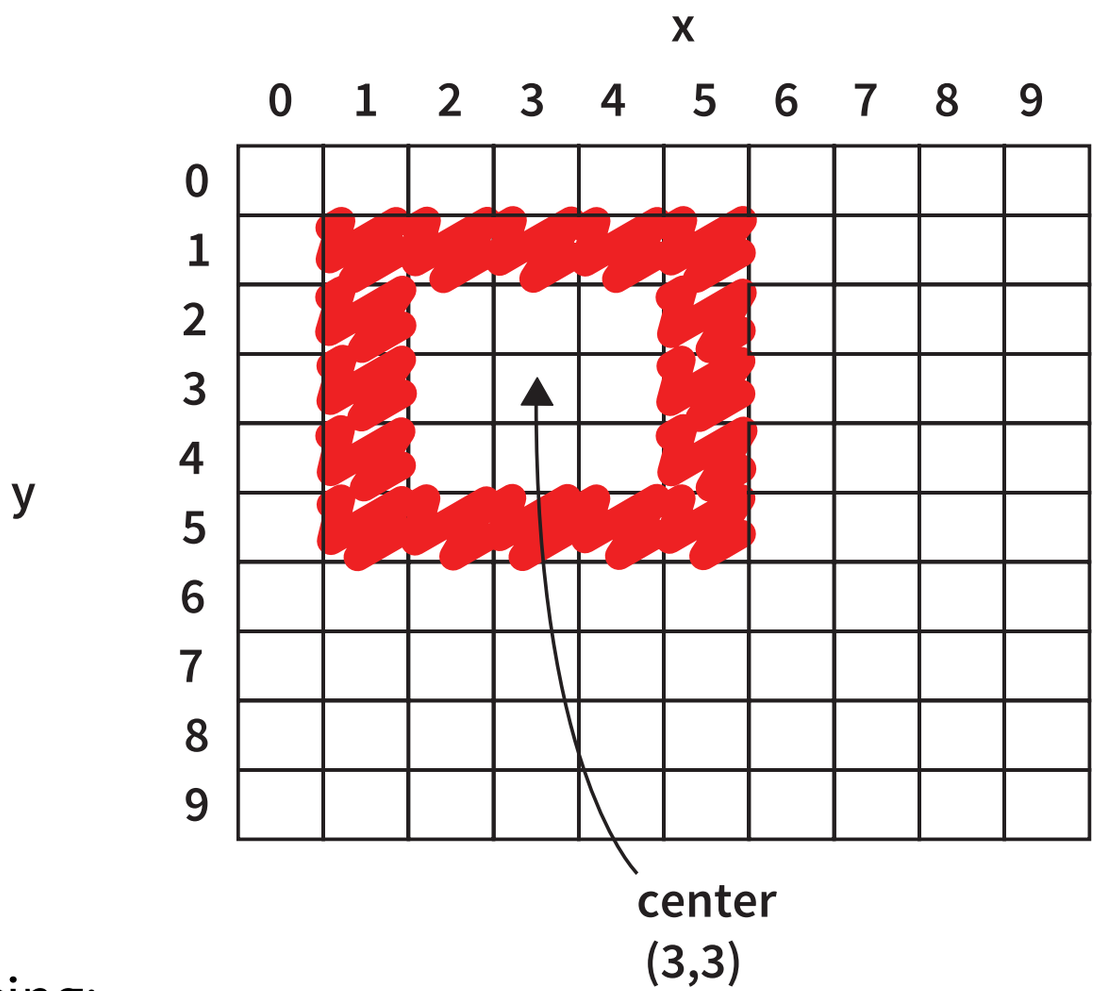
- en Processing:
  - `line(x1, y1, x2, y2);`

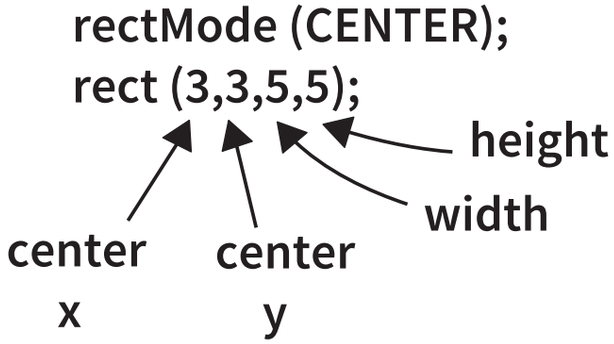
Afficher un rectangle



- en Processing:
  - `rect(x, y, width, height);`
  - NOTE: le mode par défaut est CORNER

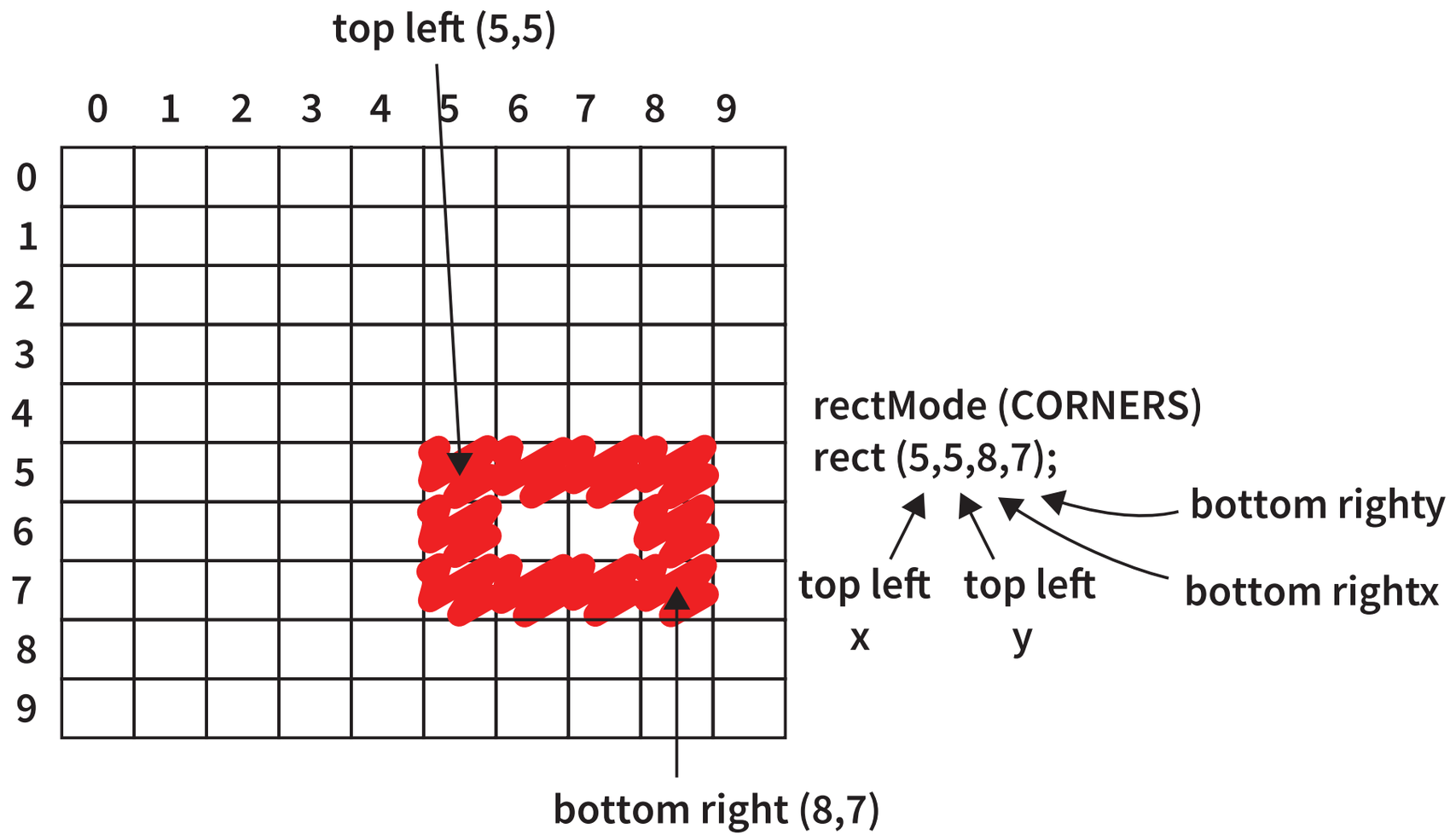
Afficher un rectangle (mode CENTER)





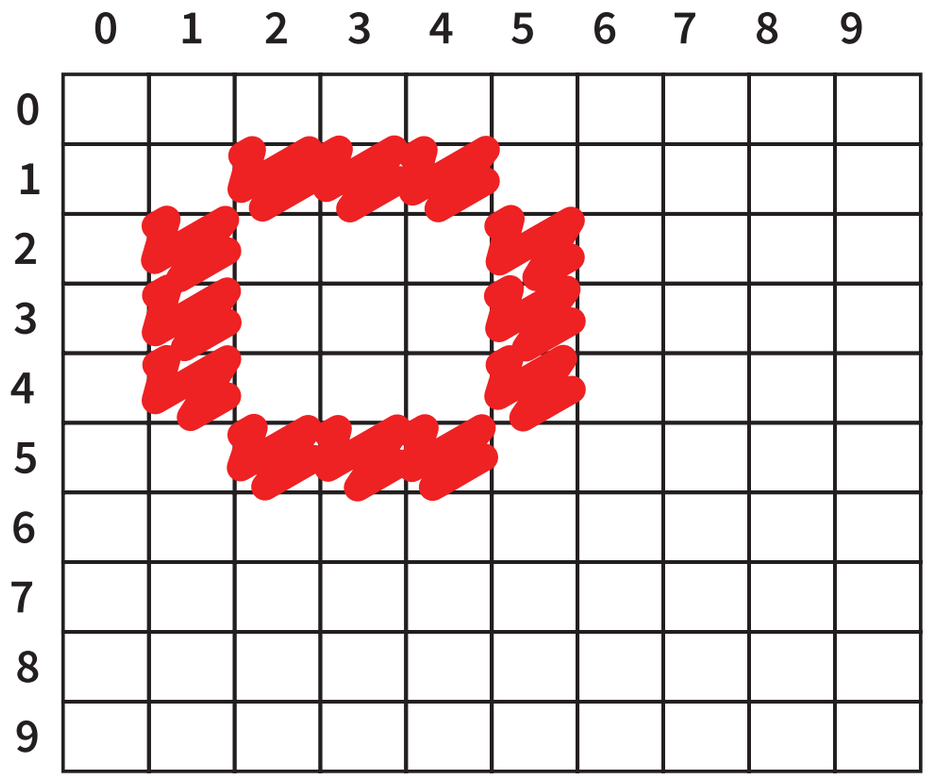
- en Processing:
  - `rectMode(CENTER);`
  - `rect(x, y, width, height);`

Afficher un rectangle (mode CORNERS)



- en Processing:
  - `rectMode(CORNERS);`
  - `rect(x1, y1, x2, y2);`

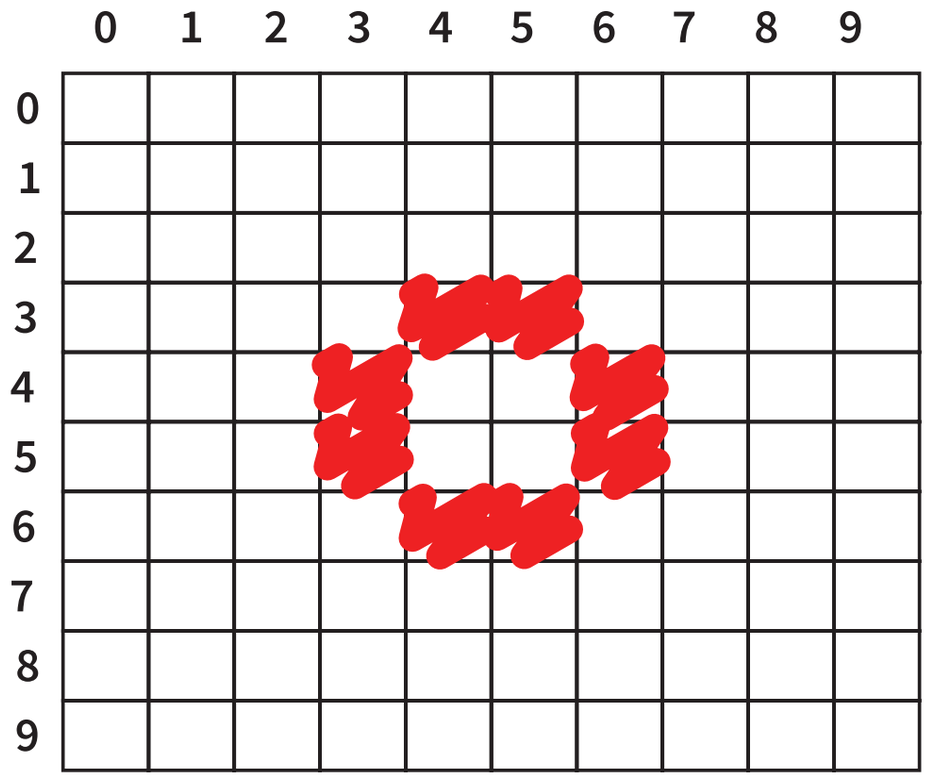
Afficher une ellipse (mode CENTER)



ellipseMode (CENTER);
ellipse (3,3,5,5);

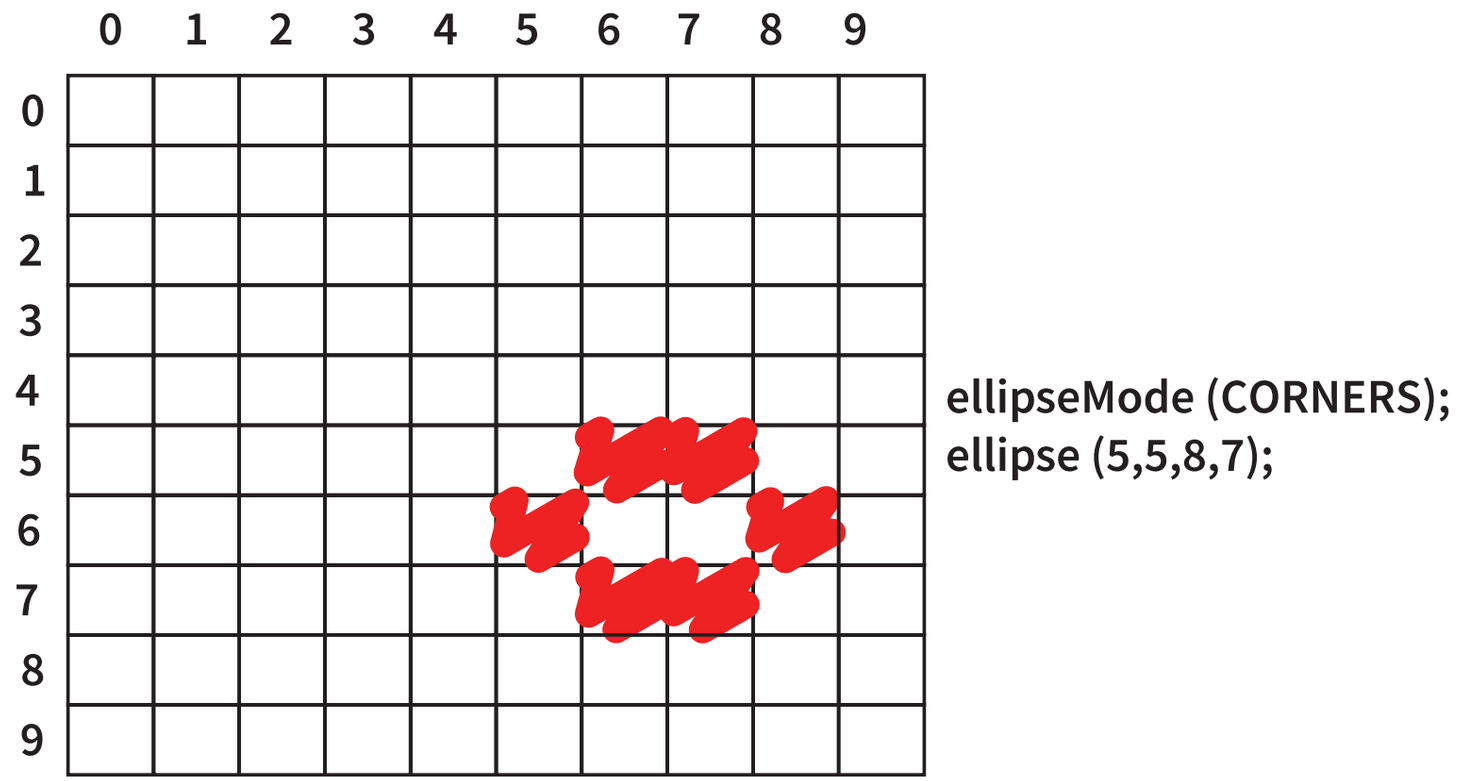
- en Processing:
  - **`ellipseMode(CENTER);`**
  - **`ellipse(x, y, width, height);`**
- NOTE: le mode par défaut est CENTER

Afficher une ellipse (mode CORNER)



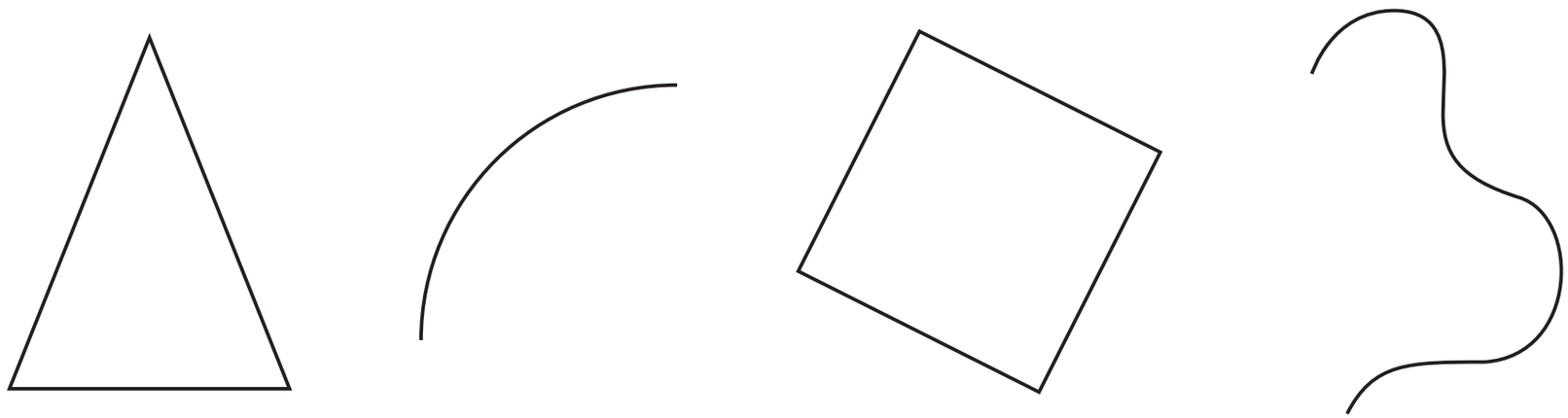
- en Processing:
  - `ellipseMode(CORNER);`
  - `ellipse(x, y, width, height);`

Afficher une ellipse (mode CORNERS)



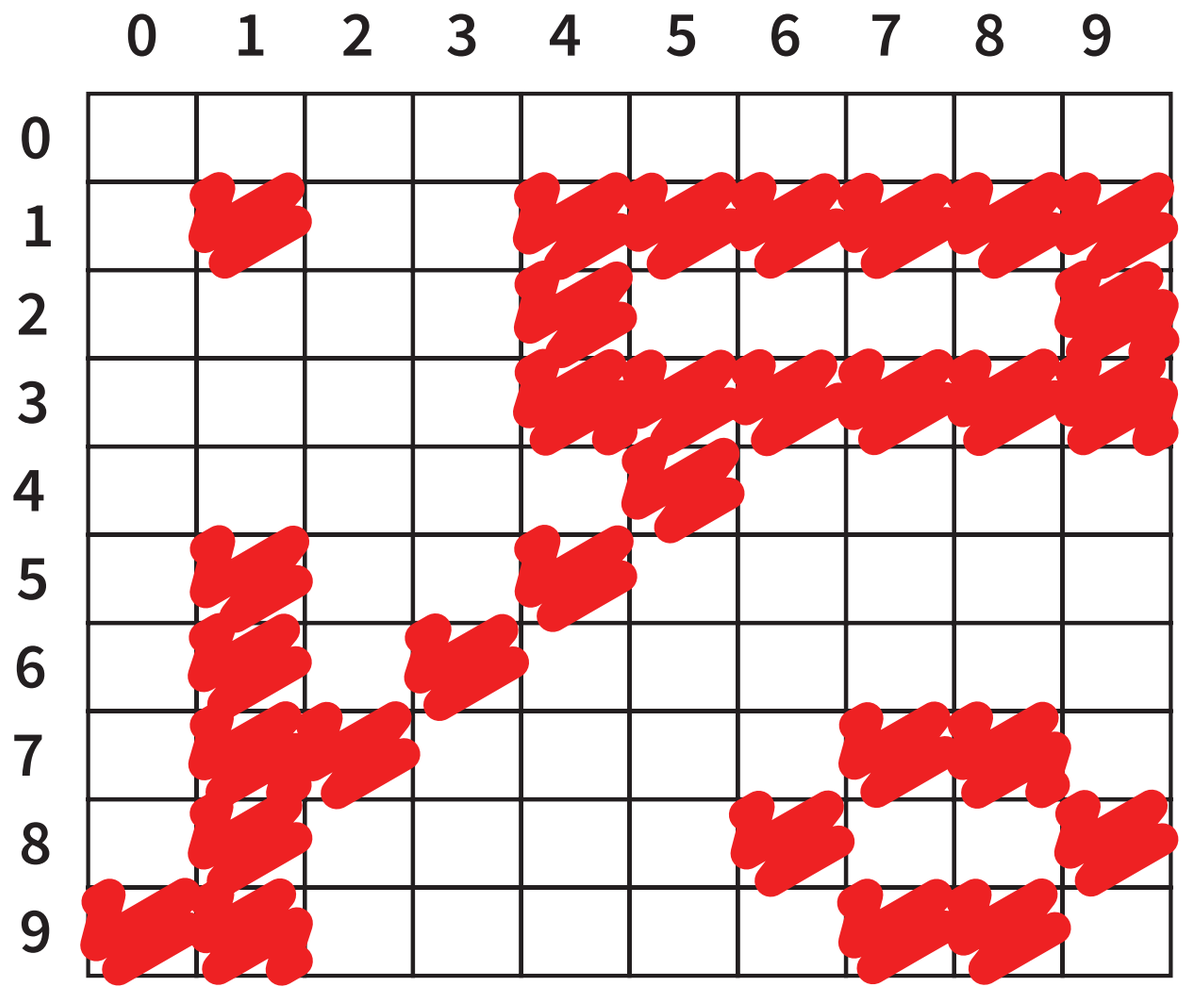
- en Processing:
  - `ellipseMode(CORNERS);`
  - `ellipse(x1, y1, x2, y2);`

Les autres formes



On reviendra dessus un peu plus tard.

Exercice

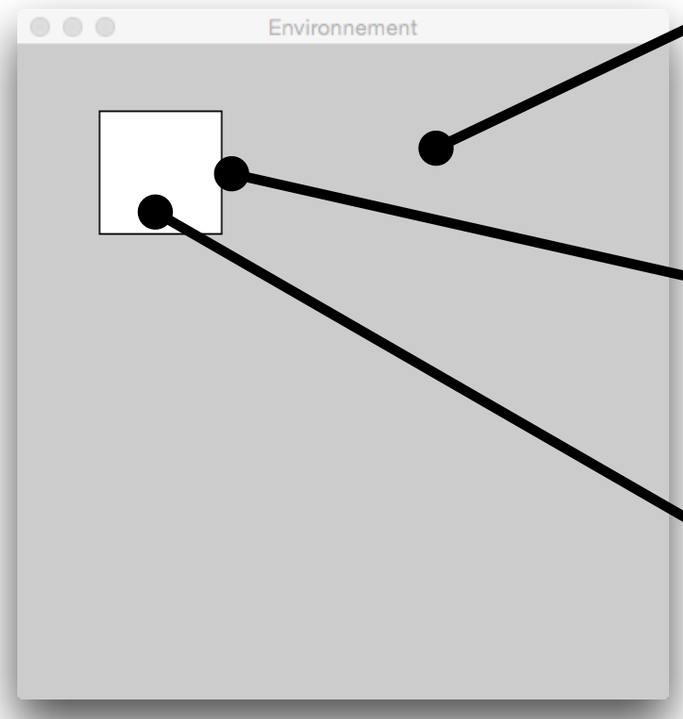


Niveau de gris

0 50 87 162 209 255

Mise en place

```
size(400,400);
rect(50,40,75,75);
```



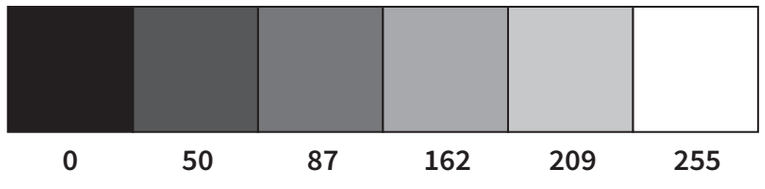
le fond est gris

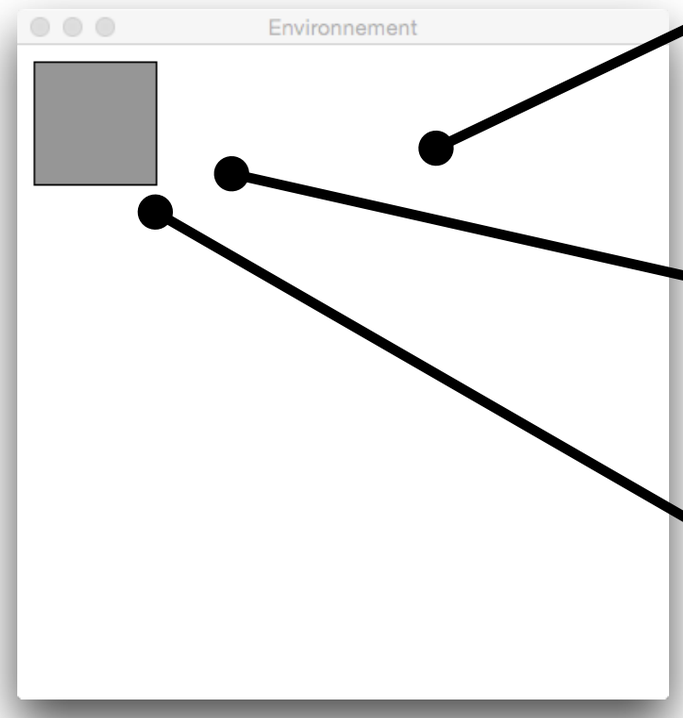
le contour est noir

l'intérieur est blanc

stroke () et fill()

```
size(400,400);
background(255);
stroke(0);
fill(150);
rect(10,10,75,75);
```





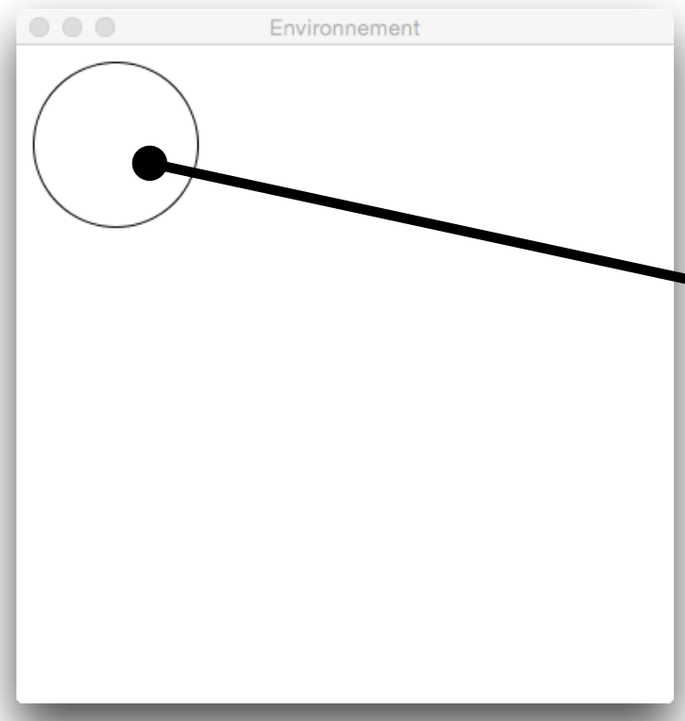
le fond est blanc

le contour est noir

l’intérieur est gris

noFill()

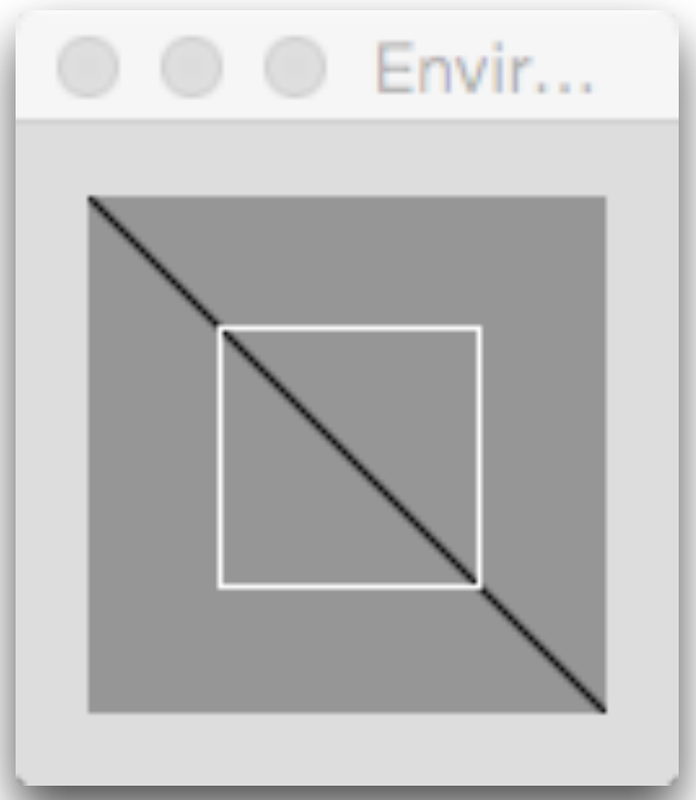
```
size(400,400);
background(255);
stroke(0);
noFill();
ellipse(60,60,100,100);
```



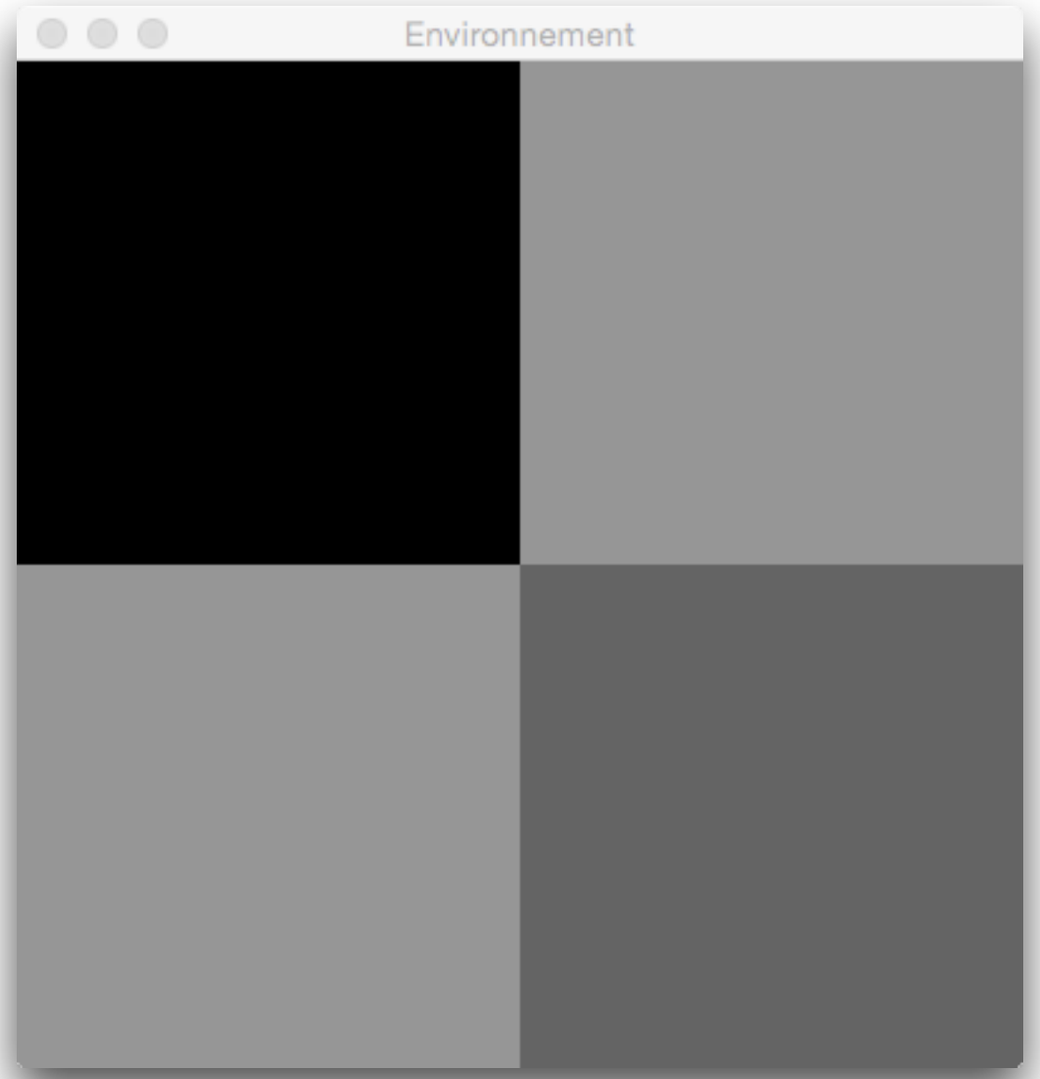
nofill() enlève
l’intérieur

Deux formes

```
background(150);
stroke(0);
line(0,0,100,100);
stroke(255);
noFill();
rect(25,25,50,50);
```



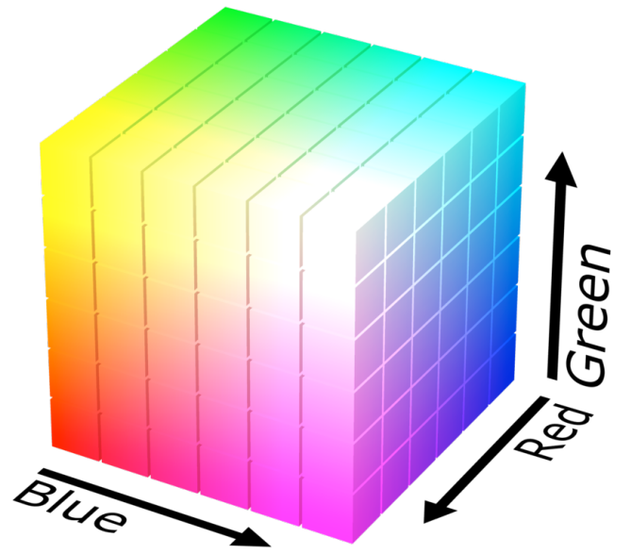
Exercice



Couleur RVB (RGB)

Codage de la teinte saumon

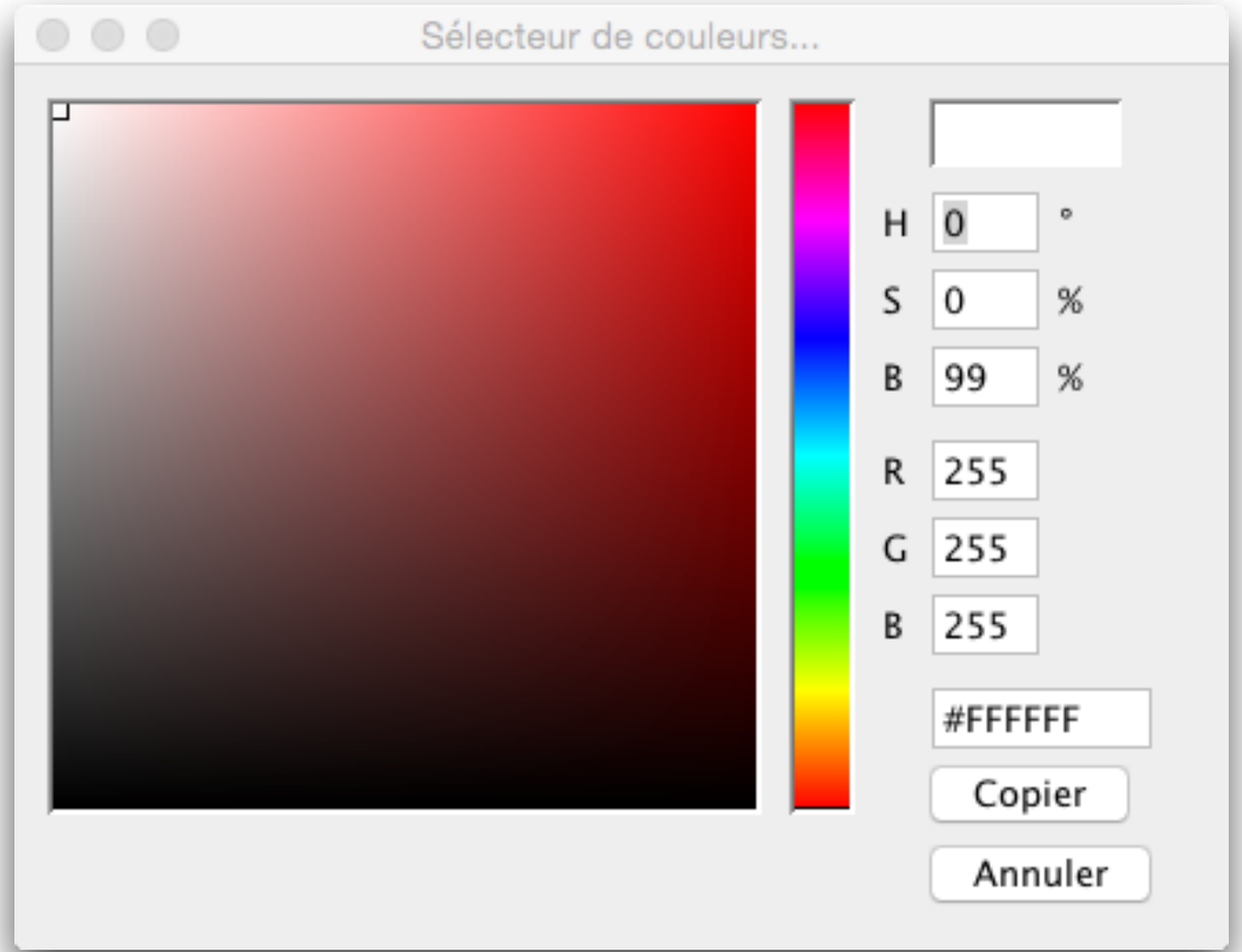
- rouge = 100%, vert = 80%, bleu = 60% ou
- rouge = 255, vert = 204, bleu = 153



Couleur RVB (RGB)

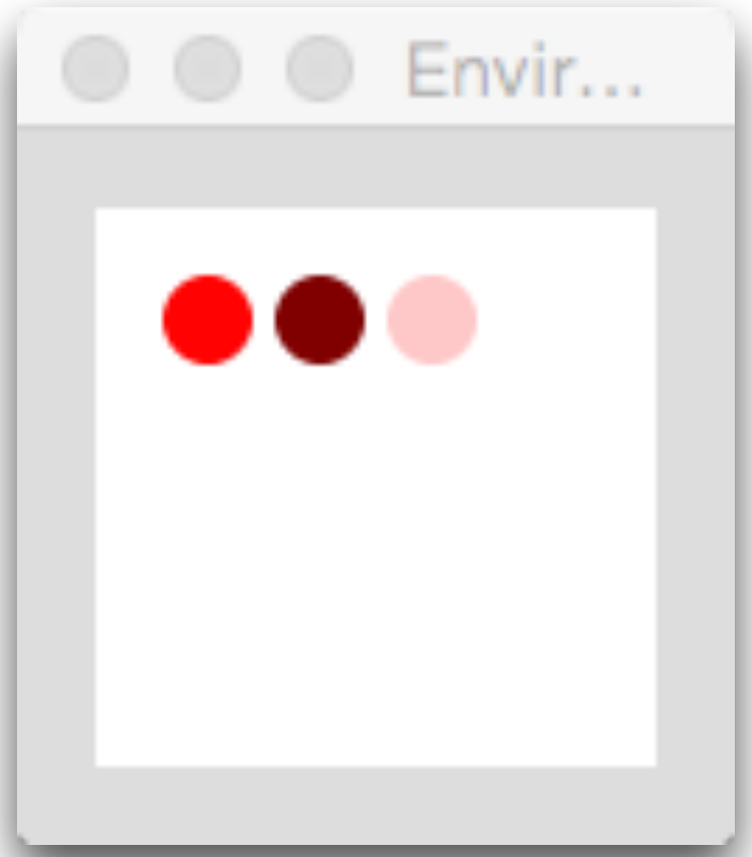
dans Processing :

Menu -> Outils ->
Sélecteurs de couleurs



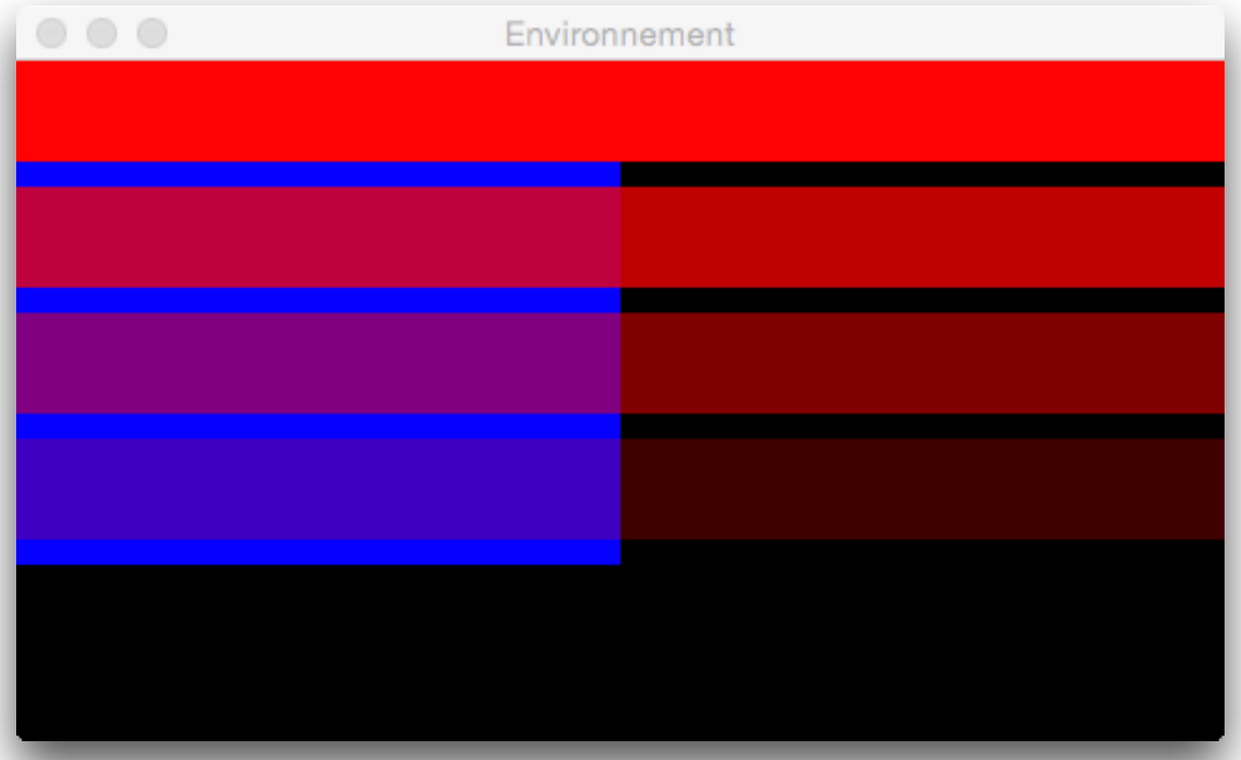
Exemple

```
background(255);
noStroke();

// Rouge
fill(255, 0, 0);
ellipse(20, 20, 16, 16);

// Rouge foncé
fill(127, 0, 0);
ellipse(40, 20, 16, 16);

// Rose
fill(255, 200, 200);
ellipse(60, 20, 16, 16);
```



Transparence

```
size(480, 270);
background(0);
noStroke();

// trois arguments donc 100% d'opacité.
fill(0, 0, 255);
rect(0, 0, 240, 200);

// 255 donc 100% d'opacité.
fill(255, 0, 0, 255);
rect(0, 0, 480, 40);

// 75% d'opacité.
fill(255, 0, 0, 191);
rect(0, 50, 480, 40);

// 55% d'opacité.
fill(255, 0, 0, 127);
rect(0, 100, 480, 40);

// 25% d'opacité.
fill(255, 0, 0, 63);
rect(0, 150, 480, 40);
```



Exercice