1-2 ► Hello World

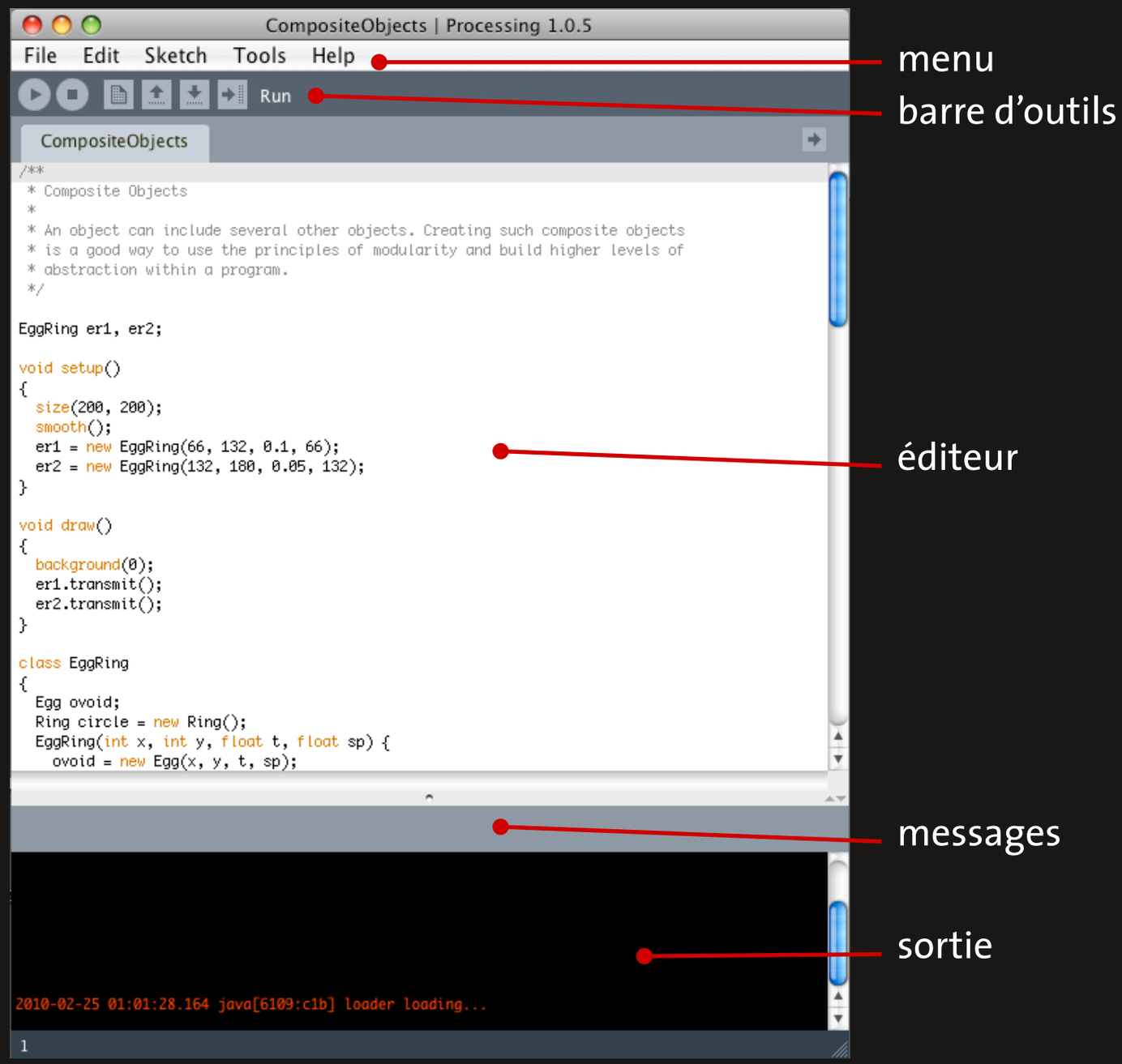
- installons Processing http://processing.org/download/

- ouvrons (enfin) Processing !

- l'interface est réduite au stricte nécessaire.

1-2 ► Hello World

Tour de l'interface



1-2 ► Hello World

- un programme, dans P5, est appelé un **sketch**

- **notre premier sketch: hello world!** (pour d'autres languages, le hello world affiche du texte, comme P5 est orienté image, on va afficher une ligne).

```
line(15, 25, 70, 90);
```

- Cliquez sur 

1-2 ► Hello World

- changeons la taille et les couleurs:

```
size(400, 400);
background(192, 64, 0);
stroke(255);
line(150, 25, 270, 350);
```

- Testons, puis ajoutons quelques commentaires:

```
// taille de l'image
size(400, 400);
//couleur du fond
background(192, 64, 0);
//couleur des lignes
stroke(255);
//tracer une ligne
line(150, 25, 270, 350);
```

1-2 ► Hello World

```
//
/* */
```

commentaires ignorés par l'ordinateur. Indispensables pour s'y retrouver dans ses sketchs.

1-2 ► Hello World

les fonctions nous permettent d'exécuter des actions (dessiner des formes, définir des couleurs, calculer des nombres,...). Une fonction est en général écrite en bas de casse.
Les infos séparées par une virgule sont les paramètres de cette fonction.

1-2 ► Hello World

; (le point virgule)

Le point virgule est un séparateur, il permet de faire comprendre à l'ordinateur que l'on passe d'une ligne d'instruction à l'autre.
L'oubli d'un point virgule déclenchera l'apparition d'un message d'erreur...

1-2 ► Hello World

- Attention, sensibilité à la casse.

- Les espaces n’ont pas d’importance.

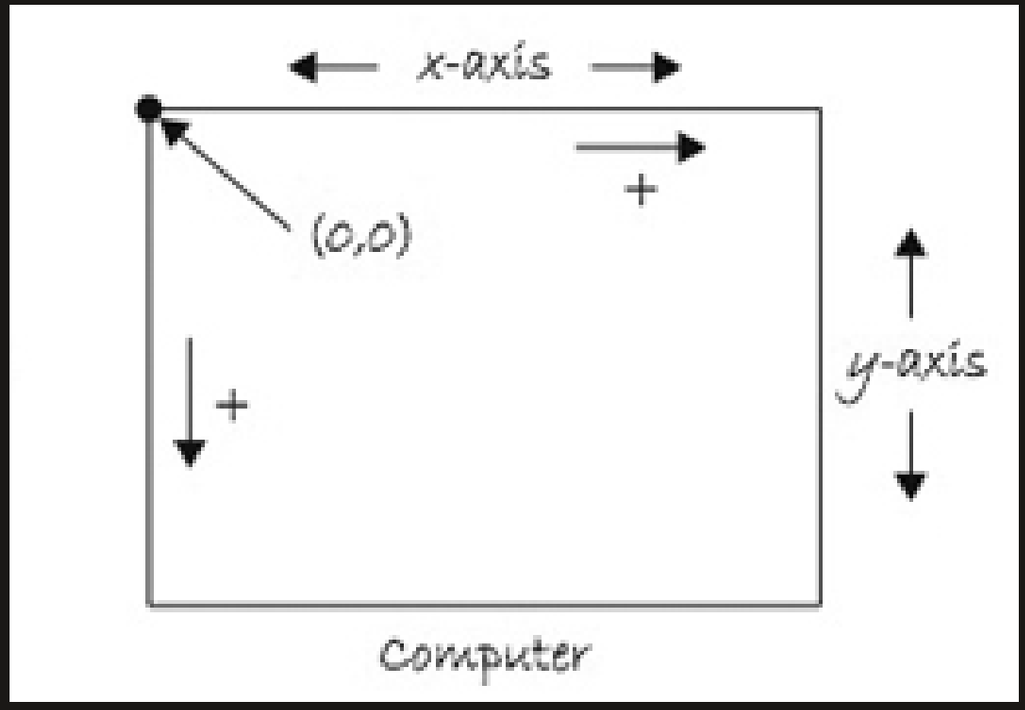
- La console nous permet de comprendre ce qui se passe pendant l’exécution du programme, grâce aux fonctions:

```
print(  )
println(  )
```

1-2 ► Hello World

Se repérer dans l'espace 2D

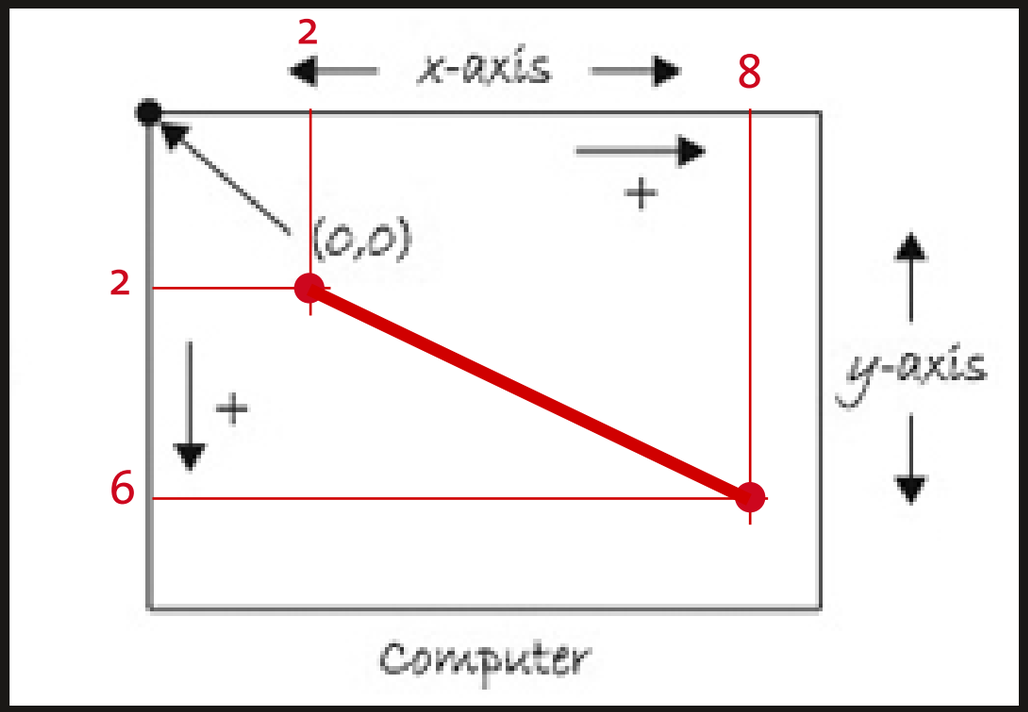
- la plus petite unité d'un écran => le pixel
c'est donc l'unité qui nous servira à définir des points dans l'espace de l'écran,
selon le schéma ci-dessous:



1-2 ► Hello World

Se repérer dans l'espace 2D

- pour tracer une ligne, il faut donc relier deux points:

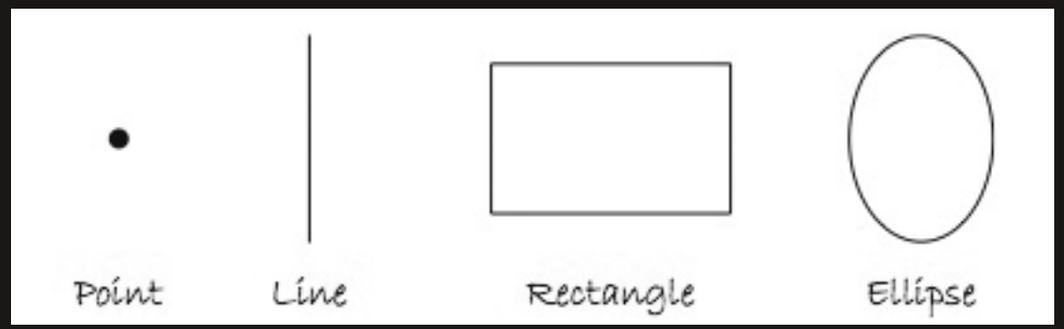


- en code, ça donne `line (2,2,8,6);`

1-2 ► Hello World

Se repérer dans l'espace 2D

- les primitives de base disponibles:



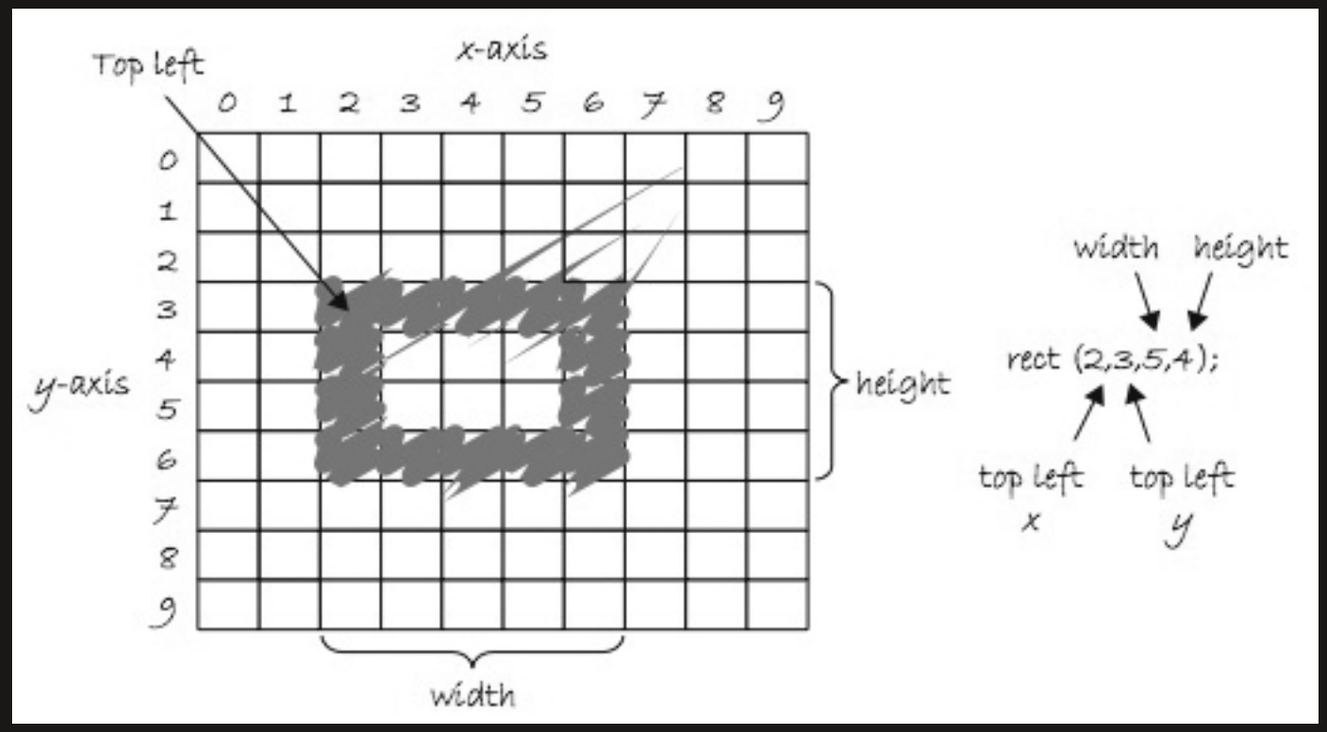
- il en existe d'autres...

1-2 ► Hello World

Se repérer dans l'espace 2D

- les modes de dessin du rectangle

- le mode par défaut CORNER: on définit la position du coin supérieur gauche, puis la largeur, puis la hauteur.

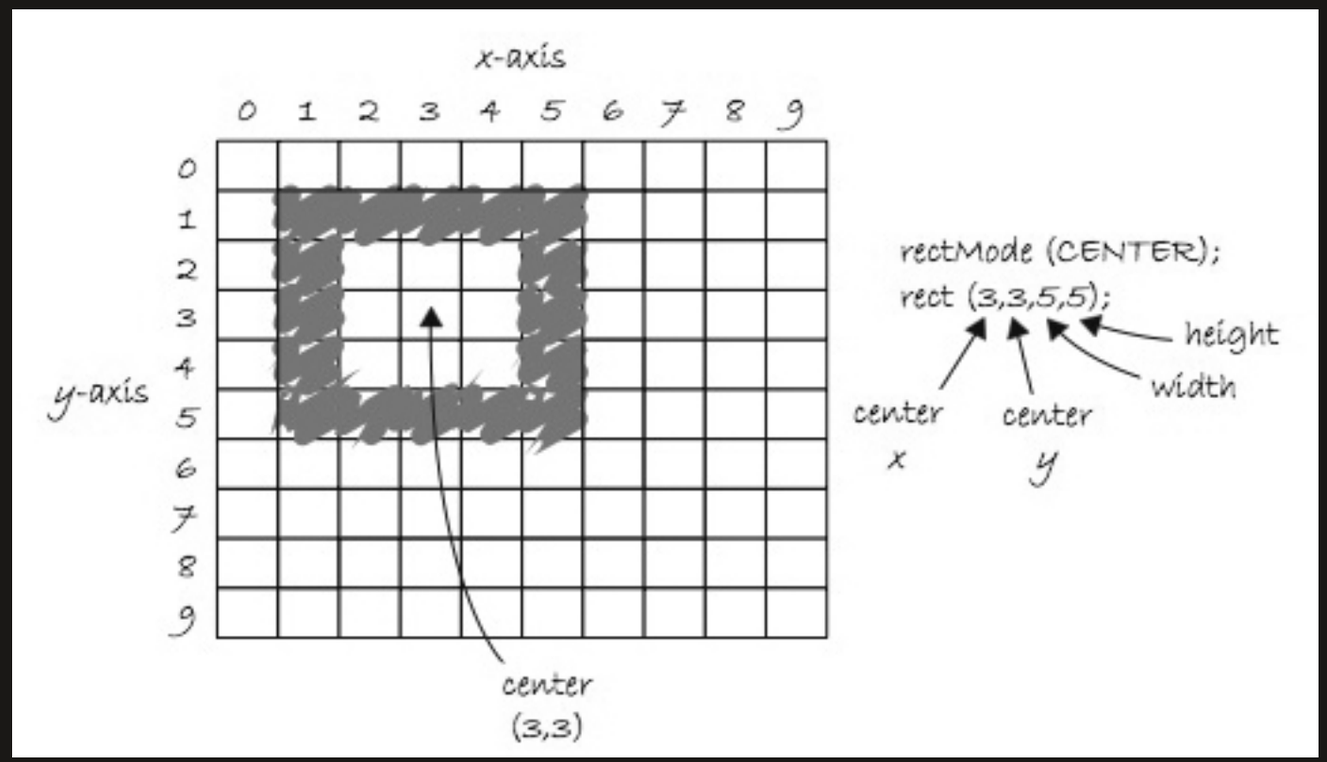


1-2 ► Hello World

Se repérer dans l'espace 2D

- les modes de dessin du rectangle

- le mode CENTER: on définit le centre, puis la largeur, puis la hauteur.

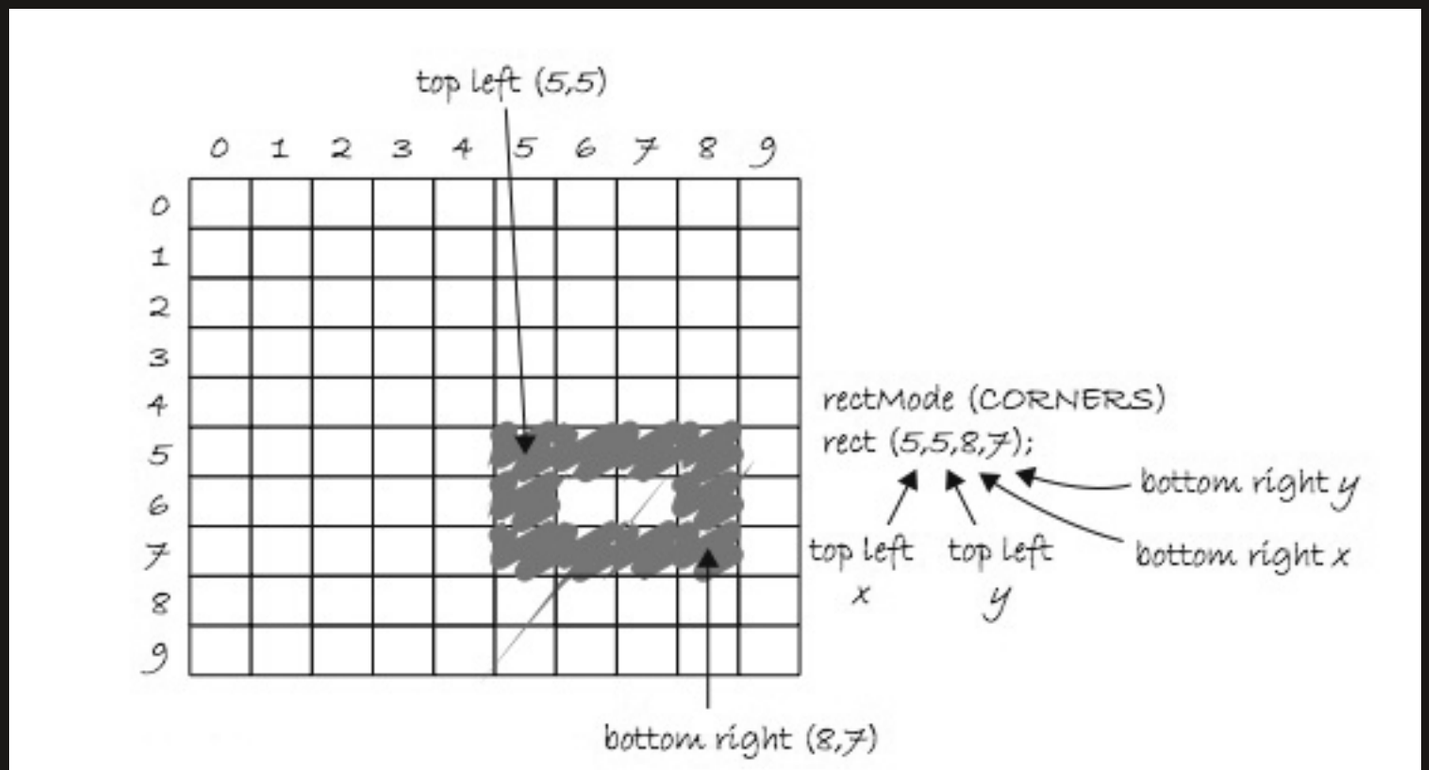


1-2 ► Hello World

Se repérer dans l'espace 2D

- les modes de dessin du rectangle

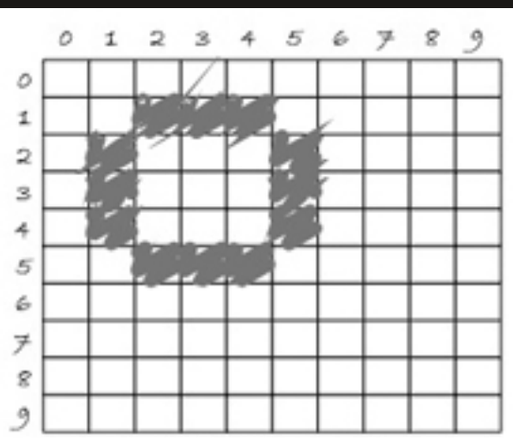
- le mode CORNERS: on définit le coin supérieur gauche, puis le coin inférieur droit.

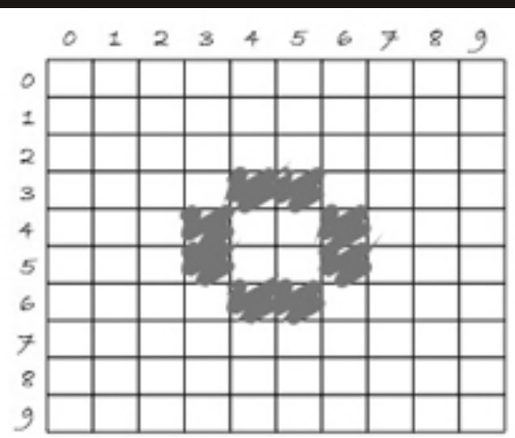


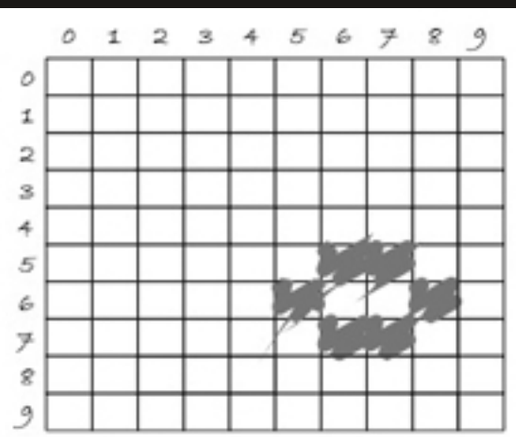
1-2 ► Hello World

Se repérer dans l'espace 2D

- les modes de dessin de l'ellipse sont les mêmes que pour le rectangle, sauf que le mode CENTER est celui par défaut.







1-2 ► Hello World

Se repérer dans l'espace 2D

- Comment savoir tout cela ?
-> Il faut aller voir dans la référence des fonctions de Processing,
sur le site ou dans le menu du logiciel.

1-2 ► Hello World

Changer les couleurs

- les niveaux de gris sont définis sur 256 niveaux.
0 = noir
255 = blanc

- chaque forme que nous avons dessiné peut avoir deux paramètres «visuels»
`stroke()` -> le contour
`fill()` -> le remplissage

- on ajoute le paramètre de couleur entre parenthèses, `fill(210);`

1-2 ► Hello World

Changer les couleurs

- les modes de dessin se définissent AVANT de dessiner la forme:

```
//fond gris
size(250,250);
background (125);

// contour noir et fond blanc
stroke(0);
fill(255);
rect(20,20,40,80);

// contour blanc et fond noir
stroke(255);
fill(0);
rect(80,20,40,80);
```

1-2 ► Hello World

Changer les couleurs

- pas de contours: `noStroke();`

- pas de remplissage: `noFill();`

- évidemment, attention de ne pas utiliser les deux en même temps! Rien ne serait affiché.

1-2 ► Hello World

Changer les couleurs

- le mode RVB
-> chaque couleur, en mode RVB, est une addition de rouge, de vert et de bleu, dans des proportions différentes.
-> comme en niveaux de gris, il y a 256 variations possibles pour chaque couleur.

Un rouge lumineux: `fill (255,0,0);`
Un rouge sombre: `fill (40,20,16);`
Un jaune: `fill (255,255,0);`

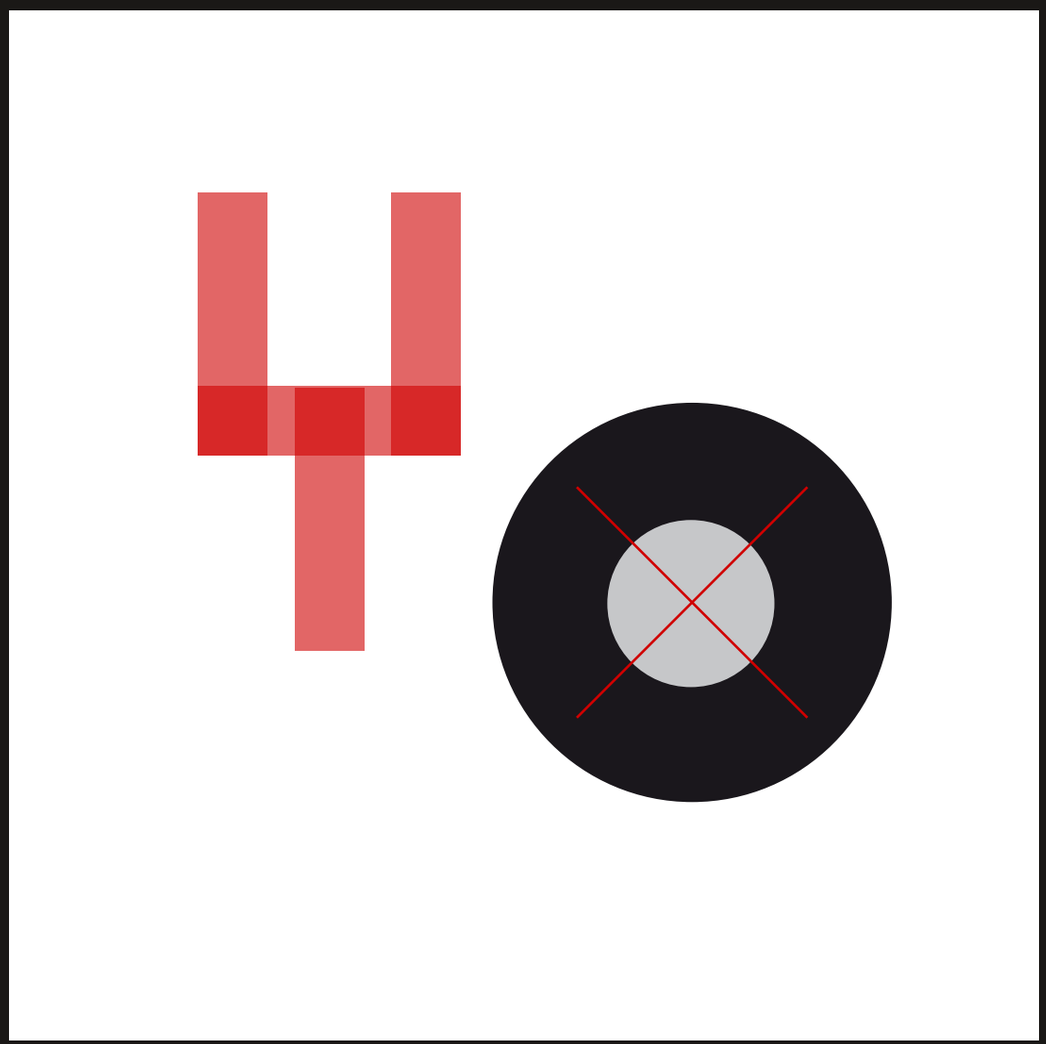
- un sélecteur de couleur dans Processing nous facilitera le choix des couleurs.

- un quatrième paramètre peut être ajouté, il définira une valeur d'alpha, de transparence
0 -> transparence totale , 127 -> transparence à 50% , 255-> opacité totale
`fill (125,20,125,200);`

1-2 ► Hello World

Exercice

- reproduire ce message de paix, ou un truc ressemblant:



1-2 ► Hello World

Publier son sketch

- lorsque le sketch est terminé, on peut désormais le publier,
comme une application autonome, que l'on pourra lancer depuis le bureau
(menu File>Export Application)

1-2 ► INTERACTION

- qui dit interaction, dit ajout d'une **DIMENSION TEMPORELLE**

- intéressons nous donc au flux d'un programme tel que nous allons le créer avec P5

- prenons l'exemple d'un jeu vidéo

-> le jeu commence: vous donnez un nom à votre personnage, le score commence à 0, et vous commencez au niveau 1. C'est l'**INITIALISATION** du programme.

-> une fois que ces conditions sont initialisées, vous commencez à jouer. L'ordinateur vérifie à chaque instant la position de la souris, calcule le comportement approprié des joueurs, et met à jour l'affichage à l'écran. Ce cycle de calculs et de dessins tourne en boucle, idéalement 30 fois par seconde, ou plus, pour une animation fluide. C'est la phase de **DESSIN** du programme.

1-2 ► INTERACTION

- Résumons :

Étape 1 - Définition des conditions de départ pour le programme (1 fois).

Étape 2 - Faire quelquechose encore, et encore, et encore, et encore... jusqu'à ce que le programme quitte.

Si je pars faire un footing:

Étape 1 - Mettre ses baskets et son short. On ne fait cela qu'une fois...

Étape 2 - Un pied devant l'autre, puis l'autre devant celui d'avant, et ainsi de suite...

1-2 ► INTERACTION

- Dans Processing, nous allons séparer ces deux étapes à l'aide de **BLOCS DE CODE**.

- un bloc de code, c'est simplement du code inclus dans des accolades.

```
{
un bloc de code;
on peut en mettre autant que l'on veut;
tout ça est regroupé dans ces accolades;
}
```

on peut les imbriquer

```
{
un bloc de code;
      {
      un autre bloc de code à l'intérieur;
      }
}
```

1-2 ► INTERACTION

```
void setup () {
      // l'initialisation
      // étape 1
      // étape 2
      // étape 3
}
```

effectué une seule fois

lorsque l'éxecution du setup() est terminée, P5 passe au draw()

```
void draw () {
      // le code qui va se répéter
      // étape 1
      // étape 2
      // étape 3
}
```

boucle jusqu'à ce qu'on quitte le programme

1-2 ► INTERACTION

Mis en application

Nous voulons attacher un carré noir de 10 pixels de côté au curseur de notre souris.
En bon français informatique, nous voulons que les coordonnées X et Y du centre de notre carré soient toujours égales à celles de notre curseur.
Quand je dis TOUJOURS, je signifie que cette information doit être mise à jour dans le draw().

J'aurai donc, dans le draw(), une ligne qui créera un carré selon cette formule:

```
rectMode(CENTER);
rect( la position X de la souris, la position Y de la souris, 10,10);
```

Il se trouve qu'en langage Processing, la position de la souris est renvoyée par les fonctions mouseX et mouseY.

1-2 ► INTERACTION

```
void setup(){
	size(200,200);
}

void draw(){
	background (255);

	noStroke();
	fill(0);
	rectMode(CENTER);
	rect (mouseX, mouseY, 10,10);

}
```

Faisons maintenant un test en déplaçant le background() dans le setup().

1-2 ► INTERACTION

- Reprenons notre YO en lui faisant suivre les mouvements de la souris...

- Arrangeons nous, maintenant, pour que les couleurs changent en fonction de la position de la souris... (idée: travailler dans un sketch de 255 pixels de côtés...)

1-2 ► INTERACTION

- Nous avons vu que mouseX et mouseY renvoyaient les valeurs X et Y de notre curseur.

- Nous avons également à disposition les fonctions pmouseX et pmouseY .
Elle renvoient la position de la souris **À L'IMAGE PRÉCÉDENTE**.

- avec ces deux nouvelles fonctions, nous pouvons facilement créer une application de dessin...

```
void setup(){
	size (200,200);
	background (255);
	//lissage des graphiques
	smooth();
}

void draw(){
	stroke(0);
	line (pmouseX, pmouseY, mouseX, mouseY);
}
```

1-2 ► INTERACTION

Détecter les clic de souris et les touches du clavier

- nous aurons besoin de deux nouvelles fonctions, qui détecterons ces **événements:**
mousePressed() -> détecte les clics de souris
keyPressed() -> détecte un appui sur une touche

- ce que nous allons mettre entre les accolades sera executé une fois et une seule, à chaque fois que l'évenement sera exécuté.

1-2 ► INTERACTION

```
void setup(){
        size (200,200);
        background (255);
}

void draw(){
}

void mousePressed(){
        stroke(0);
        fill(175);
        rectMode(CENTER);
        rect (mouseX, mouseY, 16,16);
}

void keyPressed(){
        background (255);
}
```

1-2 ► INTERACTION

- une dernière fonction avant de clore ce chapitre:
frameRate();

- cette fonction, à poser dans le setup(), sert à définir **UNE VITESSE DE RAFRAÎCHISSEMENT**.
- la boucle draw() sera donc executée autant de fois par secondes que le chiffre indiqué entre parenthèses.

- c'est une valeur maximum. Si vous essayez de dessiner un million de triangles à chaque image, elle mettra beaucoup plus de temps que prévu pour être dessinée.

- si frameRate() n'est pas défini, sa valeur par défaut est de 60 images par secondes.

```
void setup(){
      size (200,200);
      smooth();
      frameRate (30);
}
```

1-2 ► VARIABLES

Qu'est ce qu'une variable?

- Une variable, c'est un panier. Vous mettez quelque chose à l'intérieur, vous l'amenez quelque part, vous vous servez de ce que vous avez mis à l'intérieur.

- Une variable, c'est une consigne atomatique. Vous stockez quelque chose à l'intérieur, et vous venez le chercher quand vous en avez besoin.

- Une variable, c'est un post-it, il est noté dessus «je suis une variable, écrivez une information sur moi».

Daniel Shiffman

1-2 ► VARIABLES

Qu'est ce qu'une variable?

- L'ordinateur a une mémoire. Il va donc pouvoir retenir les informations dont on va avoir besoin. Une variable, c'est un nom qui pointera vers un espace de la mémoire de l'ordinateur, là où la variable sera stockée.

- Une variable pourra contenir toutes les informations dont nous aurons besoin pour dessiner des formes, par exemple: couleur, taille, position...

- Une variable est une référence textuelle a une valeur, qui pourra changer au fur et à mesure du programme.

Ex: lors d'un jeu de scrabble entre Jane et Billy.
Au début du jeu, les variables scoreBilly = 0 et scoreJane = 0.
À la fin du jeu, scoreBilly = 124 et scoreJane = 136.

- Dans cette exemple, la variable scoreBilly a un **NOM** (scoreBilly) et une **VALEUR**, évolutive. Cette valeur est d'un certain **TYPE** (un nombre).

1-2 ► VARIABLES

Déclarer et initialiser des variables

- Les variables sont déclarées en donnant d'abord le **TYPE**, puis son **NOM**
(c'est à dire que l'on réserve une partie de la mémoire de l'ordinateur,
et on lui donne un nom pour la retrouver).

- Le nom des variables doit commencer par une lettre. Il peut contenir un chiffre, mais pas en premier.

- Pas de ponctuation, pas de caractères spéciaux, sauf l'underscore _
Une convention de nommage fait que l'on commence toujours les variables par une minuscule,
et on sépare les mots avec des majuscules.
maSuperBelleVariable

1-2 ► VARIABLES

Déclarer et initialiser des variables

- Le type est la sorte de données qui va être stockée dans la variable. Cela peut être un nombre entier, un décimal, un ou des caractères.

Les types que l'on utilisera le plus souvent:

• Nombre entiers: 0, 1, 2, -125 ...
ils sont de type «integers» : on notera int dans Processing

• Nombres décimaux: 3,14159 ou 2,5 ou -9,95
ils sont de type «floating point values»: float

• Caractères: a, b ou c seront de type char
un mot (plusieurs caractères) sera de type «string», une chaîne de caractères: string

1-2 ► VARIABLES

Déclarer et initialiser des variables

Une fois qu'une variable est déclarée (et seulement une fois que c'est le cas), on peut lui assigner une valeur.

```
int compteur;
compteur = 50;
```

On peut être plus concis:

```
int compteur = 50;
```

1-2 ► VARIABLES

Déclarer et initialiser des variables

Une fois qu'une variable est déclarée (et seulement une fois que c'est le cas), on peut lui assigner une valeur.

```
int compteur;
compteur = 50;
```

On peut être plus concis:

```
int compteur = 50;
```

1-2 ► VARIABLES

Exemples de déclarations de variables

```
int compteur = 0;
char lettre = ‘a’;
boolean happy = false;
float x = 4.0;
y = x +5.2;
float z = x*y + 15.0;
```

1-2 ► VARIABLES

Exemples de déclarations de variables

Commençons par cet exemple simple:

```
void setup(){
      size (200,200);
}

      void draw(){
            background(255);
            stroke (0);
            fill(175);
            ellipse (100, 100,50,50);

      }
```

1-2 ► VARIABLES

Exemples de déclarations de variables

Nous déclarons deux variables qui viennent remplacer les valeurs que nous avions posées.

```
int circleX = 100;
int circleY = 100;

void setup(){
	size (200,200);
}

void draw(){
	background(255);
	stroke (0);
	fill(175);
	ellipse (circleX, circleY,50,50);

}
```

1-2 ► VARIABLES

Exemples de déclarations de variables

Faisons maintenant varier une des valeurs au fur et à mesure du draw().

```
int circleX = 0;
int circleY = 100;

void setup(){
	size (200,200);
}

void draw(){
	background(255);
	stroke (0);
	fill(175);
	ellipse (circleX, circleY,50,50);
	circleX = circleX + 1;
}
```

1-2 ► VARIABLES

Exemples de déclarations de variables

Que modifier dans le code précédent pour que le disque ne se déplace plus mais grossisse?

1-2 ► VARIABLES

Les variables système

Ce sont des variables prédéfinies qui n'on pas besoin d'être déclarées avant d'être utilisées.
Les plus courantes:

mouseX et mouseY
width -> la largeur de la fenêtre du sketch
height -> la hauteur de la fenêtre du sketch
frameCount -> le nombre d'images affichées par Processing depuis le début du sketch
frameRate - > fréquence de rafraichissement
screen.width -> la largeur totale de l'écran
screen.height -> la hauteur totale de l'écran
key -> la dernière touche pressée du clavier
keyPressed -> true ou false? une touche est-elle pressée?
mousePressed -> true ou false? un bouton de la souris est il pressé?
mouseButton -> quel boutton de la souris est pressé?

1-2 ► VARIABLES

Les variables système

```
void setup(){
	size (200,200);
}

void draw(){
	background(100);
	stroke (255);
	fill(175);
	rectMode (CENTER);
	rect (width/2, height/2, mouseX+10, mouseY+10);
}

void keyPressed(){
	println(key);
}
```

1-2 ► VARIABLES

Aléatoire

Nous allons maintenant modifier nos variables avec des chiffre aléatoires.
Commençons par écrire un script simple, nous ajouterons de nouvelles fonctionnalités ensuite:

```
float r = 100;
float g = 150;
float b = 200;
float a = 200;

float diam = 20;
float x = 100;
float y = 100;

void setup(){
        size (200,200);
        background (255);
        smooth();
}
(...la suite...)
```

VARIABLES

Aléatoire

```
void draw(){
        noStroke();
        fill (r,g,b,a);
        ellipse(x,y,diam,diam);
}
```

Vous voyez que tout, ici, est stocké dans des variables...
La fonction `random()` est un fonction spéciale: elle renvoie une valeur.
Elle renvoie une valeur `float` au hasard, entre deux nombres définis.

```
float w = random(1,100);
```

w peut être égal à 63.0, ou 12.0, 25.0...

Pour convertir un `float` en `int`:

```
int w = int(random(1,100));
```

1-2 ► VARIABLES

Aléatoire

Modifions donc le code précédent:

```
float r;
float g;
float b;
float a;

float diam;
float x;
float y;

void setup(){
        size (200,200);
        background (255);
        smooth();
}
(...la suite...)
```

1-2 ► VARIABLES

Aléatoire

```
void draw(){
        r = random(0,255);
        g = random(0,255);
        b = random(0,255);
        a = random(0,255);
        diam = random (0,20);
        x = random(width);
        y = random(height);
        noStroke();
        fill (r,g,b,a);
        ellipse(x,y,diam,diam);
}
```