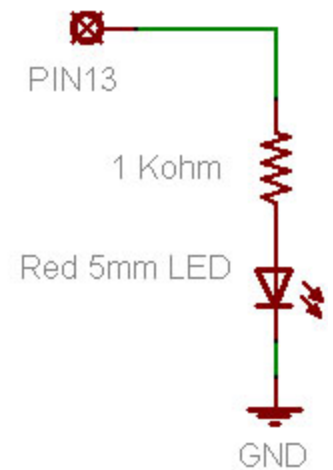
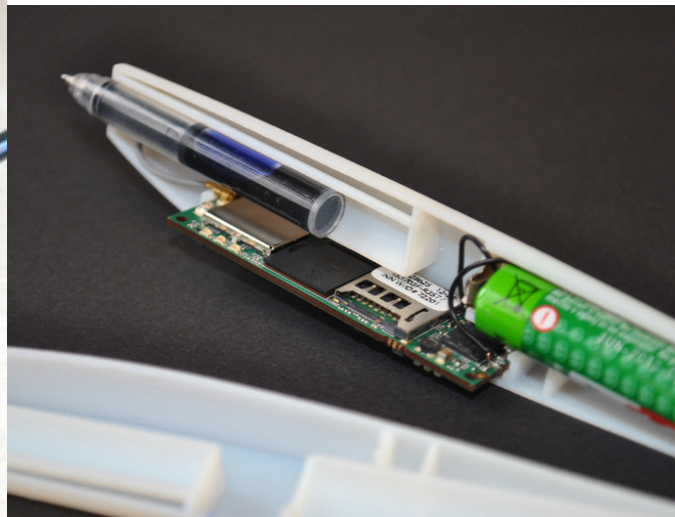
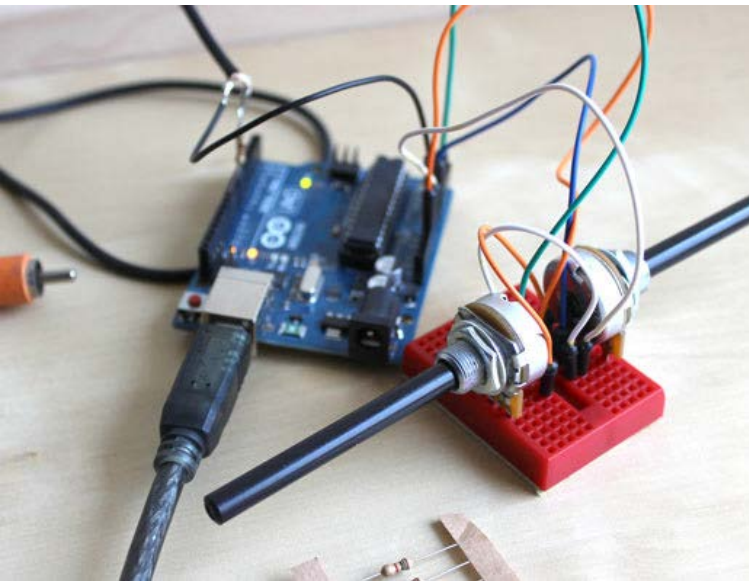
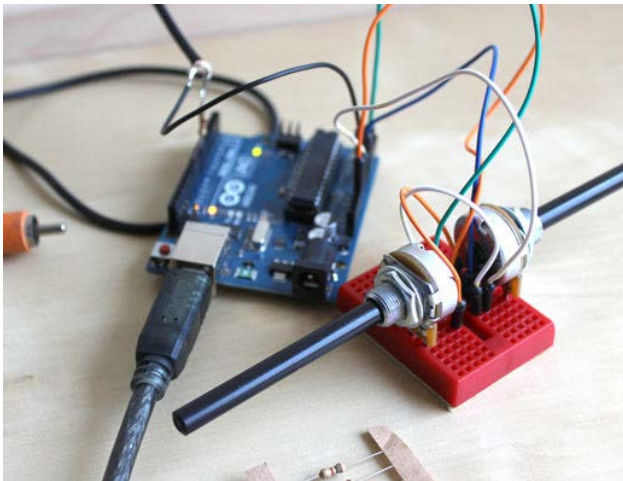


Semaine interdisciplinaire: Arduino , Electronique, Capteurs

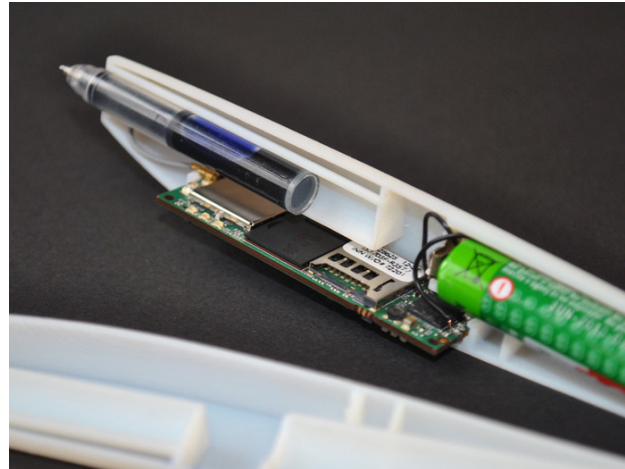


Objectifs de la semaine

- Initiation à la programmation processing et arduino
 - Initiation à l'électronique numérique et analogique
 - Initiation au traitement des données (FFT, diagramme temps fréquence)
- => Comprendre la chaine en la mesure et les données receuillis



Device 1



Device 2

Organisation de la semaine

- **Lundi** : Processing/Arduino => Pong
- **Mardi** : Arduino/Electronics 101 /Fabrication device 1
- **Mercredi** : Scilab/traitement des données (amodsen)
- **Jeudi** : Fabrication device 2
- **Vendredi** : Evaluation

Evaluation (vendredi)

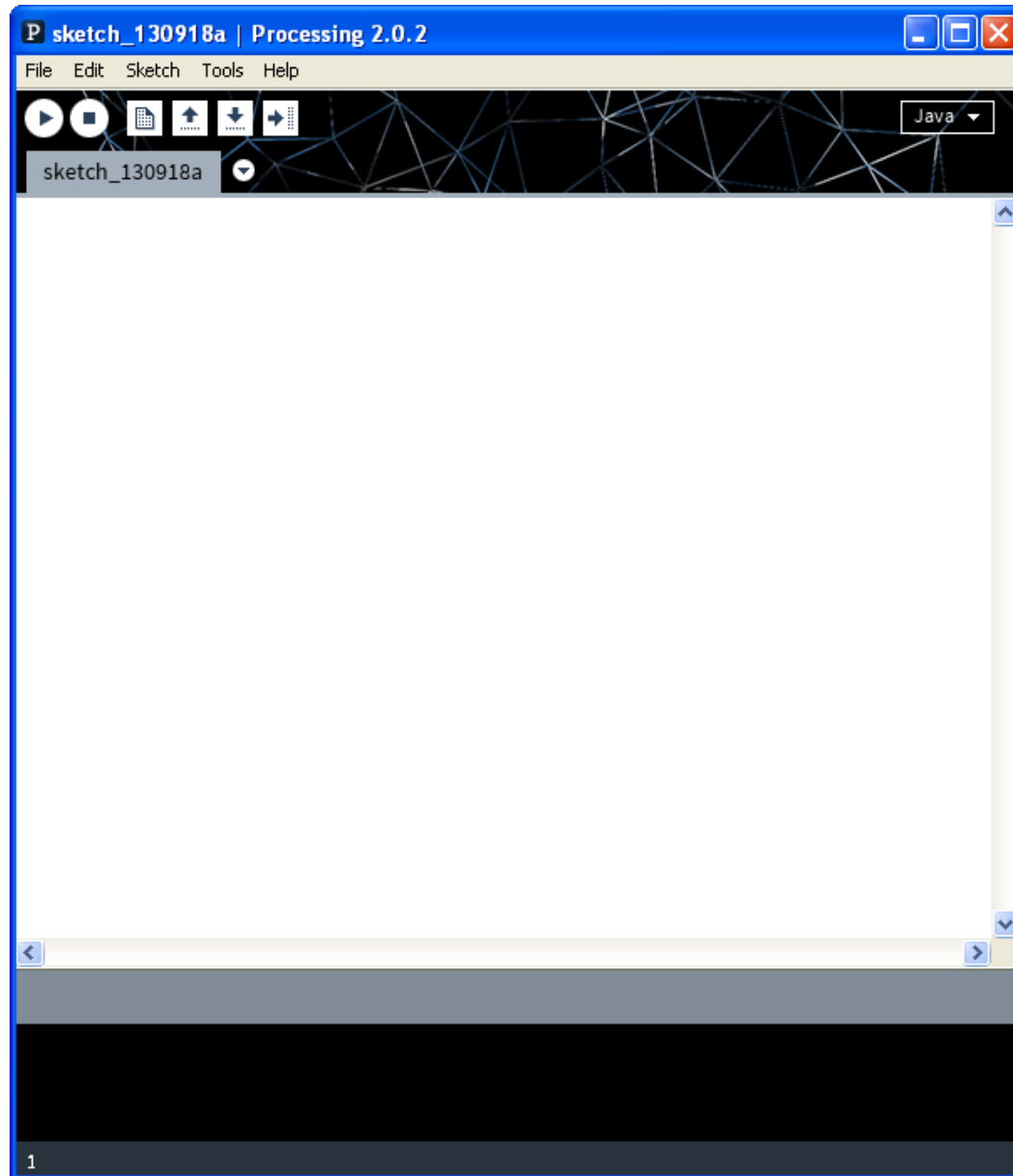
- Comprendre les bases de la programmation processing/arduino
- Savoir lire un schéma électronique
- Savoir lire une datasheet
- Savoir comment debugger un circuit /programme
- Comprendre les bases de l'électronique (Différence courant /tension)

Team up !



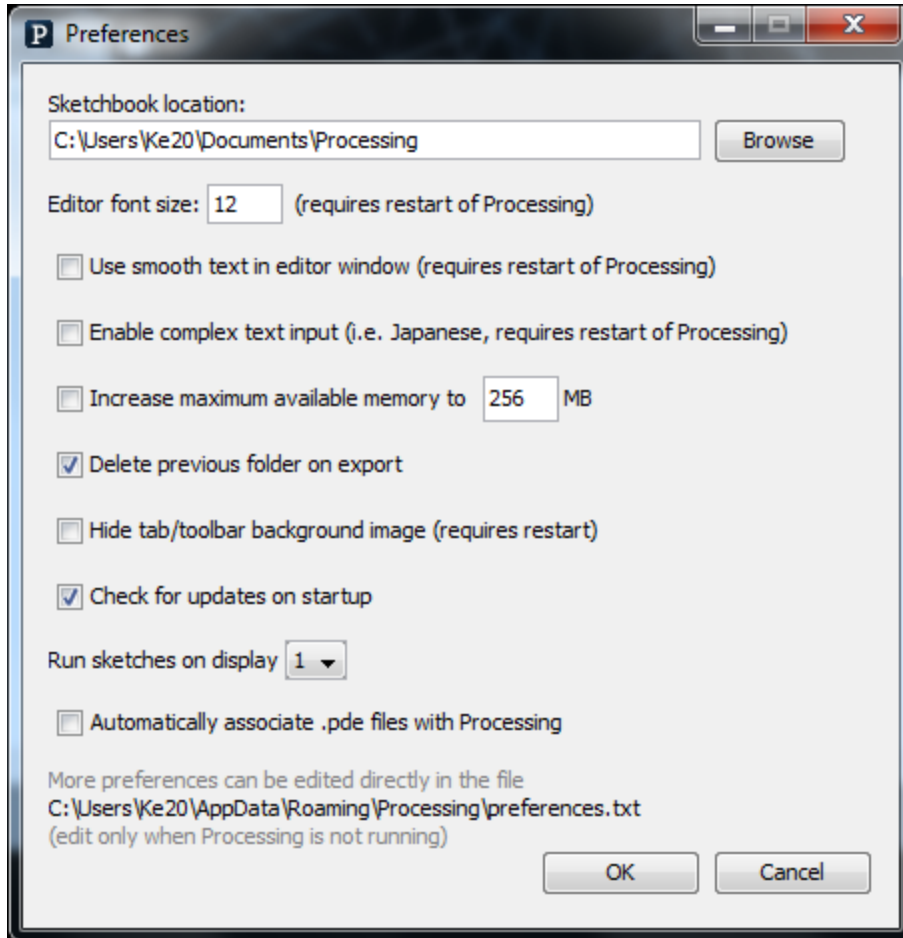
Team of 2 : 1 L1 / 1L2

Processing Installed ?



Processing sketchbook

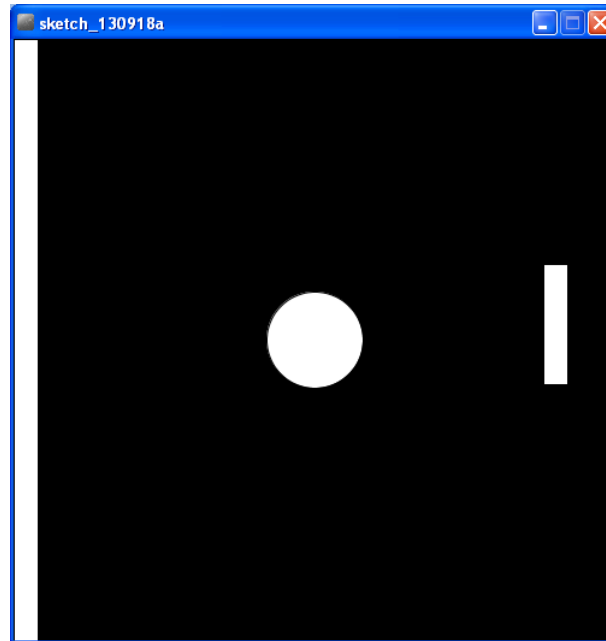
Aller dans file - > Preference - > Sketchbook location



Et changer l'emplacement vers le dossier de processing par ex Desktop/processing/

Part 1- Pong Processing

Challenge Level 0 : Trouver le pseudo code du pong (5min)



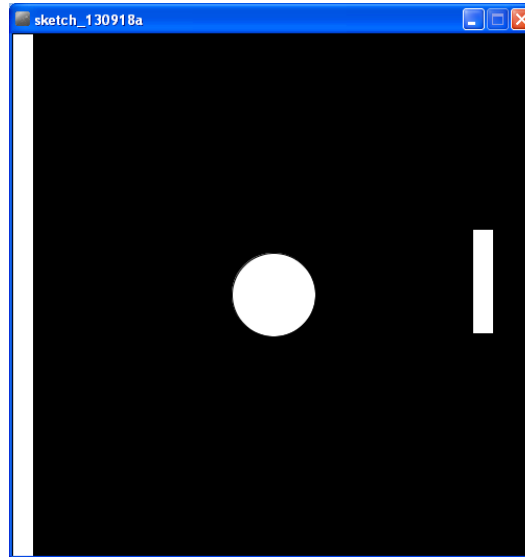
Ex:

Si balle touche la raquette

.....

Tips: il y a entre 4 (malin) et 6 cas

Challenge Level 0 : correction



6 cas

Si la balle touche la raquette Inverser le sens en X de la balle

Si la balle touche le mur de gauche inverser le sens en X de la balle

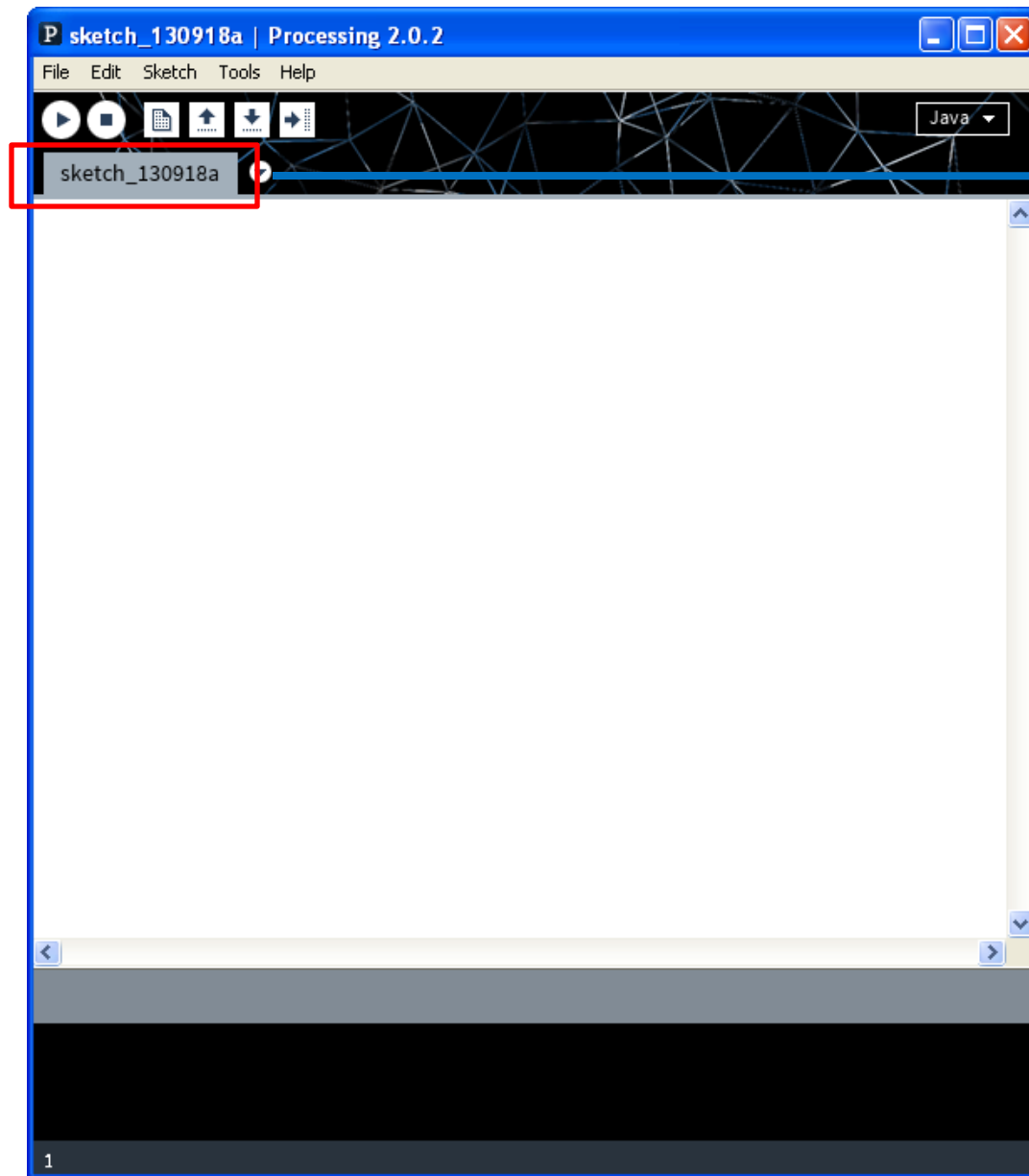
Si la balle dépasse la raquette on a perdu

Si la balle touche le mur du haut rebondir (inverser Y de la balle)

Si la balle touche le mur du bas rebondir (inverser le Y de la balle)

Sinon faire avancer la balle

Level 1- processing basics



Sketch = Program

Level 1- Processing basics



ellipse(x,y,a,b) ;

=> Dessine une elipse

X: origine en x

Y: origine en y

a: largeur

b : hauteur



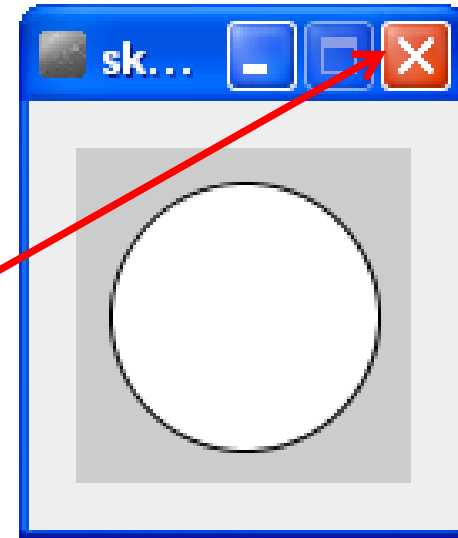
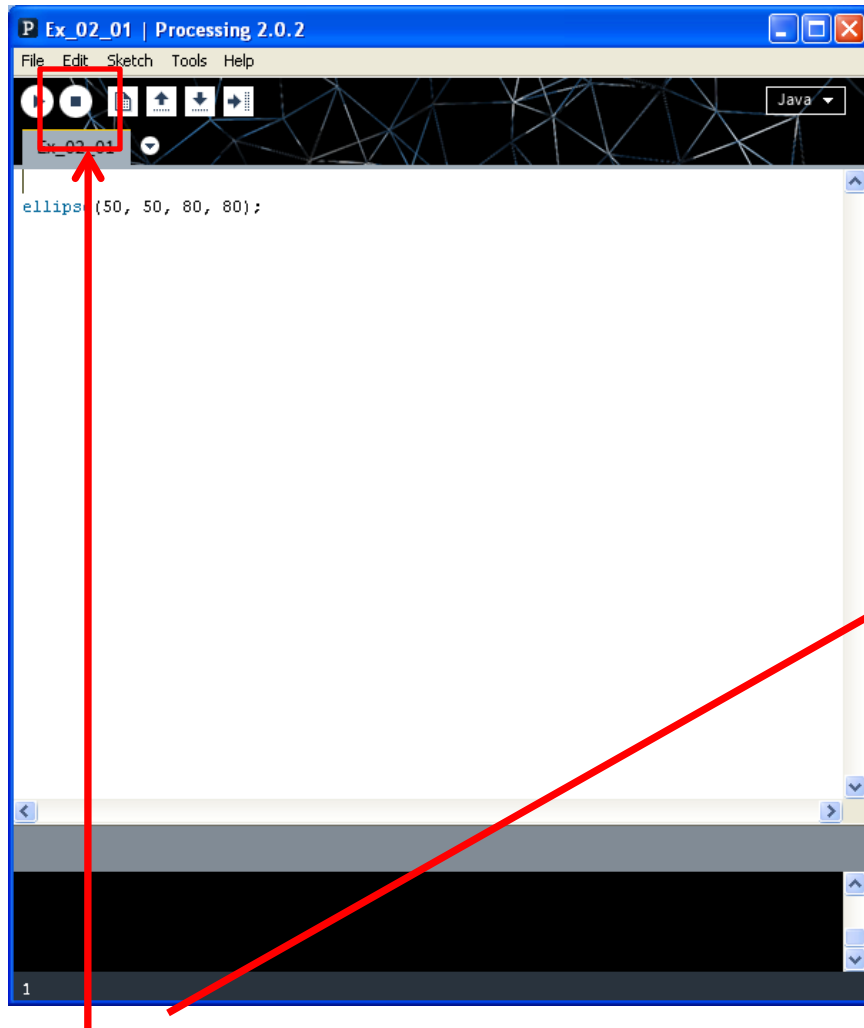
Chaque ligne de commande se termine par un ;

Level1 : Exercice 1

***Bouton executer
=> Demarrer le
programme***

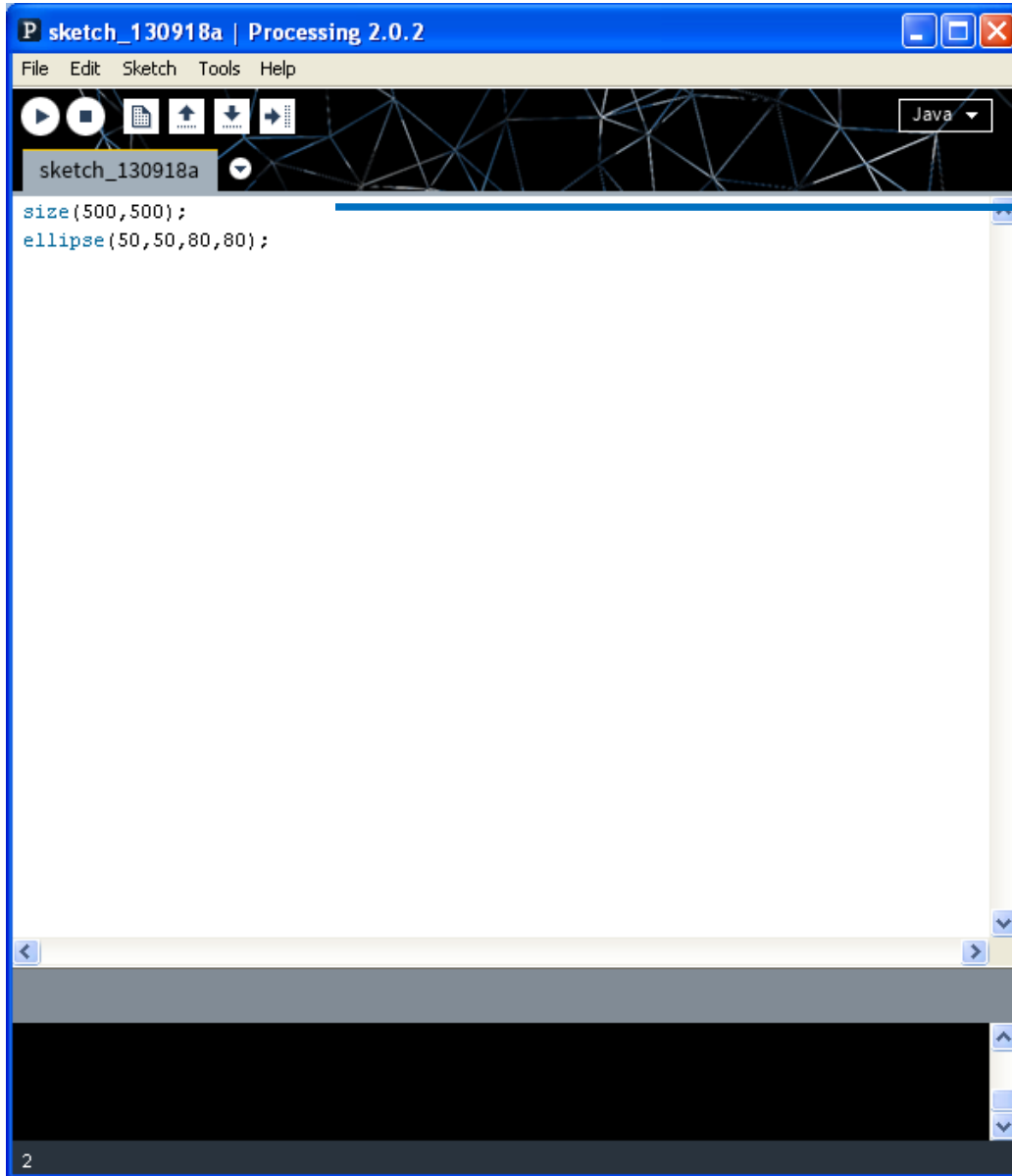


Level 1: Exercice 1



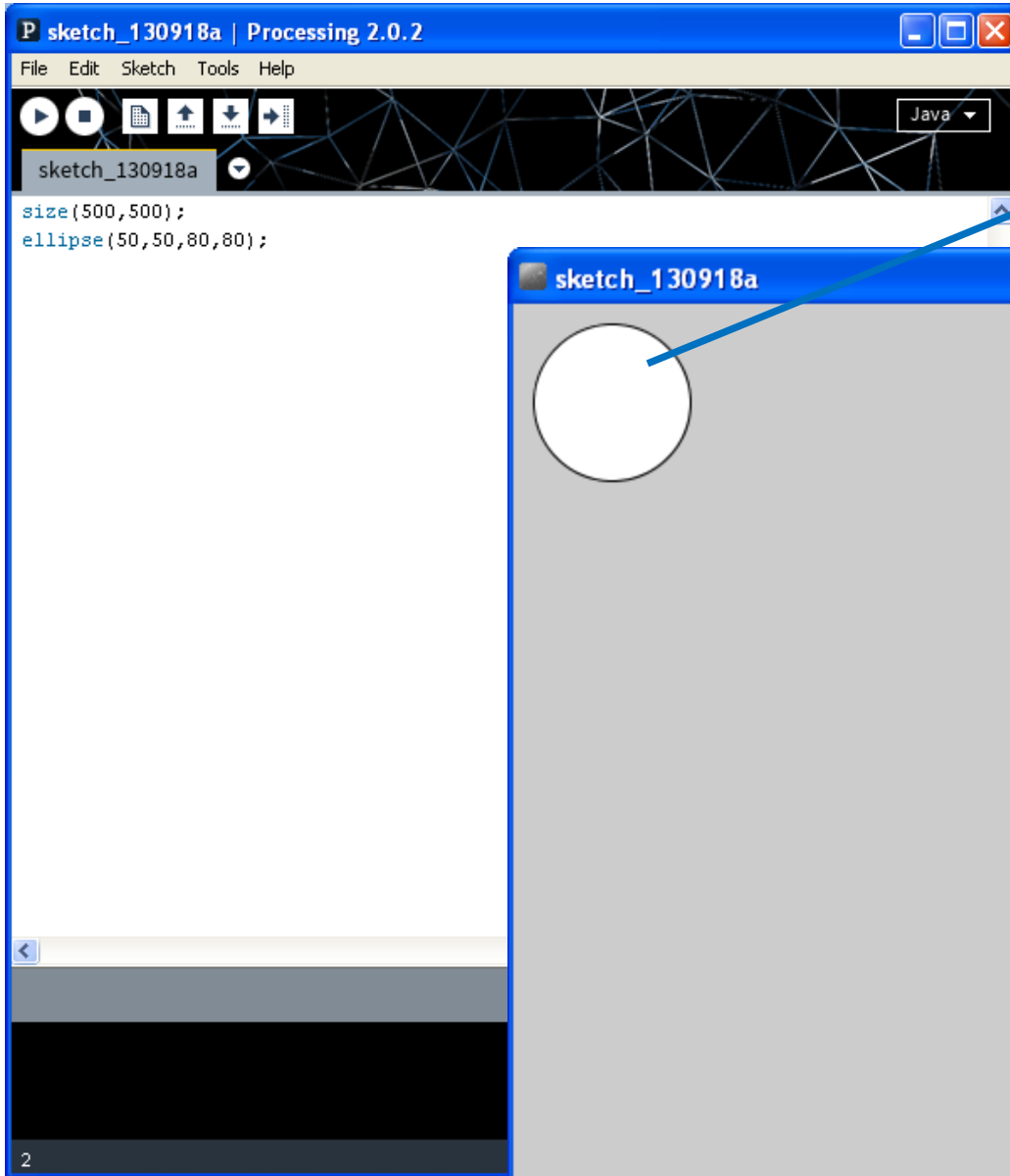
Quitter le programme : croix rouge ou bouton stop ou ESC

Level1: Exercice 2



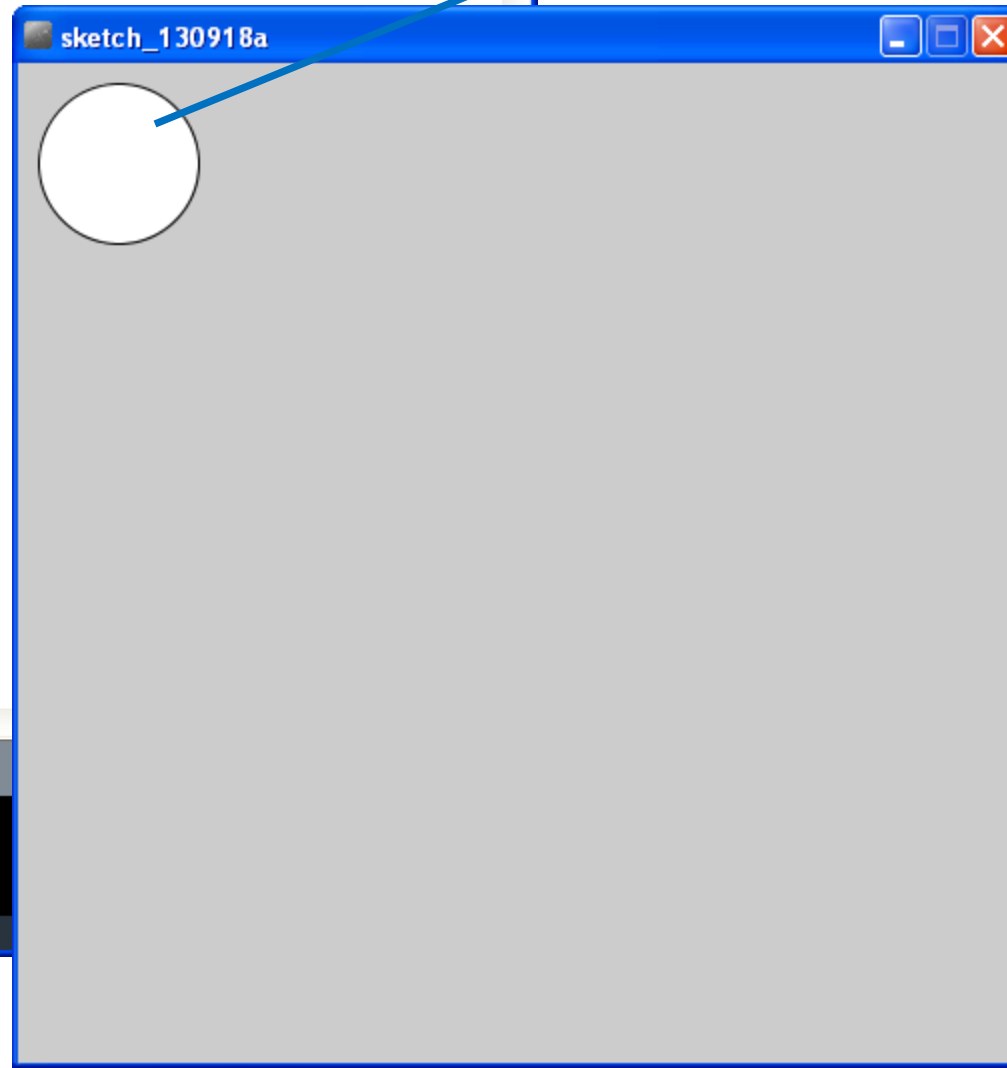
size(a,b) ;
=>Definit la taille de la
fenetre
a: largeur
b: hauteur

Level 1: Exercise 2

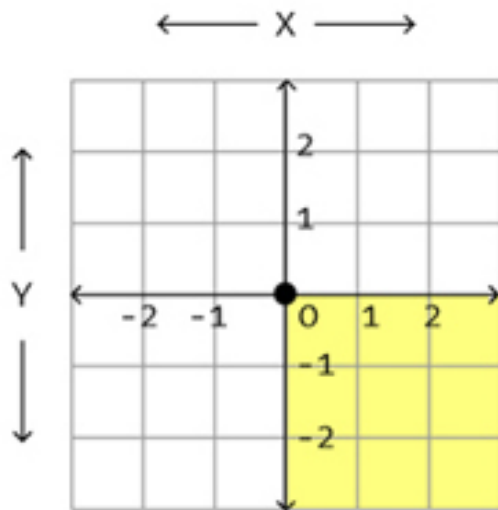


ellipse(50,50,80,80);

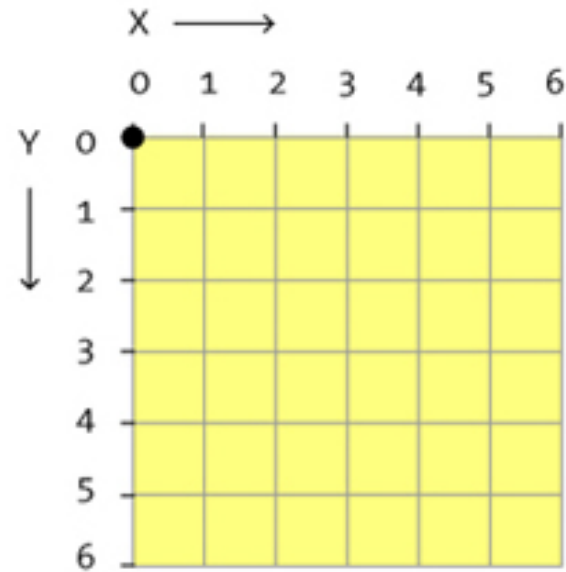
Why ??



Level1 - Processing Coordinates

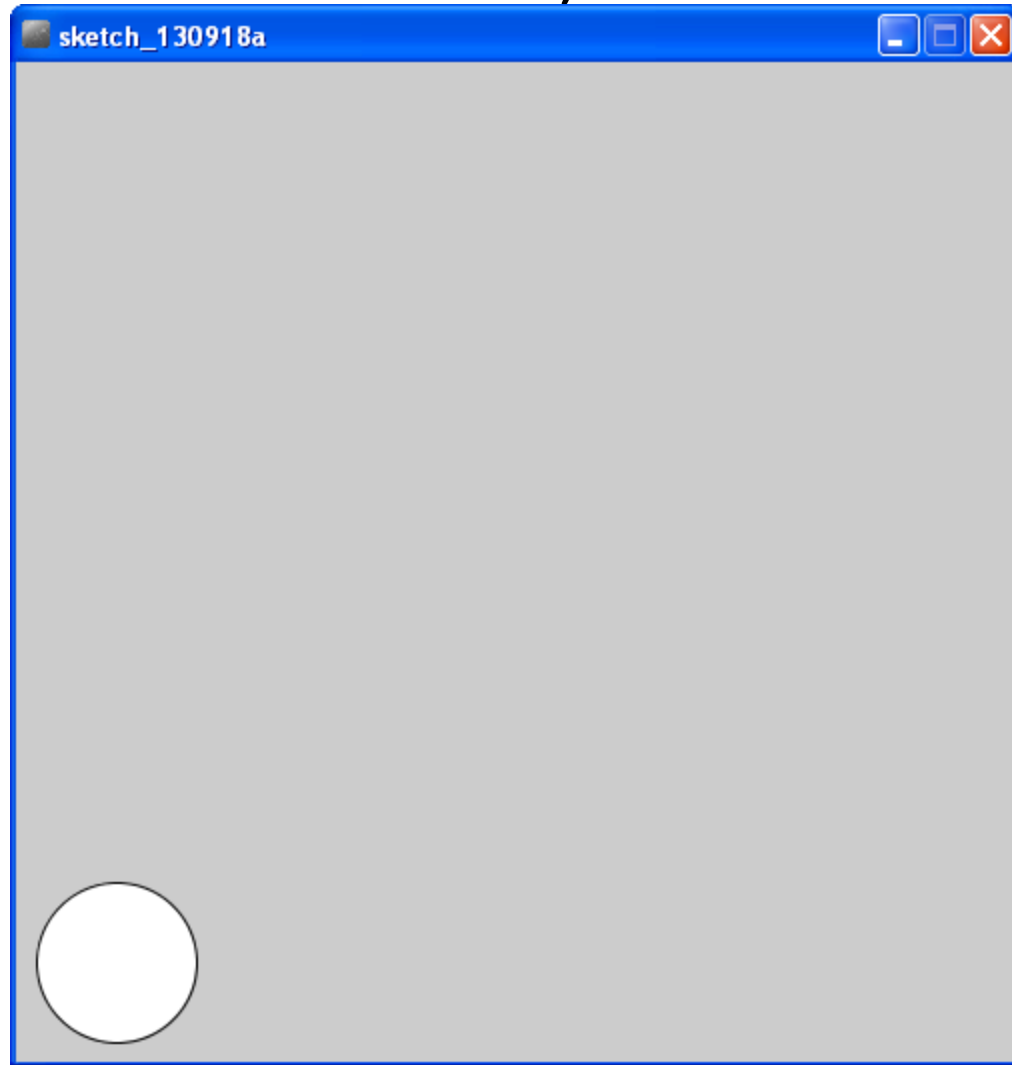


Eighth Grade

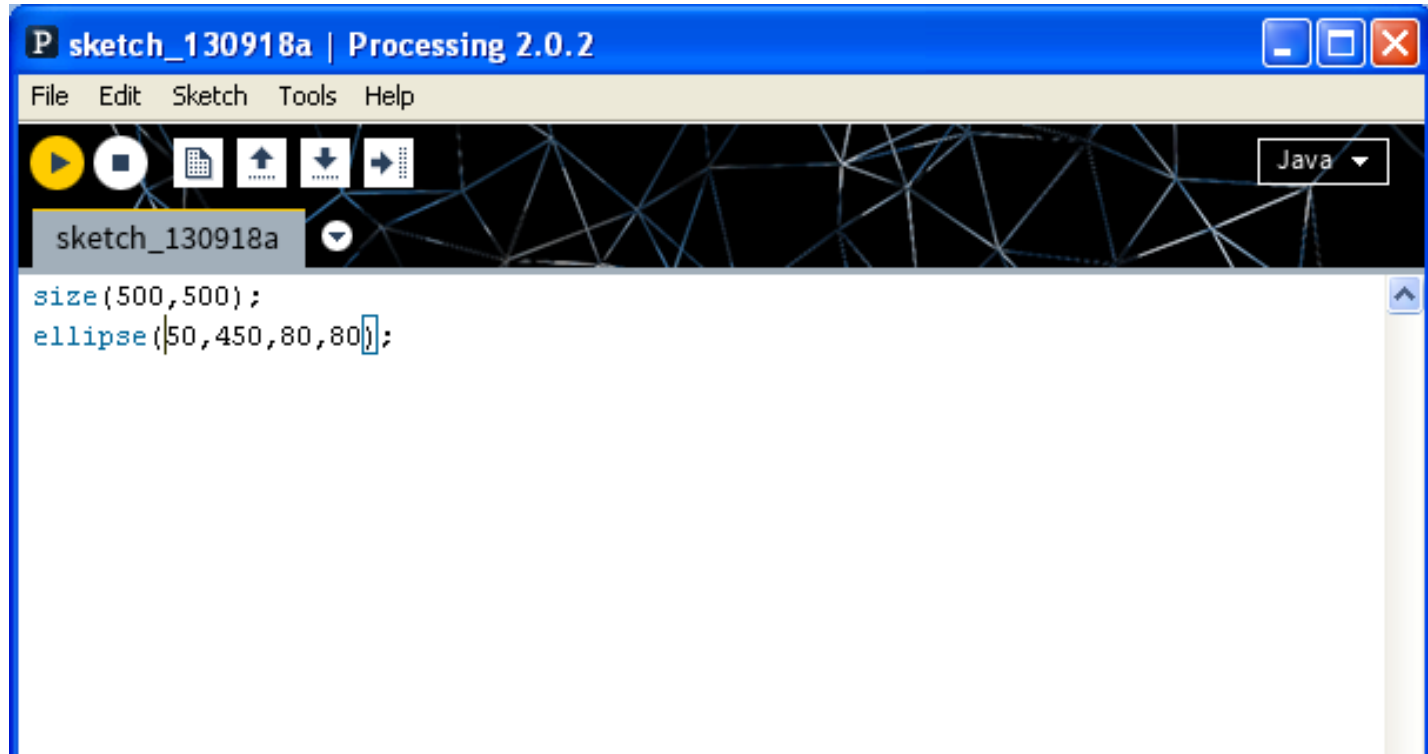


Computer

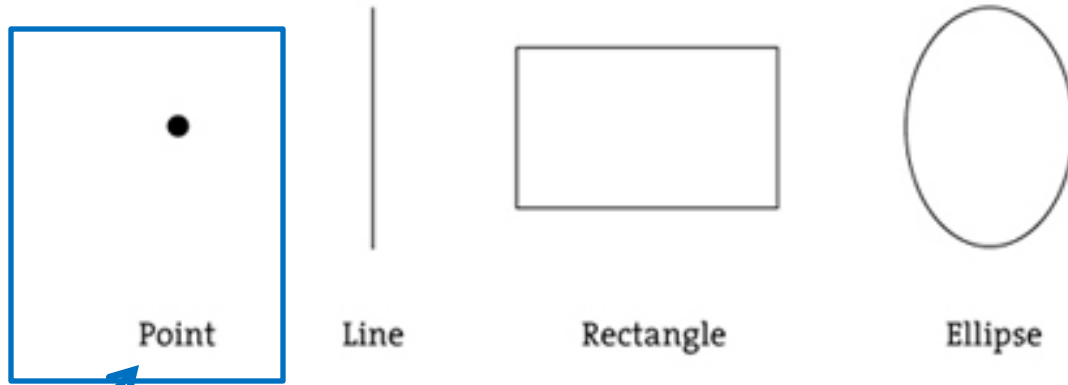
Challenge Level 1 : placer le cercle en bas à gauche (3 min.)



Challenge Level1 : Correction



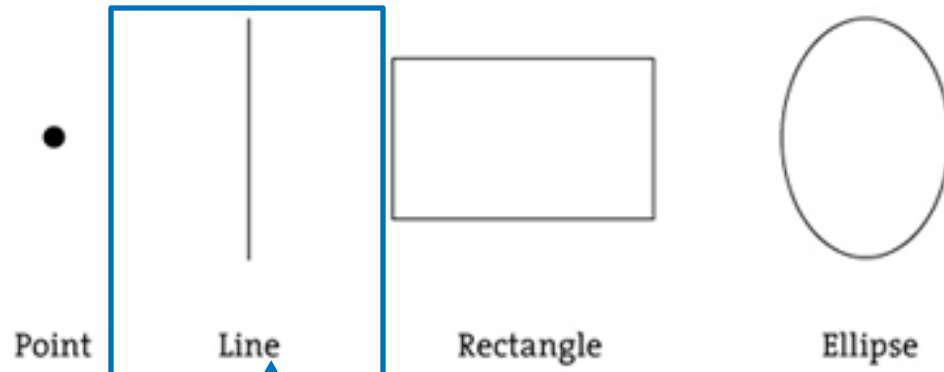
Level 2 : shapes



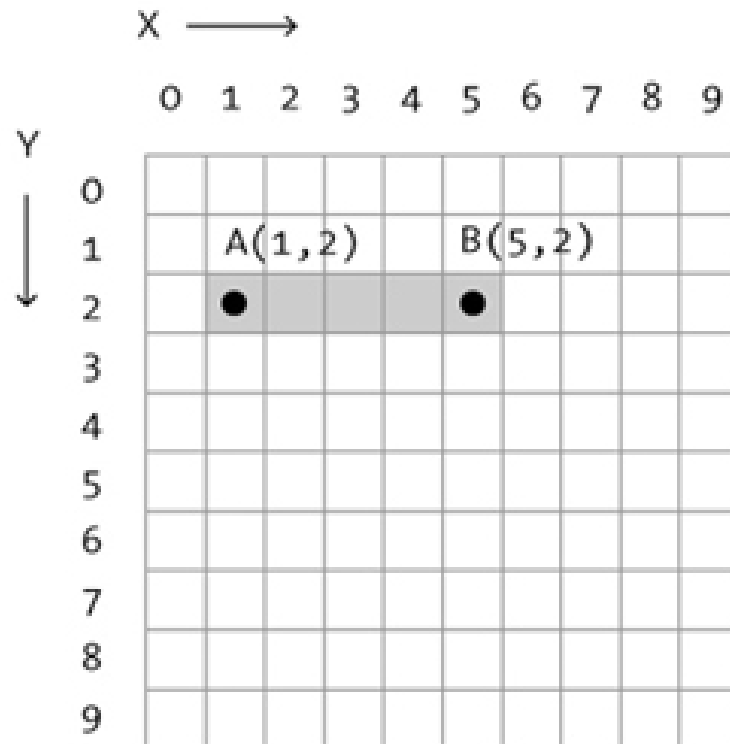
point(x,y) ;

***=> 1 point = 1 pixel =
très petit***

Level 2 : shapes



line(xa,ya,xb,yb) ;



1:

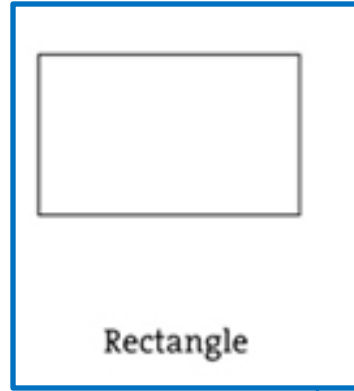
Es

1:

Level 2 : shapes



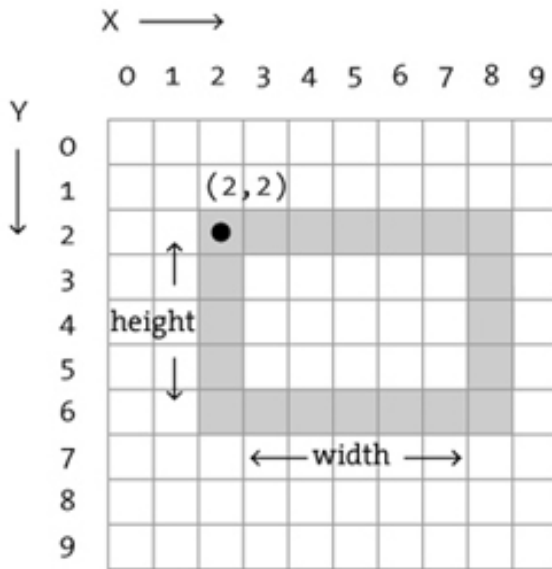
Line



Rectangle



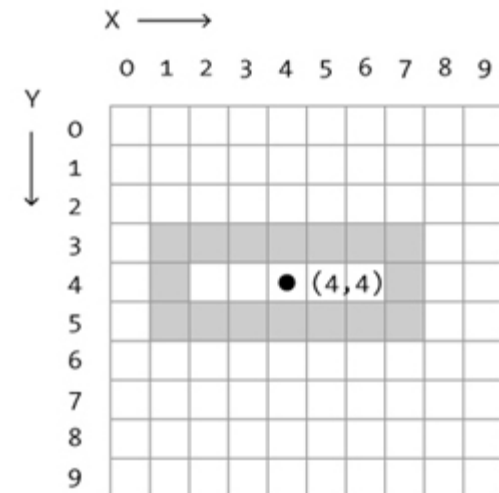
Ellipse



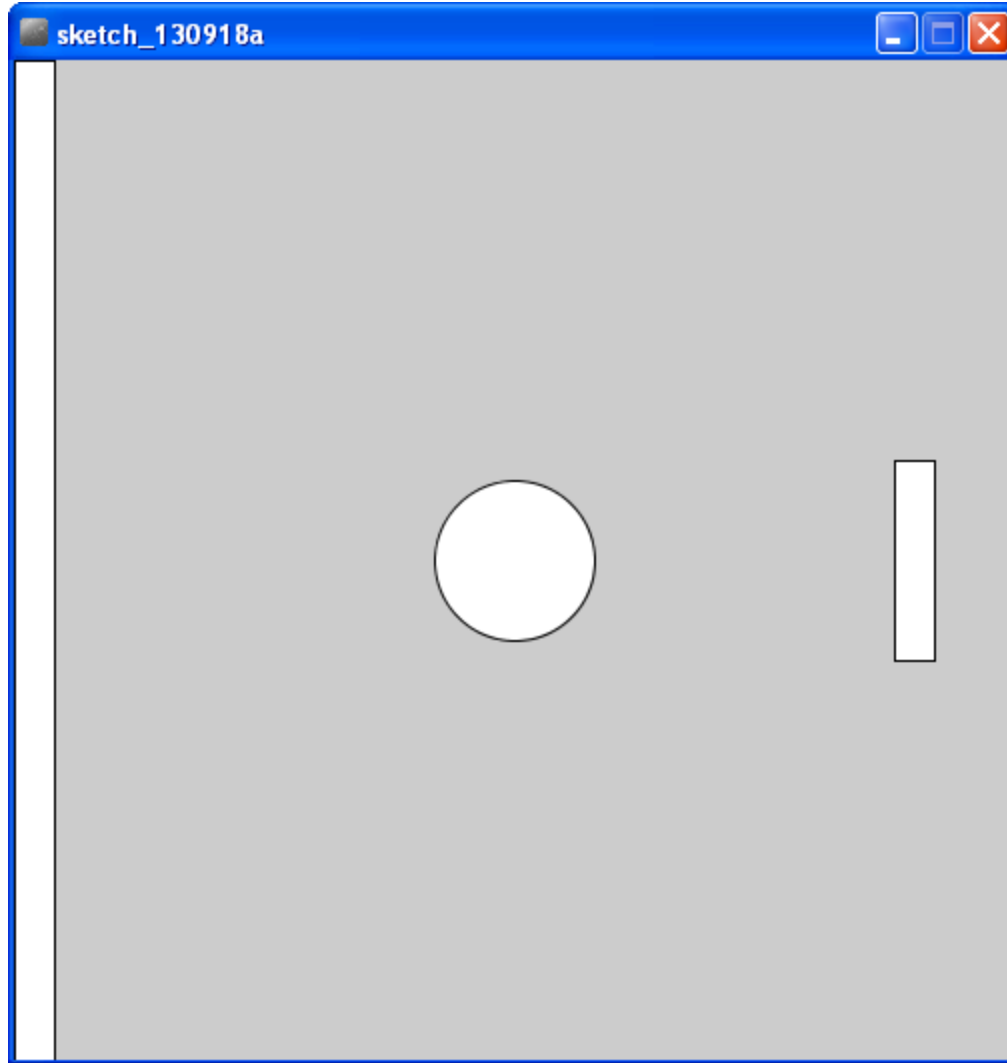
***rect(x,y, largeur,
hauteur);***

***⇒ x, y du coin haut
gauche***

***Sauf si avant on écrit:
rectMode(CENTER);***



Challenge Level 2 : dessiner une balle un mur et une raquette (10 min.)



Toolbox

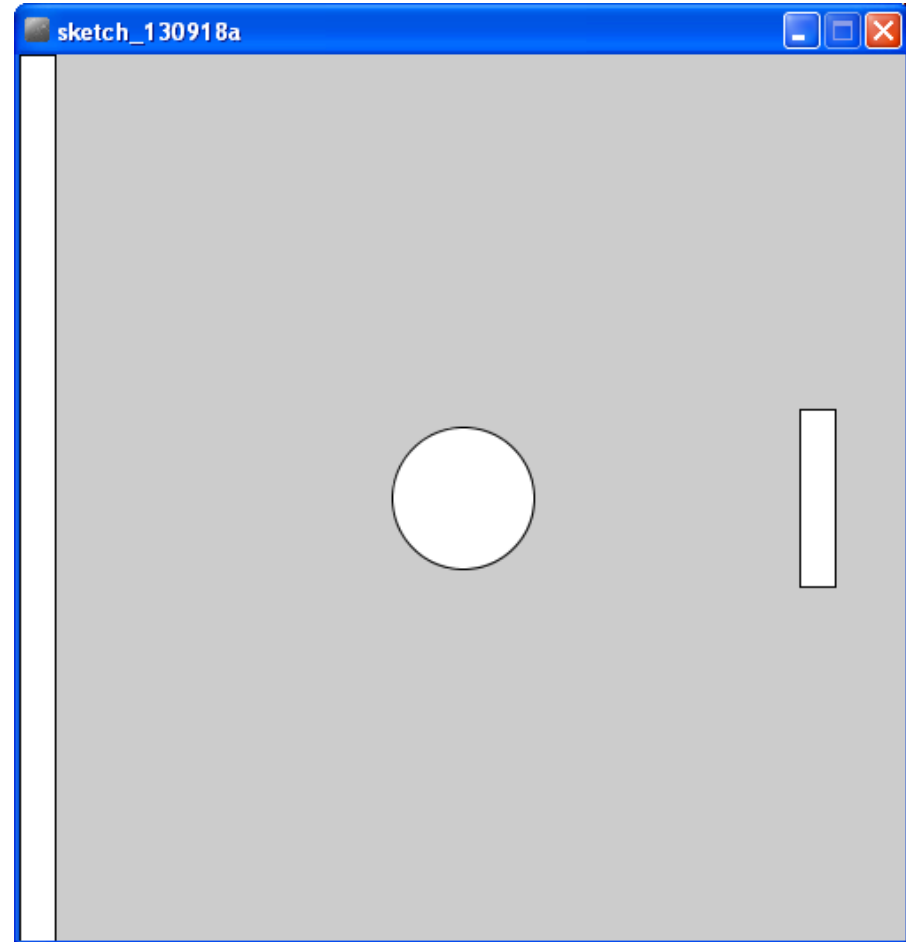
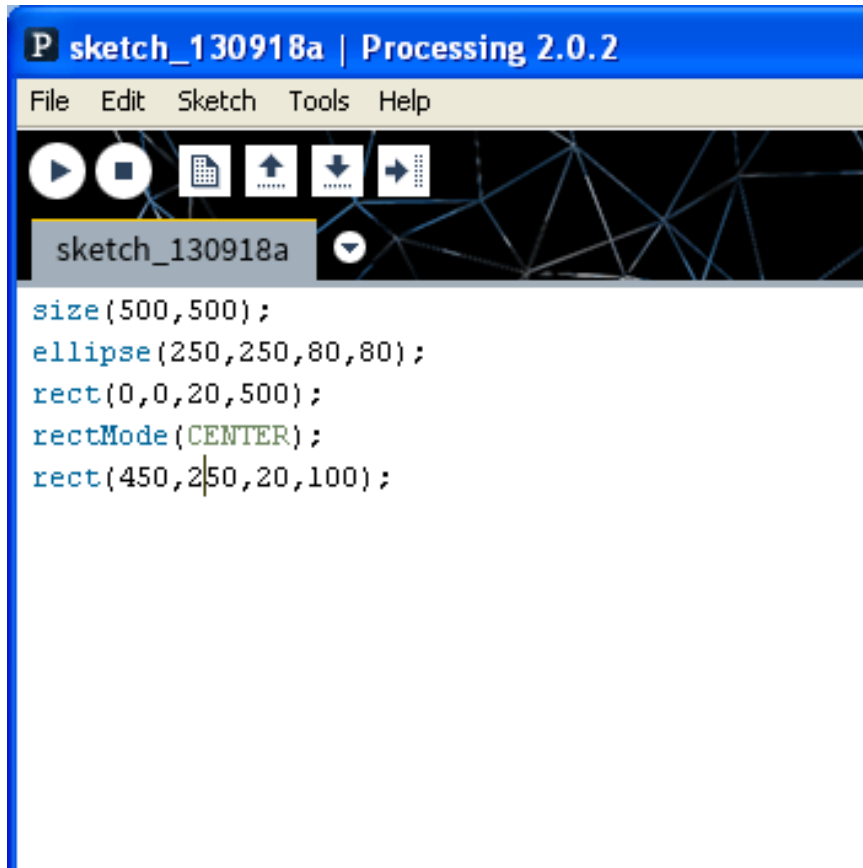
Ellipse

`ellipse(x,y,largeur,hauteur);`

Rectangles :

`rect(x,y,largeur,hauteur);`

Challenge Level 2 : Correction



Level 3 : commentaires et console

// : la ligne est un commentaire et n'est pas exécutée

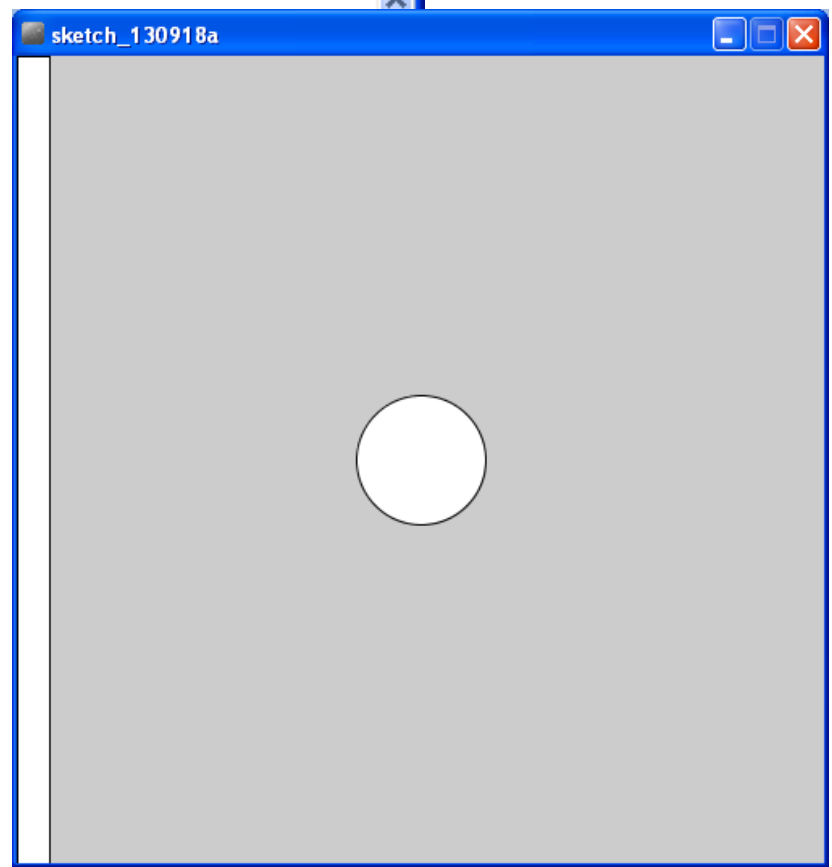
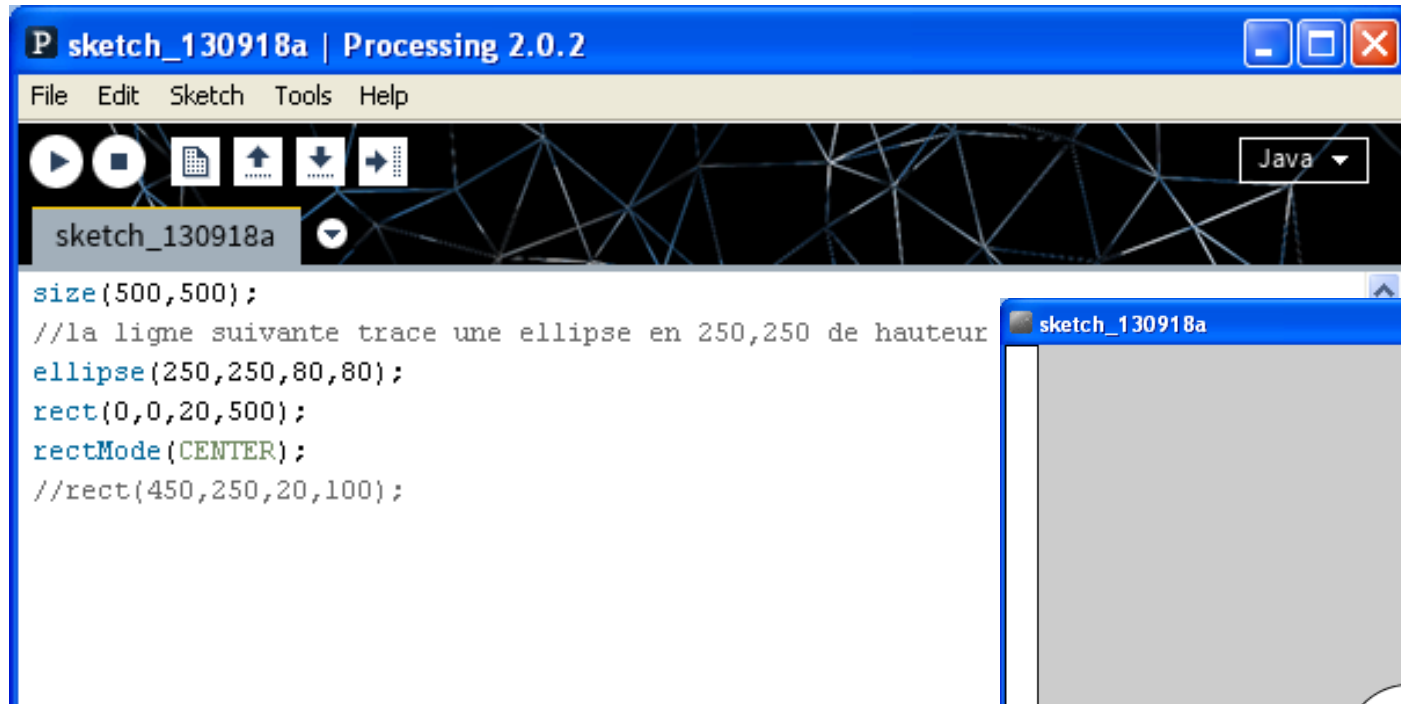
/* : debut de commentaire
Tout ce qui est ici n'est pas executé

****/ : fin de commentaire***

2 utilisations :

- documenter le code***
- tester le programme sans un morceau de code sans avoir à effacer***

Level 3 : Exercice 1 commentaires

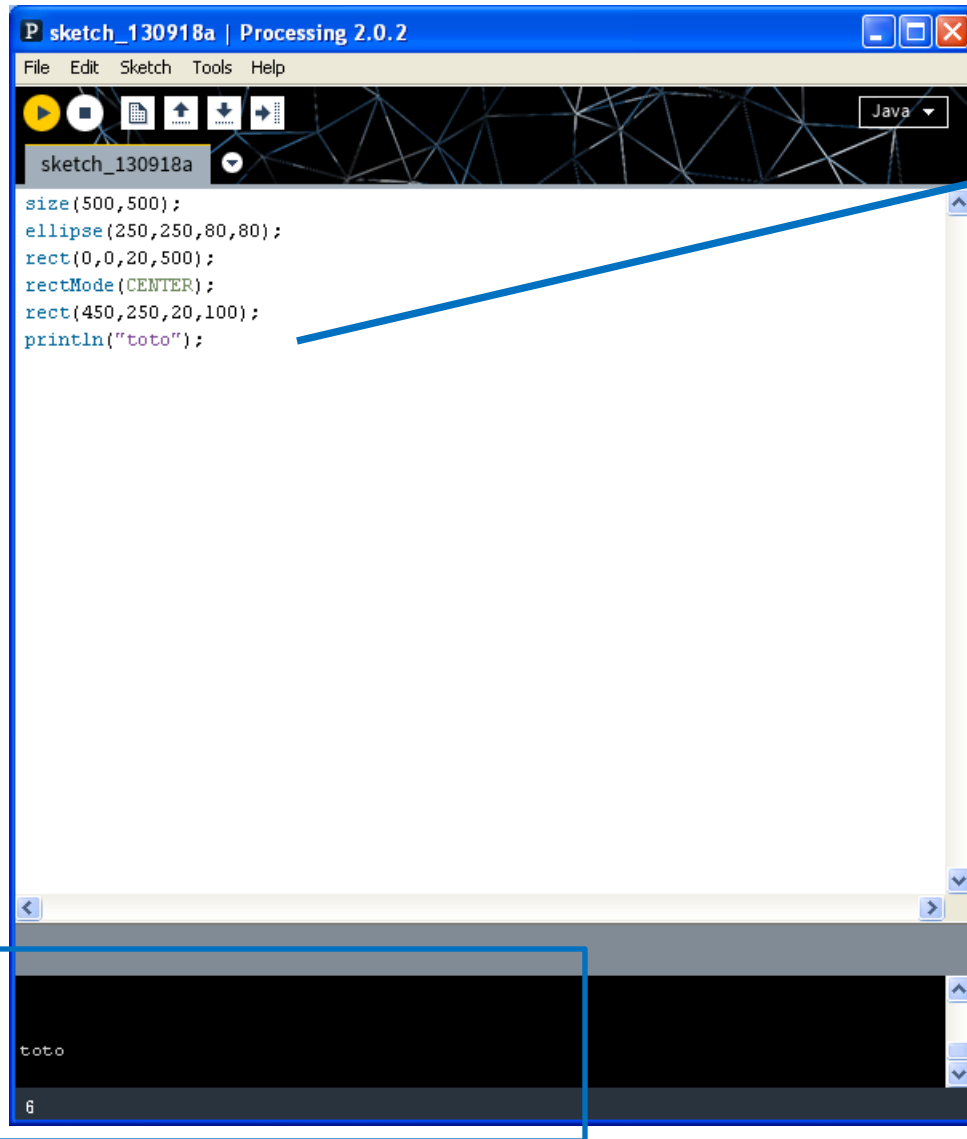


Level 3 : console et commentaires

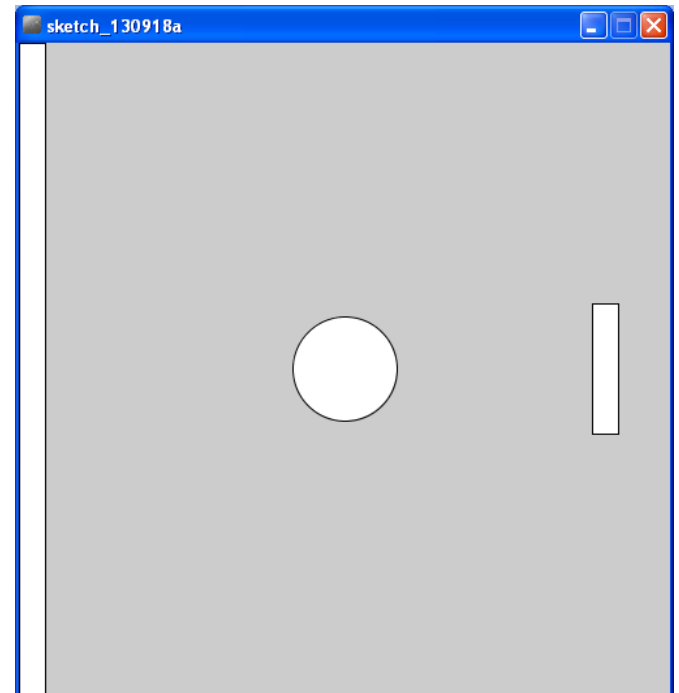


***Console de debug :
permet d'afficher
des message et
erreurs***

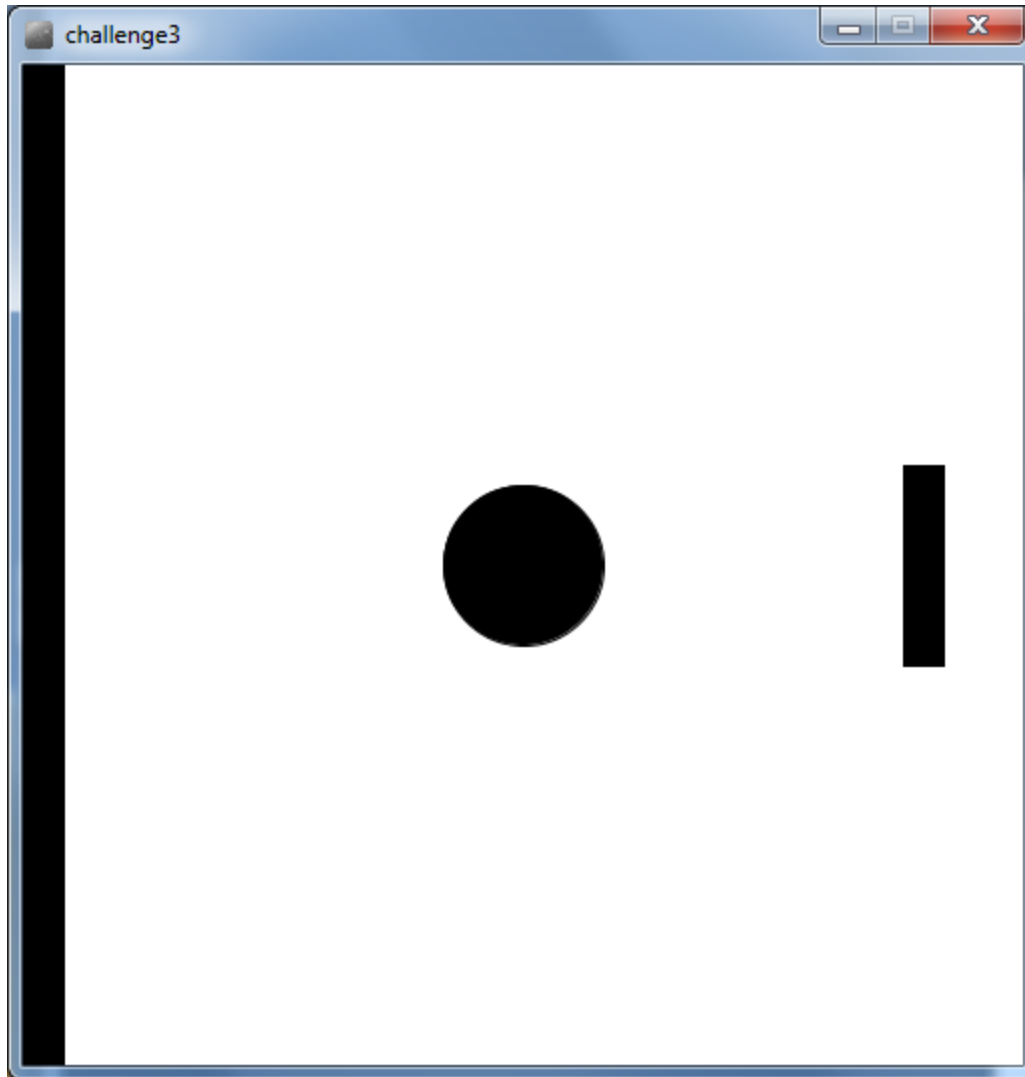
Level 3 Exercice 2 : ecrire sur la console



println("toto");

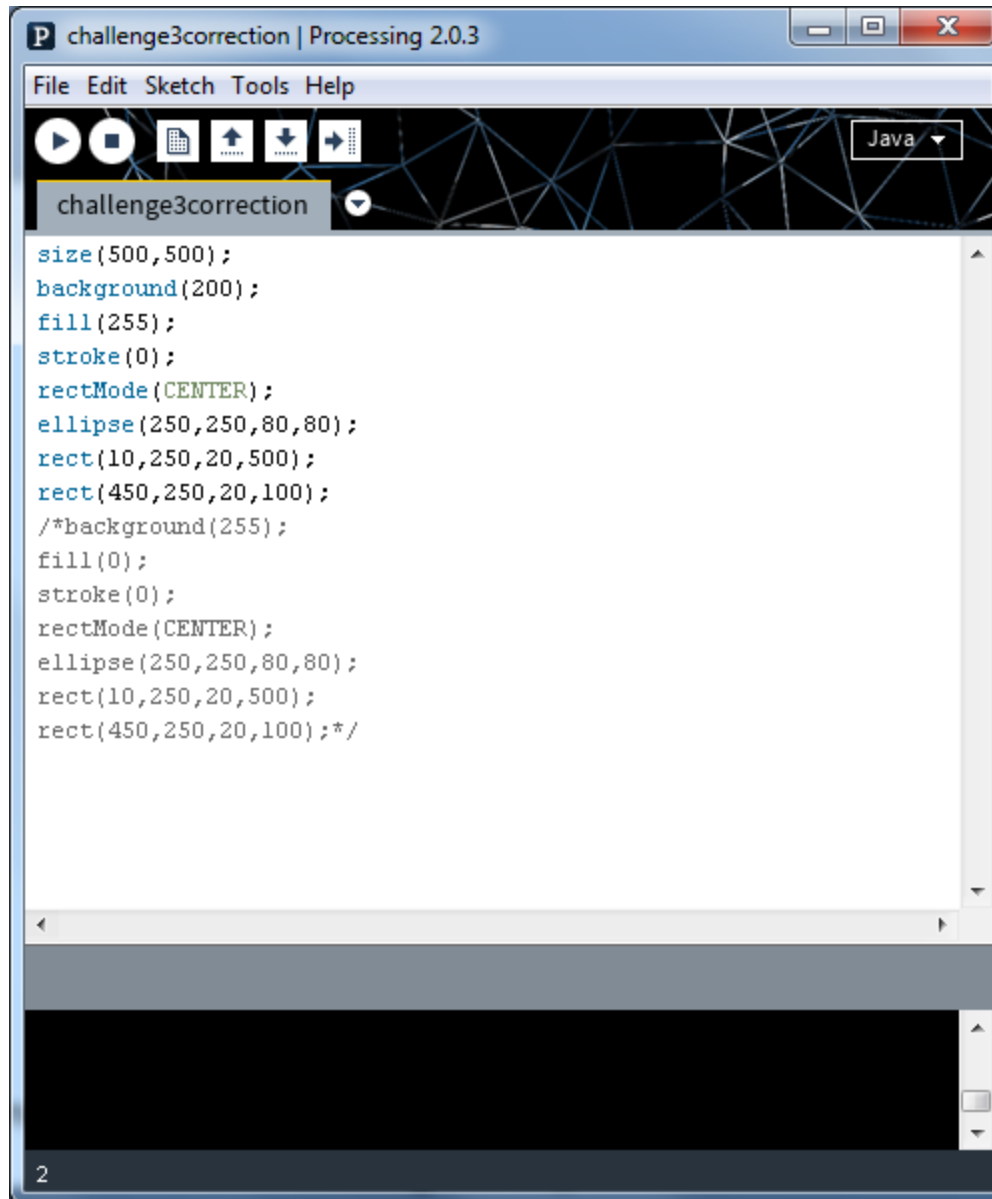


Challenge Level 3 : en utilisant juste les commentaires et en changeant les parametres retrouver l'écran d'avant



Tips : changer les parametres de fill(), stroke() , background ()

Challenge Level 3 : correction



Bonus : comprendre comment marchent fill(), stroke() ,
background ()

Level 4 : introduction GUI



*S'execute qu'une
seule fois*

:-(

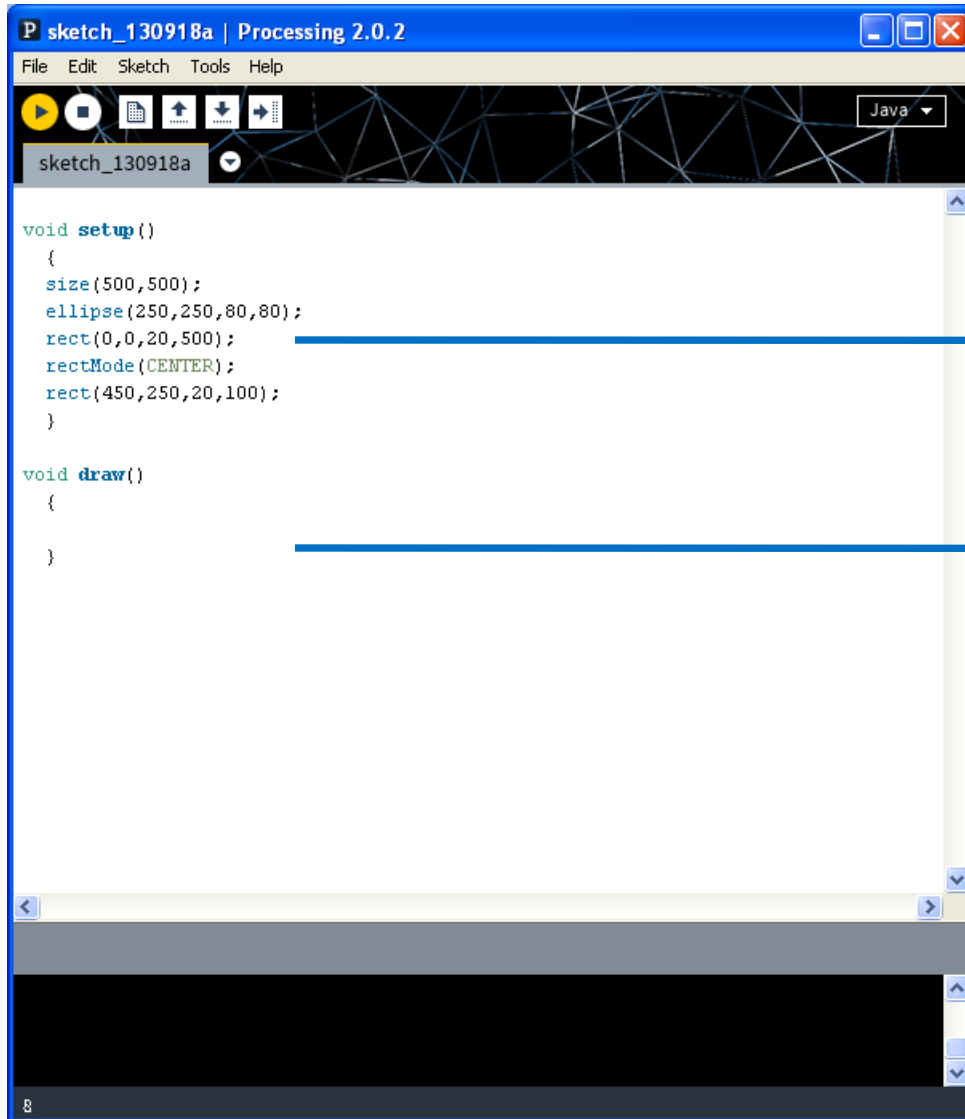
Level 4 : introduction GUI



***GUI : Graphical User
interface***

***=> Réagit en
permanence aux
input de l'utilisateur***

Level 4 : void draw() et void setup ()



void setup()

```
{
//le code ici ne s'execute qu'une fois
}
```

=> s'execute une fois

void draw()

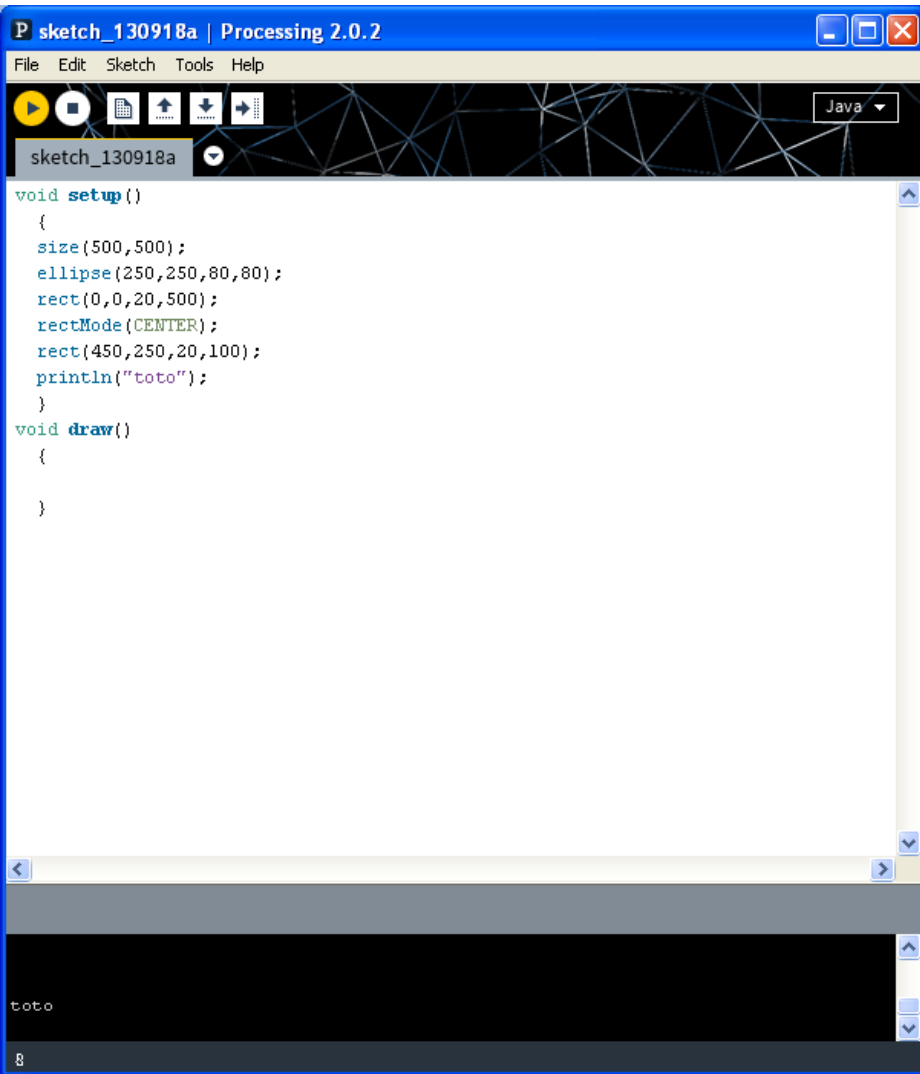
```
{
//le code ici s'execute en continu
}
```

=> s'execute une fois

Void setup et void draw

=> Block de code = fonction
(a garder en tete)

Level 4 : Exercice 1



Processing 2.0.2 IDE window showing a sketch named "sketch_130918a". The code defines a `setup()` function with the following statements:

```
void setup()
{
  size(500,500);
  ellipse(250,250,80,80);
  rect(0,0,20,500);
  rectMode(CENTER);
  rect(450,250,20,100);
  println("toto");
}
```

The `draw()` function is currently empty. The console output shows "toto".

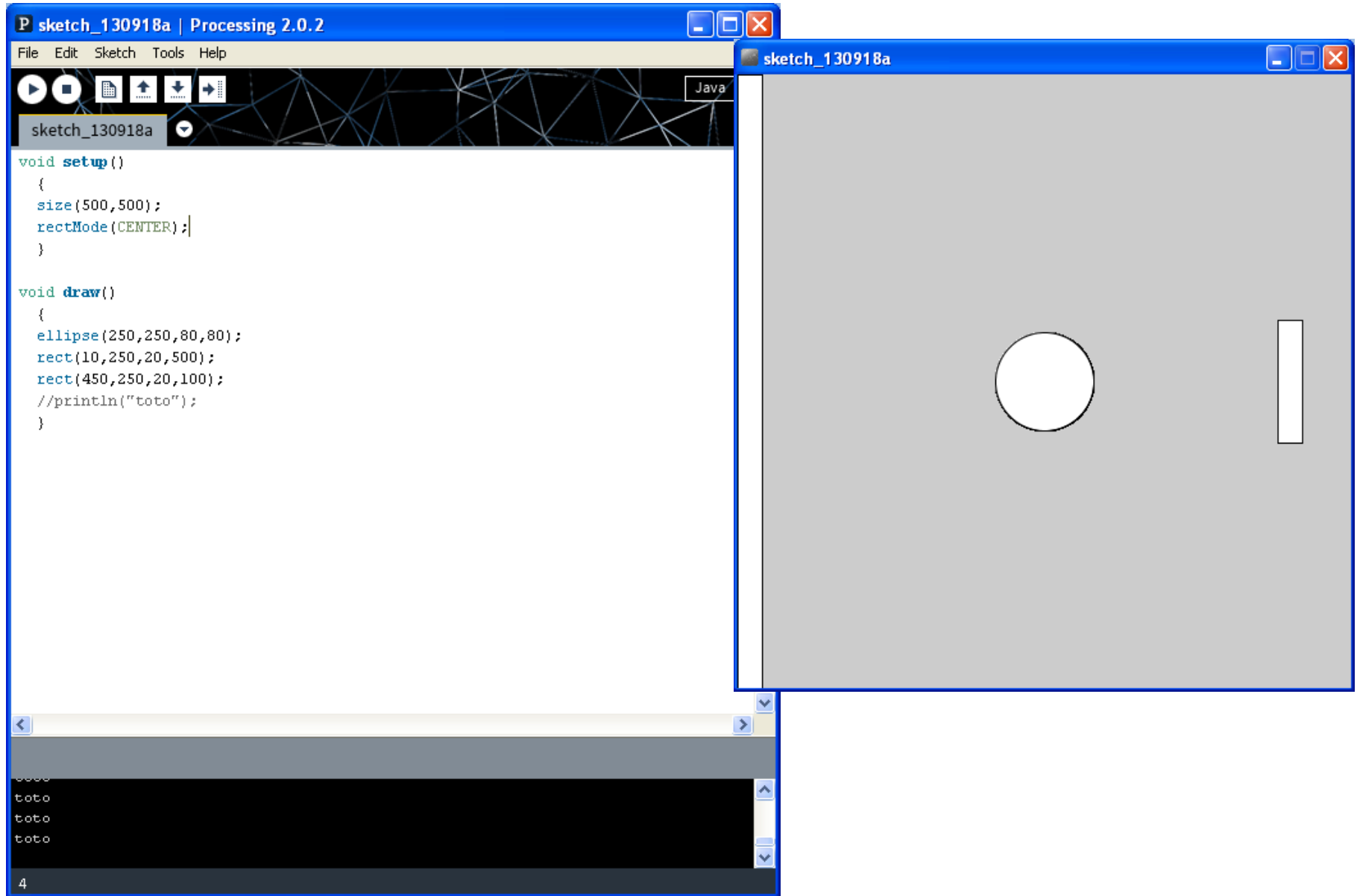


Processing 2.0.2 IDE window showing a sketch named "sketch_130918a". The code defines a `setup()` function with the following statements:

```
void setup()
{
  size(500,500);
  ellipse(250,250,80,80);
  rect(0,0,20,500);
  rectMode(CENTER);
  rect(450,250,20,100);
}
```

The `draw()` function now contains a `println("toto");` statement. The console output shows "toto" repeated four times, indicating the sketch is running in a continuous loop.

Level 4 : Exercice 2



Level 4 : mouse input

mouseX : coordonnée en X de la souris

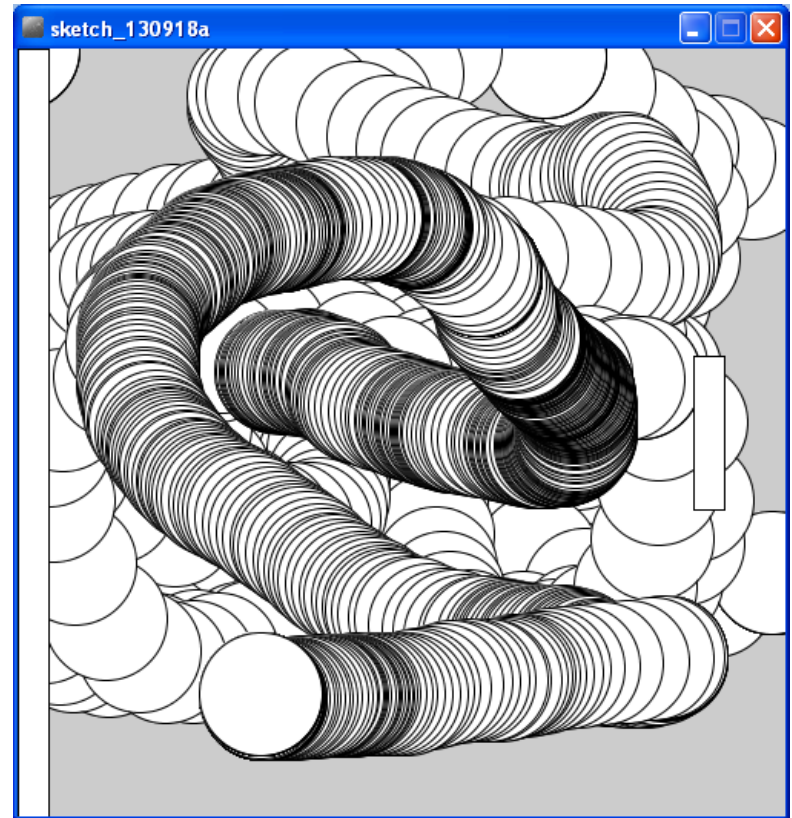
mouseY : coordonnées en Y de la souris

Ex:

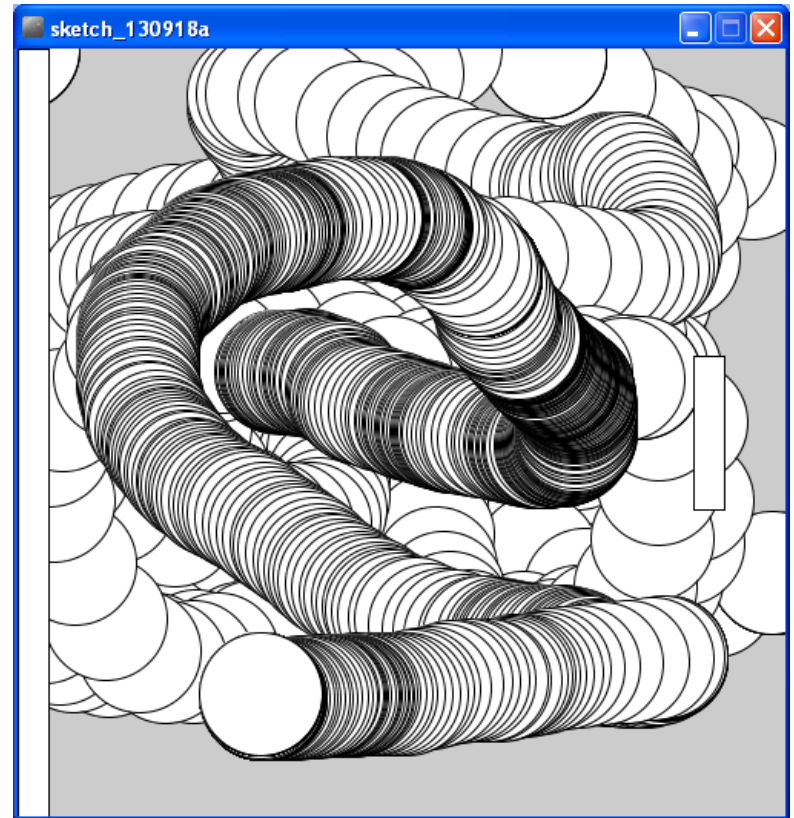
ellipse (mouseX, mouseY, 80,80) ;

=> Trace une ellipse aux coordonnées x y de la souris

Level 4 : Exercice 3

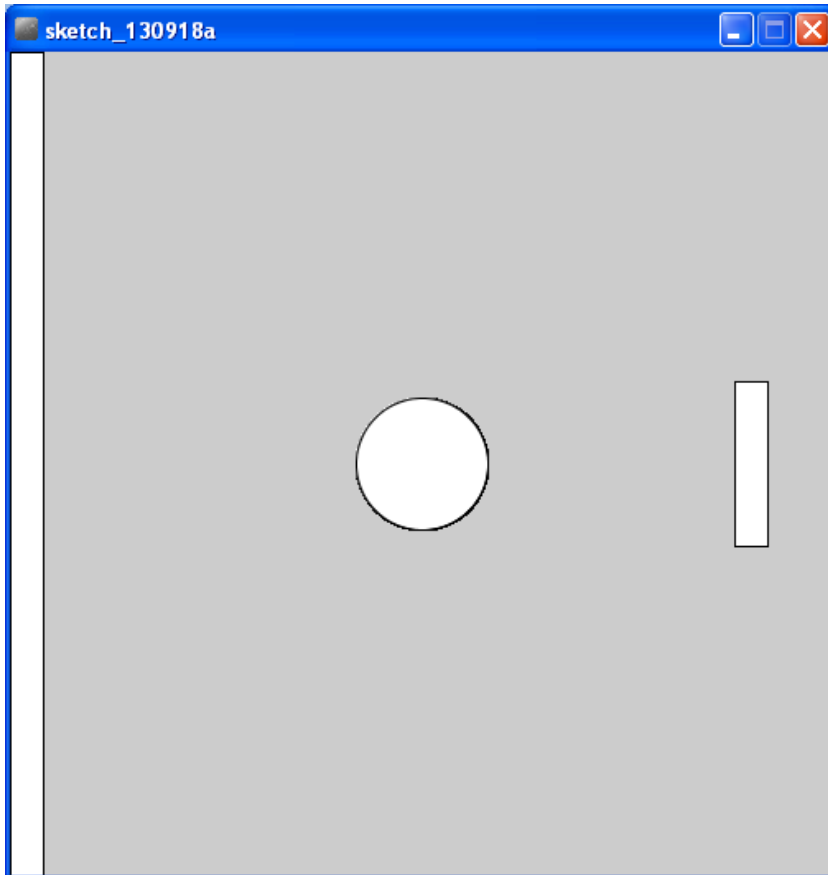


Level 4 : Exercice 3



A la fin de l'exercice remettre ellipse à :
Ellipse(250,250,80,80)

Challenge 4 : Afficher la raquette aux qui suit la souris que en Y et afficher ces coordonnées (10 min.)

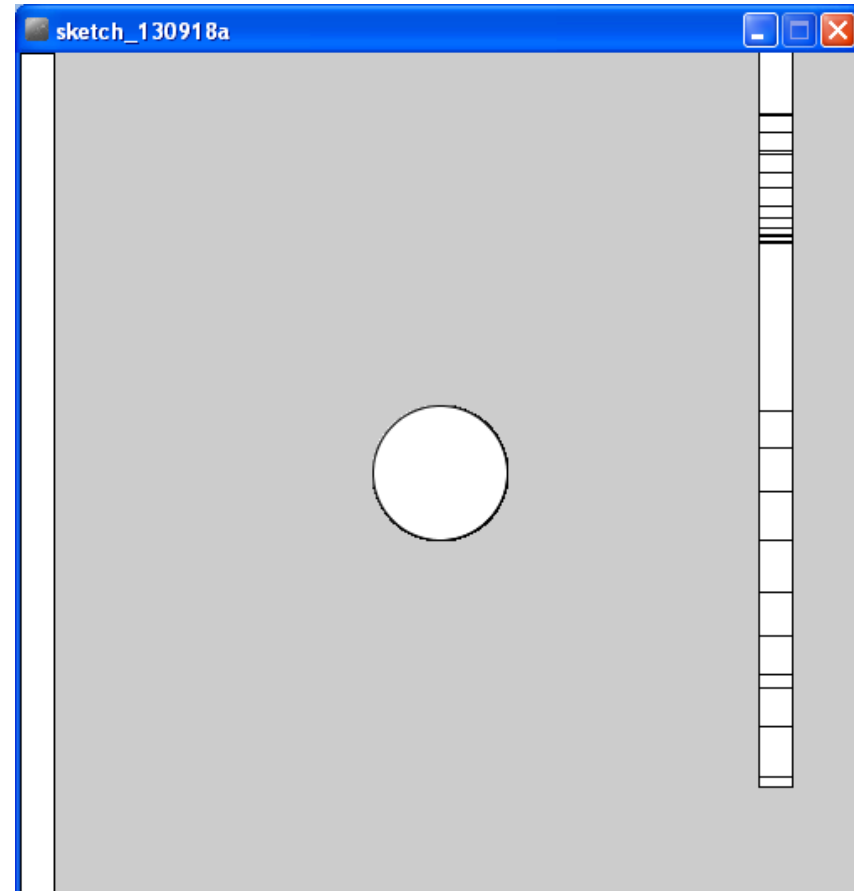


Toolbox

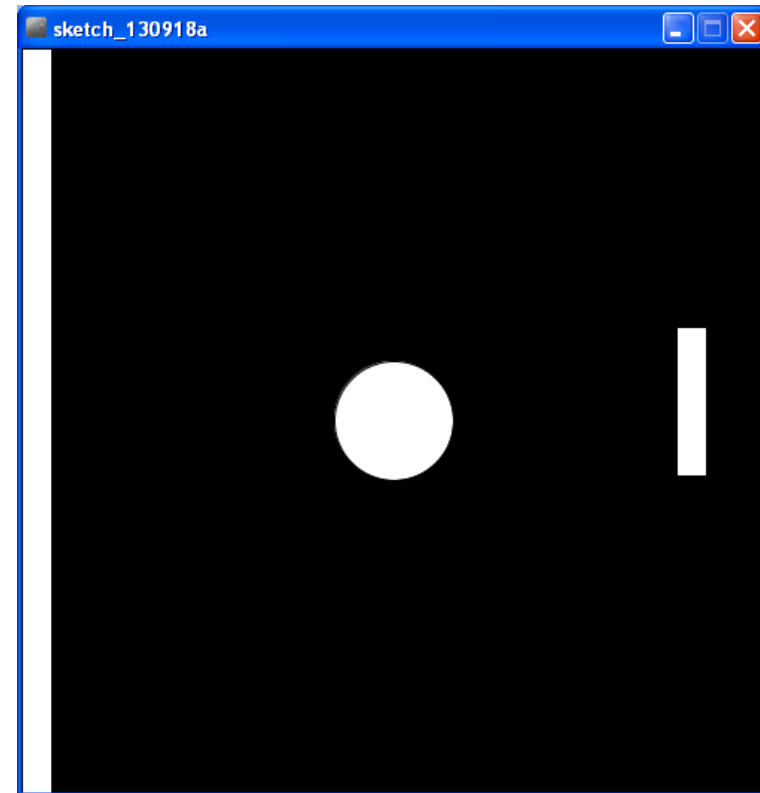
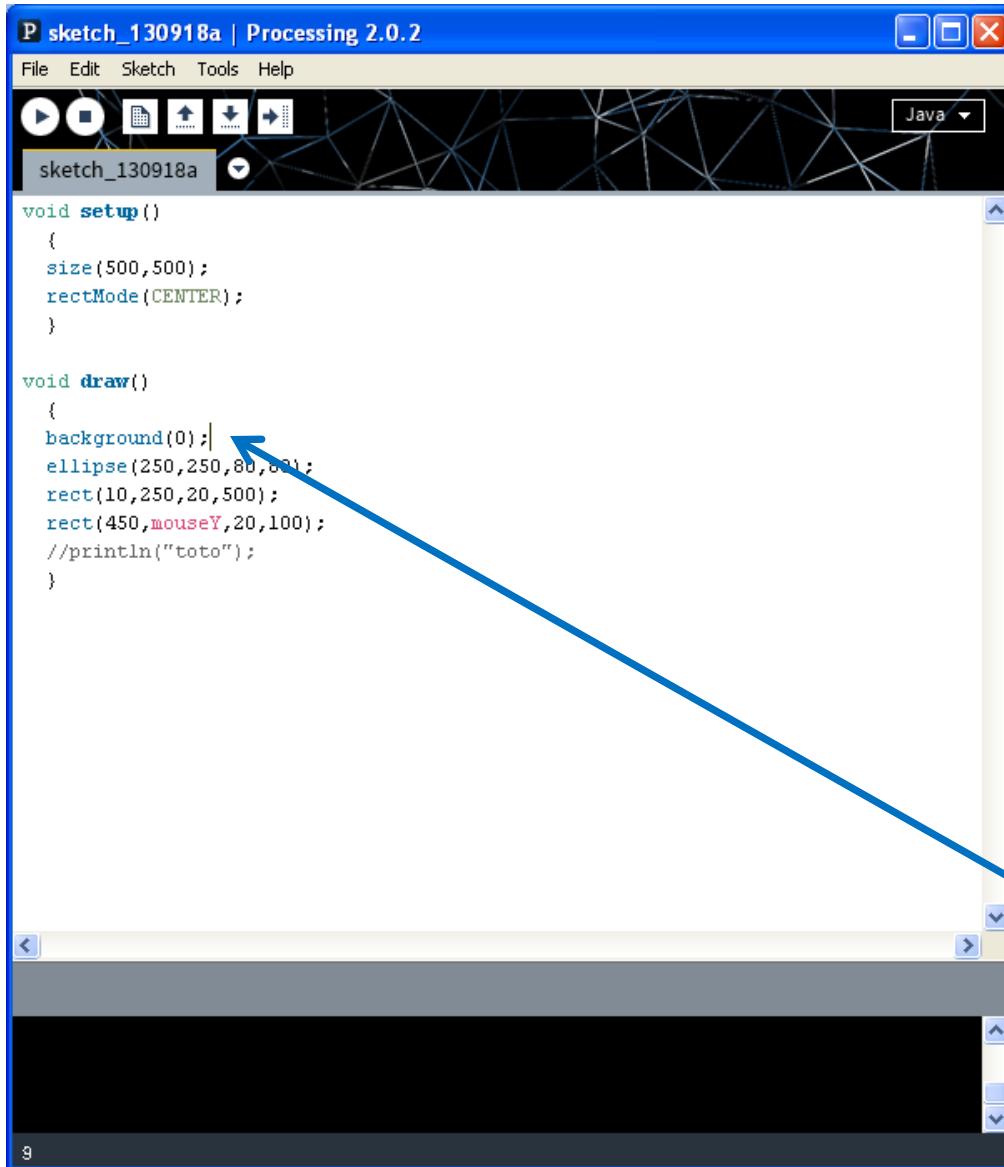
mouseX : coordonnée en X de la souris

mouseY : coordonnées en Y de la souris

Challenge 4 : Correction



Comment effacer l'écran ?



background(0);

***: mettre le fond en
noir (0) en continu***

Level 5 : variables

Variable : valeur d'un certain type stockée en mémoire vive qui est modifiée par le programme

Une variable a toujours un type. Si une variable ne change pas on l'appelle constante

Ex : le score , la position de la souris , la position de la balle

Types :

Int : entier ex: 1 ,2,3,- 4,0 ...

Float : nombre à virgule ex : 0.01 , 1.56 , 7.89

Boolean : 0 ou 1

Char : caractère ex : a

String : chaîne de caractères ex : « toto »

Level 5 : variables

Comment utiliser les variables :

A faire une fois au début du programme : déclaration et initialisation

int score = 0 ; // déclare et initialise la variable score

↑ ↑
type Nom de la variable



Case sensitive : score et Score sont différents

Pas d'espaces ni d'accents ni +-/=*

Pas de nom qui seraient pris par le système ex : ellipse

IL FAUT DONNER DES NOMS INTELLIGIBLES AUX VARIABLES
(pas toto !!)

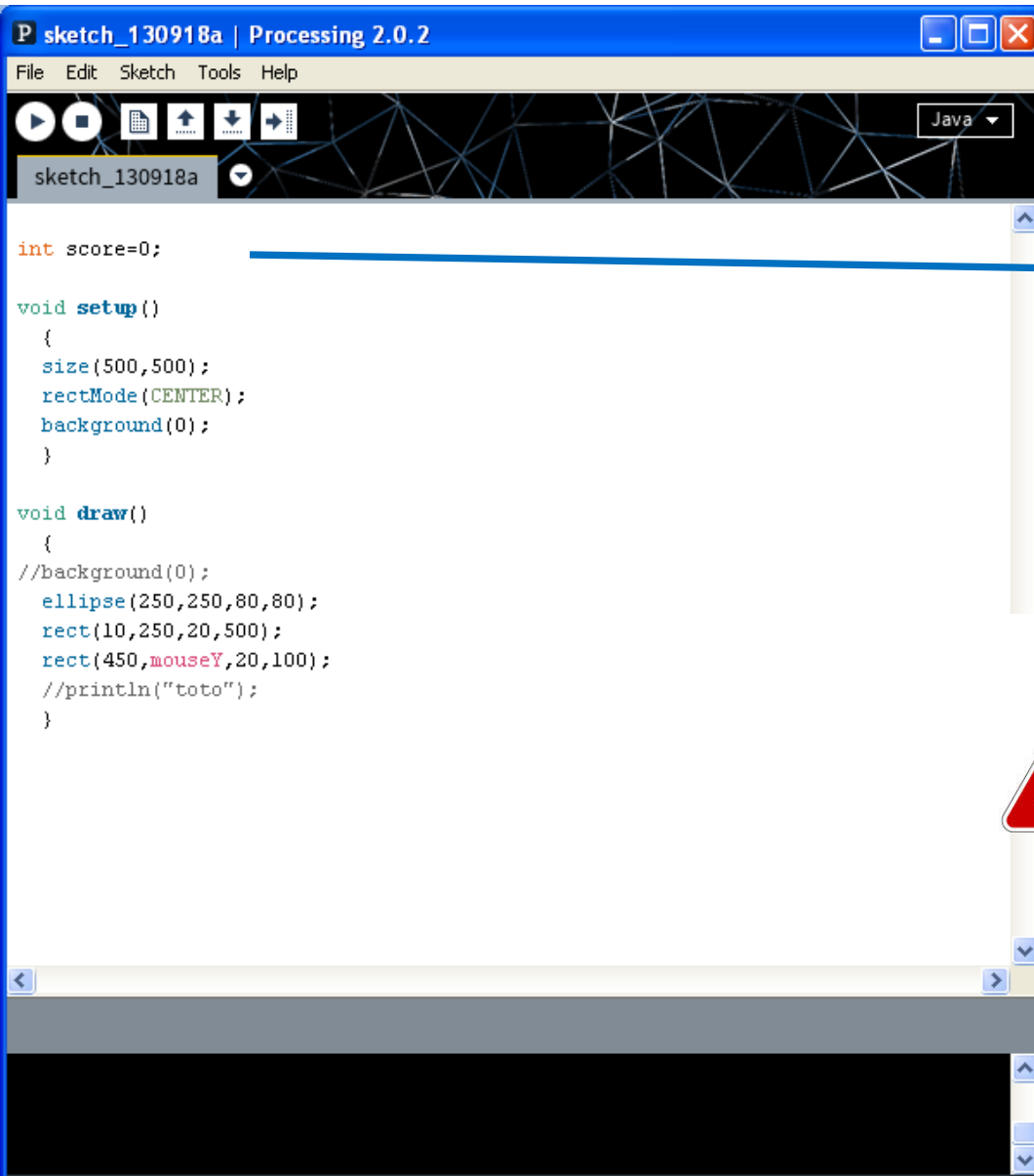
Ensuite quand on veut changer la valeur :

score = 10; //le nouveau score est 10

score = score +1; // on ajoute 1 à la variable score

Level 5 : variables

Portée des variables:



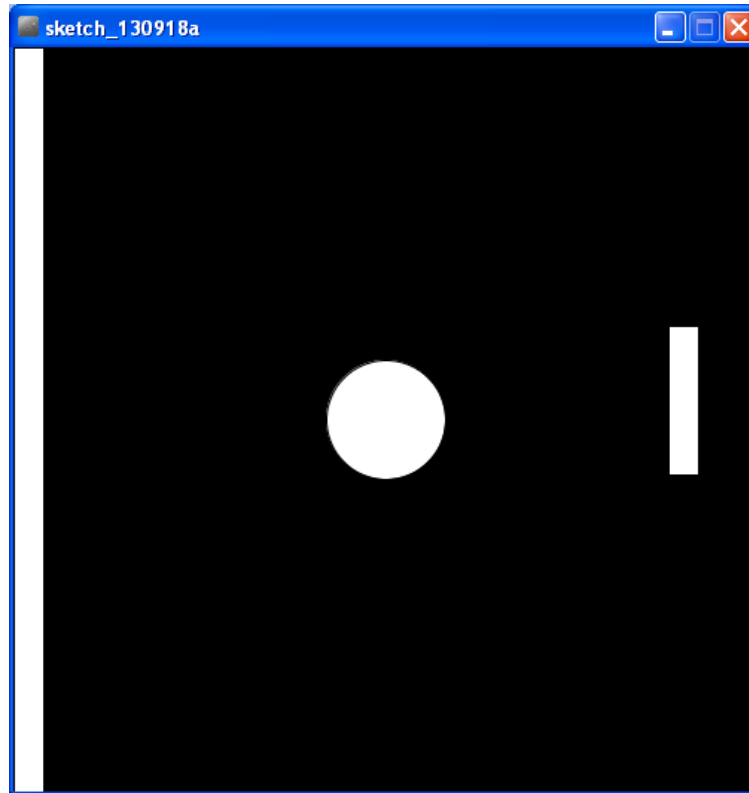
En simplifiant:

On déclare les variables avant le setup pour que tout le monde puisse y accéder

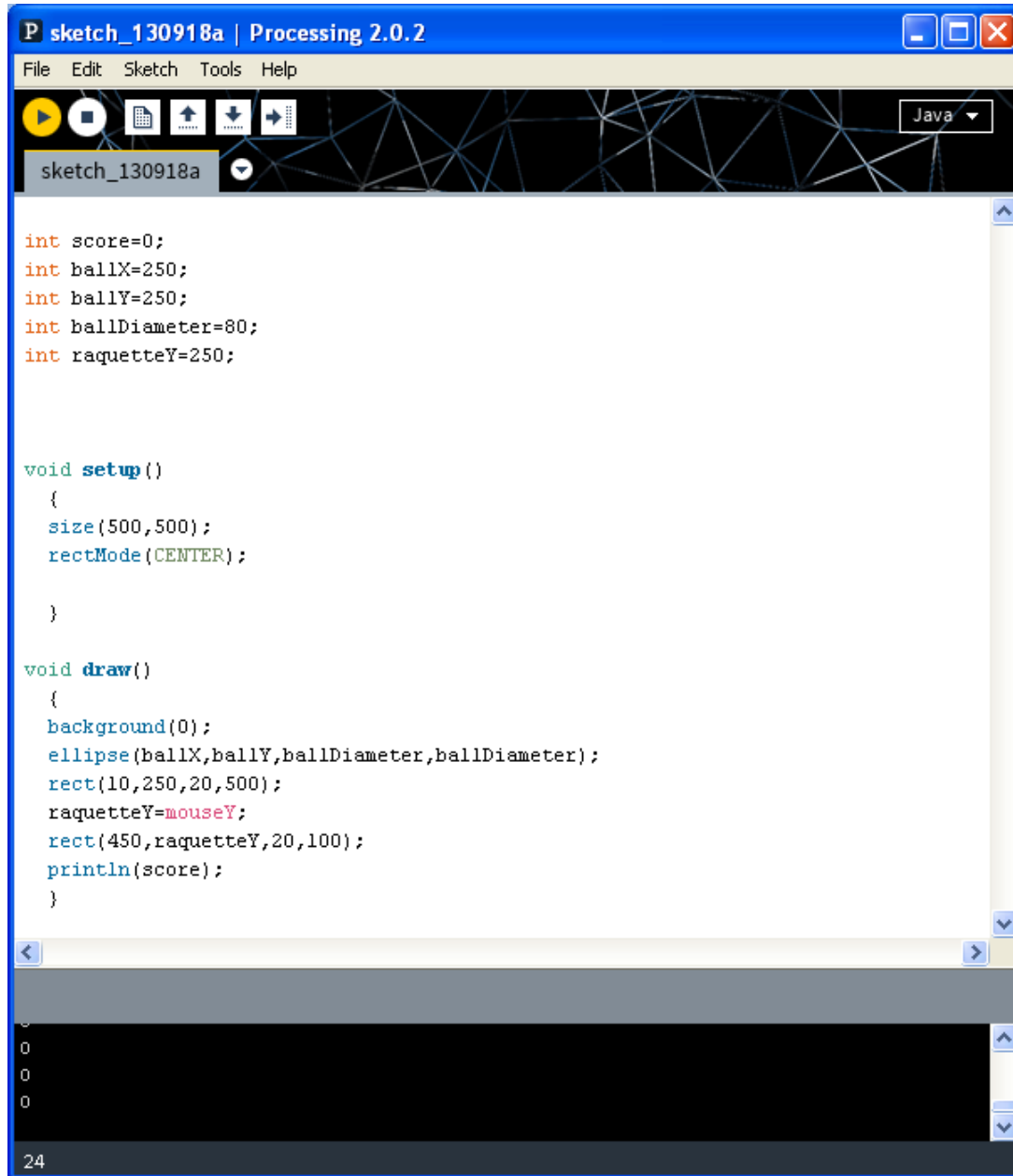
On ne peut pas accéder à des variables si on ne les pas déclarées auparavant

On ne peut pas changer le type des variables après leur déclaration

Challenge Level 5 : trouver les variables (et leurs types) dont on a besoin pour le pong et les déclarer – (10 min.)



Challenge Level 5 : correction



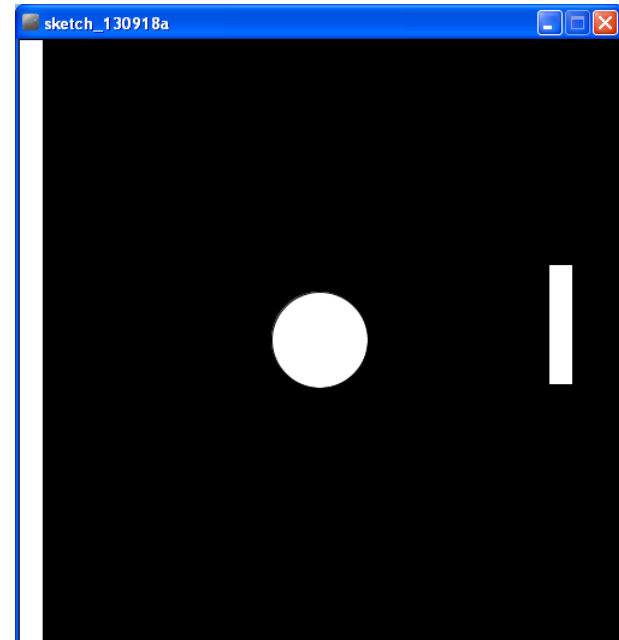
The screenshot shows the Processing IDE interface for a sketch named 'sketch_130918a'. The code is written in Java and defines a simple game environment. It includes variables for score, ball position, ball diameter, and racket position. The setup function initializes the canvas size and mode. The draw function updates the background, draws the ball and racket, and prints the score.

```
int score=0;
int ballX=250;
int ballY=250;
int ballDiameter=80;
int raquetteY=250;

void setup()
{
  size(500,500);
  rectMode(CENTER);
}

void draw()
{
  background(0);
  ellipse(ballX,ballY,ballDiameter,ballDiameter);
  rect(10,250,20,500);
  raquetteY=mouseY;
  rect(450,raquetteY,20,100);
  println(score);
}
```

The console output at the bottom shows the score being printed as 0, 0, 0, and 24.



Level 6 : boucles = Loops

```
for(int i=0; i<=10 ; i++)
```

```
{
```

```
//mettre ici le code que l'on veut executer plusieurs fois
```

```
}
```

Initialisation du
compteur (paramètre de
sortie de la boucle)

condition de sortie de la
boucle ici si i est plus petit
ou eagal à 10

Pas de la boucle ici i est
incrémenté de 1 à
chaque tour de boucle
(i++ => i=i+1)

Level 6 : boucles = Loops

```
for(int i=0; i<=10 ; i++)  
{  
    //mettre ici le code que l'on veut executer plusieurs fois  
    println(i);  
}  
println(« toto »);
```

```
Int i =0 => println(0)  
i = 1  println(« 1 »);  
i = 2  println(« 2 »);  
.  
.  
.  
i=10 println(« 10 »);  
Int i=11 => sortie  
println(« toto »);
```


Level 6 Exercice 1



**i<10 va s'arreter a 9
alors que
i <=10 va s'arreter à 10**

```
Processing 2.0.2
sketch_130918a

File Edit Sketch Tools Help

sketch_130918a

int ballDiameter=80;
int raquetteY=250;

void setup()
{
  size(500,500);
  rectMode(CENTER);
  for(int i=0;i<=5;i++)
  {
    score=score+1;
    println(score);
  }
}

void draw()
{
  background(0);
  ellipse(ballX,ballY,ballDiameter,ballDiameter);
  rect(10,250,20,500);
  raquetteY=mouseY;
  rect(450,raquetteY,20,100);
}
```

```
for(int i=0; i<=?? ; i++)
{
  //mettre ici le code que l'on veut executer plusieurs fois
}
```

Level 7 : ifs

```
If( condition1)
{
  //code a executer pour la condition 1
}
else if (condition2)
{
  //code a executer si on la condition 2
}
else
{
  //code a executer si ni condition 1 ni condition 2 ne sont réunies)
}
```

Level 7 : ifs



On met 2 = car c'est un test et nom une initialisation

Exemple :

```
If( toto==0)
{
    //code a executer quand toto=0
}
```

```
else if (toto<0)
{
    //code a executer si toto <0
}
```

```
else if (toto>1 && toto<10)
{
    //code a executer si toto <0
}
```

```
else
{
    //code a executer si ni condition 1 ni condition 2 ne sont réunies)
}
```

Operateur ET on en met 2 aussi &&

Level 7 : ifs

opérateurs:

&& : ET

|| : OU

Ex :

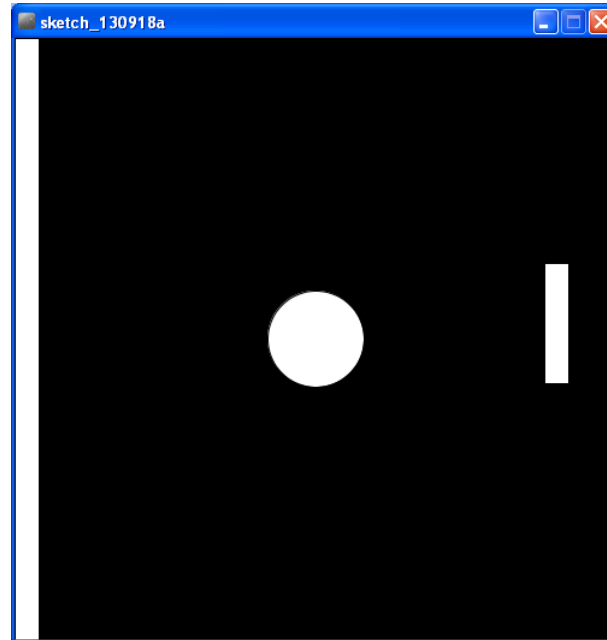
toto 1 = 10;

toto 2 = 11;

If(toto1<10 && toto2<10) //ne rentrera pas le if

If(toto1<10 || toto2<10) // rentrera dans le if

Challenge Level 7 : trouver le code du pong (15 min.)



TIPS :

```
if( ballX ==raquetteX &&  
.....)
```

```
{  
}
```

```
else if (.....)
```

```
{
```

```
}
```

Code pour la trajectoire :

```
ballX=ballX+speedX;
```

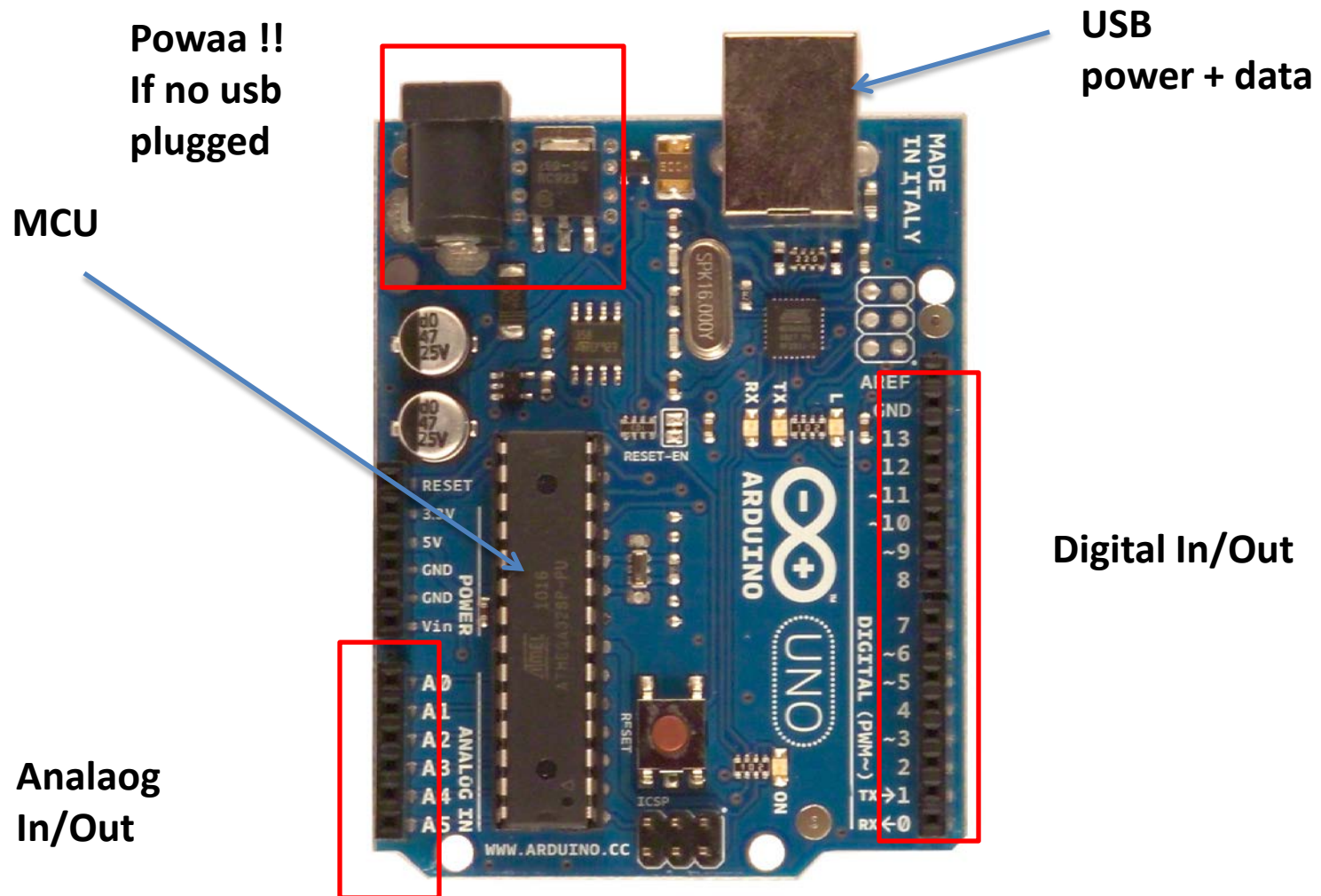
```
ballY=ballY+speedY;
```

Challenge Level 7 : correction

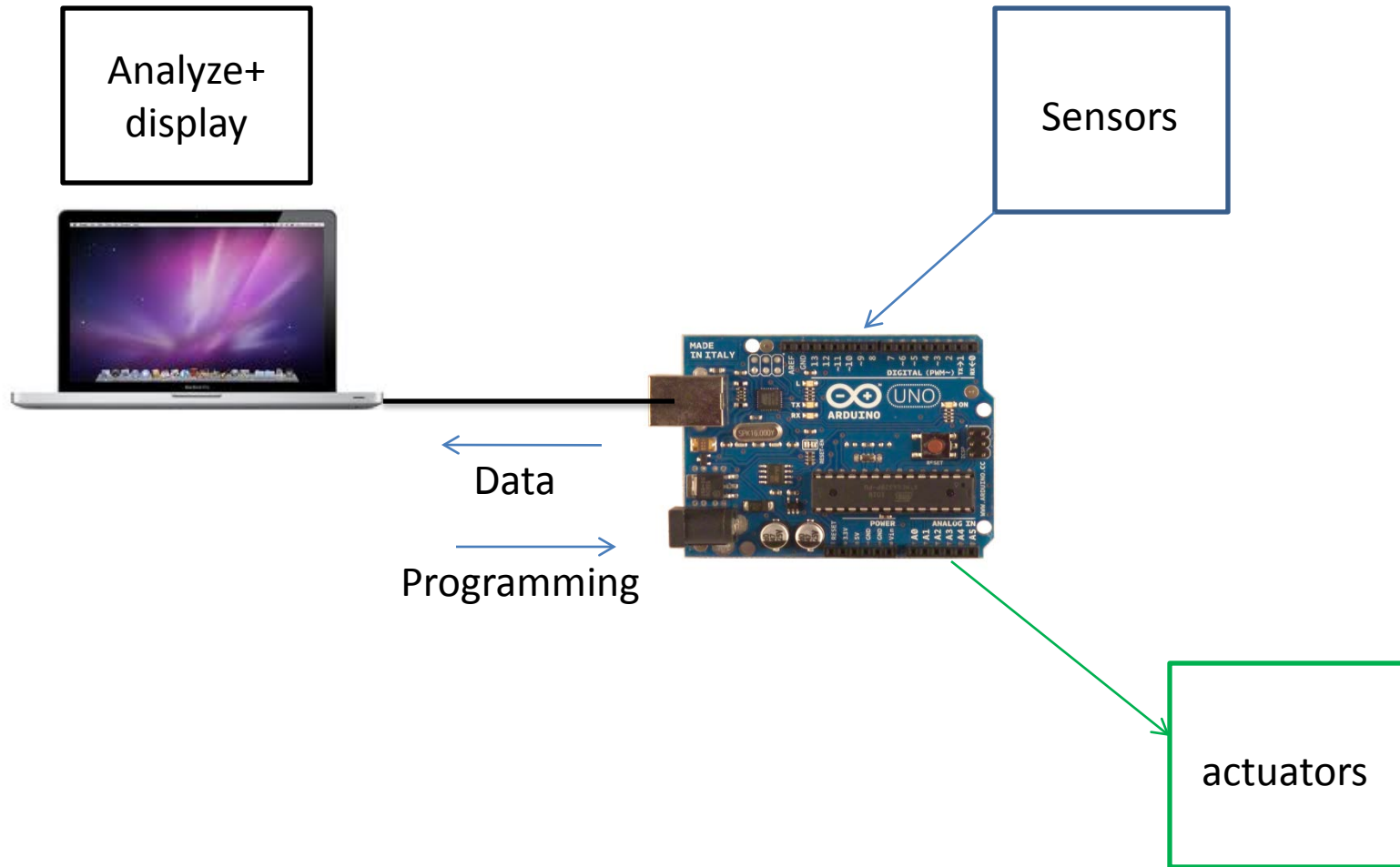
```
if(gameStarted==1)
{
    ballX=ballX+speedX;
    ballY=ballY+speedY;
    if((ballX+(ballDiameter/2)==raquetteX) && (ballY<raquetteY+50)
    &&(ballY>raquetteY-50))
    {
        speedX=speedX*-1;
    }
    else if(ballX-(ballDiameter/2)==0)
    {
        speedX=speedX*-1;
    }
    else if(ballX+(ballDiameter/2)==500)
    {
        println("perdu");
        gameStarted=0;
        ballX=250;
        ballY=250;
    }
    if(ballY==500)
    {
        speedY=speedY*-1;
    }
}
```

Part 2- Arduino

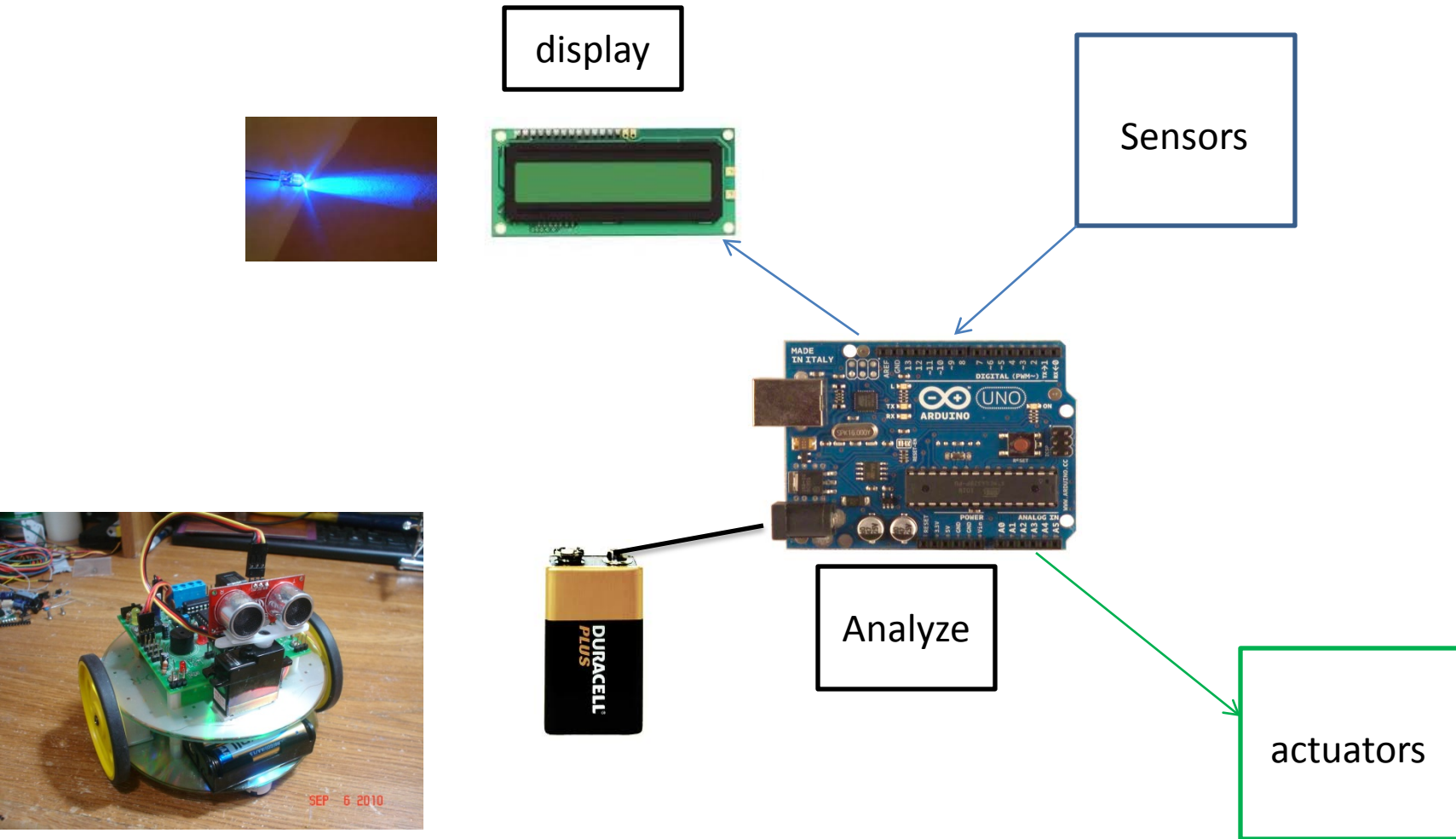
What is the Arduino ?



Use case 1: interface card

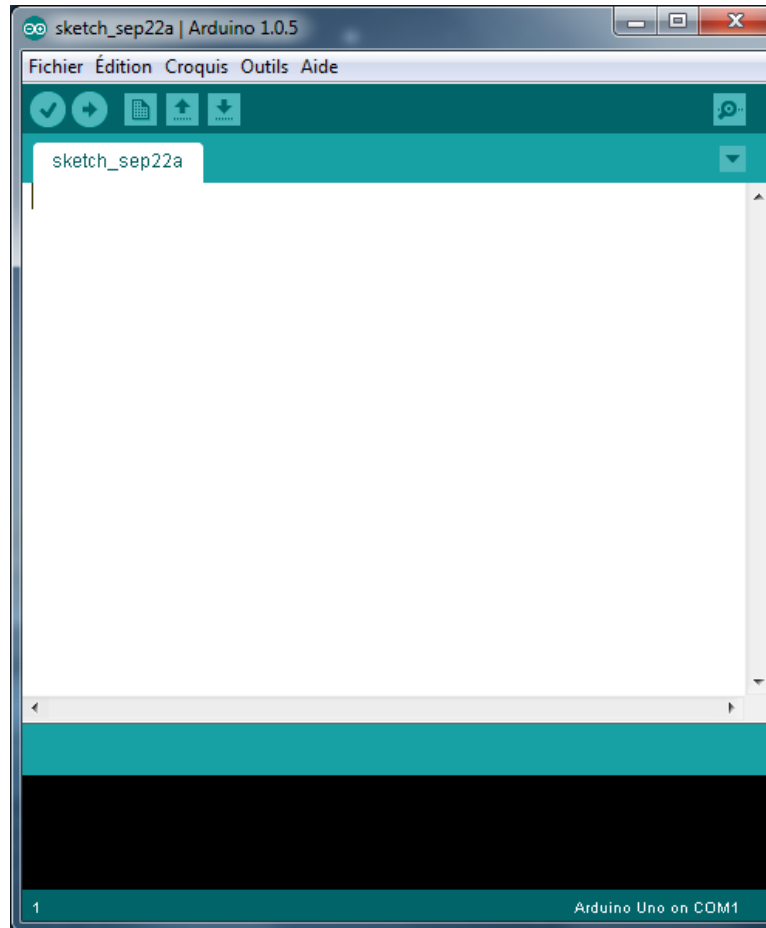


Use case 2 : autonomous *



*but need to be programmed with a computer once

Arduino Installed ?



Pour que tout marche : Ouvrir avec terminal

Sudo arduino

Demarrer Arduino

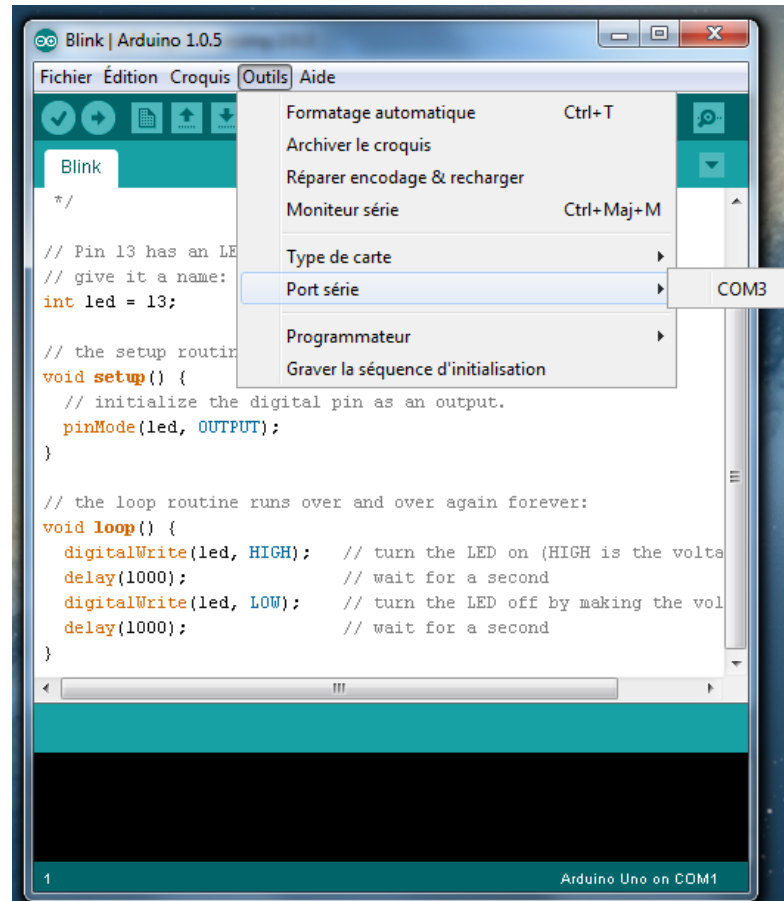


Choisir la bonne carte et le bon port (90%) de erreurs

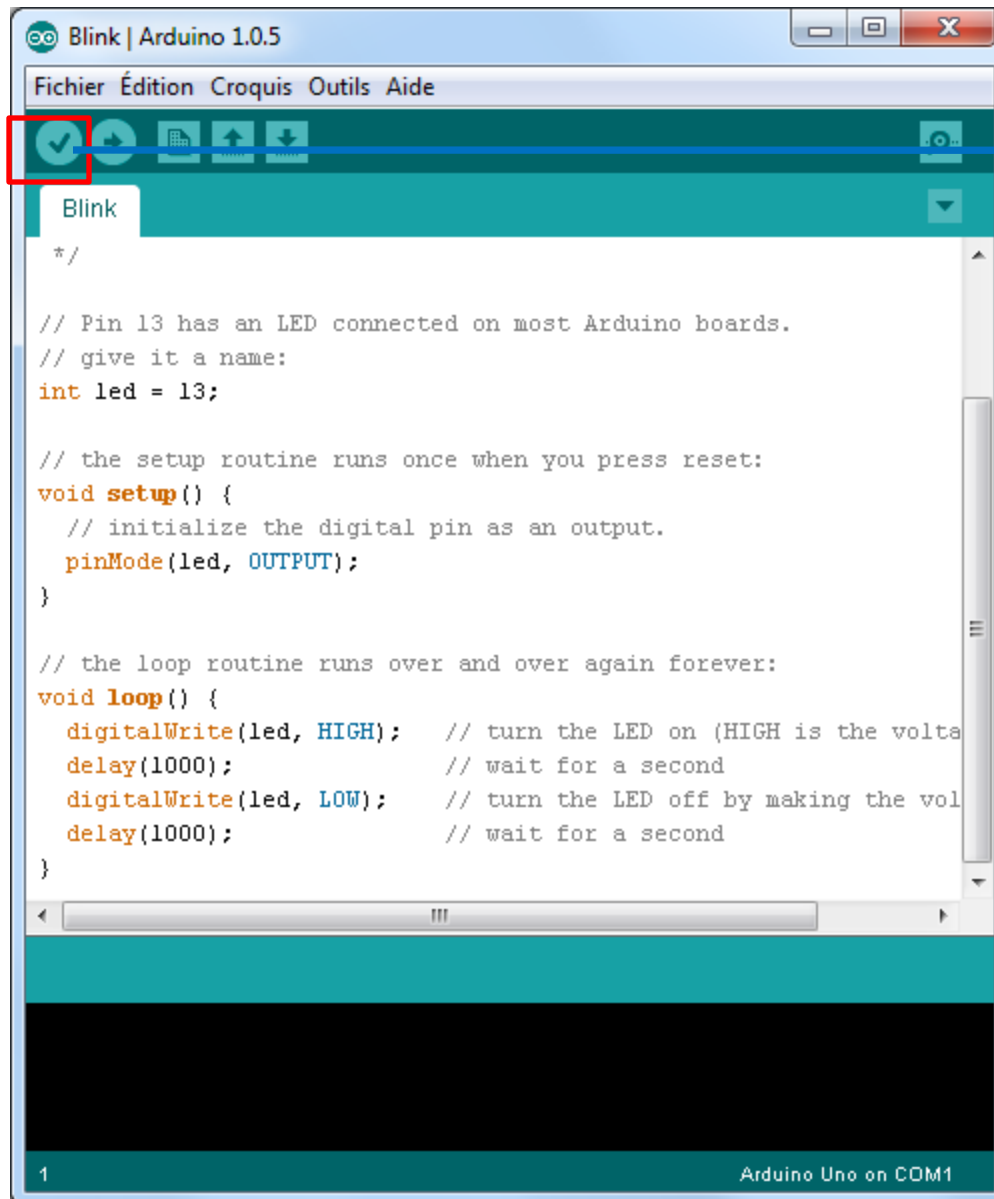
Outils -> type de carte -> uno

Outils -> port -> dev/ttyACM0/

Si aucun port disponible probleme ! (ouvrir arduino en SUDO ! ! !)

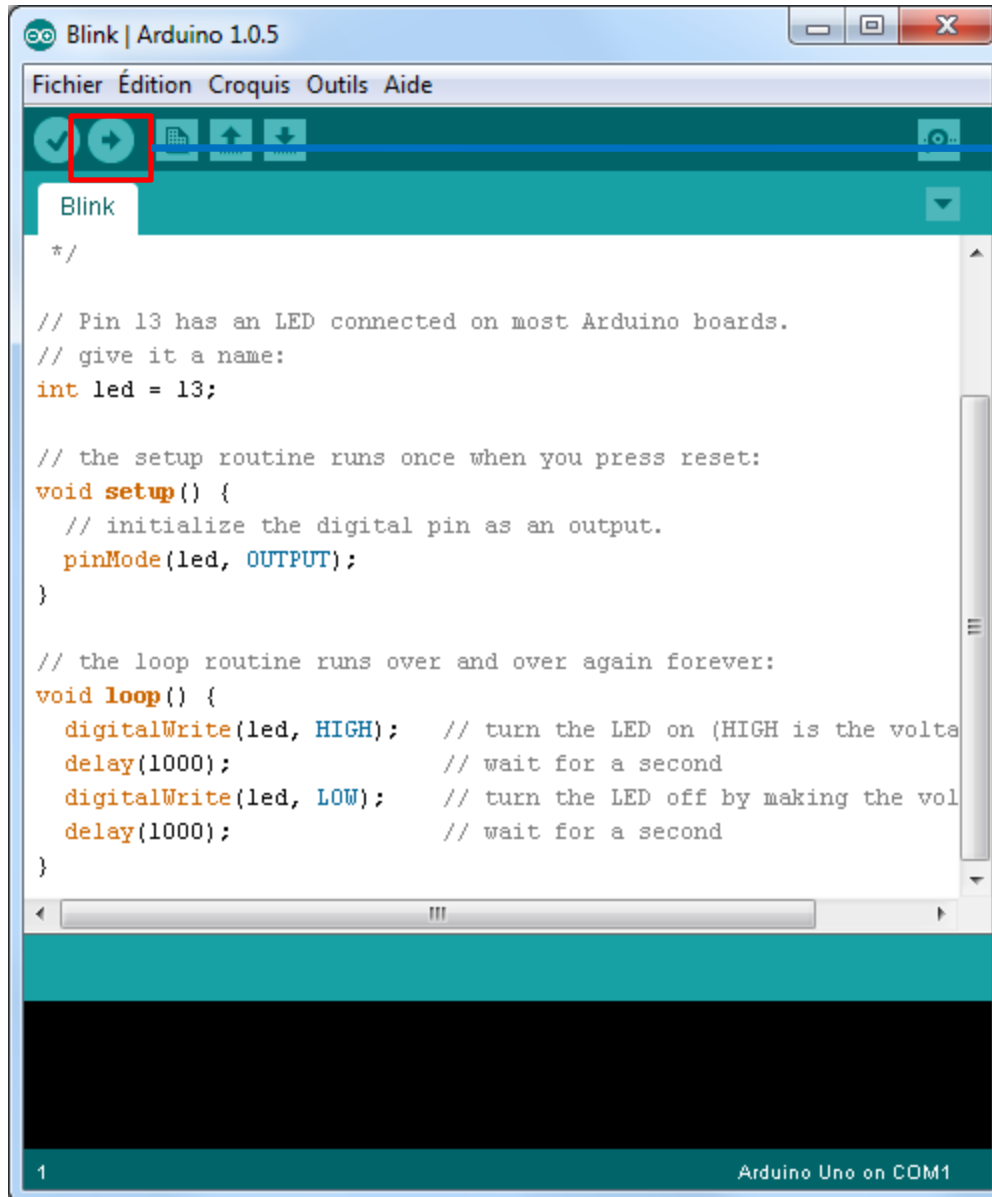


Arduino exercice 1 : Blink



Compiler

Arduino exercice 1 : Blink

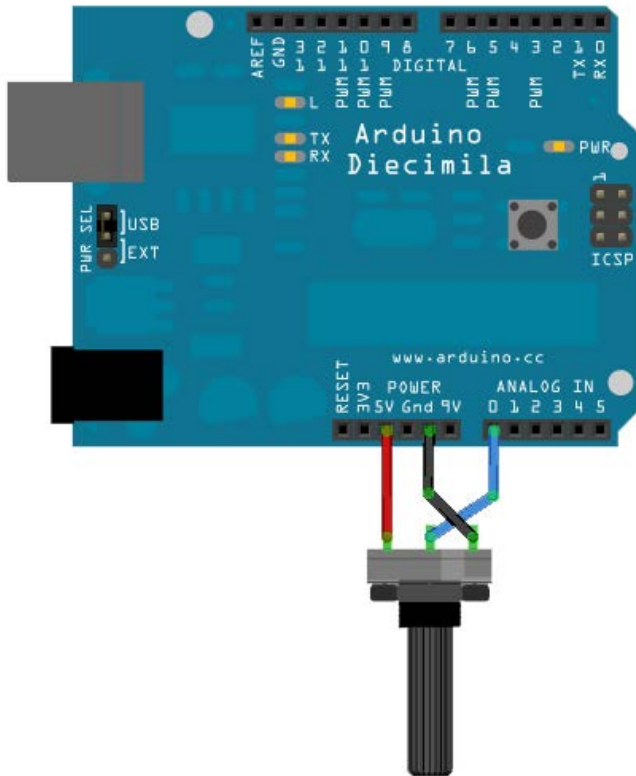


Compiler

+

**Envoyer sur la
puce**

Arduino exercice 2 : Potentiometer



```
Ex2_Potentiometer | Arduino 1.0.5
Fichier Édition Croquis Outils Aide

Ex2_Potentiometer

/*
  Analog input, analog output, serial output

  Reads an analog input pin, maps the result to a range from 0 to 2
  and uses the result to set the pulsewidth modulation (PWM) of an
  Also prints the results to the serial monitor.

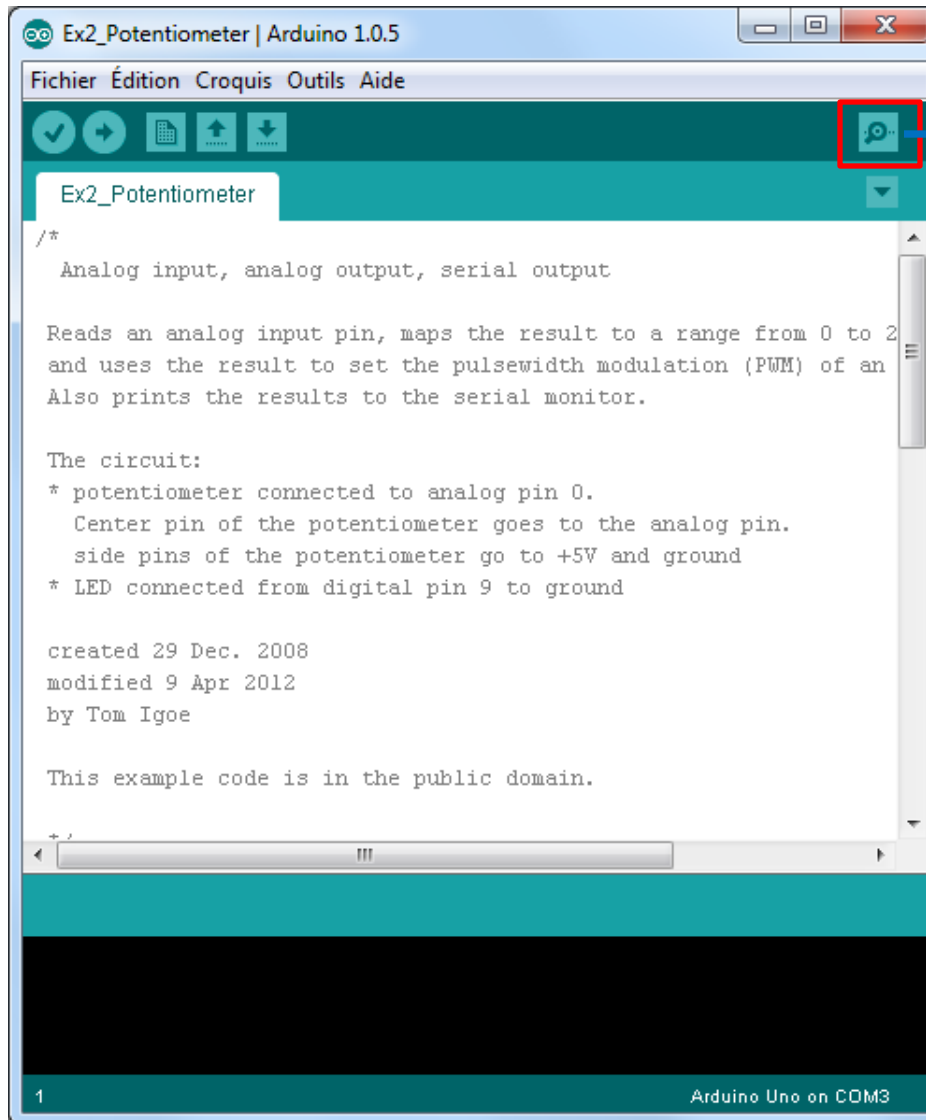
  The circuit:
  * potentiometer connected to analog pin 0.
    Center pin of the potentiometer goes to the analog pin.
    side pins of the potentiometer go to +5V and ground
  * LED connected from digital pin 9 to ground

  created 29 Dec. 2008
  modified 9 Apr 2012
  by Tom Igoe

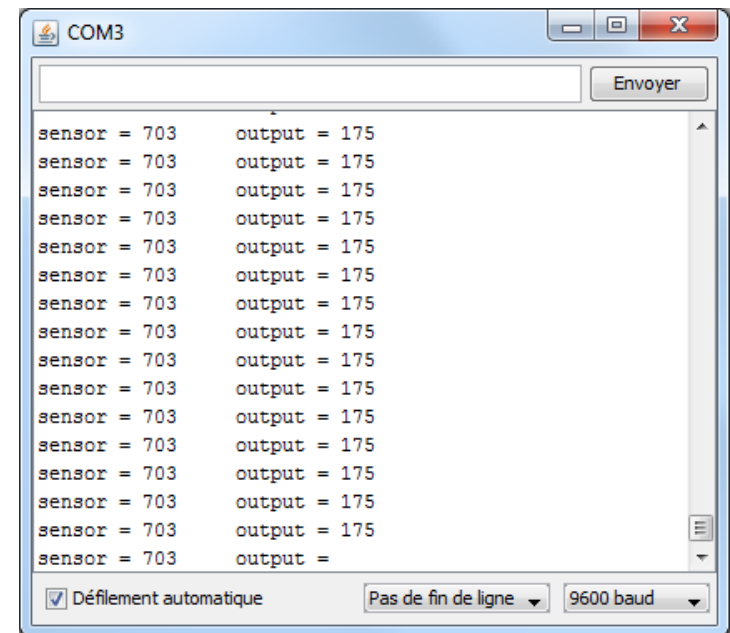
  This example code is in the public domain.
*/

1 Arduino Uno on COM3
```

Arduino exercice 2 : Potentiometer



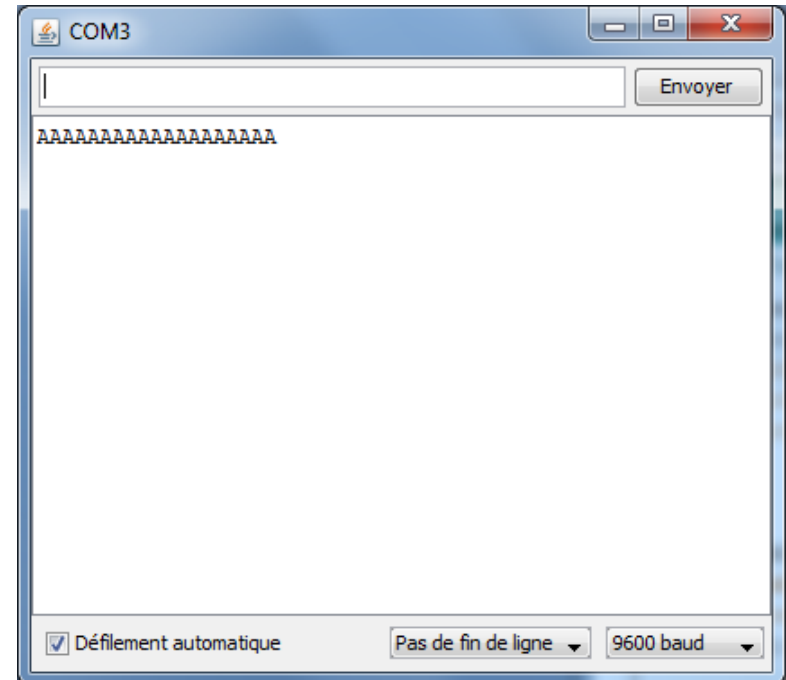
Moniteur série



Part 3- Pong Processing + arduino controllers

Potentiometer to processing

```
void establishContact() {  
  while (Serial.available() <= 0) {  
    Serial.print('A'); // send a capital A  
    delay(300);  
  }  
}
```



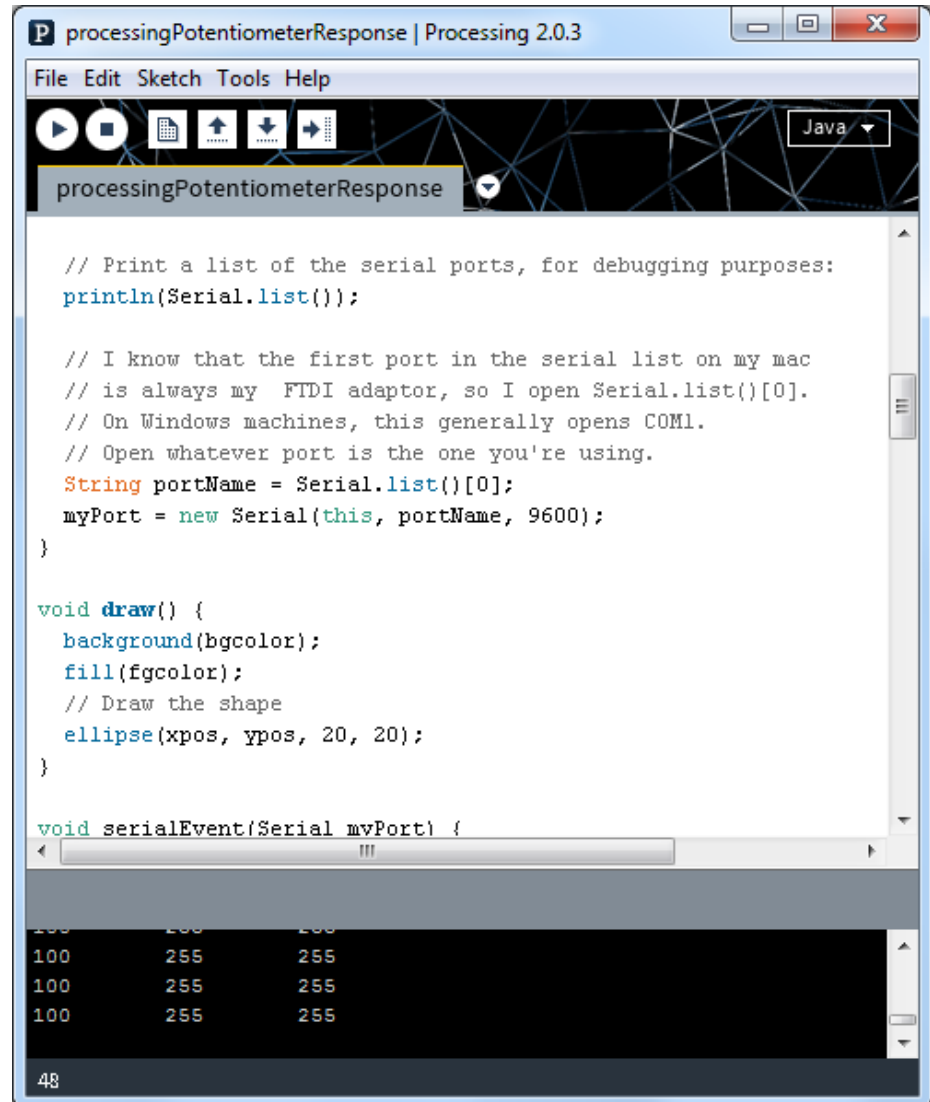
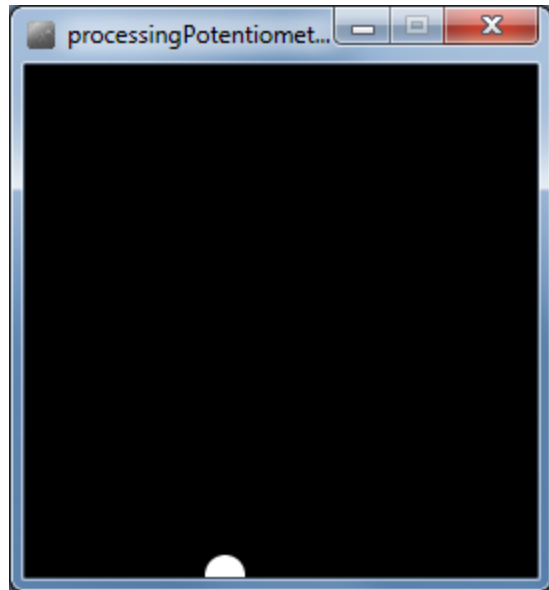
Processing potentiometer response



Pour avoir accès au port série Ouvrir
processing avec :

Cd « chemin processing »

Sudo ./processing



Epic Challenge : Pong Potentiometer

