ISN S4 : Initiation à la programmation : Processing 2

But :
- Programmation de tableau sous processing 2
- Programmation des différentes structures
- Programmation des évènements extérieurs (souris et clavier)

1. Les tableaux :

Un tableau est ce que l'on peut aussi appeler une liste. Le tableau est un élément de programmation assez abstrait ! C'est un ensemble de variables dans lequel on stocke plusieurs informations. Ces informations sont classées par cases, d'où a notion de tableau. Les informations contenues dans un tableau doivent être du même type : donc soit des caractères, soit des chiffres...

Syntaxe : **Type[] nom_du_tableau = {"élément1", "élément2", "élément3", "élément4", "élément5"};**

String[] arbre = {"chene", "sapin", "pommier", "cedre", "erable"};

a) Appeler un élément du tableau :

Syntaxe : **print (nom_du_tableau[]);** Entre les crochets, il faut renseigner le numéro, ou la position, si vous préférez, de l'élément dans la tableau

```
String[] arbre = {"chene", "sapin", "pommier", "cedre", "erable"};
print (arbre[3]);
```

I) Copier dans Processing cette partie de programme puis appuyer sur Run.
Qu'obtenez-vous?

Processing considère que le premier élément a pour numéro "0" et non "1" (Comme beaucoup de logiciel de programmation).

b) Remplacer un élément du tableau par un autre :

Syntaxe : **nom_du_tableau[] = valeur;**

```
String[] arbre = {"chene", "sapin", "pommier", "cedre", "erable"};
arbre[2] = "bouleau";
```

c) Ajouter un élément dans le tableau:

Syntaxe : **nom_du_tableau = splice(nom_du_tableau, valeur, position);**

```
String[] arbre = {"chene", "sapin", "pommier", "cedre", "erable"};
arbre = splice(arbre,"bouleau",4);
println(arbre[0]);
println(arbre[1]);
println(arbre[2]);
println(arbre[3]);
println(arbre[4]);
println(arbre[5]);
```

II) Copier dans Processing cette partie de programme puis appuyer sur Run.
Qu'obtenez-vous?

d) Supprimer un élément dans le tableau:

Il est possible de supprimer un élément d'un tableau, mais ce sera la dernière valeur
Syntaxe : **shorten(nom_du_tableau);**

e) Découper une chaine de caractère et la mettre dans un tableau

Syntaxe : **String[] nom_tableau = split(nom_de_la_chaine_de_caractères, delimitant);**

```
String arbres = "chene,sapin,pommier,cedre,erable";
String[] tableau_arbres = split(arbres, ',');
println(tableau_arbres[0]);
println(tableau_arbres[1]);
println(tableau_arbres[2]);
println(tableau_arbres[3]);
println(tableau_arbres[4]);
```

III) Copier dans Processing cette partie de programme puis appuyer sur Run.
Qu'obtenez-vous?

f) Autre façon de créer un tableau

Exemple :

```
int[] tableau = new int[2]; //le new int[2] signifie que l'on crée un tableau de deux cases
tableau[0] = 10;
tableau[1] = 25;
```

On peut aussi avoir plusieurs colonnes :

```
int[][] tableau = new int[2][3]; //le new int[2] signifie que l'on crée un tableau de deux cases
tableau[0][0] = 10;
tableau[0][1] = 20;
tableau[0][2] = 30;
tableau[1][0] = 25;
tableau[1][1] = 26;
tableau[1][2] = 27;
println(tableau[0][0]+ " " + tableau[0][1]+ " "+ tableau[0][2]);
println(tableau[1][0]+ " " + tableau[1][1]+ " "+ tableau[1][2]);
```

IV) Copier dans Processing cette partie de programme puis appuyer sur Run.
Qu'obtenez-vous?

2. L'incrémentation :

L'incrémentation est un processus particulier que l'on utilise assez souvent dans les boucles. Il consiste à augmenter la valeur d'une variable. Il est aussi possible d'utiliser la décrémentation, qui est le processus inverse, qui consiste à baisser la valeur d'une variable.

Incrémentation :

```
int a=0;
```

a=a+1; on peut aussi écrire **a++;** ou **a+=1;**

```
println(a);
```

Décrémentation :

```
int a=0;
```

a=a-1; on peut aussi écrire **a--;** ou **a-=1;**

```
println(a);
```

On a le même principe pour la multiplication, la division ...

a=a*2;⇔ a*=2;
a=a-2; ⇔ a-=2;
a=a/3; ⇔a/=3;

3. Les boucles :

Le fonctionnement des ordinateurs repose sur assez peu de fonctions :

- effectuer des calculs numériques
- accéder à des valeurs
- tester des conditions
- et répéter des actions

Les boucles permettent de répéter des actions.
Il existe deux types de boucles :

- les boucles conditionnelles crées avec la fonction while()
- les boucles itératives crées avec la fonction for()

While : de l'anglais, signifie "pendant"/"jusqu'à".
For : de l'anglais, signifie "pour"/"durant".

a) Les boucles conditionnelles :

Tant que la condition est respectée, le programme exécute les instructions contenues dans la boucle

Syntaxe :

```
while(condition)
{
 instruction 1;
 instruction 2;
 ...
}
```

```
int a = 0;
while ( a < 5 )
{
  println(a);
  a++;
}
```

V) Copier dans Processing cette partie de programme puis appuyer sur Run.
Qu'obtenez-vous?

b) Les boucles itératives :

De la condition de départ, jusqu'à la condition de fin, j'effectue les instructions contenues dans la boucle et l'itération. L'itération correspond à l'incrémentation ou la décrémentation d'une variable..

Syntaxe :

```
for ( condition de départ ; condition de fin ; itération)
{
  instruction 1;
  instruction 2;
  ...
}
```

```
int a=1000;
for(int i=0;i<5;i++)
{
  a/=2;
  println(a);
}
```

VI) Copier dans Processing cette partie de programme puis appuyer sur Run.
Qu'obtenez-vous? Est-ce normal d'après vous?

4. Les structures de contrôle :

Pour réaliser des tests conditionnels sur une variable, on utilise :

- **if :** de l'anglais qui signifie "si"
- **else :** de l'anglais qui signifie "autre"/"sinon"

Une condition peut être traduite ainsi : Je veux que cette/ces actions soi(en)t réalisée(s) si ... sinon, cette/ces action(s) ... sera(ont) exécutée(s).

Syntaxe :

```
if (condition)
{
  instruction 1;
  instruction 2;
  ...
}
else
{
  instruction 1;
  instruction 2;
  ...
}
```

Ce qu'il faut mettre entre parenthèse dans la condition :

```
if(variable_a==valeur) {} //condition égal à
if(variable_a>valeur) {}  //condition supérieur à
 if(variable_a>=valeur) {} //condition supérieur ou égal à
 if(variable_a<valeur) {} //condition inférieur à
 if(variable_a<=valeur) {} //condition inférieur ou égal à
 if(variable_a!=valeur) {} //condition n'est pas égal à
 if(condition 1 || condition 2) {} //condition  1 ou condition 2
 if(condition 1 && condition 2) {} //condition  1 et condition 2
```

Exemple :

```
int TF = 400;
int x;
 void setup()
{
  x=0;
  size(TF,TF);
}
 void draw()
{
  dessiner();
}
void dessiner()
{
  ellipse (x,200,10,10);
  x++;
  if(x>395)
  {
    x=5;
  }
}
```

VII) Copier dans Processing cet exemple puis appuyer sur Run.
Qu'obtenez-vous? Est-ce normal d'après vous?
L'ellipse laisse une grosse trace toute moche derrière elle, la solution consiste a créer une fonction nettoyer en faisant simplement background(255).
Modifier le programme dans ce sens.

5. Les fonctions Loop() et noLoop() :

Loop() est une fonction qui permet l'exécution d'une animation. En réalité, on n'en a pas besoin directement. Son "inverse" noLoop() est lui plus utile. Dans une animation, prenons l'exemple du pong, si la balle touche le mur, et bien vous perdez. Vous pouvez donc utiliser la fonction noLoop() pour arrêter le programme en finissant les tâches en cours.

Syntaxe :
noLoop;// la fonction s'utilise très simplement, mais est généralement placée dans une condition

6. La souris :

a) La position de la souris :

A tout moment, on peut avoir une position de la souris suivant X et Y à l'aide de la fonction mouseX et mouseY
Syntaxe :
mouxeX; // position en X
mouseY; // position en Y

```
int x,y;
x=mouseX;
y=mouseY;
```

VIII) Créer un programme qui ouvre une fenêtre blanche de 200*200, qui affiche un cercle de diamètre 10 pixels qui se déplace avec la souris (Cercle jaune au contour noir).

b) Clique de souris:

A tout moment, on peut avoir si l'utilisateur a cliqué.
Syntaxe :

```
if (mousePressed == true ou false)
{                                   // souris appuyée
 instruction;
}
else
{
 instruction;
}

void draw()
{
  if (mousePressed == true)
  {
    fill(200,0,155);
  }
  else
  {
    fill(255);
  }
  background(255);
  smooth();
  ellipse(50, 50, 50, 50);
}
```

```
mouseReleased ()
 {                                  // souris relachée
 instruction;
}

int couleur = 0;
void draw()
{
  fill(couleur);
  rect(25, 25, 50, 50);
}
void mouseReleased()
{
  couleur = couleur + 15;
  if (couleur > 255)
  {
    couleur = 0;
  }
}
```

IX) Modifier le programme précédent en rajoutant, si l'utilisateur clique, on nettoie la fenêtre.
Donner la différence entre mousePressed et mouseReleased?

c) Souris déplacée :

Détecte s'il y a une déplacement de la souris sans donner la valeur de la position.

Syntaxe :

```
mouseMoved()
 {                    // souris non appuyée
 instruction;
}
```

```
mouseDragged ()
 {                    // souris appuyée
 instruction;
}
```

```
int couleur = 0;
void draw()
{
  fill(couleur);
  rect(25, 25, 50, 50);
}
void mouseMoved()
{
  couleur = couleur + 15;
  if (couleur > 255)
  {
    couleur = 0;
  }
}
```

```
int couleur = 0;
void draw()
{
  fill(couleur);
  rect(25, 25, 50, 50);
}
void mouseDragged()
{
  couleur = couleur + 15;
  if (couleur > 255)
  {
    couleur = 0;
  }
}
```

7. Le clavier:

a) Touche appuyée :

A tout moment, on peut savoir si une touche et appuyée

Syntaxe :

```
if (keyPressed == true ou false)
{
 }
```

```
void draw()
{
  background(255);
  smooth();
  ellipse(50,50,90,90);
}
void keyPressed()
  {
   if (key == '1')    // chiffre 1 du clavier appuyé
  {
    fill(200,50,150);
  } else {
    fill(255);
  }
 }
```

X) Copier dans Processing cette partie de programme puis appuyer sur Run.
Qu'obtenez-vous?

a) keyCode :

La fonction keyCode permet de spécifier une touche spéciale : UP, DOWN, LEFT, RIGHT, ALT, CONTROL, SHIFT, BACKSPACE, TAB, ENTER, RETURN, ESC, et DELETE)

Syntaxe :

```
keyCode
```

```
void draw()
{
  background(255);
  smooth();
  ellipse(50,50,90,90);
}
```

```
void keyPressed()
  {
   if (keyCode == ENTER)
   {
    fill(200,50,150);
  } else {
    fill(255);
  }
}
```

8. Lien hypertexte :

Avec Processing, il est possible de faire des liens vers des URL internet ! Imaginons que vous ayez un carré gris , si vous cliquez dans ce carré, on ouvre une page internet : http://www.lyceemermoz.com/spip/spip.php

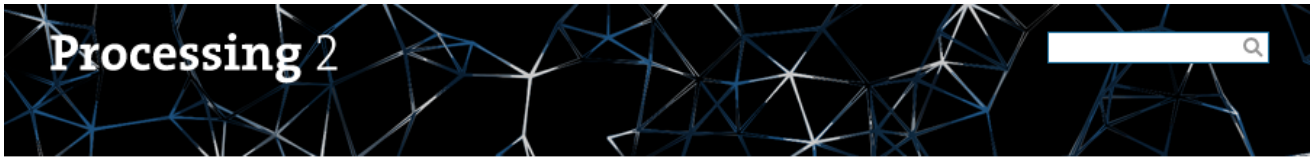
Syntaxe :
link("URL");

link ("http://www.lyceemermoz.com/spip/spip.php");

XI) Créer un programme qui affiche une fenêtre de 500*500 à fond blanc.
Dans cette fenêtre, dessiner un rectangle gris de 200*100 au milieu.
Si l'utilisateur clique dans ce carré, on ouvre la page web de lycée Jean Mermoz.

9. Aide sur les fonctions :

Vous pouvez avoir la liste de toutes les fonctions disponibles sous Processing en faisant Help puis Référence, chaque fonction est détaillée.



Language (A-Z)
Libraries
Tools
Environment

Reference. The Processing Language was designed to facilitate the creation of sophisticated visual structures.

Structure

() (parentheses)
, (comma)
. (dot)
/* */ (multiline comment)
/** */ (doc comment)
// (comment)
; (semicolon)
= (assign)
[] (array access)
{} (curly braces)
catch
class
draw()
exit()
extends
false
final
implements
import
loop()
new

Shape

createShape()
loadShape()
PShape

2D Primitives
arc()
ellipse()
line()
point()
quad()
rect()
triangle()

Curves
bezier()
bezierDetail()
bezierPoint()
bezierTangent()
curve()
curveDetail()

Color

Setting
background()
clear()
colorMode()
fill()
noFill()
noStroke()
stroke()

Creating & Reading
alpha()
blue()
brightness()
color()
green()
hue()
lerpColor()
red()
saturation()

10. Pour aller plus loin :

XII) Créer un programme qui affiche une fenêtre de 500*500 à fond blanc.
Dans cette fenêtre, dessiner ET (programme donné) qui suivra les mouvements de la souris .
Dès qu'on clique, tout est effacé.
Enregistrer sous Prog_XIII.pde