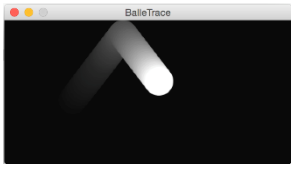
ISN S14: La programmation orientée objet

Objectif :
Découvrir l'utilisation de la programmation orientée objet sous processing.

Il y a 2 manières principales de programmer :
- La programmation **procédurale** car on appelle de nombreuses procédures.
- La **programmation orientée objet** ou **POO**.

Nous allons simplement montrer la différence sur un exemple, le rebond d'une balle sur les bords de la fenêtre.

1. Programmation procédurale :



I) Compléter et tester le programme sous forme procédurale pour une balle qui rebondit contre les parois d'une fenêtre de 400 * 200 pixels.
Sauvegarder sous Exemple1.pde

```
float x, y; //position
float vitesseX, vitesseY; // vitesse
color couleur; // couleur
void setup() {
  smooth(); //Lissage des dessins
  size(____,_____); //Taille de la fenêtre
  x=________; //balle au milieu de la fenêtre
  y=________;
  vitesseX = 2 + random(-1,1);
  vitesseY = 2 + random(-1,1);
  couleur=color(random(0,255), random(0,255), random(0,255));
  noStroke();
}
void draw() {
  fill(0, 10); // voile noir (alpha=10)
// on recouvre la fenêtre d'un voile noir
// pour estomper la trace de la balle
  rect(0, 0, width, height);
  //Déplacement et affichage de la balle
  bouger();
  rebondir();
  afficher();
}
// Dessin de la balle
void afficher() {
  fill(_______);
  ellipse(___,___, 40, 40);
}
// Déplacement
void bouger() {
  x = x + vitesseX;
  y = y + vitesseY;
}
// Rebondir
void rebondir() {
  //Si la balle touche un mur, elle rebondit
  if (x > _______ || x < ______) {
    vitesseX = vitesseX * -1;
  }
  if (y > _______ || y < _______) {
    vitesseY = vitesseY * -1;
  }
}
```

Déclaration des variables, globales pour simplifier

Initialisation des variables

Boucle principale :
on estompe les traces précédentes, et on appelle 3 procédures pour gérer l'évolution de la balle

Les 3 procédures afficher, bouger et rebondir.

2. Programmation orientée objet :

Si l'on observe que les variables globales ne concernent qu'une balle, et que les procédures sont également dédiées à la gestion de balles, on pourrait vouloir rassembler tout cela dans un **objet Balle**.

Une balle possède des caractéristiques (les coordonnées de son centre, sa couleur…) appelées **propriétés** et des actions (se dessiner, rebondir…) appelées **méthodes**.

Toutes ces propriétés et méthodes sont définies dans une **classe**.

Voici la classe Balle par exemple:

```
class Balle {
  //Déclaration des paramètres de base de la balle
  float x; //position
  float y;
  float vitesseX; // vitesse
  float vitesseY;
  color couleur; // couleur
```

Mes **propriétés** sont conservées dans ces variables

```
  //Constructeur de la balle, appelé quand on fait new Balle(...)
Balle(float nouvX, float nouvY, color nouvCouleur) {
  x          = nouvX;
  y          = nouvY;
  couleur    = nouvCouleur;
  vitesseX = 2 + random(-1, 1);
  vitesseY = 2 + random(-1, 1);
}
```

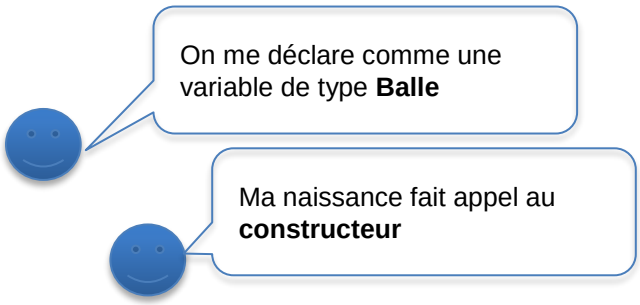
La naissance d'une balle demande que ses propriétés soient définies. C'est le rôle du **constructeur** qui a le même nom que la classe, ici Balle.

```
//Dessin de la balle
void afficher() {
  fill(couleur);
  ellipse(x, y, 40, 40);
}

void bouger() {
  x = x + vitesseX;
  y = y + vitesseY;
}

void rebondir() {
  //Si la balle touche un mur, elle rebondit
  if (x > width-20 || x < 20) {
    vitesseX = vitesseX * -1;
  }
  if (y > height-20 || y < 20) {
    vitesseY = vitesseY * -1;
  }
 }
}
```

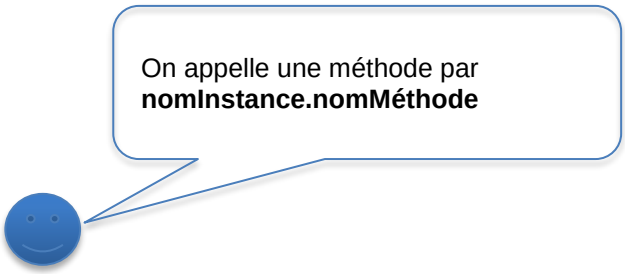
Voici mes méthodes ! Je suis donc capable de m'afficher, bouger et rebondir.

Maintenant que Balle est définie comme un objet, on peut en obtenir des **instances**, c'est à dire des copies ayant leurs caractéristiques propres, mais pouvant faire toutes les mêmes actions. Je pourrais ainsi avoir N balles différentes, disposées en des endroits différents mais toutes capables de s'afficher, de bouger, de rebondir.

Le programme principal devient :




```
//Déclaration d'une instance de Balle
Balle maBalle;
void setup() {
  smooth(); //Lissage des dessins
  size(400, 200); //Taille de la fenêtre
  noStroke();
  // on initialise une Balle nommée maBalle
  maBalle = new Balle(width/2, height/2, color(random(0,255), random(0,255),
random(0,255)));
}
void draw() {
  fill(0, 10);
  rect(0, 0, width, height);
  // appel des méthodes
  maBalle.bouger();
  maBalle.rebondir();
  maBalle.afficher();
}
class Balle {…}
```



II) Tester le programme orienté objet pour une balle. Obtient-on la même chose ?
Sauvegarder sous Exemple2.pde

Finalement, les 2 programmes sont très proches. Mais si je veux gérer plusieurs balles, c'est facile en POO, moins naturel en programmation procédurale.

III) Modifier le programme pour créer un tableau de 3 balles en mouvement ?
Sauvegarder sous Exemple3.pde