

TP : Exercice 2 :

Énoncé :

1. Exercice sur les structures de contrôle simple if-then-else à une condition

1) Écrivez un sketch mettant en scène une balle (« balle.png ») chutant verticalement du milieu de la fenêtre et rebondissant sur le sol. À chaque rebond, la balle atteint une altitude de plus en plus basse, jusqu'à devenir immobile (Vous diviserez par 2, ou bien par 3, la hauteur de 2 rebonds successifs).

2) Jouez un son au moment de la collision avec le sol :

a) Installez la librairie permettant de jouer du son :

- Sketch > importer une librairie > ajouter une librairie :

- Installez la librairie « Sound » de « The Processing Foundation ».

b) Importez la librairie dans votre sketch en écrivant en haut du programme :

```
import processing.sound.* ;
```

c) Pour lire un son :

```
// chargez le fichier son (qui est dans data/ du répertoire du sketch)
```

```
SoundFile sound = new SoundFile(this, dataPath("bounce.mp3"));
```

```
// jouez le son
```

```
sound.play() ;
```

3) Ajouter la possibilité de définir la position initiale de la balle à l'aide de la souris, avant de lâcher la balle (« drag and drop »). Utilisez les fonctions «mousePressed», «mouseDragged» and «mouseReleased».

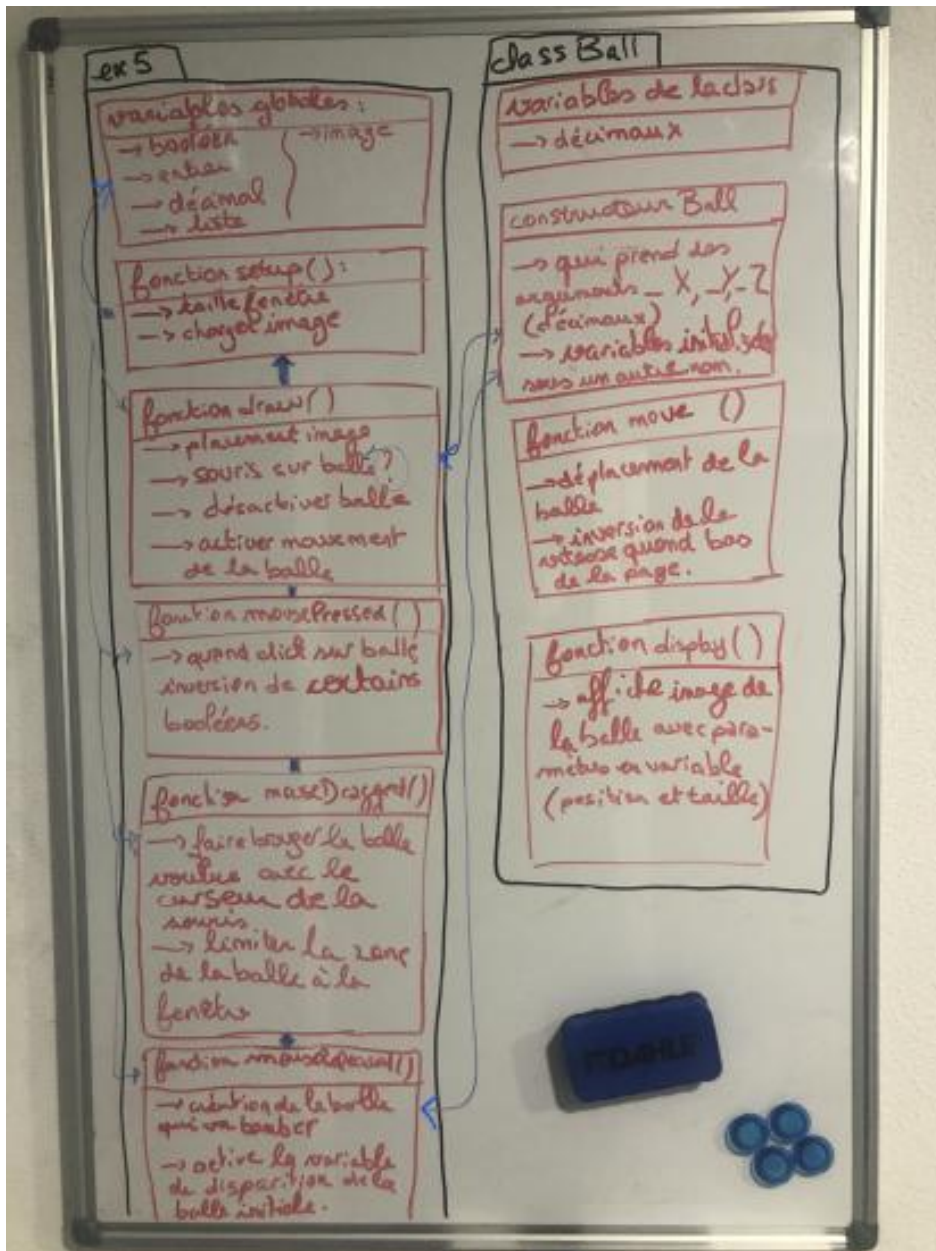
4) Ajoutez une deuxième balle (sans gestion des collisions entre balles).

5) En quoi la définition et l'utilisation d'une classe « Balle » réduit la longueur et la complexité du programme facilitant ainsi l'ajout de plusieurs balles ?

Brouillon :



Diagramme de classe :



Exemples de ce qui se passe (graphique et console) :

Il ne doit rien se passer dans l'interface de la console.

Dans l'interface graphique nous pouvons donc donner des exemples de ce qui se passe :

- La balle en question doit être une image et non un cercle généré par processing ;
- Si l'on maintient clique gauche sur une balle et que l'on bouge la souris, la balle doit sur le pointeur ;
- Si on lâche la souris la balle doit retomber ;
- La balle doit faire un son quand elle rentre en contact avec le sol ;
- La balle doit remonter de la moitié de la distance de descente précédente.

Capture d'écran du code final :

```
30 PImage img;
31 import processing.sound.* ;
32 SoundFile sound;
33 //déclaration d'une liste appelée "liste" d'éléments de la class Ball
34 ArrayList<Ball> liste= new ArrayList<>();
35 //déclaration de variables de type entier
36 int ballWidth = 30;
37 int locY=180, locX=500;
38 int larg1=30, larg2=30;
39 int locX2=140, locY2=180;
40 //déclaration d'une variable de type décimal
41 float souris1,souris2,compteur;
42 //déclaration de variables de type booléen
43 boolean overBall1=false;
44 boolean overBall2=false;
45 boolean locked1=false;
46 boolean locked2=false;
47 boolean disparu1=false;
48 boolean disparu2=false;
49
```

```

50 //déclaration de la fonction setup:
51 void setup() {
52     //initialisation taille de fenêtre
53     size(640, 360);
54     //chargement de l'image sous le nom défini dans les fichiers
55     img = loadImage("balle.png");
56     sound = new SoundFile(this, dataPath("ball.wav"));
57 }
58 }
59
60 //ouverture de la fonction draw:
61 void draw() {
62     background(255);
63     //centrer les images
64     imageMode(CENTER);
65     //déclaration des images avec les coordonnées x,y puis la taille z,q
66     image(img, locX, locY, larg1, larg1);
67     image(img, locX2, locY2, larg2, larg2);
68     /*calculs permettant de vérifier si la souris se trouve sur l'image
69     si cette variable est négative on considère que nous sommes sur l'image*/

```

```

70     souris1=sqrt(sq(mouseX-locX)+sq(mouseY-locY))-13;
71     souris2=sqrt(sq(mouseX-locX2)+sq(mouseY-locY2))-13;
72     //si on est sur l'image1 la variable voulue passe sur vraie
73     if (souris1<0){
74         overBall1=true;
75     }
76     //si on est sur l'image2 la variable voulue passe sur vraie
77     if(souris2<0){
78         overBall2=true;
79     }
80     //si la variable disparu1 ou 2 passe à vrai l'image disparaît (largeur=0)
81     if(disparu1){
82         larg1=0;
83         //assure de ne pas pouvoir reposer une seconde balle en cliquant nul part
84         locked1=false;
85     }
86     if (disparu2){
87         larg2=0;
88         locked2=false;
89     }

```

```

90 //si une balle est créé, la liste va être de taille 1 et va donc animer
91 //la balle avec les fonction ball.move et display définis dans la class Ball
92 //Pour sélectionner la bonne balle on utiliser liste.get(i) = bon élément dans la liste.
93 for (int i = liste.size()-1; i>=0; i--) {
94     Ball ball = liste.get(i);
95     ball.move();
96     ball.display();
97 }
98 }
99
100 //ouverture de la fonction mousePressed() (quand on appuie sur la souris):
101 void mousePressed() {
102     //quand on est sur la balle la variable est à vraie
103     if(overBall1){
104         /*débloque d'autres conditions quand la souris est pressée (des id dans
105         d'autres fonction plus basses dans le programme)*/
106         locked1 = true;
107         //repasse à faux pour ne pas pouvoir recliquer sur ses coordonnées
108         overBall1=false;
109     }else{
110         //sécurité qui n'a pas forcément de rôle
111         locked1=false;
112     }

```

```

113     if(overBall2){
114         locked2 = true;
115         overBall2=false;
116     }else{
117         locked2=false;
118     }
119 }
120
121
122 //ouverture de la fonction de mouvement de la souris:
123 void mouseDragged(){
124     //si la balle1 est bien sélectionnée avec la souris
125     if (locked1){
126         //on prend en coordonnées la souris qui bouge
127         locX=mouseX;
128         locY=mouseY;
129         /*Les 4 prochains if permettent de ne pas faire sortir la balle
130         de la fenêtre afin de toujours l'avoir en vision*/
131         if(locX<0){
132             locX=0;

```

```

133     }
134     if(locX>width){
135         locX=width;
136     }
137     if(locY<0){
138         locY=0;
139     }
140     //le -14 permet de ne pas aller au plus bas de la page et que
141     //l'effet ne soit pas "irréel" dans le pied de page.
142     if(locY>height-14){
143         locY=height-14;
144     }
145 }
146 if (locked2){
147     locX2=mouseX;
148     locY2=mouseY;
149     if(locX2<0){
150         locX2=0;
151     }
152     if(locX2>width){
153         locX2=width;
154     }
155     if(locY2<0){

```

```

156         locY2=0;
157     }
158     if(locY2>height-14){
159         locY2=height-14;
160     }
161 }
162 }
163
164 //déclaration de la fonction de relachement de la souris
165 void mouseReleased(){
166     //si on a la balle en sélection (l'une ou l'autre=else if)
167     if(locked1){
168         //la variable qui active la disparition plus haut
169         disparu1=true;
170         /*on ajoute une nouvelle balle en utilisant le constructeur de la classe
171         avec les paramètres que l'on veut et que l'on pourra réutiliser
172         dans notre cas la balle part de la position à laquelle on lache la balle
173         ainsi que sa taille de la balle.*/
174         liste.add(new Ball(locX, locY, ballWidth));
175     }else if (locked2){
176         disparu2=true;
177         liste.add(new Ball(locX2, locY2, ballWidth));
178     }
179 }

```

Il y a aussi une classe Ball :

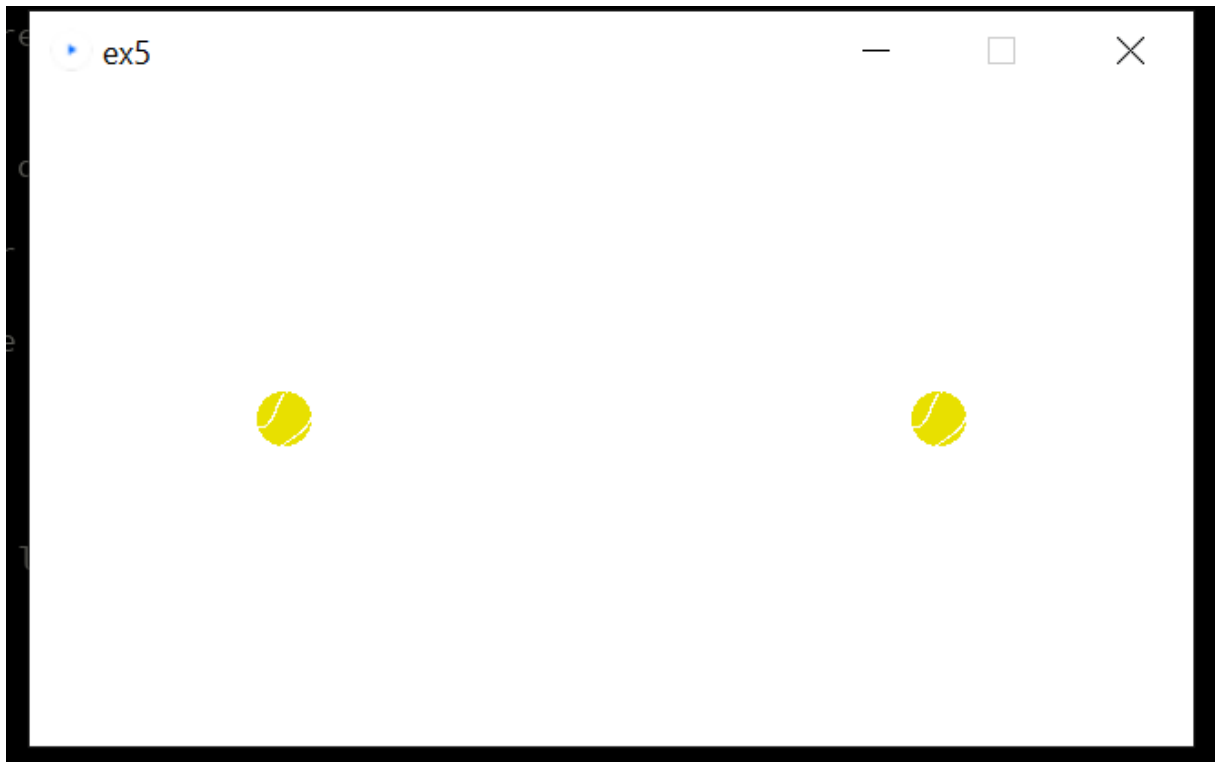
```

1 //ouverture de la class Ball
2 class Ball {
3     //déclaration des variables globales à la classe
4     float x;
5     float y;
6     float speed;
7     float gravity;
8     float sizeBall;
9     boolean test=true;
10
11     //ouverture du constructeur avec le même nom que la class
12     //qui récupère les arguments rentrés dans le programme
13     Ball(float _X, float _Y, float _W) {
14         //on récupère les variables dans des variables de la class déclaré plus haut
15         x = _X;
16         y = _Y;
17         sizeBall = _W;
18         //vitesse de la balle
19         speed = 0;
20         //déclaration de la gravité
21         gravity = 0.1;
22     }
23     //ouverture de la fonction mouve()
24     void move() {
25
26         //calcul permetant d'augmenter la variable de la "vitesse"
27         speed = speed + gravity;
28         //on ajoute à la position la variable "vitesse" qui fait avancer la balle
29         y = y + speed;
30         //mettre le son en avance que la balle touche le sol afin d'éviter des problèmes de délais
31         //il y a une valeur max afin d'éviter que le son sature à la fin
32         if(y<345 && y>185 && speed==abs(speed) && test){
33             sound.play() ;
34             //variable qui empêche de rejouer le son en remontant
35             test=false;
36         }
37         //si on arrive au bas de page (-14 pour éviter que la balle soit coupée)
38         if (y > height-14) {
39             //la vitesse va en sens inverse pour remonter
40             speed = speed * -0.70;
41             //la hauteur est donc celle du bas de la page puis va remonter etc...
42             y = height-14;
43             test=true;
44         }
45         //ouverture de la fonction display qui s'occupe de l'affichage de l'image avec
46         //les paramètres voulus en position et taille.
47
48         void display() {
49             image(img, x, y,sizeBall,sizeBall);
50         }
51     }
52 }

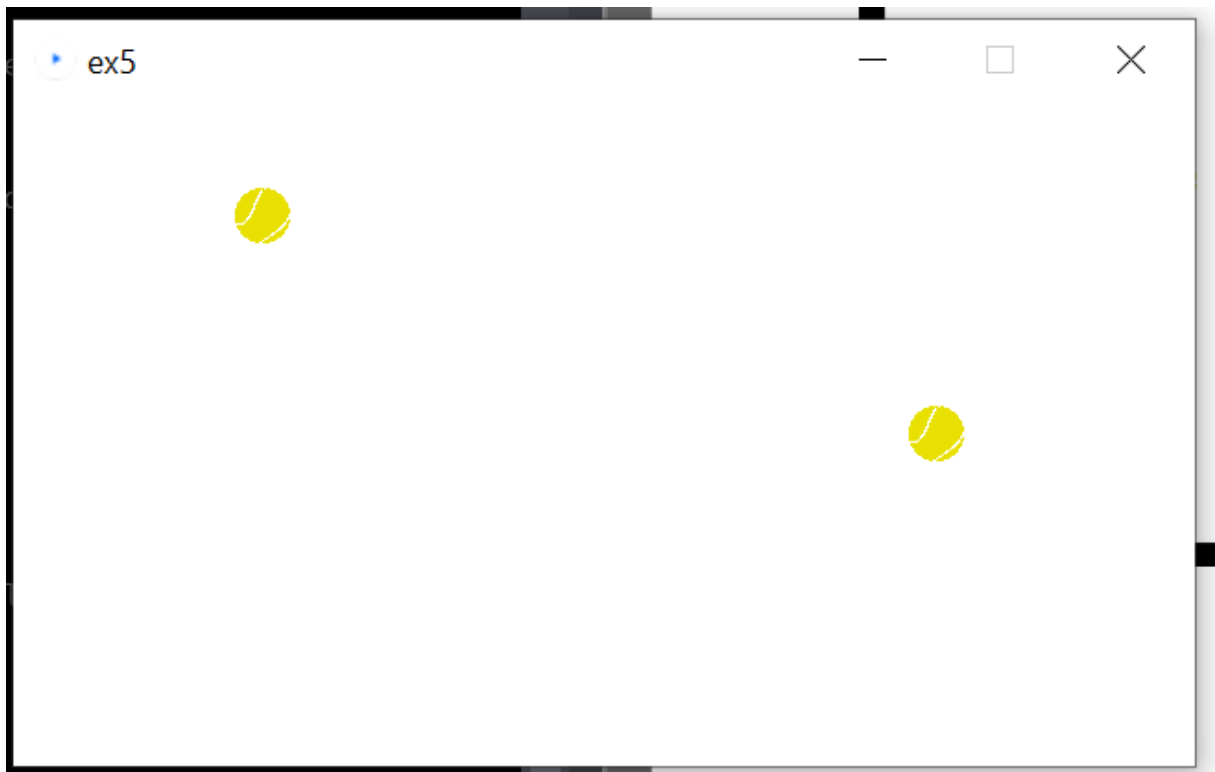
```

Capture d'écran du l'exécution (interface graphique) :


Au départ :



Puis on sélectionne et fait bouger une balle :




La balle tombe et rebondit en faisant du son :

 ex5

— ☐ ✕



Et enfin la balle reste au sol (nous pouvons faire de même avec la seconde balle) :

 ex5

— ☐ ✕



Avis/difficultés/problèmes rencontrés/piste d'amélioration/remarques :

- J'ai eu du mal à comprendre comment fonctionnaient les listes et surtout comment elle fonctionnait dans le cas de notre code ;
 - Je n'ai pas éprouvé de problème pour que la balle est une allure réelle avec une accélération lors de la chute. Au contraire, faire remonter la balle à $\frac{1}{2}$ de la distance à été plus compliqué, et j'ai dû procéder par des tests successif avec un affichage en console ;
 - L'utilisation et l'application du son a aussi été complexe afin d'éviter d'une saturation lors de la stagnation de la balle au bas de l'écran, en plus du décalage de son qui apparait (il est possible qu'il reste une erreur quand les deux balles sont lâchées en même temps mais je n'ai jamais eu ce cas lors de mes tests) ;
 - L'implémentation de l'image ne m'a pas posé trop de problème ;
 - Cet exercice est selon moi ma plus grande réussite dans cette matière, je suis passé par de nombreux codes et j'ai dû utiliser de nombreuses ressources (votre cours ou bien de la documentation officielle Processing). J'y ai passé beaucoup de temps pour avoir un résultat qui me convient et je pense que je vais poursuivre ce code dans le futur pour y intégrer le lancement des balles par rapport à la vitesse du curseur qui la lance. Je pourrais aussi implémenter la collision de balle et éventuellement un décor à l'arrière-plan.
- ⇒ Dans ce code l'intérêt d'avoir intégrer créer une class pour les balles est que ça simplifie grandement le code. Ça le réduit aussi en longueur car nous n'avons pas besoin de mettre le code de chaque balles mais une seul code pour deux balles suffit.