TP Balles en mouvement

Allan BOOZ

13 novembre 2018

Résumé

Dans ce TP, il s'agira d'écrire un programme qui affiche des balles en mouvement dans une fenêtre.

1 Introduction

Le but de ce TP consistera à créer un programme capable de faire bouger des balles dans une fenêtre tout en utilisant des Threads comme vu précédement durant le TP. Ainsi pour permettre le bon fonctionnement de ce projet, nous devons respecter quelques conditions :

— Les balles ne peuvent sortir du cadre de la fenêtre et ricochent contre les bords.
— Un bouton permet de démarrer/arrêter le mouvement de toutes les balles. Lorsque les balles sont en mouvement, ce bouton a pour étiquette "stop" et lorsque les balles sont à l'arrêt il a pour étiquette "start".
— Un bouton "+" permet d'ajouter une nouvelle balle de couleur aléatoire. Lorsqu'un seuil fixé au préalable est atteint, plus aucune balle ne peut être ajoutée.
— Un bouton "-" permet de supprimer une balle.
— Lorsqu'une collision se produit, les balles impliquées sont supprimées.
— Un score est affiché en permanence et est incrémenté à chaque collision.
— Une horloge affiche le temps écoulé pendant que les balles sont en mouvement ; cette horloge doit stopper son décompte lorque les balles sont arrêtées et reprendre son décompte lorsque les balles sont à nouveau en mouvement.

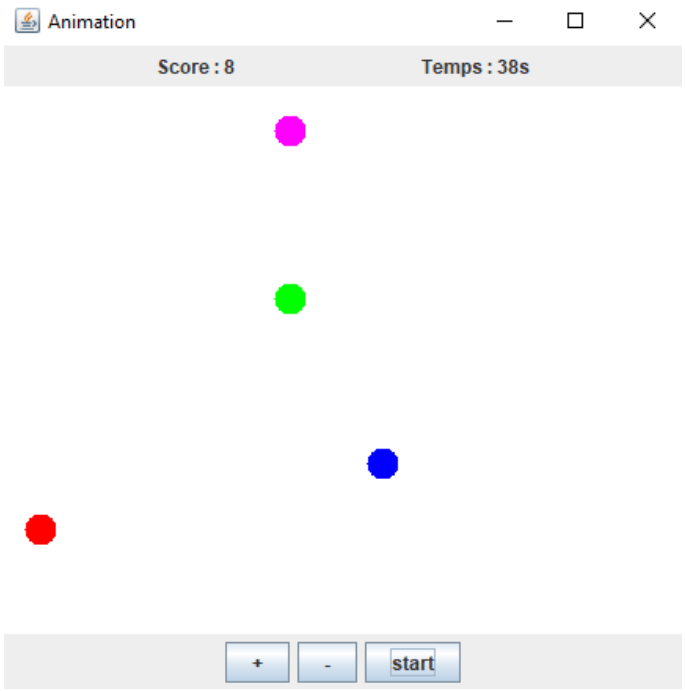

FIGURE 1 – Vue d'ensemble du programme

2 Le corps du projet

2.1 L'Objet Cercle

Dans un premier temps, nous aurons besoin d'un Objet Cercle afin de représenter nos balles. Pour cela, notre programme sera entièrement écrit en Java : il nous faudra un constructeur Cercle qui aura en paramètre des coordonées X et Y ainsi que d'un diamètre pour représenter la balle.

```
int diametre,x,y;
private Color couleur;
boolean backX = false;
boolean backY = false;

public Cercle(int x, int y, int r) {
    super();
    this.x = x;
    this.y = y;
    this.diametre = r;
}
```

FIGURE 2 – Constructeur du cerlce

Et puis quelques méthodes, afin de récuperer ou modifier les paramètres d'un cercle mais aussi rajouter une méthode de mouvement en utilisant des conditions if/else pour rediriger la balle lorsqu'elle touchera l'un des bords de notre fenêtre.

```
public int getX() {return x;}
public int getY() {return y;}

public void setX(int x) {this.x = x;}
public void setY(int y) {this.y = y;}

public int getDiametre() {return diametre;}
public Color getColor() {return couleur;}

public void setDiametre(int r) {diametre = r;}
public void setColor(Color c) {couleur = c;}

public void draw(Graphics g) {
    g.fillOval(x, y, diametre, diametre);
}
```

FIGURE 3 – Des méthodes utiles

Ici, les méthodes utiles nous permmettent de récupérer ou de changer les informations d'un Cercle, et chaque cercle aura ainsi ça propre méthode de mouvements.
Par ailleurs, nous aurons besoin de créer 2 classes : l'une permettant de dessinner l'objet Cercle en mouvmement (Panneau.java) et l'autre une fenetre (Fenetre.java) où nous pourrons visualiser les balles, le timer, les boutons, ... etc.

```java
public void  move(Panneau pan) {
    int x = getX(); int y = getY();

    // Si la coordonnée x est inférieure à 1, on avance.
    // Si la coordonnée x est supérieure à la taille du Panneau moins la taille du rond, on recule
    if (x < 1) backX = false;
    if (x > pan.getWidth() - getDiametre()) backX = true;

    // Idem pour l'axe y
    if (y < 1) backY = false;
    if (y > pan.getHeight() - getDiametre()) backY = true;

    // Si on avance, on incrémente la coordonnée
    if (!backX) {setX(++x);}
    else setX(--x);

    // Idem pour l'axe Y
    if (!backY) {setY(++y);}
    else setY(--y);
}
```

FIGURE 4 – La méthode de mouvement

2.2 L'Objet Panneau

Par le biais de cette classe, on pourra dessiner nos balles. Il faudra récuperer les differents objets Cercle dans une liste globale déclarée dans cet objet Panneau, et utiliser la méthode paintcomponent() pour pouvoir dessiner nos balles dans la fênetre. Cependant les balles devront être en mouvement, elle vont donc changer de position et laisser derrière elles la marque de leur passage. Pour éviter cela, on doit d'abord dessiner un rectangle blanc puis faire appel dans la méthode paintcomponent() la méthode draw(g) de l'objet Cercle.

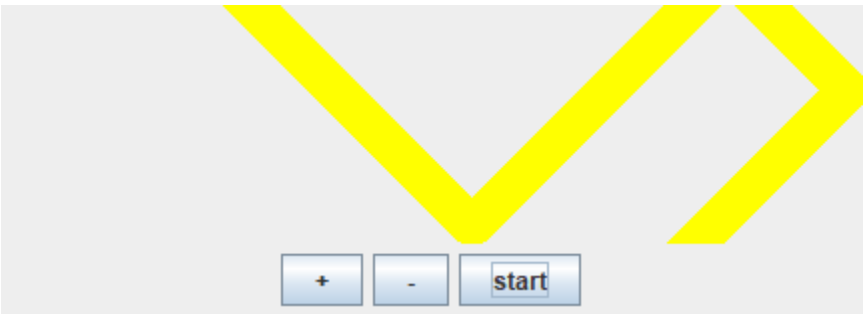

FIGURE 5 – Balle en mouvement sans le rectangle blanc.

FIGURE 6 – Balle en mouvement avec le rectangle blanc.

2.3 L'Objet Fenetre

C'est ici que nous devont mettre en place une fenêtre pour contenir notre objet Panneau car autrement, nous pourront jamais la visualiser. Cet objet Fenêtre va aussi nous servir pour le placement des boutons : dont toute action est géré par une boucle switch ; le placement du score, du temps et des actions ou écoute faite sur cette fenêtre.

```
public Fenetre() {
    JFrame fen = new JFrame();
    Container conteneur = fen.getContentPane();
    conteneur.setLayout(new BorderLayout());
    conteneur.add(pan);

    //[gestion de la fermeture de la fenêtre
    fen.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

    fen.setTitle("Animation");
    fen.setSize(450, 450);
    fen.setLocationRelativeTo(null);
    fen.setVisible(true);
```

FIGURE 7 – Initialisation d'une fenetre (avec "pan" l'objet panneau)

FIGURE 8 – Affichage des boutons, du score et du temps

2.4 Le Thread Mouvement

Pour ce qui est de la classe Mouvement, c'est un thread qui va parcourir notre liste de Cercle écrite dans la classe Panneau.java, et appliquer dans une méthode run(), sur chaque Cercle, la méthode move(), voir section 2.1, et cela dans une boucle pour continuer le mouvement. Tant qu'un booléen "pause" reste faux, les balles seront en mouvement sinon elle seront en "attente". Ceci qui d'ailleurs est géré dans notre classe Fenêtre grâce aux boutons "+" et "-".

De plus, ce thread contient la méthode collision qui calcule la distance entre deux centres des cercles présent dans la liste des Cercles, et regarde si cette distance est supérieure ou inférieure à la somme des rayons. Si cette distance est inférieur alors il y a bien une collision, on sauvegarde l'indice de ces cercles, et on les supprime de la liste. Cependant l'un de nos objectifs est aussi de pouvoir stopper le programme. par conséquent on introduit une méthode synchronized relance() qui ne contient juste qu'un notifyAll() et à notre méthode run() on ajoute également "synchronized" pour permettre le l'arrêt et le redémarrage du programme.

2.5 Le Thread ThreadCollisionAffiche

Ensuite, on souhaite pouvoir afficher le nombre de collision durant le programme. Pour cette raison, on crée un thread qui va permettre l'affichage en lui donnant en paramètre une JLabel, un objet qui contient du texte, et un nombre correspondant à notre nombre de collision. Il suffira ensuite dans une méthode run() de changer à chaque fois le texte de la JLabel pour afficher le score. On démarre ensuite notre thread dans notre classe Mouvement lorsqu'on vérifie qu'il y a bien une collision.

```java
public class ThreadCollisionAffiche extends Thread{
    private JLabel lab = new JLabel();
    private int n;

    public ThreadCollisionAffiche(JLabel l, int n) {
        this.lab = l;
        this.n = n;
    }
    public void run() {
        try {
            this.lab.setText("Score : "+Integer.toString(n));
            sleep(2);
        }catch(InterruptedException e) {

        }
    }
}
```

FIGURE 9 – Nombre de collision

2.6 Le Thread DesactivePause

Enfin, il est nécessaire d'avoir 2 threads pour permettre à notre programme de se relancer. Or plus haut, nous avons mentionné la création d'une méthode synchronized, voir section 2.4. On utilisera dans ce cas, la méthode synchronized "relance" dans un nouveau thread nommé "DesactivePause". Grâce à la méthode run(), on vérifie dans une boucle que le programme n'est pas en mode pause, dans le cas contraire, on on fait appel à la méthode synchronzed relance() de la classe Mouvement.

```java
public synchronized void run(){
    boolean a = move.getPause();
    while(true) {
        if(a == false) move.relance();
    }
}
```

FIGURE 10 – Relancer le mouvement des balles

3 Conclusion

En conclusion, ce TP avait pour but de manipuler des Threads à travers un projet, c'est-à-dire savoir gérer le mouvement des balles, savoir établir un programme qui détecte des collisions, afficher le nombre de collisions dans la fenêtre ou encore stopper puis redémarrer le mouvement des balles grâce à un bouton. De plus, si nous devrions améliorer quelque chose, se serait de remplacer l'objet Timer par un Thread qu'on aurait fait nous-même.

4 Bibliographie

4.1 Balle en mouvement

1. https://openclassrooms.com/fr/courses/26832-apprenez-a-programmer-en-java/23193-le-fil-rouge-une-animation

4.2 Collision

1. https://openclassrooms.com/forum/sujet/collision-entre-2-cercles
2. http://sdz.tdct.org/sdz/eorie-des-collisions.html

4.3 Timer

1. https://docs.oracle.com/javase/7/docs/api/javax/swing/Timer.html