Balles en mouvement

Ettouil Boudemia Yanis, L3 informatique

13 novembre 2018

Résumé

Le but de ce programme est d'afficher une fenetre qui contient des balles en mouvement

1 Introduction

- 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.

2 Corps

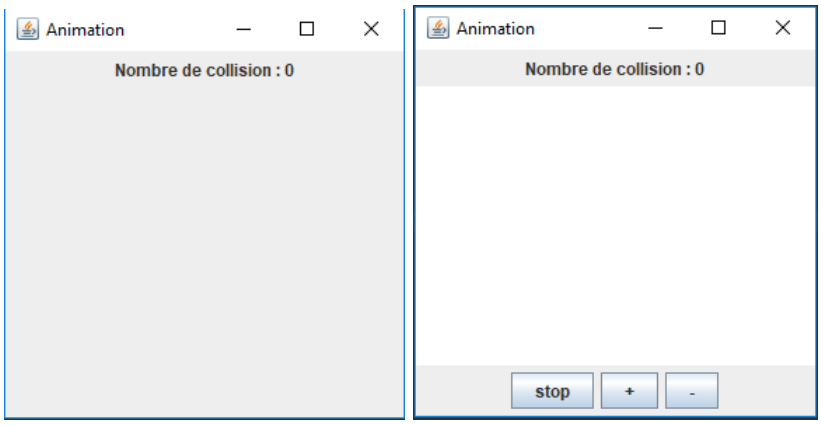
2.1 Architecture de l'application

Afin de produire le programme demandé, il a fallu réfléchir en détail à la façon de s'y prendre et essayer de déceler les différentes parties du code que j'ai décidé de réaliser en Java :

- Une Classe **Cercle** pour créer et gerer leurs déplacements.
- Une classe **Panneau** dont le but sera de de remplir la fenêtre avec les cercles ainsi que les boutons.
- Une classe **Fenetre** qui servira à créer notre fenêtre dans laquelle tout va se passer, et qui fera également office de fichier principal contenant le **main**.

2.2 Création et msie en place de la fenêtre

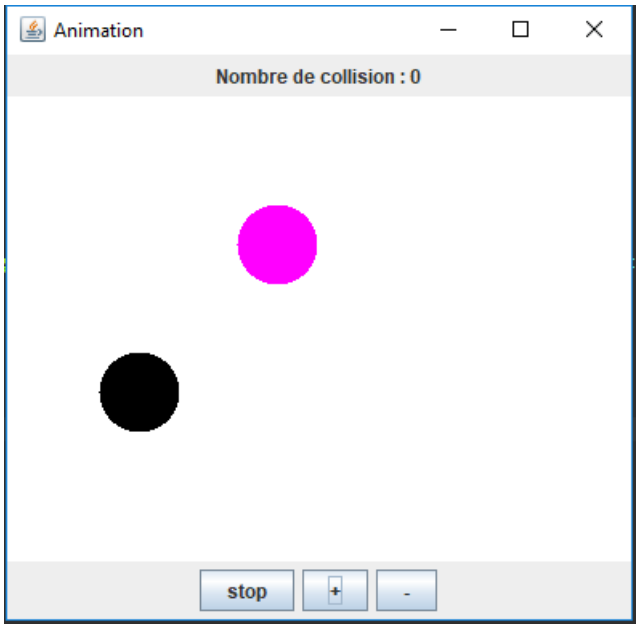
Dans un premier temps il faut créer une fenêtre fonctionelle (*image de gauche*).
Une fois cette fenêtre réalisée nous pouvons passer à l'implémentation de nos boutons(*image de droite*).



2.3 Deplacement des cercles

Notre environnement de travail fait, passons à l'intégration des cercles en mouvement. En effet il suffit de modifier les coordonnées d'un cercle pour le voir bouger, cependant il doit rester dans l'enceinte de la fenêtre et pour ce faire nous devons faire en sorte qu'il rebondisse sur ses bords. Nos variables *backX* et *backY* sont donc nos pilliers sur lesquels reposent ces rebonds.

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

if (x < 1) backX = false;
if (x > pan.getWidth() - getRayon()) backX = true;

if (y < 1) backY = false;
if (y > pan.getHeight() - getRayon()) backY = true;

if (!backX) {setX(++x);}
else setX(--x);

if (!backY) {setY(++y);}
else setY(--y);

}
```



2.4 Ajout et suppresion de cercles

Plus tôt nous avons intégré des boutons "+" et "-", dont leur fonction respective est d'ajouter et de supprimer un cercle de la fenêtre. Ici nous avons choisi de stocker les cercles dans une liste d'objets de type **Cercle** et de les faire apparaitre aléatoirement lors de leur ajout.

```
switch(bt_titre) {
case "+":
if(pan.cercle.size() < 10) {
Cercle c1 = new Cercle((int)(Math.random()*(300-0)+1),(int)(Math.random()*(300-0)+1),50);
c1.setColor(Tab[(int)(Math.random()*8)]);
pan.cercle.add(c1);
}
break;

case "-":
if(pan.cercle.size() > 0) {
pan.cercle.remove(pan.cercle.get(pan.cercle.size()-1));
pan.repaint();
}
break;
case "stop":
start_stop.setText("start");
t1.interrupt();
break;
case "start":
start_stop.setText("stop");
t1.start();
break;
}
```

3 Conclusion

Nous avons donc au final une fenêtre fonctionelle qui nous permet d'y ajouter des cerlces mobiles, qui, lors d'une collision avec un bord vont rebondir ou bien s'effacer à la rencontre d'un autre cercle. Des boutons pour gere l'ajout ou bien la suppression de cercles.

Références

[1] https://docs.oracle.com/javase/tutorial/uiswing/index.html