TP en temps Libre de Programmation Orientée Objet

2025-2026

Mohammed-Amine Jellil

Table des matières

1 Conception de la partie “Balles”

Dans cette première partie du projet, l'objectif était d'implémenter un simulateur simple permettant de manipuler et d'afficher des balles se déplaçant dans une fenêtre graphique. Cette section décrit les choix de conception réalisés, la structure des classes développées ainsi que les résultats obtenus.

1.1 Représentation des balles

Le sujet impose qu'une balle soit simplement définie par ses coordonnées (x, y). Pour cela, le choix a été fait d'utiliser la classe `java.awt.Point`, parfaitement adaptée puisque celle-ci fournit déjà les attributs et opérations nécessaires, notamment la méthode `translate`, évitant ainsi de créer une classe spécifique.

Les balles sont regroupées dans une classe `Balls`[1], qui encapsule une liste de points. Cette classe joue le rôle de module de calcul : elle gère l'état des balles mais n'effectue aucun affichage. Deux listes sont maintenues : la liste courante des balles et la liste des positions initiales. Cela permet d'implémenter efficacement la fonctionnalité `reInit`, qui restaure l'état initial sans devoir recalculer les coordonnées. La classe fournit également des méthodes pour ajouter une balle, translater toutes les balles et générer une représentation textuelle de l'état.

Ce design respecte le principe d'encapsulation : aucune structure interne n'est exposée en écriture et l'intégrité des données est préservée.

1.2 Séparation calcul / affichage

Conformément au sujet, l'affichage et les interactions utilisateur sont séparés du calcul. La classe `BallsSimulateur`[2] joue le rôle de simulateur graphique en réalisant l'interface `Simulable`. Elle possède une instance de `GUISimulator` et une instance de `Balls`.

La méthode `next()` sollicite uniquement le gestionnaire d'événements pour actualiser la simulation, puis appelle une méthode dédiée, `drawBalls`, pour redessiner les cercles. La méthode `restart()` réinitialise les balles, réinitialise le gestionnaire d'événements et réaffiche l'état initial. Ce découplage permettrait facilement de tester les règles de déplacement de manière indépendante de l'interface graphique.

1.3 Gestion du mouvement et rebonds

Le simulateur gère des translations individuelles pour chaque balle. Deux tables de hachage (`HashMap`) stockent les vitesses horizontales et verticales associées à chaque balle. Cette approche facilite la mise à jour individuelle des trajectoires.

La méthode `rebonBalls` calcule les déplacements et détecte les collisions avec les bords de la fenêtre. Lorsqu'une balle dépasse un bord, sa position est corrigée pour rester visible, et la direction est inversée. Ce mécanisme garantit un rebond naturel et prévient les effets de “téléportation” lors des collisions.

Le rayon des balles est stocké dans la classe `Balls` et utilisé dans la détection de collisions, ce qui permet un affichage cohérent entre physique et rendu visuel.

1.4 Tests et résultats obtenus

Deux classes de test ont été développées :

[1] Fichier `Balls.java`
[2] Fichier `BallsSimulateur.java`

- `TestBalls`[3] teste le module de calcul seul : ajout de balles, translation, réinitialisation. Ce test a permis de vérifier l'intégrité de la liste des positions initiales et la cohérence de la méthode `translate`.
- `TestBallsSimulateur`[4] initialise un simulateur avec plusieurs balles de coordonnées différentes. Les rebonds sur les bords ont été validés visuellement et la continuité de l'animation confirmée.

[3]Fichier `TestBalls.java`
[4]Fichier `TestBallsSimulateur.java`