MINES Douai

Tests + Observateur

Noury Bouraqadi

1 Balle Simple

Il s'agit de faire une simulation de la chute d'une balle simple soumise à la gravité. Vous allez procéder suivant une démarche de développement dirigée par les tests.

Définissez la classe TestBalleSimple sous-classe de TestCase qui teste les scenarios ci-dessous. Implémentez la classe BalleSimple qui répond aux tests au fur et à mesure de leur définition.

- Une balle a une position (résultat du message position) qui est initialement (0@0).
- Soit une balle dont la position est (86@29). Après réception du message position : (30@40), la position la balle simple devient (30@40).
- Soit une balle dont la position est (73@64). Après réception du message position : (21@ -34), la position la balle simple devient (21@0).
- La vitesse d'une balle (résultat du message vitesse) est initialement de (0 @ -1). Il s'agit en fait d'un vecteur vitesse qui donne la vitesse selon les axes x et y.
- Soit une balle initialement en (10 @ 10) avec une vitesse de (3 @ 2). Après un pas de simulation (message unPasDeSimulation), la position de la balle devient (13 @ 12). La vitesse reste inchangée.
- Soit une balle initialement en (20 @ 20) avec une vitesse de (-5 @ 6). Après un pas de simulation (message unPasDeSimulation), la position de la balle devient (15 @ 26). La vitesse reste inchangée.
- Soit une balle initialement en (23 @ 50) avec une vitesse de (-3 @ 5). Après un pas de simulation (message unPasDeSimulation), la position de la balle devient (20 @ 55). La vitesse reste inchangée.
- Soit une balle initialement en (42 @ 10) avec une vitesse de (8 @ -9). Après un pas de simulation (message unPasDeSimulation), la position de la balle devient (50 @ 1). La vitesse reste inchangée.
- Soit une balle initialement en (34 @ 4) avec une vitesse de (21 @ -10). Après un pas de simulation (message unPasDeSimulation), la position de la balle devient (55 @ 0). La vitesse reste inchangée.
- Soit une balle initialement en (100 @ 0) et dont la vitesse est (0 @ -5). Après un pas de simulation (message unPasDeSimulation), la position et la vitesse restent inchangées.
- Soit une balle initialement en (35 @ 0) et dont la vitesse est (10 @ -7). Après un pas de simulation (message unPasDeSimulation), la position de la balle devient (45 @ 0). La vitesse reste inchangée.
- Soit une balle initialement en (38 @ 0) et dont la vitesse est (2 @ 10). Après un pas de simulation (message unPasDeSimulation), la position de la balle devient (40 @ 10). La vitesse reste inchangée.

2 Affichage de la balle

Complétez les classes fournies pour réaliser l'affichage de la balle.

— Faites passer les tests fournis pour coupler l'affichage de la balle avec l'afficheur de balle fourni. Utilisez pour cela le *pattern* Observateur et plus spécifiquement avec la bibliothèque *Announcement* de Pharo.
— Vérifiez à l'aide de des tests que vous avez écrits dans l'exercice précédent que la balle continue de fonctionner correctement après vos modifications.
— Toujours avec le *pattern* Observateur, liez la balle avec une courbe (instance de la classe Courbe fournie) de manière à tracer la trajectoire de la balle. Vous devez avoir en même temps l'affichage de la balle et le tracé de la courbe. Vous devez avoir une seule classe d'événement utilisée aussi bien pour l'afficheur de la balle que pour la courbe.
— Complètez la méthode initialize de la classe Simulation fournie pour lier une balle simple avec un afficheur et une courbe. Vous pourrez voir une simulation s'exécuter en évaluant dans un *workspace* le code suivant :
```
sim := Simulation new.
sim lancerSimulation.
```

3 Balle Rebondissante

Les besoins de votre client ont évolué, il souhaite que la balle soit désormais rebondissante. Elle doit vérifier les scénario ci-dessous. Définissez la classe TestBalleRebondissante comme une copie de la classe TestBalleSimple (clic droit sur la classe, puis choisir le menu *Copy*)

Définissez la classe BalleRebondissante pour qu'elle réponde aux tests. L'implémentation doit se faire au fur et à mesure de l'introduction des tests.

— Une balle a une position (résultat du message position) qui est initialement (0@0).
— Soit une balle dont la position est (86@29). Après réception du message position : (30@40), la position la balle simple devient (30@40).
— Soit une balle dont la position est (73@64). Après réception du message position : (21@ -34), la position la balle simple devient (21@0).
— La vitesse d'une balle (résultat du message vitesse) est initialement de (0 @ -1). Il s'agit en fait d'un vecteur vitesse qui donne la vitesse selon les axes x et y.
— Soit une balle initialement en (10 @ 10) avec une vitesse de (3 @ 2). Après un pas de simulation (message unPasDeSimulation), la position de la balle devient (13 @ 12). La vitesse reste inchangée.
— Soit une balle initialement en (20 @ 20) avec une vitesse de (-5 @ 6). Après un pas de simulation (message unPasDeSimulation), la position de la balle devient (15 @ 26). La vitesse reste inchangée.
— Soit une balle initialement en (23 @ 50) avec une vitesse de (-3 @ 5). Après un pas de simulation (message unPasDeSimulation), la position de la balle devient (20 @ 55). La vitesse reste inchangée.
— Soit une balle initialement en (42 @ 10) avec une vitesse de (8 @ -9). Après un pas de simulation (message unPasDeSimulation), la position de la balle devient (50 @ 1). La vitesse reste inchangée.
— Soit une balle initialement en (34 @ 4) avec une vitesse de (21 @ -10). Après un pas de

simulation (message unPasDeSimulation), la position de la balle devient (55 @ 0). La vitesse devient (21 @ 5). C'est le début du rebond : changement de signe et de la valeur la vitesse suivant l'axe y.

— Soit une balle initialement en (100 @ 0) et dont la vitesse est (0 @ -5). Après un pas de simulation (message unPasDeSimulation), la position reste inchangée. La vitesse devient (0 @ 2).
— Soit une balle initialement en (35 @ 0) et dont la vitesse est (10 @ -7). Après un pas de simulation (message unPasDeSimulation), la position de la balle devient (45 @ 0). La vitesse devient (10 @ 3).
— Soit une balle initialement en (38 @ 0) et dont la vitesse est (2 @ 10). Après un pas de simulation (message unPasDeSimulation), la position de la balle devient (40 @ 10). La vitesse reste inchangée.

4 Balle rebondissante réaliste

Toujours en vous appuyant sur des tests, développez la classe BalleRealiste. Il s'agit d'une classe de balles rebondissantes dont la vitesse évolue avec la gravité et le frottement de l'air. La gravité ralenti la montée et accélère la chute de la balle. Le frottement ralenti dimune la valeur absolue de la vitesse dans toutes les directions. Nous faisons l'hypothèse que :

— la gravité est de 1 suivant l'axe Y ;
— le frotemment est constant et a pour valeur 0.25 suivant l'exe X et également 0.25 suivant l'axe Y.

5 Mini-Pong

Développez un mini-jeu de pong décrit ci-dessous.

Il s'agit d'un jeu avec une balle qui rebondit sur les murs d'un terrain rectangulaire. La balle peut rebondir sur 3 murs. Mais, si elle touche le quatrième, la partie se termine.

Le joueur dispose d'une raquette qu'il peut déplacer le long du quatrième mur. Il peut ainsi faire renvoyer la balle et faire durer la partie le plus longtemps possible.

La vitesse initiale de la balle en x comme en y est choisi aléatoirement parmi les valeurs suivantes : -50, -30, 30 et 50. Sa position a un x abitraire, mais son y est toujours égal à la moitié de la hauteur du terrain.

A chaque rebond le joueur marque 1 point. A la suite d'un rebond, la vitesse de la balle augmente de 10 soit en x soit en y suivant le mur sur lequel a lieu le rebond. Puis. la balle change d'orientation. Cela est effectué en inversant le signe de la coordonnée de la vitesse qui a changé.