Laboratoire final : le jeu de pong

# Objectifs

Au terme du laboratoire, les étudiants seront capables de :

* D’expliquer les concepts de classes d’objets et d’objets ;
* De définir des classes simples et d’instancier des objets ;
* De comprendre et expliquer la portée des différents éléments programmés ;
* De faire interagir plusieurs objets ;
* De faire appels aux méthodes d’un objet sur base de la documentation proposée.

Pour atteindre ces objectifs, nous réaliserons un jeu vidéo : Pong !

# Le jeu de Pong

« Le concept original de Pong est une simulation simpliste de tennis de table (ping-pong), repris pour le nom du jeu. Au tennis de table, les joueurs se tiennent de chaque côté d'une table et manient une raquette pour frapper une petite balle qui se déplace entre eux dans les deux sens. Ce concept de base est celui de Pong. Une petite balle, se déplace à travers l'écran, rebondissant sur les rebords du haut et du bas, et les deux joueurs commandent chacun une raquette, la faisant glisser verticalement entre les extrémités de l'écran à l'aide des contrôles. Si la balle frappe la raquette, elle rebondit vers l'autre joueur. Si elle manque la raquette, l'autre joueur marque un point. La balle rebondit de différentes manières selon la façon dont elle touche la raquette. »[[1]](#footnote-1)

Il s’agit donc de faire interagir deux raquettes, une balle et une zone de jeu.

# Semaine 1 : définition de la balle, mise à jour et dessin de la scène

Téléchargez le projet préconfiguré **Pong** depuis Moodle et ouvrez-le dans Netbeans.

## Définir ce qu’est une Balle et créer une balle

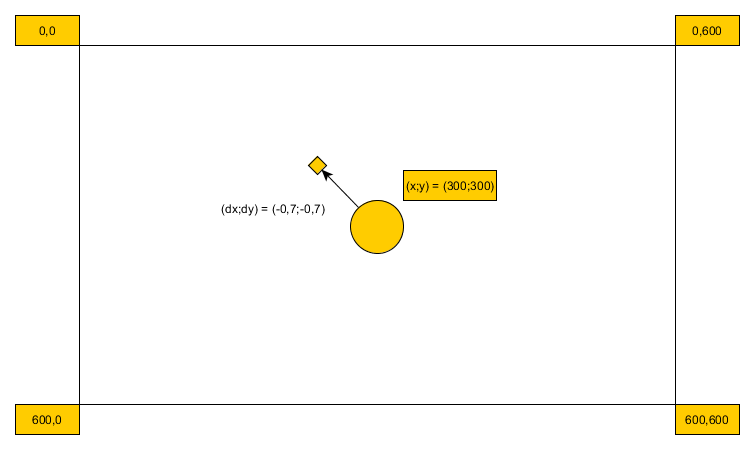
La balle est un objet caractérisé par une position matérialisée par deux flottants x et y et par une direction matérialisée par deux flottants dx et dy[[2]](#footnote-2).

Définissez une classe Balle dont les données membres sont les flottants x, y, dx et dy.

Dans la classe prédéfinie JeuPong, déclarez une référence sur une Balle et instanciez cet objet dans le constructeur par défaut de JeuPong.

Où déclarerez-vous la référence ? Cette dernière doit être accessible à toutes les méthodes de la classe Pong.

Au moment de sa création, une balle est placée au centre de la scène et est orientée aléatoirement, comme le montre la figure suivante :



Plus précisément, les valeurs acceptables pour dx et dy sont comprises entre 0 et 1 inclus. En outre dx et dy doivent former un vecteur normalisé. Enfin, on veillera à éviter les vecteurs suivants :

1. (-1 ;0)
2. (1 ;0)
3. (0 ; 1)
4. (0 ; -1)
5. (0 ;0)

Concevez un algorithme à cette fin et concrétisez le dans le constructeur par défaut de la classe Balle. Avant de programmer, détaillez les étapes de votre algorithme :

Pour initialiser aléatoirement la direction d’une balle, il faut :

# Dessiner la balle

La méthode public void dessinerScene(javafx.scene.canvas.GraphicsContext peintre ) est utilisée par le nhpack pour dessiner la scène sur la fenêtre. Cette méthode est appelée 30 fois par seconde : la scène est mise à jour et dessiner 30 fois pendant une seconde.

En plus d’appeler la méthode mettreAJour, nous allons utiliser le peintre pour dessiner la balle. A cette fin, nous utiliserons la position de la balle et supposer qu’elle possède un rayon de 30 pixels.

Où allez-vous stocker la taille de la balle ? Est-ce une donnée locale à la méthode dessinerScene ou devez-vous définir la donnée ailleurs ? Pour vous aider, pensez qu’il existe des balles de tailles différentes.

Les méthodes GraphicsContext.setFill(Paint p) et GraphicsContext.fillOval(double x,double y, double w, double h) s’occuperont du dessin à l’écran. Consultez la documentation officielle pour les utiliser à bon escient[[3]](#footnote-3). Soyez particulièrement attentif deux premiers paramètres de fillOval…

# Mettre à jour la balle

La balle ne reste pas statique à l’écran. Au contraire, elle bouge continuellement.

Du point de vue de l’état de l’objet, cela signifie que sa position change à chaque mise à jour de la scène. Cela signifie également qu’en cas de collision avec un autre élément, la direction prise par la balle change.

## Déplacer la balle

Ajouter une méthode public void deplacer(float vitesse) à la classe balle. Cette dernière met à jour la position de la balle en ajoutant respectivement aux données membres x et y le produit de dx\*vitesse et celui dy\*vitesse.

Appeler cette méthode dans la méthode mettreAJour et vérifiez que la scène s’anime.

## Gérer les collisions avec le bord de la scène

Lorsqu’elle atteint le bord de la scène, la balle doit changer de direction. Ce changement est fonction de la direction initiale et du mur avec lequel la balle est entrée en collision.

En cas de collision avec le bord supérieur ou le bord inférieur, dx ne change pas mais le signe de dy est inversé. En cas de collision avec le bord gauche ou le bord droit, dy ne change pas mais le signe de dx est inversé.

Comment détecter une collision avec l’un des bords ? Pensez à éviter les chevauchements de la balle avec l’un des bords du jeu.

Où allez-vous placer la logique de détecter des collisions ? Avant ou après le déplacement ? Dans la méthode mettreAJour ou dans une nouvelle méthode de Balle ? Pouvez-vous justifier votre choix ?

Avant de coder, écrivez les différentes étapes de votre algorithme :

Pour détecter et gérer les collisions avec les bords, il faut :

1. Source : http://fr.wikipedia.org/wiki/Pong [↑](#footnote-ref-1)
2. Mathématiquement, x et y forment un point tandis que dx et dy forment un vecteur normalisé. [↑](#footnote-ref-2)
3. <https://docs.oracle.com/javase/8/javafx/api/javafx/scene/canvas/GraphicsContext.html#fillOval-double-double-double-double-> et <https://docs.oracle.com/javase/8/javafx/api/javafx/scene/canvas/GraphicsContext.html#setFill-javafx.scene.paint.Paint-> [↑](#footnote-ref-3)