Examen Final

Développement de programmes dans un environnement graphique – SIM – A24

Nicolas Hurtubise - Salwa Mbarek

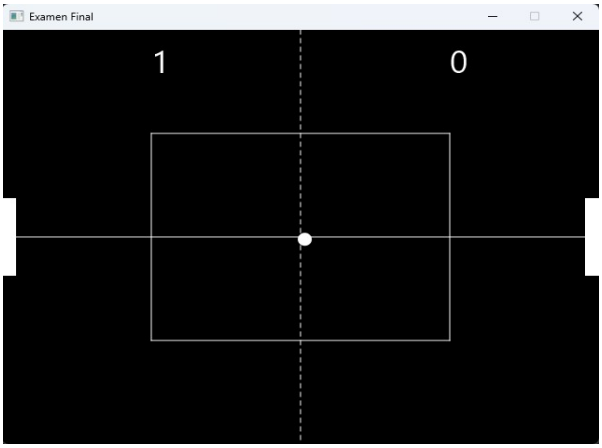
Barème

Pondération	Éléments d'évaluation
10%	Instruction 1
15%	Instruction 2
20%	Instruction 3
10%	Instruction 4
10%	Instruction 5
5%	Instruction 6
10%	Instruction 7
20%	Qualité et organisation du code

Contexte

Votre tâche sera de coder une version simplifiée du jeu de Pong avec JavaFX sur IntelliJ.

Le jeu doit mettre en scène deux raquettes et une balle sur un terrain de jeu en 2D.



Projet de base

Une petite base de code vous est fournie :

- **Main** : le point d'entrée de l'application
- **Input** : une petite classe utilitaire pour savoir quelle touche est appuyée à chaque instant (même classe qu'on a vu en cours)
- **ObjetDuJeu** : une classe abstraite qui définit la base de ce qu'un objet du jeu peut faire
- **Partie** : qui ne contient presque rien pour le moment, mais qui devrait contenir la logique de votre code

Vous ne devriez pas avoir besoin de modifier le Main.

Instructions

Vous avez 3 heures, **incluant le temps pour remettre le projet sur Léa.**

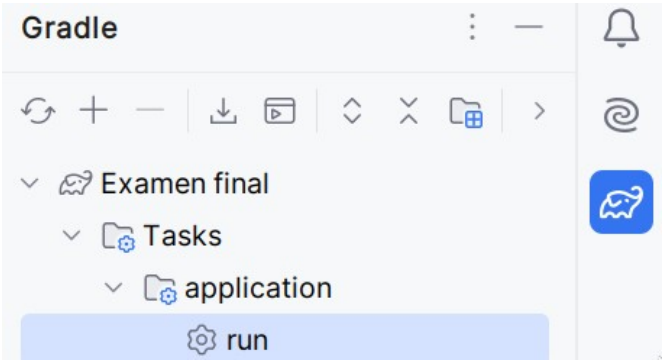
Si vous n'arrivez pas à tout faire, vous serez évalués sur ce que vous avez complété.

Mieux vaut un code qui marche mais qui ne fait pas tout plutôt qu'un code qui ne marche pas mais qui essaie de (mal) tout faire.

Allez-y une seule étape à la fois, dans l'ordre où c'est écrit dans les instructions.

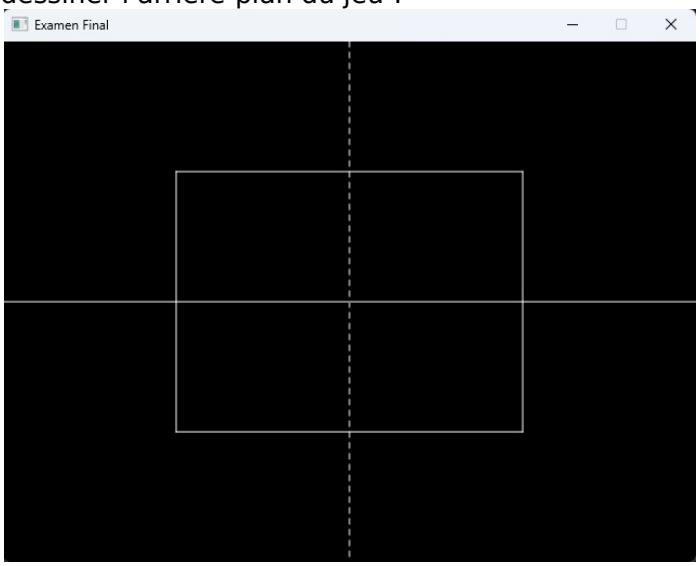
Si vous bloquez sur une des choses à faire, c'est ok de passer à la prochaine, mais assurez-vous simplement de ne pas tout essayer de faire en même temps.

ATTENTION : Pour lancer le projet, vous **devez** passer par le menu **Gradle** à droite :



1. Arrière-plan

Commencez par dessiner l'arrière-plan du jeu :



- Dessiner un rectangle noir occupant tout l'écran.
- Dessiner une ligne en pointillée centrée verticalement et de couleur blanc.
- Dessiner une ligne continue centrée horizontalement et de couleur blanc.
- Dessiner un cadre central de couleur blanc.

Pour faire des pointillés, utilisez :

```
context.setLineDashes(5);
```

Pour enlever les pointillés, utilisez :

```
context.setLineDashes(0);
```

2. La raquette gauche (joueur)

- Position : côté gauche du terrain. Initialement : centrée verticalement.

- Forme : rectangle blanc de largeur=15px et de hauteur=90px.

- Déplacement : vertical contrôlé par le joueur

  - Elle se déplace vers le haut et vers le bas avec les touches du clavier ↑ et ↓
  - Son déplacement est fluide avec une vitesse constante de 600 pixels par seconde.
  - Le déplacement doit être continu lors du maintien des touches.
  - Elle ne peut pas dépasser les bords haut et bas de l'écran.

3. La raquette droite (déplacements automatiques)

- Position : côté droit du terrain. Initialement : centrée verticalement.

- Forme : rectangle blanc de largeur=15px et de hauteur=90px.

- Déplacement : automatique de manière aléatoire.

- Son déplacement est fluide avec une vitesse constante de 200 pixels par seconde
- Les déplacements se font au hasard selon l'algo suivant :
    - On choisit une direction au hasard (haut ou bas)
    - On choisit une durée au hasard entre 1s et 2s
    - On avance dans la direction choisie jusqu'à la fin de la durée choisie
    - Une fois la durée terminée, on choisit une nouvelle direction au hasard et une nouvelle durée. *Notez : la nouvelle direction pourrait être la même que la précédente, ce qui ferait en sorte que la raquette continue de se déplacer comme avant*
- Lorsqu'elle frappe le haut ou le bas de l'écran, elle rebondit : sa direction s'inverse et la durée est réinitialisée au hasard

4. La balle

La balle est un cercle blanc de largeur=15px et de hauteur=15px.

Initialement elle est positionnée au centre du terrain.

Au début d'une ronde, la direction est aléatoire (haut ou bas/droite ou gauche), la vitesse horizontale est de 300 pixels par seconde et la vitesse verticale est de 180 pixels par seconde.

Rebond sur les murs (haut et bas)

Quand elle entre en collision avec le bord supérieur ou le bord inférieur, elle inverse sa direction verticalement. On doit également s'assurer que la balle reste à l'intérieur du terrain.

Rebond sur une raquette

Quand la balle entre en collision avec une raquette, la balle inverse sa direction horizontalement.

Verticalement, la vitesse est réinitialisée au hasard entre -180 et 180.

Pour détecter une collision entre une raquette et la balle, on considère que la balle est un rectangle. Utilisez le code suivant, qui retourne ***true*** quand il y a une collision entre deux rectangles :

```
return !(this.getDroite() < autre.getGauche() ||
         this.getBas() < autre.getHaut() ||
         autre.getDroite() < this.getGauche() ||
         autre.getBas() < this.getHaut());
```

5. Balle hors terrain et gestion du score

Quand la balle sort du terrain (lorsqu'elle est complètement hors de l'écran à droite ou à gauche), elle revient au centre et se relance avec une direction aléatoire (même chose qu'expliqué au début du point 3).

Un score est affiché pour chaque joueur. Les scores sont à 0 au début de la partie.

Incrémenter le score à droite quand la balle dépasse la raquette gauche (sort de la fenêtre du jeu).

Incrémenter le score à gauche quand la balle dépasse la raquette droite (sort de la fenêtre du jeu).

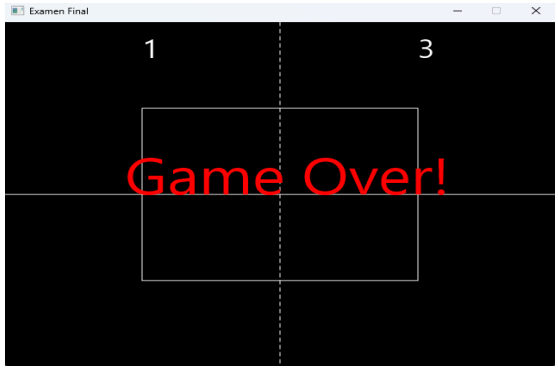
Afficher les scores de manière claire et lisible (en gros, en blanc, en haut de l'écran et respectivement à 25% et 75% de la largeur de la fenêtre).

Truc : utilisez context.fillText() pour dessiner du texte sur l'écran. **Ne perdez pas de temps à modifier les composantes JavaFX pour ajouter des VBox, des Text et autres.**

6. Fin du jeu

Dès que le score d'un des joueurs atteint 3 alors la partie est terminée.

À ce moment, on affiche « Game over! » en rouge sur l'écran pendant 3 secondes, puis on recommence une nouvelle partie de zéro.



7. Animations pendant la fin de la partie

Pour rendre la fin de la partie plus intéressante, on va faire « tomber » les raquettes pendant qu'on affiche le message de fin de partie.

Leur vitesse en Y est remise à zéro quand on commence à afficher le message « Game over! ». Les raquettes « tombent » ensuite vers le bas avec une gravité de 500px/s^2. On doit voir les deux raquettes tomber jusque plus bas que l'écran. Elles ne rebondissent plus sur les côtés pendant cette animation.

Précisions

Utilisez la structure du projet fourni pour ne pas perdre de temps.

Vous serez évalués sur le respect du découpage MVC, mais si vous suivez la structure proposée, ça ne devrait pas être très difficile à faire. ***Vous ne devriez pas avoir besoin de modifier le* Main.**

Si quelque chose n'est pas spécifié dans l'énoncé, c'est que ce n'est pas grave. Par exemple : je ne serai pas sévère sur la position au pixel près du mot "Game Over!" quand on termine la partie.

Annexes

La documentation à même IntelliJ vous sera utile.

Vous aurez principalement besoin des méthodes du GraphicsContext :

```
// Dessiner un oval plein (un cercle quand w=h)
context.fillOval(x, y, w, h);

// Dessiner un rectangle plein
context.fillRect(x, y, w, h);

// Dessiner un rectangle (contour seulement)
context.strokeRect(x, y, w, h);

// Changer la couleur de remplissage des dessins
context.setFill(Color.BLUE);

// Changer la couleur des tracés de lignes
context.setStroke(Color.BLUE);

// Changer la taille de la police d'écriture utilisée :
context.setFont(Font.font(55));

// Changer l'alignement du texte
context.setTextAlign(TextAlignment.CENTER); // LEFT et RIGHT existent aussi

// Dessiner du texte en (x, y)
context.fillText("Bonjour tout le monde!", x, y);
```

Pour obtenir des nombres aléatoires, vous pouvez utiliser une de ces méthodes :

```
// Retourne un nombre à virgule entre 0 et 1
double x = Math.random();

// Retourne un nombre à virgule entre min (inclus) et max (exclus)
double x = ThreadLocalRandom.current().nextDouble(min, max);

Random random = new Random();
double x = random.nextDouble(min, max);
```