Collège de Maisonneuve
Formation continue

Professeur: Ahmed Imed Eddine Rabah
Mail: arabah@cmaisonneuve.qc.ca
Cours: Dev App : Bureau

Projets JavaFX

- **Respectez strictement les instructions mentionnées dans cet énonce.**
- À rendre au plus tard le Samedi 08 Février 2025 à 23h59, Aucun retard accepté.
- Ce projet vaut 20% de la moyenne du cours.
- **Fichier à remettre** :
  - Vous devez remettre sur **Teams** uniquement un fichier **.zip** par le chef de groupe seulement.
  - Nom du fichier : `projet<avec numéro de projet et juste les nom des personne dans le groupe>.zip`.
  - **Ne soumettez pas** de fichier PDF, Word ou tout autre format.
- Dans le fichier `.zip`, on trouve le `Code` et le `Rapport` :
  - Dans le rapport mentionnez le noms et prénoms de chaque membre de l'équipe au début pour introduire votre projet, incluant :
    - Nom du projet.
    - Description du projet.
    - Votre nom et prénom.
  - Faites des captures d'écrans de chaque partie mentionne dans le barème.
- **Structure du fichier** :
  - Le dossier qui contient les fichiers code doit être organisé comme on a vu pendant la seance de JavaFX, partie code java / et partie resources pour les fichier .FXML .
  - Assurez-vous que votre code s'exécute correctement.
  - Aucun Template n'est donné donc soyez organiser et créative.
- **Distribution des projets** :
  - Si vous travaillez seul, vous avez le droit de choisir le projet qui vous plaît.
  - Si vous travaillez en groupe de deux, vous pouvez choisir uniquement le projet 1 ou le projet 3.
  - Si vous travaillez en groupe de trois personnes, vous ne pouvez travailler que sur le projet 3.

Introduction

Ce document présente trois projets de jeux développés en **JavaFX** avec des niveaux de difficulté différents. Chaque projet inclut une description, une liste des fonctionnalités, les éléments d'interface utilisateur à utiliser, ainsi qu'un barème d'évaluation.

Projet 1 : Tic-Tac-Toe (Facile à Moyenne)

Description

Le jeu de Tic-Tac-Toe (Morpion) consiste en une grille 3x3 où deux joueurs jouent à tour de rôle en plaçant un "X"ou un "O". Le premier qui aligne trois de ses symboles horizontalement, verticalement ou en diagonale gagne la partie. Si toutes les cases sont remplies sans vainqueur, la partie est déclarée nulle.

Objectifs Techniques :

Utilisation de JavaFX pour créer une interface utilisateur fluide et interactive. Gestion des événements utilisateurs pour enregistrer les coups des joueurs. Détection automatique de la victoire ou de l'égalité. Réinitialisation de la partie via un bouton "Recommencer".

Interface Graphique avec Scene Builder :

Éléments JavaFX à utiliser :

- GridPane : Pour structurer la grille de 3x3.
- Button (9 boutons) : Chaque case de la grille sera un bouton sur lequel les joueurs pourront cliquer.
- Label : Pour afficher les instructions et le statut du jeu (ex: "Joueur X joue", "Joueur O joue", "Victoire de X", "Match nul").
- HBox ou VBox : Pour organiser l'affichage du statut et des contrôles.
- Button ("Recommencer") : Pour réinitialiser la grille après une partie.

Détails Techniques et Logique :

Gestion de la grille :

- Une matrice 2D char[ ][ ] (ou String[ ][ ]) pour stocker les mouvements des joueurs.
- Les boutons sont mis à jour en fonction des valeurs de la matrice.
- Désactiver les boutons déjà utilisés.

Déroulement d'une partie :

- Au démarrage, la grille est vide et le joueur "X"commence.
- Lorsque le joueur clique sur une case, le symbole correspondant s'affiche.
- Après chaque coup :
  - Vérifier si un joueur a gagné en parcourant la matrice.
  - Vérifier s'il y a une égalité (si la grille est pleine).
  - Passer au tour suivant.
- Afficher un message si un joueur gagne ou si la partie est nulle.
- Le bouton "Recommencer"réinitialise la grille et relance le jeu.

Gestion des événements :

- Associer un **EventHandler<ActionEvent>** à chaque bouton pour enregistrer les coups.
- Mettre à jour dynamiquement le Label pour afficher le tour du joueur suivant.
- Ajouter des effets visuels (ex: mise en évidence des cases gagnantes).

Extensions possibles :

- Mode IA : Implémenter une IA basique (random) ou avancée (algorithme Minimax).
- Mode multijoueur en ligne avec Sockets et Networking en Java.
- Animation des boutons avec FadeTransition ou ScaleTransition.

Barème détaillé

Critères d'évaluation	**Points**
1. Interface graphique (5 points)	
La grille 3x3 est correctement affichée avec des boutons.	2
Les éléments (boutons, labels) sont bien positionnés avec `GridPane`.	2
Esthétique et ergonomie de l'interface utilisateur.	1
2. Gestion des événements et interactions (5 points)	
Les boutons réagissent bien aux clics (ajout de "X"et "O").	2
Les cases déjà utilisées ne sont plus cliquables.	1
Le statut du jeu (tour du joueur) est bien affiché.	2
3. Détection de victoire et égalité (5 points)	
Vérification correcte de la victoire (lignes, colonnes, diagonales).	3
Détection de l'égalité si la grille est remplie sans gagnant.	2
4. Réinitialisation et gestion du jeu (3 points)	
Un bouton `"Recommencer"` réinitialise correctement la partie.	2
Après une victoire, la grille devient inactive jusqu'à une nouvelle partie.	1
5. Bonus et améliorations (2 points, optionnel)	
Ajout d'animations (ex : effet sur les boutons, changement de couleur des cases gagnantes).	1
Ajout d'une IA basique pour jouer contre l'ordinateur.	1
Total	**20**

Projet 2 : Pierre-Papier-Ciseaux (Facile)

Description

Le joueur affronte l'ordinateur dans un duel de Pierre-Papier-Ciseaux. Le jeu suit les règles classiques :

- Pierre bat Ciseaux
- Ciseaux battent Papier
- Papier bat Pierre
- Le joueur clique sur l'une des trois options, et l'ordinateur choisit aléatoirement. Un score est mis à jour après chaque manche.

Objectifs Techniques :

- Utilisation de JavaFX pour une interface fluide.
- Génération aléatoire du choix de l'ordinateur.
- Détection et affichage du résultat de chaque manche.
- Mise à jour dynamique du score.

Interface Graphique avec Scene Builder :

Éléments JavaFX à utiliser :

- ImageView (ou Button avec une image) : Pour afficher les icônes de Pierre, Papier et Ciseaux.
- Button (3 boutons) : "Pierre", "Papier"et "Ciseaux", permettant au joueur de faire son choix.
- Label : Pour afficher le score et le choix de l'ordinateur.
- HBox ou VBox : Pour organiser l'affichage.
- Button ("Rejouer") : Pour réinitialiser les scores.

Détails Techniques et Logique :

Gestion du choix du joueur :

- Un EventHandler est associé aux boutons "Pierre", "Papier"et "Ciseaux".
- Lorsqu'un bouton est cliqué, le choix du joueur est enregistré et affiché.

Génération aléatoire du choix de l'ordinateur :

- Utilisation de Random pour sélectionner entre "Pierre", "Papier"ou "Ciseaux".
- Affichage de l'image correspondante.

Détermination du gagnant :

- Comparer le choix du joueur et celui de l'ordinateur via une condition if-else.
- Afficher le résultat ("Victoire", "Défaite", "Égalité").
- Mettre à jour le score en conséquence.

Gestion des événements :

- Boutons cliquables pour chaque choix.
- Bouton ”Rejouer”pour réinitialiser les scores.
- Animation des images pour dynamiser l'interface (FadeTransition, RotateTransition).

Extensions possibles :

- Ajouter un mode ”Meilleur des 5 manches”.
- Ajouter des animations pour rendre le jeu plus immersif.
- Ajouter une option ”Statistiques”pour suivre l'historique des parties.

Barème détaillé

Critères d'évaluation	**Points**
1. Interface graphique (5 points)	
Présentation soignée avec trois boutons pour ”Pierre”, ”Papier”, ”Ciseaux”.	2
Affichage clair du choix du joueur et de l'ordinateur.	2
Bonne organisation de l'interface avec `HBox` et `VBox`.	1
2. Gestion des événements et interactions (5 points)	
Les boutons fonctionnent correctement et permettent au joueur de choisir une option.	2
Le choix de l'ordinateur est généré aléatoirement et affiché.	2
L'affichage du résultat de chaque manche est clair (victoire, défaite, égalité).	1
3. Gestion du score et affichage (5 points)	
Mise à jour du score après chaque manche (victoires, défaites, égalités).	2
Score correctement affiché sous forme de `Label` et mis à jour dynamiquement.	2
Ajout d'un message clair indiquant le premier à atteindre un certain nombre de points (ex: ”Premier à 5 victoires”).	1
4. Réinitialisation et gestion du jeu (3 points)	
Un bouton `"Rejouer"` permet de réinitialiser les scores et relancer une partie.	2
L'affichage du jeu est bien réinitialisé après un redémarrage.	1
5. Bonus et améliorations (2 points, optionnel)	
Ajout d'animations sur les boutons ou effets visuels (ex: animation sur le choix du joueur et de l'ordinateur).	1
Ajout d'un mode ”Meilleur des 5 manches”avec affichage du gagnant global.	1
Total	**20**

Projet 3 : Rogue-Like 2D (Difficile)

Description

Le joueur incarne un personnage qui doit s'échapper d'un labyrinthe rempli de monstres et d'objets interactifs. Il se déplace case par case et doit collecter des clés, éviter des pièges et combattre des ennemis pour atteindre la sortie.

Objectifs Techniques :

- Utilisation de JavaFX pour un affichage 2D fluide.
- Génération procédurale du labyrinthe pour varier chaque partie.
- Déplacement du personnage avec gestion des collisions.
- Système de combat basique contre les ennemis.
- Gestion de l'inventaire avec objets ramassés (potions, clés, armes).
- Système de victoire et de défaite (sortir du labyrinthe ou mourir).

Interface Graphique avec Scene Builder :

Éléments JavaFX à utiliser :

- Canvas : Pour dessiner dynamiquement la carte du jeu.
- ImageView : Pour afficher les sprites du personnage et des objets.
- GridPane : Pour organiser la carte en tuiles.
- VBox/HBox : Pour l'interface (barre de vie, inventaire, score).
- Label : Pour afficher le statut du joueur (PV restants, clé obtenue, etc.).

Détails Techniques et Logique :

Génération du labyrinthe :

- Représenter le niveau sous forme d'une matrice 2D (int[ ][ ] où 0 = mur, 1 = chemin, 2 = monstre, 3 = clé, 4 = porte etc.).
- Générer dynamiquement une carte en utilisant un algorithme de génération (ex : algorithme de backtracking pour un labyrinthe parfait).

Déplacement du joueur :

- Associer les touches directionnelles (Z, Q, S, D) ou flèches à un KeyEvent.
- Vérifier si le mouvement est possible en fonction de la matrice.
- Rafraîchir l'affichage du personnage et du labyrinthe après chaque mouvement.

Gestion des collisions et interactions :

- Si le joueur entre en contact avec un monstre, un combat aléatoire est déclenché.
- Si le joueur ramasse une clé, elle est ajoutée à son inventaire et permet d'ouvrir une porte.
- Si le joueur trouve une potion, il récupère des PV.
- Si le joueur atteint la sortie avec la clé, il gagne la partie.
- Si ses PV tombent à 0, il perd et doit recommencer.

Système de combat basique :

À l'entrée en collision avec un monstre :

- Lancer un combat tour par tour où le joueur peut attaquer ou fuir.
- Chaque attaque inflige des dégâts aléatoires basés sur une force de base + arme.
- Si le monstre tombe à 0 PV, il disparaît de la carte.
- Si le joueur fuit, il perd des PV et recule d'une case.

Gestion des événements :

- Événements KeyPressed pour le déplacement et les actions.
- Événements MouseClicked pour interagir avec l'inventaire et les objets.
- Mise à jour dynamique des Label et Canvas pour l'affichage du statut du joueur.

Extensions possibles :

- IA des monstres : Ajouter des ennemis qui se déplacent aléatoirement ou poursuivent le joueur.
- Amélioration du combat : Ajouter des attaques spéciales, des armes et des boucliers.
- Mode multijoueur local : Un deuxième joueur peut apparaître et coopérer.
- Quêtes secondaires : Ajouter des PNJ qui donnent des objectifs avant la sortie du labyrinthe.
- Système de sauvegarde : Permettre au joueur de reprendre là où il s'est arrêté.
- Animations et sons : Ajouter des transitions pour le combat, des bruitages pour le mouvement.

Barème détaillé

Critères d'évaluation	Points
1. Interface graphique et affichage du labyrinthe (5 points)	
Affichage correct du labyrinthe sous forme de grille (`Canvas` ou `GridPane`).	2
Affichage du personnage et des objets interactifs (monstres, clés, potions).	2
Organisation de l'interface avec statut du joueur (PV, inventaire, etc.).	1
2. Gestion du déplacement du joueur et des collisions (5 points)	
Déplacement du joueur avec les touches du clavier (`KeyEvent`).	2
Gestion des collisions avec les murs du labyrinthe.	2
Gestion correcte des déplacements en fonction de la matrice du labyrinthe.	1
3. Interactions avec les objets et gestion de l'inventaire (5 points)	
Ramassage des objets (clés, potions) et mise à jour de l'inventaire.	2
Utilisation des objets (ex: clé pour ouvrir une porte, potion pour récupérer des PV).	2
Mise à jour dynamique de l'inventaire affiché à l'écran.	1
4. Système de combat et logique de jeu (5 points)	
Combat contre les monstres avec un système de tour par tour.	2
Gestion des PV et affichage des dégâts infligés/reçus.	2
Détection de victoire ou défaite (mort du joueur ou sortie du labyrinthe atteinte).	1
5. Bonus et améliorations (2 points, optionnel)	
Ajout d'une IA pour que les monstres se déplacent ou attaquent automatiquement.	1
Ajout d'effets visuels et sonores (animations de déplacement, attaques, bruitages).	1
Total	**20**

Autre aide et recommandations Générales :

- ChatGPT : Utilisez-le intelligemment.

Outils collaboration et travail en groupe :

- **Gestion de temps et de tache**: Trello