Snake

Dans ce TP, vous allez développer votre propre jeu snake. Les instructions qui suivent sont détaillées, mais pas exhaustives. Si vous ne comprenez pas comment faire certaines parties ou utiliser certaines classes, *utilisez Google et lisez la documentation avant de poser une question.*

https://www.youtube.com/watch?v=DekS8Pgb1qc

Règles du jeu

1. Grille :
    - La grille du jeu est généralement représentée comme une grille bidimensionnelle. Chaque cellule de la grille peut être soit vide, soit occupée par une partie du corps du serpent, soit par la nourriture que le serpent doit manger pour grandir.
2. Éléments dans la grille :
    - Serpent : Il est représenté par une série de segments (généralement des carrés) qui forment le corps du serpent. Chaque segment occupe une cellule de la grille. Le serpent grandit à chaque fois qu'il mange de la nourriture.
    - Nourriture : Un élément de nourriture apparaît aléatoirement sur la grille. L'objectif du joueur est de guider le serpent pour qu'il "mange" cette nourriture et ainsi augmente sa taille.
3. Update du jeu :
    - Le jeu fonctionne selon une boucle continue où le serpent se déplace en continu dans une direction. Vous gérer la mise à jour de la position du serpent à chaque itération de la boucle.
    - Lorsqu'une commande de déplacement est entrée (par exemple, le joueur appuie sur une touche directionnelle), le serpent change de direction dans la grille. Vous devez également s'assurer que le serpent ne peut pas faire demi-tour instantanément pour éviter des collisions avec son propre corps.
    - À chaque itération, le programme doit vérifier si le serpent a mangé de la nourriture. Si c'est le cas, le serpent grandit en ajoutant un segment à sa queue et un nouvel élément de nourriture apparaît à un emplacement aléatoire.
    - Le programme doit également gérer les collisions du serpent avec les bords de la grille (ce qui peut entraîner la fin du jeu) et les collisions avec son propre corps.
4. Conditions de perte :
    - Le jeu se termine si le serpent entre en collision avec les bords de la grille.
    - Le jeu se termine également si le serpent entre en collision avec son propre corps.

Mise en place

1. **Choisissez un Layout Pane dans lequel vous mettrez les cases du jeu :**
    - Créez un `GridPane` dans le FXML pour représenter la grille du jeu. Chaque cellule du `GridPane` représentera une case du jeu. Pensez bien à changer le type de fxmlRoot en Gridpane pour avoir

accès aux fonctions spécifiques au type GridPane.

2. **Initialisation du GameManager :**
    - Créez une classe `GameManager` qui gérera la logique du jeu, y compris la gestion de la grille, du serpent, du fruit, et de la boucle d'update.
    - Il vous faudra des variables et des méthodes pour représenter ces éléments de jeu.
    - Créez des classes distinctes si nécessaire, par exemple, une classe `Snake` pour représenter le serpent, une classe `Fruit` pour le fruit, etc... Attention à ne pas surcompliquer vos classes. Si vous pouvez représenter les éléments de jeu avec des classes disponibles dans Java ou JavaFX, vous travaillerez plus vite. Ces classes sont robustes et bien testées.

3. **Choisissez un élément graphique à répéter dans le layout pour représenter les éléments du jeu (serpent, fruit, case vide) :**
    - Utilisez des `Rectangle` pour représenter les segments du serpent et le fruit. Vous pouvez utiliser différentes couleurs pour les distinguer. Les cases vides peuvent être représentées par des rectangles avec une couleur différente.
    - Vous pouvez aussi utiliser des images pour les éléments de jeu si vous voulez un rendu plus beau.
    - Ne créez pas les rectangles dans le FXML, faites-le directement dans votre code. Cela vous permettra de varier la taille de la grille avec des variables et de ne pas perdre du temps à l'écrire à la main.

4. **Récupérez l'input utilisateur :**
    - Utilisez des Event Handlers pour capturer les entrées utilisateur, par exemple, les touches de direction. Mettez à jour la direction du serpent dans le GameManager en fonction de l'entrée utilisateur. Vous pouvez utiliser une énumération pour les directions possibles.

5. **Spawn du serpent et du fruit :**
    - Au lancement du jeu, créez une instance de `GameManager` et appelez une méthode d'initialisation/constructeur qui crée les éléments de la grille et place le serpent et le fruit sur cette grille.
    - Il n'est pas obligatoire de spawn le fruit en dehors du snake pour le moment. Il est plutôt conseillé de faire cette fonctionnalité après que votre jeu soit plus complet.

6. **Créez la boucle d'update dans la Timeline :**
    - Utilisez la `Timeline` pour créer une boucle d'update qui appelle la méthode de mise à jour du `GameManager` à des intervalles réguliers. Le code dans la `Timeline` est appelée toutes les 0.6 secondes indéfiniment et est considéré comme du code modifiant l'interface par JavaFX.
    - Les instructions suivantes se passent dans la boucle d'update.
    - Pensez à appeler d'autres fonctions dans la boucle afin de la rendre plus lisible et d'éviter de réutiliser du code.

7. **Déplacez le serpent :**
    - À chaque mise à jour, déplacez le serpent dans la direction la plus récente.

- Il vous faudra une direction par défaut quand le jeu commence, et réutiliser la direction précédente si il n'y pas d'input, ou la nouvelle direction est invalide (le serpent ne peut pas tourner à 180 degrés en un tour.)

8. **Vérifiez si le serpent entre en collision ou mange un fruit :**
    - Lorsque la tête du serpent atteint la position du fruit, agrandissez le serpent et générez un nouveau fruit à une position aléatoire.
    - Si la tête du serpent entre en collision avec le bord de la grille ou un segment du serpent, interrompez l'update du GameManager et lancez la procédure de fin de jeu.
9. **Affichage du serpent et du fruit :**
    - Mettez à jour les éléments graphiques en fonction de la position actuelle du serpent et du fruit.
10. **Fin de partie :**
    - Affichez un message de fin de jeu lorsque l'une de ces conditions est rencontrée, puis fermez le jeu.