CPA Projet

Conception et Pratique de l'Algorithmique

Rédigé par :

ZHANG Zimeng

LIU Yajie

Année universitaire 2021/2022

TABLE DES MATIÈRES

1 Introduction

2 Partie Introductive

2.1 Membres de l'Equipe

-ZHANG Zimeng, étudiante en STL M2(21108900)

-LIU Yajie, étudiante en STL M2(21108014)

2.2 Objectif

Implémentation d'un simple jeu de serpent 2D appelé GreedySnake en utilisant l'approche de l'architecture MVC.

2.3 Règles du jeu

Le joueur utilise les touches fléchées pour contrôler la direction de déplacement du serpent. Chaque fois que le serpent mange un aliment, le score est augmenté et le corps du serpent grandit d'un carré. Si le serpent se mord ou frappe un mur, le jeu est terminée. Nous avons fixé un score maximum de 200 points et le jeu se termine lorsque le joueur obtient 200 points. Cependant, le jeu peut être redémarré en appuyant sur le bouton "R".

2.4 Analyse Fonctionnelle

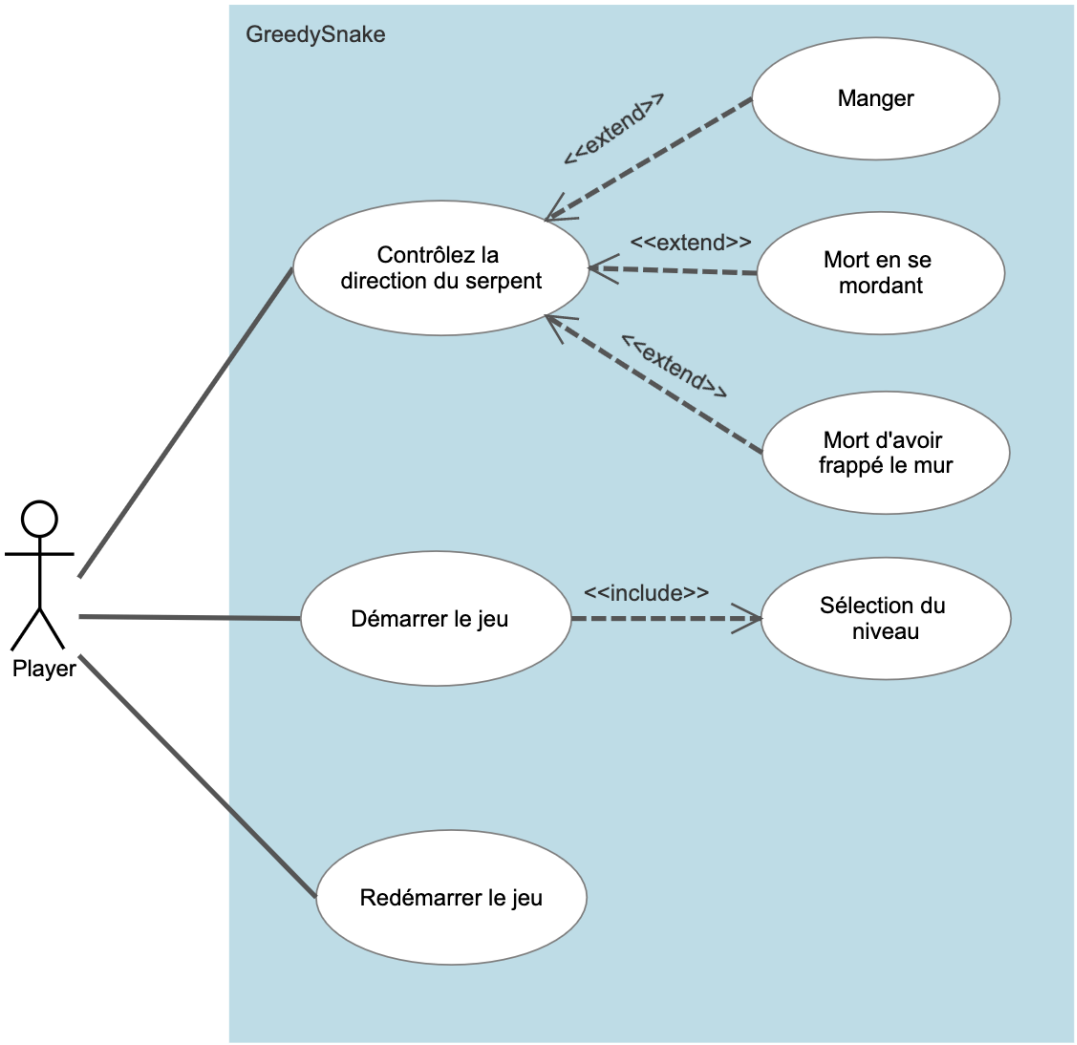

Figure 1: Usecase de GreedySnake

Voici le usecase de notre jeu. L'utilisateur peut mettre en œuvre certaines fonctions de base telles que contrôler la direction du serpent, démarrer le jeu etc.

2.5 Illustration

Dans cette section, nous avons choisi figma pour implémenter la maquette du jeu.

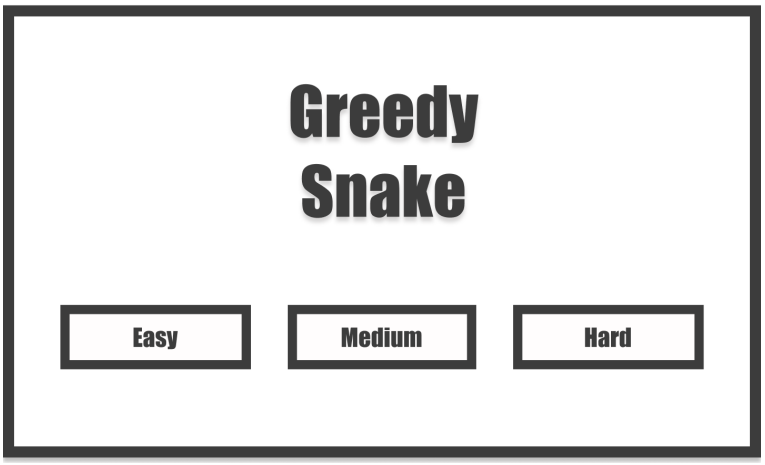

Figure 2: Le menu principal du jeu.

Il s'agit du menu principal du jeu, qui apparaît en premier lorsque vous démarrez le jeu et vous permet de choisir la difficulté du jeu, qui est facile, moyenne et difficile.

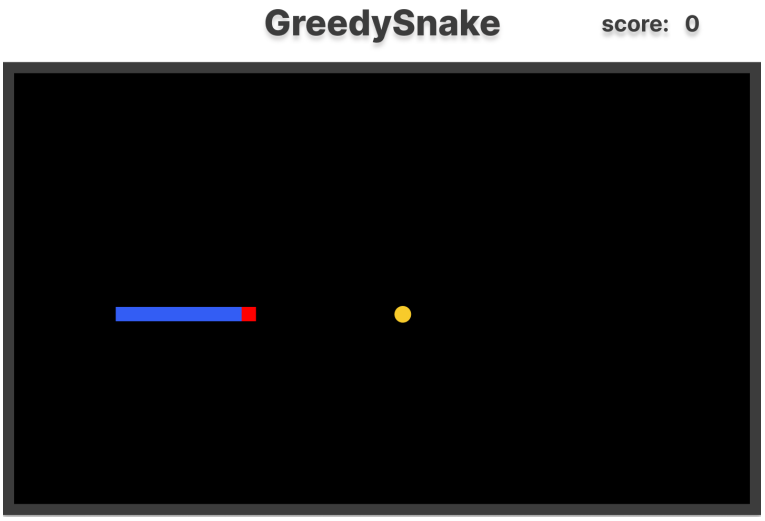

Figure 3: Écran de démarrage du jeu.

Voici l'écran de démarrage, où le serpent est immobile et où l'utilisateur sélectionne la direction dans laquelle le serpent va commencer à se déplacer.

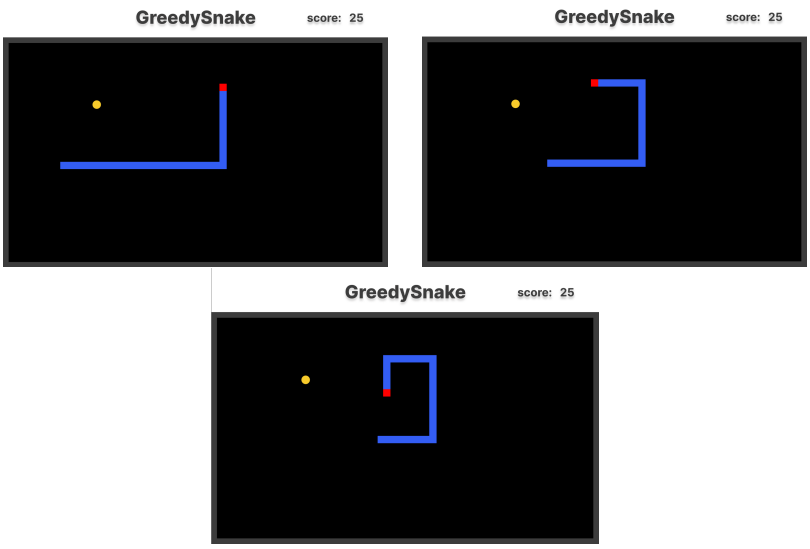

Figure 4: direction

Le schéma ici montre le serpent contrôlé par l'utilisateur pour changer sa direction de déplacement

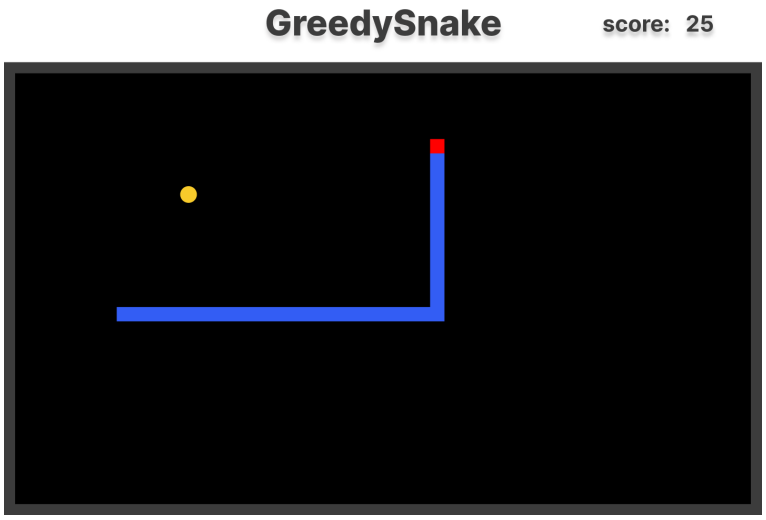

Figure 5: Le serpent mange des aliments.

Ce diagramme montre que le score du serpent augmente et que son corps s'allonge lorsqu'il mange des aliments.

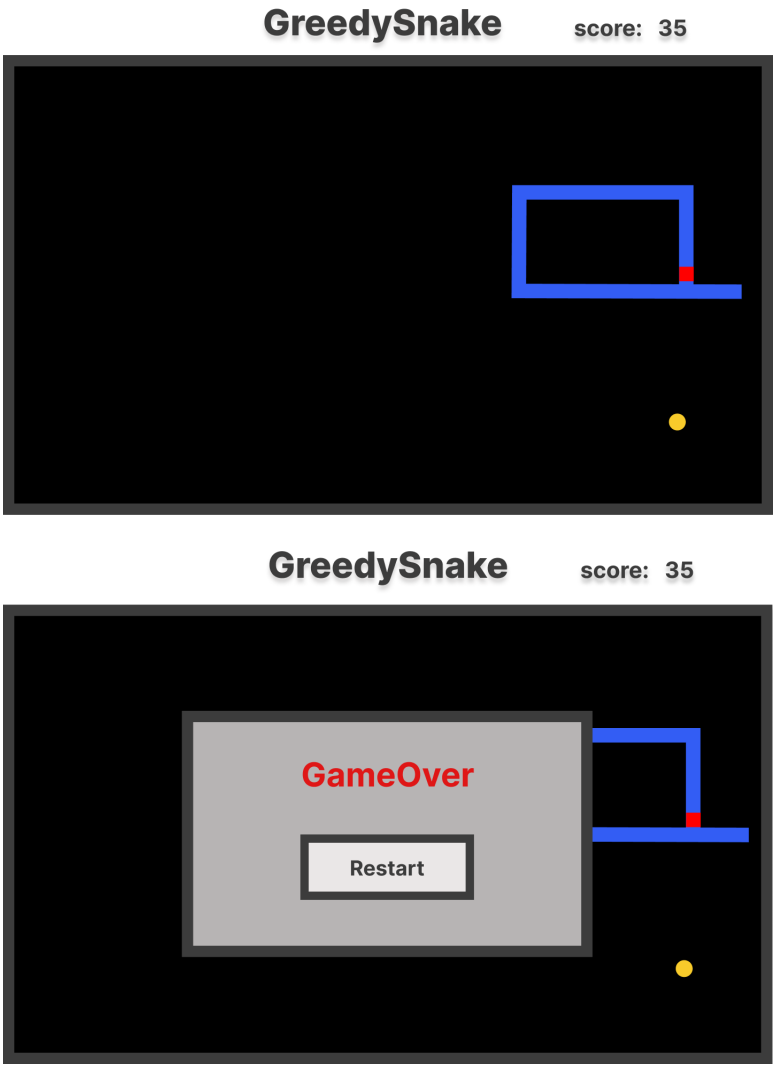

Figure 6: Le serpent se mord lui-même.

C'est le premier scénario dans lequel le jeu se termine où le serpent s'est mordu lui-même et où le jeu est terminé.

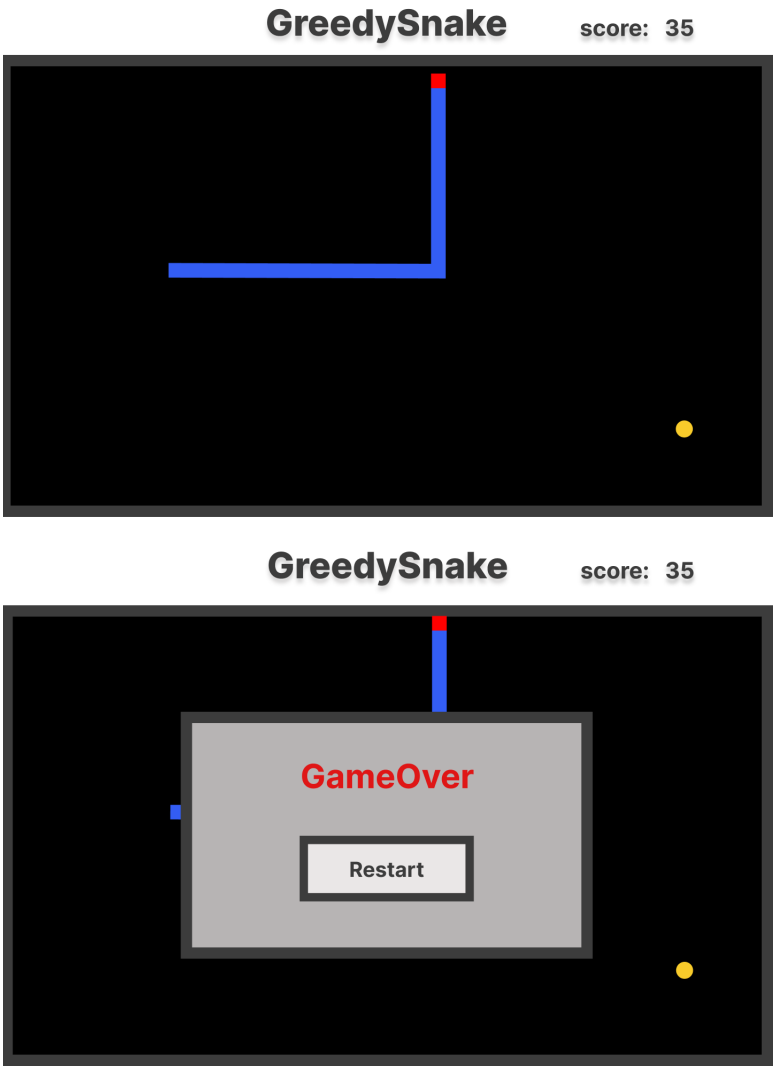

Figure 7: Le serpent frappe le mur.

C'est la deuxième scénario, le serpent frappe le mur et le jeu est terminé.

2.6 Technologies Utilisées

	C++	JavaFX	Python	TypeScript
Avantages	Proche du matériel physique, permettant de r éaliser nombreuses fonctions sous-jacentes	Prise en charge multiplateforme et riche en composants, Convient pour développer des jeux de table simples et de petite taille	Syntaxe simple, bibliothèque riche	Portable, adapté à l'écriture de grandes applications et à un meilleur dé veloppement collaboratif
Inconvénients	Plus complexe et difficile de réaliser un jeu en un temps très court	Moins de personnes l'utilisent et il est difficile de résoudre les problè mes	Fonctionnement plus lent	Le processus de dé veloppement est plus lourd et prend plus de temps

Figure 8: Un tableau comparatif entre les technologies les plus fréquentes sur le marché.

Après avoir analysé les avantages et les inconvénients de chaque langage et compte tenu des exigences et des objectifs du projet (créer un jeu 2D simple en peu de temps), nous avons choisi JavaFX pour implémenter notre jeu, qui a l'avantage du support multiplate-forme et de notre familiarité avec la syntaxe java, de sorte que nous avons pu commencer et progresser dans notre développement plus rapidement qu'avec TypeScript.

2.7 Bilan

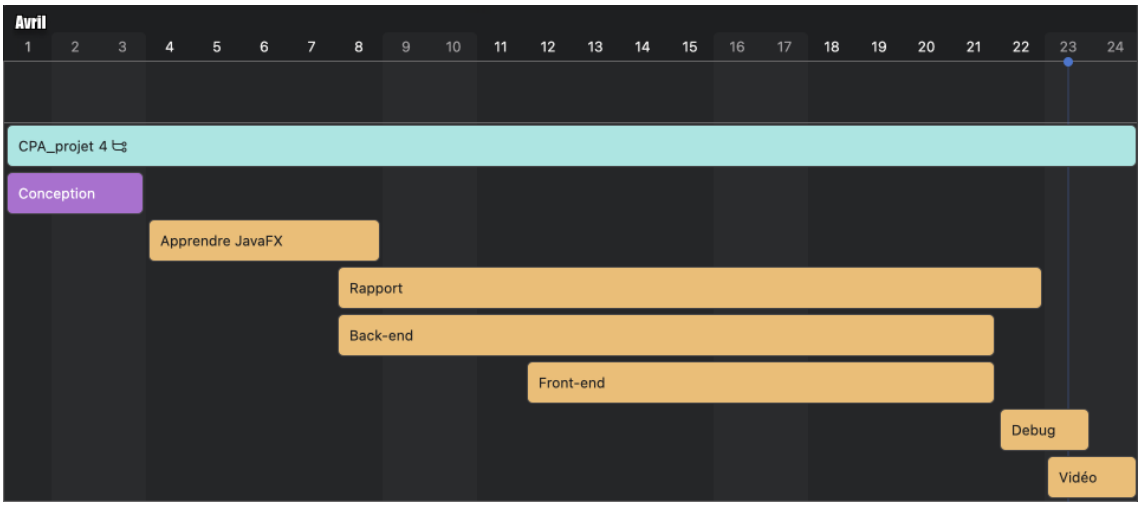

Figure 9: Gantt chart de GreedySnake

3 Partie technique

3.1 Architecture Générale

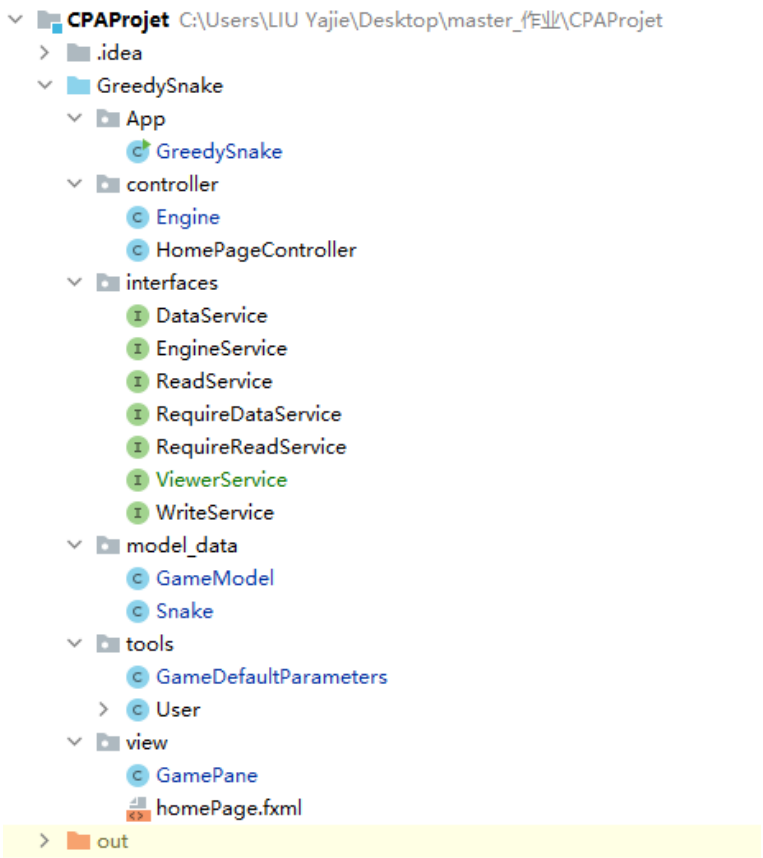

Figure 10: La structure du projet.

Notre projet utilise l'architecture MVC(modèle-vue-contrôleur).Pour notre application, coche client ≈ partie Vue, couche data ≈ partie Modèle et couche Serveur ≈ Partie Contrôleur.

Modele: Snake.java, GameModel.java

Vue: HomePage.fxml, GamePane.java

Controleur:Engine.java (moteur de jeu), HomePageController.java Main :GreedySnake.java (moteur de rendu graphique,commandes)

Le code source se trouve dans le package GreedySnake du répertoire CPAProjet, il contient 6 sous package, qui sont le package "APP", le package "model_data", le package "controller", le package"view", le package"tools",et le package"interfaces".

Le package “App” ne contient qu’une seule classe Java appelée GreedySnake, elle hérite de la classe Application du Javafx et est utilisée pour démarrer l’application . La classe GreedySnake joue le rôle de moteur de rendu graphique , À intervalles réguliers, il appellera la fonction paint() de l’interface viewService pour redessiner l’écran du jeu.

Le package “model_data” contient les classes Java de modèle,il sont utilisées comme la couche data.

Le package “ view”contient les deux interfaces graphiques utilisées par le jeu, l’une est un fichier “.fxml”, l’autre est une classe Java .

Le package “ controller” contient deux classes Java qui jouent le rôle de contrôleur. L’une s’appelle “Engine”, qui est le moteur du jeu,qui peut lire et modifier les données de la couche data(modèle), l’autre est s’appelle “HomePageController”, qui est le contrôleur des boutons de fichier ”HomePage.fxml”,il contrôle la transition entre les deux interfaces graphiques.

Le package “tools” contient deux classes , une classe appelée “GameDefaultParameters“ stocke les valeurs par défaut des paramètres du jeu,l’autre s’appelle “User”, il stocke l’énumération “Command” qui sera indiqué la direction dans laquelle le serpent se déplace.

Le package “interfaces” contient toutes les interfaces utilisées par l’application, telles que l’interface ReadService qui sert à lire les donne depuis le Modèle du jeu (couche data), l’interface DataService qui est implémenté par la classe “GameModel” et sert à lire et modifier les données dedans etc.
La relation entre les classes de notre projet est présentée dans le diagramme de classe suivant .Pour faciliter la visualisation, le diagramme suivant n’inclut pas les classes du package “tools”, car elles ne sont pas la partie la plus importante du jeu

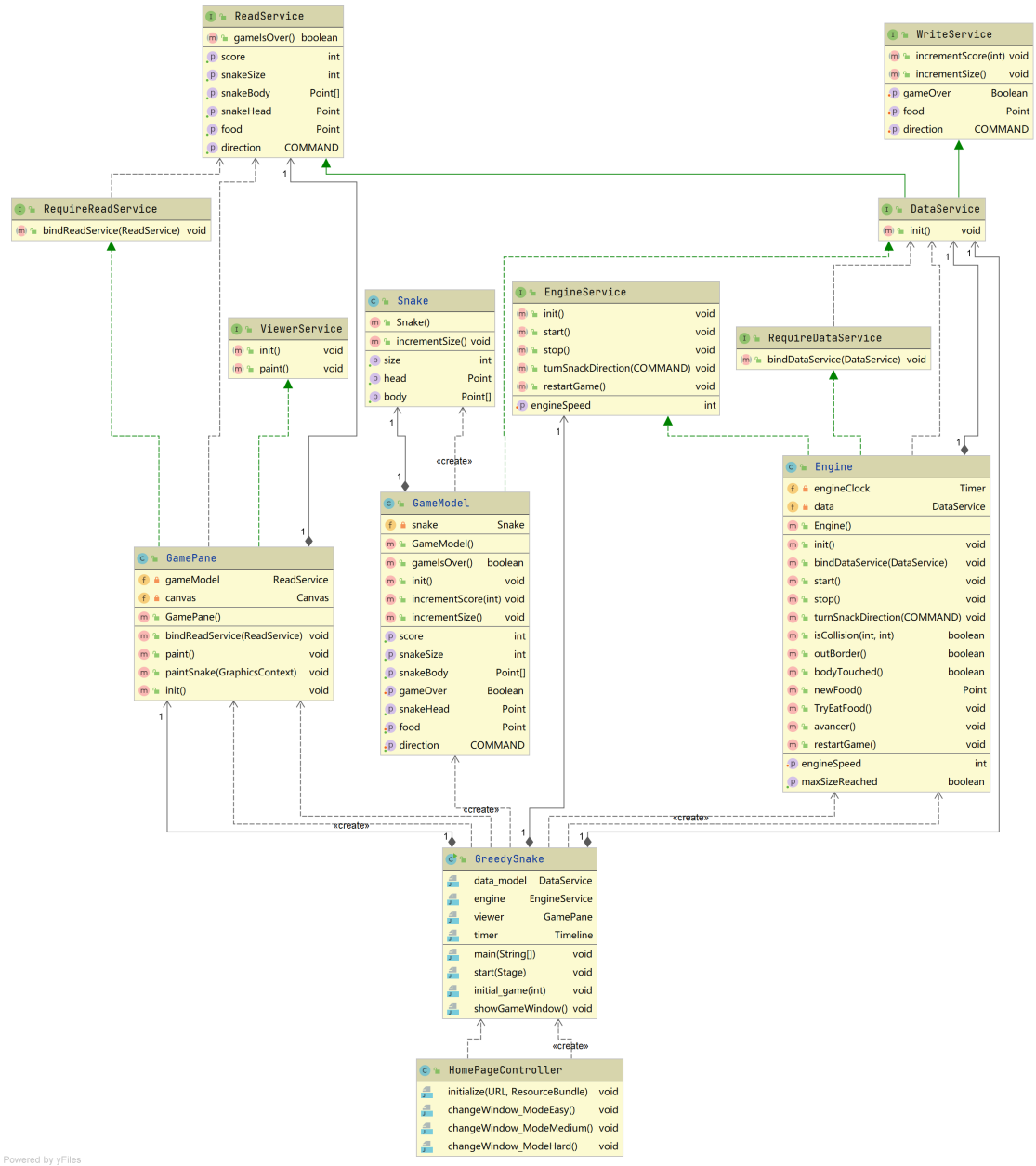

Figure 11: La diagramme de class du projet.

3.2 La couche client

Notre jeu contient deux interfaces graphiques : HomePage.fxml, GamePane.java

Le “HomePage.fxml” est une vue statique, il contient un label et trois boutons, Le label sert à afficher le nom du jeu et les trois boutons consistent à lancer le jeu avec trois niveaux différents(Plus le niveau est élevé, plus l’intervalle entre les rafraîchissements du moteur est court et plus le serpent se déplace rapidement.) cette interface graphique

est implémenté comme le maquette de notre de conception , nous avons rien changé

La classe “GamePane.java” est une vue dynamique, elle lit les données mais ne les modifie pas,elle implémente l’interface “ViewService” et l’interface “RequireReadService” afin de pouvoir lire les données depuis la couche data(modèle). Elle a un Canvas qui est utilisé pour afficher des objets changeants tels que du serpent, de la nourriture, du score, etc.

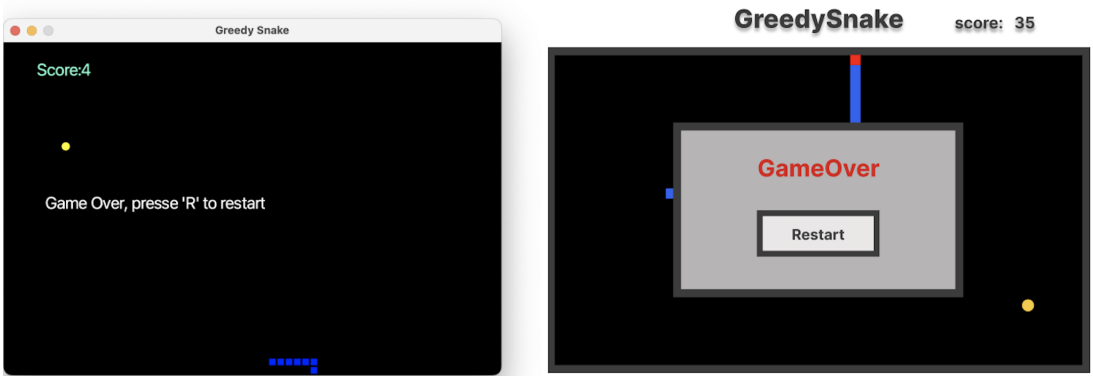

Figure 12: Rendu graphique vs. maquette de départ.

Elle est un peu différente de notre idée originaledans le maquette au depart, lorsque le jeu se termine, une fenêtre apparaîtra pour indiquer la fin du jeu et un bouton sera inclus pour permettre à l’utilisateur de redémarrer le jeu. Dans notre jeu, pour simplifier l’implémentation, lorsque le jeu se termine,au lieu d’une fenêtre , une ligne de texte apparaît sur le canvas indique la fin du jeu et informe le joueur qu’il peut appuyer sur la touche “R” pour redémarrer le jeu.

GreedySnake.java est le moteur de rendu graphique , À intervalles réguliers, il appellera la fonction paint() pour redessiner l’écran du jeu.Elle est aussi le moteur de commandes,lorsqu’il reçoit un commande , Il appelle la fonction de moteur du jeu pour qu’il soit traité.

Le changement entre les deux interfaces graphiques est implémenté de la manière suivante Dans la fonction start(Stage stage) de la classe GreedySnake, le vue “HomePage” est lancée, et lorsque le joueur clique sur le bouton de HomePage, le contrôleur des boutons va appelle la fonction showGameWindow() de la classe GreedySnake pour afficher la deuxième vue.

3.3 La couche data

La couche data de notre jeu contient deux classes: Snake.java et GameModel.java

Snake.java stocke des données concernant le serpent comme la taille ,le corps de serpent. Le corps est représenté par une tableau de point,la tête du serpent est l'indice 0 du tableau.

GameModel.java stocke l'ensemble des données utilisées dans le back-end du jeu,il implémente l'interface DataService.il contient cinq attribut :Snake snake, Command direction,boolean gameOver, Point food et int score. Le booléen gameOver est utilisé pour déterminer si le jeu est terminé. Lors de la initialisation, il vaut false;Ce booléen est utilisé de nombreuses fois dans le moteur.

3.4 La couche serveur

La couche serveur(controleur) de notre jeu se trouve dans le package "controller".

Engine.java est le moteur de jeu. Il implémente l'interface EngineService et l'interface RequireDataService. Il peut à la fois lire les données de la couche de data et les modifier.

Il contient un Timer,le timer est initialisé dans la fonction init( ), est lancé dans la fonction start(), est terminé par la fonction stop(). Après le démarrage de Timer, il appelle la fonction avancer() à intervalles réguliers.

Lorsque la vue de front-end détecte l'événement de pression de touche de flèche , la fonction turnSnackDirection()du moteur sera appelée pour changer la direction de l'avancée du serpent.

La fonction avancer() intègre l'ensemble des actions à effectuer dans chaque itération. Elle avance d'abord chaque point du tableau représentant le corps du serpent d'une case (sauf l'indice 0 qui représente la tête du serpent) ,et puis en fonction du sens de l'avance,il détermine la nouvelle position de la tête.

Elle appelle ensuite les fonctions outBorder(),bodyTouched() pour déterminer si le déclencheur de mort a été atteint .Et puis, elle appelle la fonction TryEatFood() pour déterminer si le serpent a mangé la nourriture, et si c'est le cas, elle ajoute un point au tableau du corps du serpent,la position de "Food" sera également mis à jour.Ensuite

,elle vérifie si le serpent a atteint sa longueur maximale,si c'est le cas , le jeu sera également terminé. A la fin, elle appelle gameIsOver() pour déterminer si elle doit appeler la fonction stop() pour arrêter le moteur.

Le mécanisme de détection des collisions utilisé dans le jeu (outBorder(),bodyToched(), isCollision() etc) n'est pas complexe. Comme dans le back-end, le "Food", le corps de "Snake" etc sont tous représentés par des points,détecter s'ils sont entrés en collision signifie détecter si deux points ont exactement les mêmes coordonnées.

Le HomePageController.java est le contrôleur des composants du fichier "HomePage.fxml". Lorsqu'un bouton est cliqué dans la page d'accueil, le contrôleur appelle différentes fonctions pour régler la vitesse du moteur en fonction du bouton cliqué et appelle la fonction ShowGameWindow() pour afficher l'écran de jeu.

4 Conclusion

Grâce à ce projet, nous avons acquis une compréhension plus claire de l'architecture MVC du jeu. Nous avons également amélioré nos compétences en matière de codage, de gestion de projet et de structure de projet de nombreuses façons. En plus de cela, nous avons appris les caractéristiques des différents langages et leurs forces et faiblesses en les comparant, ainsi que les outils les plus populaires sur le marché aujourd'hui, ce qui nous a beaucoup aidés et nous a donné une direction plus claire pour nos études futures.