Rapport de développement

Table des matières

Architecture du projet

Tout d'abord notre projet est composé de 3 packages principaux :

- Contrôleur : Contient tous les contrôleurs (réponse utilisateur) ainsi que les différents éléments d'interaction (boutons personnalisés)
- Model : Contient toutes les classes concernant les objets et les données utilisés dans les contrôleurs.

  Nous avons divisé ce package en 3 sous packages :

  - Fight : Contient toutes les classes liées au combat (combat, dresseur, bot, League)
  - Pokédex : Contient la classe Pokédex
  - Pokémon : Contient les classes concernant la construction d'un Pokémon (Pokémon, types, capacités, stats...)
- Vue : Contient notre classe main, une classe mainView ainsi que des fichiers FXML que charge JavaFX contenant nos interfaces.

Nous avons donc utilisé l'architecture MVC car nous l'avions déjà mise en place par le passé, notamment avec JavaFX, et nous semblait aussi correspondre à nos besoins dans ce projet.

Répartitions des tâches

Nous avons commencé en imaginant ensemble l'architecture du projet.

Nous avons ensuite travaillé de manière parallèle en 2 phases :

- Objets et construction des données :
  - Fabien : Classe Pokédex

    Lucas : Classes du package model.pokemon
- Combats et Pokédex
  - Fabien : FightController, classes du package model.fight sauf League et interfaces respectives + Rapport et Guide d'utilisation
  - Lucas : Classes PokedexController, LeagueController et GameController et interfaces respectives + Rapport et README

Difficultés rencontrées et résolution

Utilisation de Set d'énumération :

- Nous avons voulu utiliser un set d'une énumération pour la gestion des types mais cela posé problème lorsque nous voulions nous en servir dans des collections, il a fallu utiliser des ENUMSET. Mais cela nous à poser problème car l'ENUMSET est un objet non sérialisables nous avons donc créer notre propre Set de Types.

Utilisation de Java Fx :

- Pour utiliser Java Fx nous avons dû implémenter des contrôleurs, mais la communication entre les différents contrôleurs était compliquée car à chaque chargement de page un nouveau contrôleur est créé. Nous avons donc eu recours à l'utilisation d'une Classe contenant des champs Static ce qui nous a permis la communication entre les différents composant.
- De plus, nous voulions rendre un projet fonctionnel mais aussi beau au niveau de l'interface. Nous avons donc été contraint de passer du temps sur l'interface et à trouver des solutions pour créer de l'attente, jouer de la musique, changer de scènes...

Lier fonctionnement et interface :

- Dans les contrôleurs, le fonctionnement du jeu passait beaucoup par l'interface et l'action utilisateur. Il était donc difficile de créer des fonctions génériques qui permettaient à la fois de lier les conditions, la mise à jour des données et leur affichage.

Alternance, partiels et répartition des tâches :

- En effet, nous étions tous les deux en alternance, ce qui fait que je (Fabien) n'ai eu qu'une semaine de vacances et Lucas aucune. Nous n'avions d'ailleurs pas pris en compte cette différence de disponibilité à la base ce qui fait que la répartition des tâches et la communication fut compliquée au milieu du projet. De plus, nous devions réviser nos partiels pour la rentrée.

Ce qui a été implémenté

Les différentes tâches qui ont été implémenté pour ce projet sont :

- La lecture des fichiers pour permettre la création des Pokémons, ce qui inclut de gérer les statistiques, les capacités et les types des Pokémons et au final obtenir un Pokédex.
- Permettre par la suite au joueur de pouvoir sélectionné au sein de ce Pokédex les Pokémons et les capacités qui veut pour combattre.
- Création d'un système de sauvegarde du dresseur en cours pour que le joueur n'a pas à recréer son Dresseur à chaque partie.
- Création du mode combat permettant au joueur d'affronter l'ordinateur, qui sera représenté par un dresseur aléatoire.
- Création des League qui représente une suite de combat.
- Ajout de la possibilité de créer sa propre League et de pouvoir la sauvegarder puis la recharger par la suite.

Ce qui reste a implémenté

Les différentes tâches qui restent à implémenté pour ce projet sont :

- Un mode multijoueur
- Un interface de sauvegarde de Dresseur pour pouvoir en jouer plusieurs
- La gestion des objets lié à l'univers Pokémon
- Modifier l'état des Pokémon et implémenté les capacités de type spécial
- Permettre au Pokémon de gagner de l'expérience pour passer des niveaux et de pouvoir faire évoluer ses statistiques et au final évolué lui-même.
- Permettre suite à ce système de niveau de pouvoir acquérir de nouvelles capacités lors des passages de niveau
- Ajoutés un mode permettant de combattre avec des équipes de dresseurs.

Evolution possible du projet

Ce projet pourrait être notamment réutilisé pour :

- Créer un jeu Pokémon dédié aux combats et à leur mécanique

  Créer de nombreux styles de combats et de nouvelles mécaniques (combat Pokémon style Sumo, combat avec Obstacles, combat avec positon stratégique…)
- Recréer un vrai jeu Pokémon avec monde ouvert et qui engloberait toutes les générations de Pokémon.

Développeur Externe

Pour tout développeur externe souhaitant améliorer et continuer ce projet.

Interface

Il devra se servir de Java Fx pour ce qui est de la création de l'interface. Un fichier Fxml et un Contrôleur seront nécessaire pour tout nouvel écran. Pour changer d'écran il devra faire appel à la fonction statique changeScene de la classe MainView. Et pour ce qui ai de la communication d'information, il faudra passer par le Classe GameControllerStatic et utiliser des champs et méthodes statique.

Interdiction

Le projet se construit grâce à la lecture des fichiers de Pokémon ou de types, il ne faudra donc absolument pas toucher à ce fichier, ou si vous désirez ajouter des informations dans les fichiers il faudra bien respecter le formatage du fichier.

Ajout de fonctionnalité

L'ajout de fonctionnalité pourra se faire sans soucis car tous les modes de base sont dérivables et grâce au système de contrôleur ou une interface correspond à une fonctionnalité l'ajout de nouveau mode ne posera pas de problèmes.

Point d'Attention

Il faudra faire bien attention à respecter les packages créer et correspondant au Model View Controller afin d'avoir un affichage le plus fonctionnel possible.