Rapport projet PFA 2018-2019

Lam NGUYEN THIET Kenyu KOBAYASHI

11 mai 2019

1 Introduction

Ce projet implémente un jeu dans le langage OCAML, en utilisant les fonctionalités fonctionelles (en majorité), impératives, orientée objet.

2 Le jeu

2.1 But

Le but du jeu est de détruire les factions ennemies. Pour se faire, il suffit de tuer toutes leur unités.

2.2 Factions

Il y a 3 factions dans le jeu. Les cagoulés, les vert et les bleus. Vous contrôlez les bleus.

2.3 Unités

Il existe deux types d'unités.

— Le soldat est l'unité de base. Elle peut se déplacer, attaquer et récupérer des items.

— La ville permet de faire apparaître des soldats. Cette unité est très importante car si on la perd, on ne peut plus produire de soldat.

2.4 Items

Il existe deux items.

— Le pack de soin régénére la vie des unités. Il est possible d'avoir plus de points de vie que l'on avait au départ.

— La bombe nucléaire détruit tout dans un rayon de 3 cases.

2.5 Plateau du jeu

Le plateau est une grille d'hexagone. La carte est une île deserte dans l'océan avec des biômes variés.

2.5.1 Type de cases

Il y a 3 types de cases. A l'heure actuelle, elle ne sont que cosmétiques, par la suite elle peuvent avoir un impact sur l'efficacité de combat de tel unité de tel faction, mais par manque de temps je n'ai pas pu les faire.

Il y a la neige, le desert et l'herbe.

2.5.2 Caractèristiques du terrain

En plus du biome, il y a des caractèristiques sur le terrain pour chaque case.

— Les forêts et les collines coûtent plus cher pour le movement.

— Les montagnes et les lacs sont des obstacles ne peuvent pas être traversés.

2.6 Tour par tour

Le jeu se déroule en tour par tour, similaire au jeu *Civilization*. C'est d'ailleurs sur quoi je me suis inspiré pour le jeu.

2.6.1 Mouvement

Chaque unité a un nombre de movement. Lorsqu'il se déplace d'une case, il consomme n points selon la case sur laquelle il atterit. Dans le jeu, les soldats sont les seuls à pouvoir se déplacer. Les villes ne peuvent pas.

2.6.2 Attaque

Pour tuer les autres unités il faut les attaquer. Encore une fois de manière analogue à *Civilization*, les unités disposent d'une force d'attaque et d'une force de défense.

Par la suite, src et dst représente respectivement l'unité qui attaque et l'unité qui défend.

Lorsque deux unités s'attaquent, les nouveaux point de vie se calculent de cette manière :

$$healthpoints_{dst,new} = max\{0, healthpoints_{dst,old} - strength_{attack,src}\} \quad (1)$$

$$healthpoints_{src,new} = max\{0, healthpoints_{src,old} - strength_{defense,dst}\} \quad (2)$$

Et leurs nouvelles positions :

$$x_{dst,new}, y_{dst,new} = \begin{cases} \texttt{null}, \texttt{null} & \text{si } healthpoints_{dst,new} = 0, \\ x_{dst,old}, y_{dst,old} & \text{sinon.} \end{cases} \quad (3)$$

$$x_{src,new}, y_{src,new} = \begin{cases} \texttt{null}, \texttt{null} & \text{si } healthpoints_{src,new} = 0, \\ x_{dst,old}, y_{dst,old} & \text{si } healthpoints_{dst,new} = 0, \\ x_{src,old}, y_{src,old} & \text{sinon.} \end{cases} \quad (4)$$

3 Répartition du travail

3.1 NGUYEN THIET

- Gestion des appels à la bibliothéque SDL (*e.g.,* gestion du render, chargement des textures, initialisation du windows, *etc.*)
- Dessins (tuiles, caractèristiques de terrain, soldats, ville, interfaces, fond d'écran, titre, items, effets spéciaux)
- Boucle principale. Gestion d'un type `context`, comment le mettre à jour et comment récupérer informations contenus pour avancer le jeu dans le temps
- Définition du type des `unités` et les méthodes associées
- Implémentation du plateau de jeu et la grille d'héxagone. (Les pseudo-codes se trouvent ici https://www.redblobgames.com/grids/hexagons/)

3.2 KOBAYASHI

- Calcul des PV et mise à jour du plateau pendant les attaques
- Réglages de la taille de la carte dans le menu
- Fond d'écran des réglages (juste le dégradé)