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 :

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

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

2.7 Intelligence Artificielle

Les IA sont assez simple. De base, elles se déplacent au hasard. Si elles rencontrent un ennemi, elle va se diriger vers cet ennemi en priorité. Si il y a un autre ennemi elles s'attaquent, même si ce n'était pas l'ennemi en priorité.

En dessous d'un certain seuil, les IA vont chercher à fuire et chercher un pack de soin. Mais si il y a un ennemi tout prés, elles vont se suicider et attaquer cet ennemi, car elles savent qu'elles vont mourir et vont préférer attaquer pour donner une chance à leur alliés.

Sinon, en mode patrouille, si elles voient une bombe nucléaire, elles la prennent et l'utilise sur un ennemi au hasard.

3 Répartition du travail

4 Développement du jeu

Chaque phase sera développée plus en profondeur dans la suite du rapport. Chaque implémentation est listé par ordre chronologique dans le développement.

Phase 1 : Fondation

1. Familiarisation avec `SDL` et factorisation de code qui était assez récurrent.
2. Implémentation de la boucle principale
3. Généralisation du type `context` et sa mise à jour

Phase 2 : Instances de jeu

1. Définition des instances du jeu (la boucle du menu, du jeu etc.)
2. Création de boutons temporaires pour lancer le jeu et le quitter si on appuie sur la croix

Phase 3 : Plateau de jeu

1. Définition du plateau de jeu (représenté par une matrice)
2. Implémentation des fonctionalités de la grille d'hexagone

3. Définition des tuiles et des `enum` qui la définissent
4. Les dessins du plateau (tuiles, forêts, etc.)

Phase 4 : Unités

1. Définition des unités
2. Définition des constantes qui les définissent
3. Les dessins des unités

Phase 5 : Actions

1. Formalisation et généralisation des actions et leur retour pour qu'ils puissent tous être du même type
2. Implémentation des actions de déplacement et attaques
3. Interaction temporaire avec les unités avec le clavier
4. Interaction du retour des actions avec le système de contexte

Phase 6 : IA

1. Formalisation d'un comportement d'une unité
2. Systèmes similaire aux automates d'états finis pour sélectionner le comportement de chaque unités selon son environment
3. IA qui se déplacement au hasard,
4. et attaque une cible si il y en a une qui se trouve à proximité,
5. et qui va chercher des packs de soin si elles n'a pas beaucoup de points de vie,
6. et qui va chercher les bombes nucléaires si elle peut.

Phase 7 : Interface

1. Affichage des informations liés à chaque unités (ses points de vie et points de mouvement)
2. Système d'interface avec des *event listeners*
3. Formalisation et généralisation des interactions avec les interfaces dans le contexte