Rapport Lot A

Gautier Poursin, Théo Lazzaroni, Bastian Padiglione, Thibaut Arnould–Demongeot

22 mars 2020

Introduction

Cette première partie du projet correspond à la mise en place du projet et à la réalisation de toutes les interfaces nécessaires pour le reste des modules.

Tâche A1

Cette tâche correspond à l'assemblage de tous les modules crées (**plateau.h, interface.h et carte.h**), afin de permettre le bon déroulement d'une partie.

Tout d'abord, il faut initialiser une partie, en créant donc un nouveau plateau, en allouant l'espace mémoire nécessaire et en initialisant le nombre de tours et de points de DD à 0. Ensuite, tant qu'on a pas de vainqueur et qu'on a pas atteint 30 tours, on continue de jouer d'où cette boucle **while**. On affiche un nouveau tour à chaque itération et sachant qu'il y a 2 phases par tour, la boucle **for** permet de jouer 2 phases en un tour. On choisit quel joueur joue (**Player playerWhoPlays**), (aléatoire au premier tour puis on alterne) et le nombre de carte qu'il doit tirer (**int cardDraw**). On ajoute la ou les cartes piochées à la main du joueur avec la fonction **addCard**, qui n'est pas encore défini dans un module. Le joueur choisit s'il veut une carte FISE ou FISA (**chooseCard(playerWhoPlays)**). Ensuite, le joueur effectue une action, poser ou arrêter son tour,(**actionTurn(playerWhoPlays)**), d'où la boucle **while**. Tant qu'il veut et qu'il peut jouer une carte, il joue. On met à jour le plateau (**majBoard(boardGame)**), puis le joueur adverse effectue sa phase. Enfin, on affiche le plateau (**printDraw(boardGame)**), correspondant au tour joué, on met fin au tour puis on recommence si on a pas de gagnant ou que le nombre de tours limite n'est pas atteint.

Tâche A2

Tout d'abord, nous avons décidé de créer le type **list**, qui correspond à une liste d'entier. En effet, cela nous sera utile pour représenter la main . Nous

privilégions les listes car nous aurons accès à toutes les cartes de la main alors qu'avec une pile, on aurait accès que à la tête de la pile par exemple. Nous allons donc utiliser le type **Pile** pour le deck ou la défausse, qui sont des empilement de cartes et le type **File** pour les cartes personnels. En effet, il y a au plus 3 places donc des qu'on joue une quatrième carte personnel, la premiere carte rentrée sort.

Ensuite, nous avons défini le type **Player**. Chaque joueur possède une partie de plateau avec dessus : **un deck, une défausse, une main, une pile de FISE/FISA, des emplacements de personnel.** Tout ces éléments seront représentés par des listes d'entiers, où un entier correspond à une carte précise. De plus, le joueur aura des points **d'énergie et de DD**, qui seront du type **int**.

Le type **Board** représentera le plateau de jeu complet, il possède donc 2 **Player** représentant chacun une ENSIIE et un compteur du nombre de tour du type **int**.

Concernant la mise en place des fonctions, la fonction **newBoard** permet l'initialisation d'une partie. Elle permet de créer un plateau vide. Elle ne prendra donc aucun argument mais renverra un plateau. La fonction **freeBoard** libère la mémoire qui a été alloué pour un plateau. Elle prend en argument un plateau et ne renvoie rien. La fonction **majBoard** permet de mettre à jour un plateau ie elle incrémente le nombre de tour, ajoute les nouvelles cartes mises en jeu... Elle renvoie donc un plateau. Enfin, la fonction **draw** permet à un joueur de piocher une carte. Elle renvoie la carte piochée et prend en argument le joueur qui pioche une carte.

Tâche A3

Cette partie a pour but d'écrire l'interface **carte.h** qui permettra notamment de manipuler les cartes du jeu.

On a d'abord défini le type **Card** contenant une **speciality**, un **cost**, un **effect** et un **id**. Ce type va nous permettre de pouvoir collecter toutes les informations d'une carte simplement en utilisant son id. Le type est défini dans **plateau.h**.

Ensuite, différentes fonctions permettent d'utiliser les cartes. On a **numberStudentCardsDrawn** qui permet de calculer le nombre de cartes **Elèves** qu'un joueur **player** doit piocher au début du tour. Puis, les fonctions **addFISE** et **addFISA** pour d'ajouter un FISE ou un FISA au joueur **player**. La fonction **nbPEAvailable** donne le nombre de PE disponibles du joueur **player**. Une fonction appelée **playCard** renvoie l'état de **player** après que celui-ci ait joué la carte d'**id** n.

Enfin, une fonction **endTurn** permet de renvoyer l'état du **Board** après la fin du tour et la fonction **endGame** vérifie si la partie est terminée ou non.

Tâche A4

La dernière tâche du lot A avait pour but de rédiger l'interface **interface.h** qui est en charge de tous les échanges entre les joueurs humains et le jeu : afficher le plateau, demander si le joueur veut une carte FISE ou FISA, demander s'il souhaite finir sa phase, etc.

Tout d'abord, une fonction **printNewTurn** qui prend un **board** en argument a pour but de donner toutes les informations nécessaires aux deux joueurs en début de tour. Puis, une fonction **newPhase** affichant en console un message signifiant qu'une nouvelle phase commence et l'ENSIIE associée à cette phase. Une fonction **printBoard** qui affiche sur la console l'état à l'instant t du plateau. La fonction **chooseStudentCard** permet de demander au joueur s'il choisit de tirer une carte FISE ou une carte FISA. Pour jouer une carte ou simplement passer son tour, un joueur va utiliser la fonction **actionTurn**. Enfin, à la fin de la partie, la fonction **printWinner** affiche le résultat de la partie.