Projet informatique

Romain LOIRS, Damien LU, Rémi DECOUTY, Florian LECOMTE

24 mai 2020

Table des matières

1 Les structures

Toutes les structures sont définis dans le fichier structure.h pour plus de lisibilité

1.1 Les cartes

1.1.1 Le type et le nom

Une carte est définie par deux paramètres : son nom et son type. Nous avons choisi d'utiliser un maximum de types énumérés pour les noms des cartes et les types car manipuler des entiers est plus simple que de tester une chaine de caractères. Dans les règles du jeu, avoir le nom implique de connaitre le type. Il serait donc légitime de supprimer les types de la carte pour gagner de l'espace mémoire. Mais si nous désirons complexifier le jeu en ajoutant des cartes qui ont des noms identiques mais des types différents, il est intéressant de conserver le type de la carte.

1.1.2 Le code

Ensuite, dans l'optique de créer des listes chainées, il a fallu créer une structure de carte valable pour tout les types de cartes. Cependant, les cartes n'ont pas les mêmes caractéristiques. Nous dénombrons 4 informations importantes

— les points énergies, durabilités et développement pour les cartes Élève
— le coût pour les cartes actions et Personnel

La solution la plus simple serait de créer une variable (de type int) pour une information. Par conséquent, des variables seraient inutilisées donc c'est de l'allocation mémoire inutile.

Par conséquent, nous avons compressé les informations des points développement durable, énergie , durabilité et le cout dans un code que nous avons appelé (sobrement) code de type long. Nous passons ainsi de (4 x 2 bits) 8 bits de mémoire pour la création de 4 variables de types int à 1 long de 4 bits.

Le principe est simple : nous utilisons un long à la place d'un int. Un int va jusqu'à 32600 environ alors qu'un long va jusque 2000000000 environ. Ainsi, pour le coût, comme il s'agit de la seul information pour les cartes de type personnel et action, il suffit de faire code=cout et la manipulation est simple. Pour les cartes élève, c'est ici que cela devient intéressant. Un long va jusque 2000000000 (et un peu plus). Nous n'utilisons pas le 2, donc nous pouvons utiliser 9 chiffres. (cela tombe bien nous avons 3 types de points à insérer). Donc nous utilisons les 3 premiers chiffres pour les points énergie, les trois suivants pour les points durabilité et les trois derniers pour les points développement durable. Par conséquent, on ne pourra pas aller à plus de 999 points pour chaque type de points. Mais étant donné les règles, il est déjà difficile d'atteindre plus de 100 points pour un type de point.

Exemple : prenons une carte Élève dont le code est 123.078.051 (les points ont un but esthétique). La carte possède donc 123 points énergie, 78 points durabilité et 51 points développement durable.

1.2 Le deck

Pour le deck, nous avons utilisé un format de liste chaînée où chaque maillon contient une carte. Nous n'avons pas utilisé de tableau car les decks sont de longueur variable tout au long de la partie.

Les fonctions qui manipulent les decks définis dans `structure.h` ne contiennent pas de structure de contrôle (par exemple, les fonctions ne vérifient pas si avant d'enlever un élément, la liste est non vide). Celles-ci sont définies à part et permettent, lors de leur utilisation dans d'autre fonctions, au programmeur de définir un message d'erreur plus spécifique que par exemple "le deck est vide" qui n'est pas très précis.

1.3 Le plateau

Le plateau est unique pour une partie et contient l'intégralité des decks des joueurs, c'est à dire :
— les mains du joueur A et du joueur B
— les défausses du joueur A et B
— les decks du joueur A et du joueur B
— les piles des cartes FISE et FISA des joueurs A et B
— la pile side des joueurs A et B (où sont envoyées les cartes actions et personnels)
Il contient également des indications sur le tour et le joueur afin de connaître l'état d'avancement de la partie :
— le numéro du tour
— les points développement des deux joueurs
— les points énergie des deux joueurs

Celui-ci contient également des variables qui permettent de gérer certains effets des cartes personnels et actions.

2 Répartition du travail

2.1 Lot A

	Remi	Florian	Damien	Romain
Tâche A.1- Mise en place du projet et main.c	x			
Tâche A.2 - Rédiger carte.h				x
Tâche A.3 - Rédiger plateau.h		x		
Tâche A.4 - Rédiger interface.h			x	

2.2 Lot B

	Remi	Florian	Damien	Romain
Tâche B.1 - Lier tous les fichiers	x			
Tâche B.2 - structures.h			x	
Tâche B.3 – structures.c				x
Tâche B.4 – Getters de carte.h et carte.c				x
Tâche B.5 – Getters de plateau.h et plateau.c		x		
Tâche B.6 – plateau.c		x		
Tâche B.7 – interface.c			x	
Tâche B.8- Test du code	x			

2.3 Lot C

	Remi	Florian	Damien	Romain
Rédaction rapport				x