Projet informatique

Romain LOIRS, Damien LU, Rémi DECOUTY, Florian LECOMTE

17 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 définis par deux paramètre : son nom et son type. j'ai choisis d'utiliser un maximum des types énumérés pour les noms des cartes et les types car manipuler des entiers est plus simples pour les tests qu'une chaine de caractéres. Dans les régles du jeu, avoir le nom implique de connaitre le type. Il serait donc légitime de supprimer les type 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. Ainsi, conserver le type de la carte nous paraissait important.

1.1.2 le code

Ensuite, dans l'optique de créer des listes chainés, 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 Eleve
— le coût pour les cartes actions et et Personnel

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

Par conséquent, nous avons compressé les informations des points développement durable, energie , durabilité et le cout dans un code que j'ai 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 un long à la place d'un int. Le principe de codage est simple. Un int va jursqúe 32600 environ alors qu'un long va jusque 2000000000 environ. Ainsi, pour le cout, comme c'est la seul information pour les cartes de type personnel et action, Il suffit de faire code=cout et c'est simple à manipuler. Pour les cartes élève cést ici que cela devient interessant. un long vas jusque 2000000000 (et un peu plus). Je n'utilise pas le 2, donc je peux utiliser 9 chiffres. (cela tombe bien j'ai 3 types de points à insérer). Donc j'utilise les 3 premiers chiffres pour un point, les trois suivants pour l'autre. Par conséquent, on ne pourra pas aller à plus de 999 points pour chaque type de points. Mais étant donnée les régles, il est déjà difficile d'atteindre plus de 100 points pour un type de points.

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

1.2 deck

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

Les fonctions qui manipulent les deck 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). Celle-ci sont défini à part et permet lors de leur utilisation dans d'autre fonctions de permettre au programmeur de définir un message d'erreur plus spécifique qu 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 deck des joueurs. Il contient également des indications sur le tour afin de connaitre l'état d'avancement de la partie.

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