05/01/2015

Jeremy BRESSAND
Victor CONNES
Alain ALIGNAN

HLIN 301 :

Programmation impérative avancé

[RAPPORT DU PROJET : « NAIN JAUNE »]

Note du projet « Nain jaune » contenant l'analyse du sujet ainsi que nos raisonnements et notre analyse du sujet.

Introduction :

Le but de ce projet est de créer le jeu du Nain jaune à travers le langage à objet C++. Notre jeu ouvre une fenêtre dans le terminal contenant le plateau de jeu du nain jaune, les mises, le nombre de cartes, une interface de dialogue entre les joueurs et le jeu ainsi qu'un tableau contenant la liste des joueurs, le dernier joueur ayant distribué, le joueur à qui c'est le tour de jouer et le nombre de jetons de chaque joueur . L'ordinateur affiche notre main et à l'aide du clavier nous pouvons choisir quelles cartes jouées.

Il est possible de jouer entres amis, avec des amis et des ordinateurs ou seul contre les ordinateurs. Nous avons le choix entre 6 niveaux différents et nous pouvons jouer de 3 jusqu'à 6 joueurs.

Dans la note du projet, vous trouverez les principaux axes et éléments du programme ainsi que l'explication de nos choix

I) Analyse globale

I-1) Choix de la partition en classe du programme

Au départ, ne sachant pas par où débuter, nous avons décidé de découper le travail en classe, afin de mieux se représenter le code et de pouvoir partitionner le travail.

Pour ce faire, nous avons choisi les classes suivantes :

I-1-a) Carte

C'est une des classes les plus simples du programme. Pour désigner son enseigne et son numéro, on a choisit d'utiliser des size_t car leurs valeurs se limitent à des valeurs positives ce qui peut éviter certains bug et ajoute de la clarté dans notre programme.

On a ajouté des surcharges d'opérateurs afin de rendre notre code plus lisible et plus facile à utiliser.

I-1-b) Case

Cette classe ressemble fortement à la classe Carte. Mais elle n'a pas la même fonctionnalité, en effet elle doit contenir des jetons et permet l'affichage des cases avec les jetons. De la même manière nous avons ajouté des surcharges d'opérateurs.

I-1-c) Listecarte et Cellcarte

Dans notre utilisation des cartes, on doit pouvoir les ajouter ou les enlever rapidement de notre structure de donnée. Pour optimiser la vitesse de ces opérations, on a choisi d'utiliser des listes doublement chainées. De plus comme on doit utiliser de la mémoire dynamique car comme nous utilisons beaucoup de cartes et de copies de cartes, utiliser la mémoire statique risquerait de faire déborder la pile. Ainsi les méthodes de Listecarte nous permettent de manipuler les cartes en toute sécurité.

I-1-d) Main

Main contient comme attribut un tableau de Listecarte dont on justifiera le choix ultérieurement.

Comme le joueur devra rechercher, ajouter et enlever des cartes dans sa main, on a décidé de créer la classe main qui réalisera l'interface entre le joueur et ses cartes. Ainsi on évite d'écrire trop de code dans la classe Joueur (dont nous parleront ultérieurement) et on le rend plus lisible.

Parmi les méthodes de recherche de la classe main, certaines nous sont très utiles pour créer une intelligence artificielle.

La méthode pluslongueliste() correspond au niveau 4 de l'intelligence artificielle et permet de trouver la plus longue suite de cartes consécutives. En cas d'égalité entre deux suites, la fonction garde la suite qui commence avant l'autre.

La méthode pluslonguelistespeciale() correspond au niveau 5 de l'intelligence artificielle qui étant similaire à la méthode précédente, trouve la plus longue suite qui contient une carte spéciale. A défaut de carte spéciale dans la main, elle trouve la plus longue suite avec les mêmes critères en cas d'égalité.

La méthode plus longuelisteroi() correspond au niveau 6 de l'intelligence artificielle. Elle trouve la plus longue suite finissant par un roi. A défaut de roi, elle ne fait rien.

I-1-e) Joueur

Ne pouvant utiliser l'hérédité pour distinguer les joueurs humains des joueurs virtuels, on a mis dans la classe joueur l'attribut booléen cpu. Lorsque l'on demande au joueur de jouer une carte (on fait appel à sa méthode jouerCarte()) la méthode regarde si l'instance référente de joueur est virtuelle ou non et agit en fonction.

I-1-f) Plateau

Plateau est la classe mère qui englobe toutes les autres classes. Cette classe englobe aussi les fonctions principales contrôlant le bon déroulement du jeu, cela permet d'enlever du code dans le fichier main et donc de le rendre plus lisible. Elle contient le tableau de carte ainsi que le paquet de cartes qui représentent à eux deux le matériel de jeu.

I-1-g) Score

L'écriture des scores étant une manœuvre délicate, automatiser son déroulement dans une classe permet de rendre la tâche plus facile et de rendre le code plus lisible. La classe score permet entre autre d'écrire la date et l'heure à laquelle a été réalisé l'exploit du joueur.

II) Choix des structures de données

II-1) Tableau de listecarte

Dans la classe main, non avons mis l'argument : Listecarte cartes[13].

Faire un tableau de listes de cartes a plusieurs intérêts.

- Nous n'avons plus besoin de trier les cartes : lors de l'ajout de carte, on insère la carte à la bonne place dans la liste de cartes à l'indice du tableau correspondant au numéro de la carte. De même lors de la suppression, ainsi la liste reste triée.
- Cette manière de faire correspond bien à la réalité, lorsque le joueur ajoute une carte à sa main il l'insère de sorte que sa main soit triée.
- L'accès à une carte est plus rapide et s'effectue quasiment en temps constant (puisque la liste ne contient jamais plus de 4 cartes).
- Les méthodes de l'IA sont plus simples : En effet pour tester si l'IA a une carte de numéro 7, on regarde si la case 6 du tableau de listes de cartes est vide ou non. Dans le cas d'un tableau de cartes ou d'une liste de carte globale, il aurait fallu parcourir le tableau ou la liste pour vérifier la présence ou non de tel numéro de carte.

II-2) Tableau

Dans la classe plateau, nous avons mis l'argument Joueur *joueur. On affecte ce pointeur à un tableau de joueurs créé dynamiquement lors de la construction du plateau dans le fichier main. La taille de ce tableau correspond au nombre de joueurs choisi par l'utilisateur.

Faire un tableau de joueur a plusieurs intérêts :

- Accès à tel joueur en temps constant.
- Parcourir l'ensemble des joueurs est plus facile et lisible que dans le cas d'une liste de joueur.
- Aucun danger à manipuler les joueurs : dans le cas d'une liste, lorsque l'on veut accéder à la valeur d'une cellule, il vaut mieux avoir une copie de cette valeur car si on obtient la valeur par référence, on peut alors supprimer la mémoire dynamique associée à cette valeur et avoir des cellules qui ont pour valeur quelque chose pointé à une adresse mémoire dé allouée.

II-3) Les énumérations

Nous avons choisi de ne pas faire d'énumérations car nombre de nos méthodes qui renvoient des cartes utilisent le renvoie de cartes particulières (Carte(0,0), Carte(5,5), Carte(6,6)) pour

transmettre une information comme par exemple : l'absence de carte dans la main ou l'activation du mode debug.

Nous aurions pu utiliser une énumération pour les cases mais comme les attributs enseigne et numéro des cases ne sont jamais modifiés nous avons trouvé cela peu intéressant. Enfin les autres classes ne justifiaient pas l'utilisation d'énumération.

III) Remarques particulière

III-1) L'IA

Après avoir joué plusieurs parties entre nous afin de bien comprendre les mécanismes de jeu et les points clés, nous nous sommes penchés sur le développement de l'IA. Pour ce faire, nous nous somme basés sur quelques principes élémentaires ; les voici par ordre d'importance décroissant :

- Mieux vos jouer le plus de cartes possibles en un tour.
- Il faut jouer ses cartes spéciales le plus rapidement possible.
- En cas d'égalité de longueur entre deux suites de cartes il vaut mieux jouer les cartes de la suite la plus faible.
- Quand on a le choix entre deux cartes spéciales mieux vaut jouer celle qui rapporte le plus.
- Enfin de manière générale il vaut mieux jouer une carte/liste de cartes de valeur faible.

Ces quelques principes élémentaires même s'ils sont critiquables (n'est-il pas plus intéressant de jouer une liste de 8 cartes qu'une seule carte spéciale) et entraînent une manière de jouer très sûre (se débarrasser le plus rapidement de ses cartes) nous ont permis de développer les six niveaux d'IA qui suivent.

III-1-a) Niveau 1

L'IA de niveau 1 est celle de niveau le plus faible. Elle joue la première carte de sa main (donc la plus basse) quand elle a la main et même si elle a deux cartes de même valeur dont une spéciale elle ne prend pas garde à jouer la carte spéciale. En effet là aussi elle prend le parti de jouer la première carte de la liste (liste des 7, liste des 10, ...).

III-1-b) Niveau 2

L'IA de niveau 2 est celle du niveau directement supérieur au niveau 1 ; c'est donc la deuxième plus forte. Lorsqu'elle a la main, elle joue sa première carte spéciale. Lorsqu'elle a deux cartes de même valeur dont l'une des deux est spéciale, elle choisit la carte spéciale.

III-1-c) Niveau 3

L'IA de niveau 3 réagit de la même manière que l'IA de niveau 2 à la différence que lorsqu'elle a la main elle joue la carte spéciale rapportant le plus de jetons.

III-1-d) Niveau 4

L'IA de niveau 4 réagit de la même manière que l'IA de niveau 3 à la différence que lorsqu'elle a la main elle joue la suite de cartes la plus longue de sa main indépendamment des cartes spéciales. Si elle a deux cartes de même valeur au sein de la suite dont l'une d'elle est spéciale elle choisit la carte spéciale.

III-1-e) Niveau 5

L'IA de niveau 5 lorsqu'elle a la main joue la suite de cartes la plus longue contenant une carte spéciale. A défaut d'avoir des cartes spéciales dans la main elle se comporte comme l'IA de niveau 4.

III-1-f) Niveau 6

L'IA de niveau 6 lorsqu'elle a la main joue la suite de cartes la plus longue contenant un roi. Si elle a deux cartes de même valeur au sein de la suite dont l'une d'elle est spéciale elle choisit la carte spéciale. Si elle n'a pas de roi dans la main, elle se comporte comme l'IA de niveau 5 (donc à défaut de roi et de cartes spéciales elle se comporte comme l'IA de niveau 4).

III-2) Le log

Le mode d'écriture étant relativement simple, nous avons décidé qu'il était plus judicieux d'ouvrir le fichier log.txt dans le fichier main et de donner aux fonctions qui écrivent dans le log une référence vers ce fichier, au lieu de créer une fonction pour cela dont l'implémentation serait en dehors des classes.

III-3) Le mode debug

Le mode debug permet à tout moment de changer le nombre de jetons sur les cases, le nombre de jetons des joueurs, le donneur (celui qui distribue les cartes) et l'ensemble des cartes des joueurs (à condition de ne pas ajouter des cartes aux joueurs qui sont déjà dans leurs mains et de ne pas leur enlever des cartes qui ne sont pas dans leurs mains).

Conclusion :

Le nain jaune nous a permis de découvrir beaucoup de notions en c++. Ce fût un exercice très complet qui nous a permis d'apprendre à optimiser notre programme tout en le complexifiant avec de nouvelles notions. De plus, ce projet nous a permis de beaucoup nous entraîner et d'apprendre à travailler en groupe en même et séparément sur un même projet en prenant garde de toujours sauvegarder nos données et bien synchroniser notre travail !