Projet du nain jaune par De Boisvillier Christopher, Cossin Tristan et Gaune Florian

Dès le départ, nous voulûmes faire un petit main. Ce pourquoi, comme vous le constaterez, nous fîmes appel à plusieurs fonctions.

Nous allons présenter ce dossier groupé par fonction et/ou fichier, qui eux même seront mis dans l'ordre chronologique de création, mise à niveau, ou mise à jour.

La classe carte :

Tout d'abord, nous créâmes une classe carte. Dans cette classe, nous mîmes comme attribut le nombre et la couleur. Cependant, après avoir continué à avancer dans ce projet, nous eûmes besoin de faire quelques mises à niveau. En effet, pour le tri des cartes, il nous fût indispensable de créer un nouvel attribut qui est la valeur. Ainsi, ce nouvel attribut va de 1 à 13 en fonction de la carte. De plus, afin de pouvoir faire l'affichage, nous dûmes créer deux nouveaux attributs : la valeur graphique, qui va de 1 à 52 afin de différencier toutes les cartes. Ensuite, pour l'affichage à l'écran et le choix de l'utilisateur, nous ajoutâmes l'indice graphique, qui va de 1 à n, avec un nombre de carte dans la main du joueur de n.

Une fois bien avancé dans le jeu, nous vîmes qu'il nous fallait également un nouvel attribut de valeur monétaire. En effet, au début, nous pensions juste prendre la valeur de la carte qui allait de 1 à 13, mais, d'après le polycopié, les têtes ne valent que 10 €. Ce pourquoi, nous optâmes pour la création de ce nouvel attribut au lieu de faire une grande condition dans les fonctions, car nous en avions déjà des pas mal… à un moment donné, nous créâmes un nouvel attribut nommé proprio. Comme vous pourrez le constater, on l'a enlevé. Pourquoi ? Et bien on ne sait plus… Nous l'avions créé à un moment donné, puis, pris une large pause de quelques semaines (à peine… dû à de nombreux contrôles continus), et lors de notre retour, nous ne comprîmes plus notre choix sur cet attribut. Nous ne pourrons pas vous dire non plus à quoi il était destiné au départ, car nous ne nous en souvenions plus. Mais on a dû pallier ce problème différemment …

La classe joueur :

Nous créâmes cette classe juste après la classe carte. Nous mîmes comme attribut un pseudonyme, un montant d'argent, et un tableau de carte. A un moment donné, pour connaitre le nombre de carte que le joueur possédé en main, au lieu d'avoir à parcourir tout le tableau avec un compteur, nous eûmes l'idée de rajouter un attribut de nombre de cartes restante. Ce que nous fîmes. Vers la fin de la programmation, nous créâmes l'attribut humain, afin de pouvoir différencier un joueur humain d'un joueur virtuel.

Pour les méthodes, dès le départ, certaines s'imposâmes d'elle-même : perte de l'argent, gain de l'argent, ainsi que les gets. Les sets sont peu nombreux, car nous n'en vîmes pas l'intérêt, car nous ne désirons pas modifier le joueur à part ses cartes en main, et le nombre de carte restante. Pour l'argent, pseudonyme et humain, une fois créé correctement, il ne nous est pas impératif de devoir les changer. Cependant, nous créâmes au fil du programme des méthodes nous permettant de modifier certains attributs comme le tableau de carte en main, ou l'argent ou bien encore le nombre de carte en main.

Comme dit précédemment, un moment important de mise à niveau de nos classes fût lors de la création de l'affichage. En effet, nous souhaitâmes afficher des cartes, mais pas n'importe lesquelles. En effet, nous voulûmes disposer, et modifier les cartes que le joueur possède dans son tableau de carte en main. Pour cela, il nous fallût créer des méthodes dans la classe carte, permettant de renvoyer les informations utiles, qui pouvait être appelé dans le tableau de carte en main du joueur, et donc, nous dûmes créer des méthodes dans joueur, qui appelait les méthodes de carte.

Lorsque nous terminâmes de créer cette classe joueur, nous voulûmes faire une fonction de distribution des cartes. Cependant, voyant que nous l'utiliserions que sur des joueurs, nous optâmes pour une méthode dans sa classe. Et la méthode distriCarte est née.

Mais pour cette méthode, il nous fallût créer un tableau répertoriant les 52 cartes différentes, et une fonction d'initialisation de toutes ces cartes. Nous commençâmes à coder cette méthode lorsque nous nous rendîmes compte qu'il ne fallait pas oublier les doublons. En effet, une carte donné à un joueur A ne pourra pas être donné à un joueur B. Pour cela, nous créâmes un tableau nommé repet, qui est un tableau d'entier dans lequel nous ne mettons que les indices des cartes que nous avons donné. Afin de ne pas avoir de problèmes, nous créâmes une fonction d'initialisation de ce tableau à 99 dans chaque case.

Une fois fini de coder, nous l'essayâmes à maintes reprises. Pour notre plus grand désarroi, et d'amusement, le compilateur donna quelque fois un message d'erreur nous disant : WHAT …

Après recherche, nous découvrîmes que nous avions mal écrit notre rand, qui prenait un chiffre entre 0 et 52, or le tableau des cartes n'allait que de 0 à 51… Ce pourquoi, de temps en temps, lorsque le rand nous sortait 52, le compilateur n'était pas content, ce qui est logique !

Une fois que nous réussîmes à distribuer les cartes, et commençâmes à les voir, nous nous rendîmes compte d'une autre chose : un affichage trié des cartes serait bien mieux que ce désordre. Nous créâmes donc une méthode tricarte, qui tri les carte en main d'un joueur.

La fonction prendre Mise :

Avant de commencer à créer l'articulation du jeu, nous voulûmes créer des fonctions règles. La première qui nous apparût à l'esprit est : prendre les mises.

Quelques fonctions d'allègement :

Une fois que nous eûmes créé tout cela, nous nous décidâmes de créer quelques fonctions qui nous permettrait d'alléger le main (cela tenait vraiment à cœur M. Cossin). Nous créâmes donc la fonction quantité joueur qui nous permettait de poser la question de la quantité de joueur, et de renvoyer dans le main un simple entier. En une ligne, c'était fait ! Cependant, cette fonction dû subir une mise à jour, lorsque nous avons entré en ligne de mire les joueurs virtuels. En effet, nous voulûmes à tout prix différencier les joueurs humains des joueurs virtuels. Une fois que nous avions le nombre de joueurs, nous crûmes bon de créer une fonction intitulée debutJeu (quelle imagination !) qui nous permet de créer les joueurs et de les caser dans le tableau de joueur.

En parlant de tableau de joueur, il faut avouer que ce sujet à était pour nous une certaine prise de tête. Au départ, nous ne pensions pas faire un tableau, mais lors de la création d'une fonction, dont on abordera le sujet plus loin, nous avions découvert qu'il était bien plus simple pour nous de créer un tableau de joueur et d'utiliser un indice que l'on incrémenterait tout le temps, avec un modulo nous permettant de faire le tour du tableau et de revenir à sa base.

Par la suite, nous décidâmes de créer un tableau dynamique. Mais cela, nous le verrons plus profondément dans les fonctions un peu plus complexe de l'articulation du jeu en lui-même

La(les) fonctions d'articulation du jeu :

A partir de là, lorsque nous regardâmes derrière nous et que nous vîmes ce beau petit chemin que nous avions parcouru, il nous semblâmes attingible la création d'une fonction d'articulation du jeu par les fonctions sus-créés. Nous nous attaquâmes donc à

la fonction jouer (ouille !). Ce fût la fonction qui nous prîmes le plus de temps. En effet, c'est la plus complexe, qui jongle avec le plus de fonctions, et qui prend des conditions tellement phénoménal que nous dûmes nous y reprendre par quatre fois avant de pouvoir la faire tourner correctement… Nous créâmes une grande boucle nous permettant de jongler avec les joueurs. C'est à partir de cette fonction que nous nous décidâmes des commandes principales de jeu :

* passe : pour passer son tour.

* prendre : pour prendre la mise d'une carte posable et éligible.

* indice de la carte à jouer.

En parallèle à cette fonction, nous eûmes la révélation pour créer la fonction de l'affichage. Et un en week end, elle fût conçue. Elle nous posa quelques problèmes au vu de l'affichage en terminal. Cela nous obligea à la création de l'affichage en trois bouts : le haut des cartes, le milieu des cartes, et le bas des cartes. Et c'est cette fonction d'affichage qui nous fit faire les mises à jour des valeurs graphique ainsi que des indices graphiques des cartes, mentionnés au début de notre dossier.

Et ce n'est qu'une fois terminé cette fonction d'affichage que nous pûmes terminer à son tour la fonction jouer.

Mais nous n'étions pas au bout de nos peines avec cette fonction, en effet, nous l'avions terminé alors que nous nous n'étions pas encore posé la question du joueur virtuel. Cela imposa donc une mise à jour de force de cette fonction. Nous inclûmes donc des conditions supplémentaires afin de gérer l'arrivé de ces robots. Une fois fini, nous vîmes une opportunité de réduire un peu cette fonction en créant de nouvelles fonctions.

Et la fonction gain carte vit le jour. Cette fonction assura le travail de donner au joueur l'argent d'une mise lorsque celui-ci pose une carte éligible à ce gain.

Le journal :

Une fois que nous en étions là où nous en étions décrits précédemment, nous nous sentîmes dans la nécessité de créer un journal où nous pourrions marquer toutes actions du jeu. Et là, le code commença à perdre de sa beauté, en effet, les ofstream se sont multipliés dans nos fonctions, afin d'y répertorier les actions entrepris par les joueurs.

Ce n'est qu'une fois fini cette étape que nous eûmes le besoin de gérer la répétition des manches de jeu. En effet, s'il reste au moins trois joueurs en jeu, nous devons recommencer à distribuer les cartes, et reprendre le jeu. Ce que nous fîmes donc à l'aide de boucle et d'une fonction articulant la fonction qui articulait déjà le jeu. Le nom de cette dite fonction : jeux. Cette belle (et pourtant si cruelle, à la conception) fonction, permet de gérer le tableau de joueur dynamique, en le renvoyant au main, où nous le remplaçons. Grâce à ceci, le jeu du nain jaune (si nous le souhaitons) devint quasi sans fin, car en plus de gérer les parties d'un jeu, à la fin, il permet également de recommencer un nouveau jeu.

Arrivé à ce stade, nous crûmes bon de créer un fichier de score… (et quel épouvantable samedi après-midi que nous passâmes, mais si bonne, pour être la dernière ligne droite !) Nous ne pûmes utiliser fstream, nous dûmes utiliser à la fois ifstream et ofstream, car les fonctions de lecture et écriture de fstream ne marchaient pas en notre faveur. Malgré ce désagrément, et grâce à notre motivation sans faille (les vacances !), nous réussîmes à contrecarré ce problème qui ne fût finalement que de courte durée.

Et pour finir, nous fîmes un makefile. Mais ce ne fut pas aussi facile que ce que nous pensâmes. En effet, une seul petite erreur, que nous eûmes du mal à identifier nous pris beaucoup de temps. Nous créâmes le makefile, mais nous executiâmes mal cette partie. Cependant, après maintes réflexions et recherches, nous découvrîmes qu'il fallait éxécuter le makefile avec la commande make -f... Ce que nous fimes par la suite, et qui clôtura ainsi ce jeu.

Bon courage pour la lecture du code du nain jaune.

Le groupe