Al haddaoui Marwane,
Kutlu Koray

53046-52828

Rapport de programmation d'Abalone

2020-2021

Table des matières

1. Introduction

Ce document a pour but de présenter ainsi que d'expliquer les divers choix du binôme 53046-52828, vis-à-vis de l'implémentation dans le langage c++ du jeux abalone, les règles, ainsi qu'une vidéo explicative du jeu sont joint en annotation de bas de page et nous allons vous guidez dans le parcours de notre projet au cours de ce document et tout vous expliqué.

Premièrement nous allons dire pourquoi l'observer/Observable est intéressant dans notre cas et pourquoi nous l'avons choisi.

Ensuite, nous allons expliquer quelque classe de notre projet qui parait compliqué au premier regard. Ce point permettrait de mieux comprendre la manière qu'on a codé et comprendre quelque classe (voir ce qu'ils font). Ce point permettrai aussi de voir les différentes classes que l'on a.

C'est pourquoi nous avons le second point qui est lié au précédent. Dans la même optique il permet de mettre en lumière des méthodes un peu compliqué lié au jeu et de les comprendre.

Par la suite, nous allons passé à un autre point. Puisque nous voyons à peu près à quoi ressemble le projet et que nous vous avons guidé sur les gros points, on peut désormais parlé des problèmes rencontré lors de l'implémentation des méthodes présenté dans la partie console de notre application.

De plus nous allons aussi parlé des changements amené au projet, car tout problème mène à une correction ou un changement. Donc nous allons voir ce qu'on a du vraiment changé ou ce qu'on a eu du mal a comprendre lors de l'implémentation du projet.

Et pour en finir avec la partie console, nous allons parlez de nos tests, voir ce qui passe et ce qui bloque. Parler des méthodes qu'on a pas testé et dire pourquoi on a pas pu le faire.

En dernier lieu, nous allons expliquer les problèmes rencontré dans la partie gui de notre projet et des derniers changements amenée au projet pour la finalisation de celle-ci.

Et pour finir ce rapport une conclusion avec un parcours général de notre rapport sera rédiger.

2. Choix du modèle suivis (design pattern)

Pour ce projet et après de multiple recherche le modèle le plus adéquat selon nous, est le design pattern Observer/Observable pour sa dynamique ainsi que sa réactivité pour la suite du projet, ce qui fait que nous l'avons choisi.

Nous comptons observer la class Game qui regroupe les méthodes du jeu. Ce qui permettrai dans la partie console d'afficher la vue à chaque changement amené dans le jeu, et de même pour la partie graphique.

3. Explication des différentes classes

Le projet Abalone ne comprend pas énormément de class , c'est un jeu simple qui possède une implémentation similaire à d'autre contenu ludique et donc ce jeu nécessite les classes de base d'un jeu. Tel que pour la création du plateau de jeu (class Board) qui est particulier par sa composition (plateau en 3 dimension, c'est un hexagone) ,pour la création des joueurs (class Player) ainsi que la class Piece (class Piece) qui créer une pièce avec les attribut du jeux (Une couleur par joueur ainsi que une couleur et une position par piece). De plus, la couleur ajoute également la couleur Empty dans son énumération car nous avons décidé pour la création du plateau de jeu d'avoir un plateau de pièces, composé de celles des joueurs (Noir et Blanche) mais également de pièce dite "vide" (empty) afin de gérer de manière plus facilement les pièces sur le plateau.

La class Direction est une énumération des différentes directions que peux effectuer une pièce sur le plateau et nous arrivons enfin à la class Game qui va se charger de l'implémentation du jeu complet avec toutes ses méthodes nécessaires à sa création ainsi que ses différentes règles pour l'interaction des joueurs sur le plateau de jeu lors d'une partie d'abalone.

D'ailleurs si vous voulez en savoir plus sur les règles du jeu voici une vidéo[1].

[1] Video, explicatif du jeux abalone ainsi que ses multiples regles

4. Explication des méthodes

Certaines méthodes méritent une explication supplémentaire, dans cette partie du rapport, nous tacherons d'expliquer de manière la plus claire et concise certaines méthodes qui ne paraissent pas assez explicites et intuitives.

Dans la class Board (qui représente pour rappel ,le plateau de jeu) , certaines méthodes lier au plateau de jeu y sont implémenter tel que "isFree" qui vérifie à l'aide d'une position, si une case du plateau de jeu n'a aucun joueur qui a une pièce a cette position. La méthode "isMyOwn" vérifie si une pièce appartient à l'un des deux joueurs, la méthode "isInside" vérifie si la pièce est bien dans le plateau de jeu , la méthode push quant à elle s'occupe de "placer" une pièce à une certaine position sur le plateau de jeux sans oublier la méthode "remove" qui fait le contraire de la méthode push, c'est à dire, elle "retire" une pièce du plateau de jeu.

Dans la classe Game, toutes les méthodes nécessaires au bon fonctionnement du jeu y sont présentes tel que la méthode initialize qui va initialiser le jeu (son plateau de jeu, ses joueurs, etc..) , elle comprend également les méthodes spécifiques aux règles du jeu tel que canPlay qui s'occupe de savoir si un joueur peut jouer en adéquation avec les règles du jeu, la méthode nbrOfCurrentPiece compte le nombre de pièce d'un joueur courant sur le plateau afin de savoir s'il peut toujours continuer à jouer.

Pour finir la méthode play va s'occuper de l'interaction du jeu avec le joueur, c'est-à-dire qu'il va s'occuper de faire un déplacement seulement si le déplacement est correct et concorde avec les règles du jeu. Si c'est le cas il appellera toute les méthodes concernés pour faire ce mouvement.

5. Problème rencontré dans la partie console

Nous avons eu un problème dans notre game qui n'était pas lié au codage mais à la compréhension. Même si c'était pas un très gros souci, ça nous a pris du temps de comprendre toute les règles et les mouvements possibles dans les différentes directions. Donc on a du recherché ou essayé de jouer au jeu sur des sites en ligne. On a aussi regardé des vidéos fait par des personnes passionnés par ce jeu.

Le second problème rencontré était dans la classe board car nous avons changé le tableau 3d en un tableau 2d, et nous avons optimisé le tableau 2d de sorte à ce qu'il n'y a pas de case vide et que toute les cases qui sont créés soi utilisées. Et le problème crée par cela était le fait que les positions de nos pièces n'avaient pas les mêmes positions dans notre tableau. Donc une réflexion a dû être effectuée afin d'accédé au case sans générer une quelconque erreur.

Ce qui nous mène au dernier souci qui est le tableau 3d qu'on a changé en 2d. Car même si le tableau en 2d était en pratique plus simple a utilisé et a manipulé, le fait qu'on a optimisé notre tableau de sorte à ce qu'il n'y a aucun vide, a créé un souci qui est celui mentionné ci-dessus. Mais on a surtout préféré le tableau 2d car le tableau 3d était beaucoup plus compliqué à comprendre et a utilisé. Avec tout le temps qu'on a passé a essayé de comprendre le tableau a 3d, nous aurions pu finir le projet bien plutôt. Même si cela nous a éclairé un peu plus au sujet des tableaux 3d cela nous a montré qu'elle était bien plus compliqué qu'un tableau 3d. Nous pensons qu'il serai plus intéressant d'utilisé un tableau 3d dans un plus gros projet que celui-ci, dans ce cas le tableau 3d se montrera plus efficace. Et donc au final nous avons changé notre tableau en un tableau 2d optimisé de sorte à ce qu'il y a aucun vide.

6. Les changements dans la partie console

L'un des plus gros changements est celui du passage d'un tableau 3d en 2d, donc de ce fait notre classe position a changé et notre classe board aussi. Notre classe position a maintenant que 2 attributs x et y. Et notre classe board possède maintenant un tableau a 2 dimension, ce qui fait que le parcours du tableau et son initialisation a changé.

7. Les tests

Les tests ont été fait pour toute les classes sauf quelque test qui n'ont pas pu être réalisé. Dans la classe Player et Board les tests ont été fait. Et pour ce qui est de la classe game aussi. Donc voici les méthodes qui n'ont pas été testé.
Le initialize de board n'a pas été testé car trop compliqué à être testé (si on veut le faire bien). De même pour l'initialize de game qui n'a pas été testé car trop compliqué et long à le faire (de même que initialize de game). Donc on a préféré ne pas le testé mais c'est pas pour autant qu'il a été délaissé car au cours de l'évolution de notre projet lorsqu'on on jouait à notre jeu (pour tester, débugger,etc..) on faisait attention au fait que, est ce que l'initialize a bien fait son travail et est-ce que cette position existe vraiment. Et donc nous l'avons testé indirectement.
Ce qui n'a pas été vraiment testé par contre c'est sumito car dans notre game nous pouvons pas faire un plateau comme qui répond au critère et donc pas testé ce cas. Même si on voit qu'il fonctionne a priori bien car lorsqu'on joue on voit quand ces méthodes fonctionne ou pas.

8. Problème rencontré dans la partie gui

Le seul problème rencontré au cours de la remise partie gui, était de pouvoir manipulé la partie graphique de c++. Car on avait aucune base et qu'il fallait tout recherché et trouvé nous-mêmes. Donc on a du faire beaucoup de recherche pour pouvoir manipulé les objets, mettre à jour, les positionné comme on le souhaite, les faire réagir comme on le souhaite, comprendre les méthodes appelé indirectement, etc… Même si au final nous avons compris comme cela fonctionne, ça nous a pris beaucoup de temps de comprendre le tout et le manipulé. Et à part cela nous avons riens a signalé de problématique au cours de cette remise.

9. Changements effectué dans la partie gui

Le premier changement effectué est le rajout de 3 nouveaux attributs dans notre classe game qui permet le bon fonctionnement du jeu pour la partie gui. Nous avons rajouté 2 positions qui se nomme begin_ et end_. Ils sont utilisé lorsque un joueur sélectionne 2 pièces pour faire son mouvement. Begin_ stock la première position sélectionné alors que end_ possède la position d'arrivée.

Le troisième attribut est selectedCase_ qui compte le nombre de pièce choisi par le joueur lors du clique. Et lorsqu'il atteint 2, une tentative de play est effectué et si le mouvement est possible il est joué.
Et le dernier changement est l'ajout de la méthode posIndiceToPiece dans la classe position. Il permet le fonctionnement du tableau pour la partie gui. Elle permet de traduire la position sélectionné par le joueur en une position réel de notre tableau.

10. Conclusion

Après vous avoir expliqué tout notre projet pour certain point en détail et d'autre où on l'a juste survolé, nous voyons a peu près ce à quoi ressemble notre projet. Ce qui va permettre de se balader au sein de notre projet sans être pour autant perdu. Ce document ne permet pas d'expliqué tout le projet dans chaque détail mais plutôt d'expliqué certains points plus compliqués à comprendre. Ou encore expliqué la logique de notre jeu et voir de ce qu'il est constitué.
Grâce à ce petit document, vous avez les bases de notre projet, les implémentations choisis et la définition de certaines classes. Les problèmes rencontré et les changements effectué dans la partie console et gui de notre application sont décrites et expliquées. Les tests sont fait pour pouvoir testé le projet et les différentes méthodes.