DEV4

Projet Abalon

54024 - Arno Pierre Pion
54456 - Damiano Deplano

2020 - 2021

Table des matières

Introduction

Nous allons créer une version électronique du jeu Abalone en trois étapes. Premièrement nous définirons les classes métier servant à stocker toute les informations nécessaires à une partie, deuxièmement la création d'une interface semi-graphique jouable dans une console et troisièmement une version graphique.

1. UML

Un diagramme de classe UML sur lequel le projet en C++ est basé est disponible en plus de ce rapport.

AbaloneClassDiagram.svg

2. Les classes métiers

Pour ce projet nous devrons passer d'une version avec une vue console du jeu a une version graphique, nous avons donc décidé de travailler en MVC. Avec le MVC il est plus facile de rendre interchangeable l'UI.

La façade Game

Problème avec la v1: Il manquait une méthode pour gérer les tours

Elle permet l'accès aux méthodes publiques de la classe board. Elle s'occupe également du tour, pour bouger les marbles.

La classe Board

Problème avec la v1:

- *Il manquait une grille de départ pour initialiser le board, nous avons donc fait en sorte de créer le tableau dynamiquement.*
- *Nous avons dû ajouter deux attributs et une méthode pour la gestion des joueurs par tour.*

Elle sert à initialiser le jeu et contient le tableau nécessaire aux placements des Marbles sur la grille. Elle permet aussi de déplacer les marbles ainsi que de vérifier quel joueur sera gagnant en fonction des scores obtenus.

- Avant chaque déplacement on vérifie:
    - Pour les déplacements en ligne, qu'il y a un maximum de 3 billes du joueur et 2 billes du joueur adverse.
    - Pour un déplacement en flèche et qu'il n'y ai pas de bille adverse dans le chemin.
    - Qu'il n'y ait pas de marbles ennemis entre les nôtres pour un déplacement en groupe.
    - Qu'il y ait effectivement un marble a cette coordonnée.
    - Que la coordonnée donnée soit valide et dans le tableau.

- Après chaque déplacement, nous notifions la vue avec une copie défensive de la grille.
- A chaque changement de score, nous notifions la vue, pour qu'elle puisse afficher l'information à l'utilisateur.
- A chaque changement de tour l'information sur le joueur pouvant jouer est envoyée à la vue.
- La vue est notifiée en cas de mouvement interdit ou de coordonnée illogique.

La classe Board était la plus complexe à réaliser, il y a beaucoup de manière de bouger et donc beaucoup de manières de faire crash le programme. On a donc dû mettre beaucoup de choses pour vérifier que les coordonnées données soient correctes et possibles.

La classe Score

Elle permet de stocker les scores des joueurs et de récupérer le joueur avec le plus haut score.

- On vérifie après chaque sortie de Marbles de la grille s' il n'a pas atteint le score de 6 ce qui met fin à la partie.

La classe Marbles

Problème avec la v1:

- *La gestion des couleurs qui était dédiée à la console se trouvait dans cette classe, nous avons donc remplacé la couleur par le numéro de joueur et déplacer la gestion de la couleur dans la vue.*
- *Nous avons enlevé son attribut "isFallen" car il n'était plus utilisée au final*

Elle contient le numéro du joueur propriétaire.

- Cela permet d'identifier les marbles qui peuvent être joués et à la vue de lui attribuer une couleur.

La classe Direction

Problème avec la v1: On a dû inverser les coordonnées dans deltas correspondant au direction pour que les calculs soient cohérents avec notre manière d'afficher la grille.

Elle détermine dans quel sens un Marbles ou un groupe de Marbles doit se déplacer dans la grille.

- une énumération pour faciliter le passage en paramètre.
- une méthode pour transformer l'entrée utilisateur en Direction.
- les deltas en X et Y nécessaires au déplacement du marbel sur la grille.

La classe Position

Nous avons renommé cette classe qui a la base s'appelait 'Point' pour que ce soit plus compréhensible. Elle contient les coordonnées X et Y nécessaires au déplacement sur la grille.

3. La vue console

La class Main (Projet console)

Elle permet de lancer le contrôleur de la partie console.

La class MainController

Problème avec la v1: nous avions oublié d'ajouter une méthode start pour démarrer le jeu.

Elle est le contrôleur qui fait le lien entre la vue console et les modèles, grâce au paterne Observateur-Observé. C'est elle qui lance la partie.

La classe TextView

Elle permet de gérer tout l'affichage de la vue console, la vue observe les classes games et boards et s'adapte quand celles-ci changent.

- Permet d'afficher le tableau envoyé par le modèle.
- Gérer les interactions clavier avec les joueurs.
- Afficher l'en-tête des scores.
- Afficher les messages d'erreur.
- Afficher le gagnant.

Après chaque demande de mouvement de l'utilisateur, le modèle est notifié et traite le mouvement ou renvoie un message d'erreur.

La classe Color

Problème avec la v1: La couleur était gérée au niveau du modèle et cela rendait complexe le passage des informations à la vue, de plus cette gestion était dédiée à la console et n'était pas réutilisable pour la GUI. Nous avons donc déplacé cette classe vers la vue console.

Elle contient la couleur de chaque joueur et sera utilisée pour différencier les marbles dans la vue console.

- La liste des couleurs permet de récupérer le code couleur ANSI nécessaire à l'affichage en mode console.
- Une méthode permet de récupérer un caractère correspondant à l'affiche des marbles de chaque joueur.

4. La vue graphique

La class Main

Elle permet de lancer le contrôleur de la partie graphique, ainsi que d'initialiser l'application graphique Qt.

La class MainController

Elle est le contrôleur qui fait le lien entre la vue graphique et les modèles, grâce au paterne Observateur-Observé. C'est elle qui lance la partie.

La class Hexacell

Elle a été fournie par les enseignants et permet de gérer le rendu graphique des hexagone.

Nous lui avons ajouté un attribut pour identifier à quel joueur appartient le Marbel.

La class Point

Elle a été fournie par les enseignants, elle est un Alias de la classe Pair et elle permet aussi de gérer et calculer le positionnement des Hexacell dans la MainWindow.

La class MainWindow

Elle est la vue qui permet d'afficher une fenêtre graphique avec la librairie Qt.

- Permet d'afficher la grille d'Hexacell.
- D'indiquer le tour du joueur.
- Afficher les scores.
- Les messages pour informer le joueur en cas d'erreur dans la notation ABAPRO et le gagnant.

Les déplacements se font soit via la notation ABAPRO dans le champ texte en bas de la fenêtre soit par des cliques sur les cases à déplacer pour remplir automatiquement les coordonnées ABAPRA.