

Projet de programmation orientée objet et d'interfaces graphiques

Julien COOLEN

Yaniv BENICHOU

Année 2018 - 3^e semestre L2

1 Introduction

Nous abordons dans ce rapport nos choix de modélisation et les difficultés rencontrées au cours de ce projet pour le cours de programmation orientée objet et d'interfaces graphiques.

Le but de ce projet consiste à implémenter plusieurs jeux de plateau en regroupant les fonctionnalités communes. Pour cela nous avons développé une plateforme logicielle¹ à partir de laquelle nous avons codé chaque jeu : un ensemble d'objets qui interagissent entre eux et permettent de développer des jeux de plateau dans la plus grande généralité. Ce framework devait être facilement étendu pour implémenter un nouveau jeu.

Aussitôt le projet entamé, il nous a paru évident de mettre en place un gitlab afin de collaborer au mieux à deux, en ayant un accès aux dernières versions de codes mises à jour par l'un d'entre nous, mais aussi avoir une sauvegarde des états du projet au fur et à mesure de notre avancée. Nous avons codé tous les jeux demandés : dominos que l'on assemble pour former une chaîne linéaire, dominos gomettes que l'on assemble pour former une chaîne avec des branches(dans le plan), un puzzle ainsi que le jeu du saboteur qui est un véritable jeu de société demandant plus de stratégie de la part des joueurs.

2 Modélisation

Nous avons tout d'abord remarqué que chaque jeu s'articule toujours autour des mêmes objets : un plateau, des pièces, des cartes, une pioche, des joueurs et leur main. Ce sont les composants de base de notre framework. Chacun de ces objets est éventuellement étendu pour ajouter ou redéfinir des comportements.

1. En anglais *framework*.

Chaque jeu adopte la structure imposée par le framework. Nous avons essayé d'exploiter au maximum les concepts étudiés ainsi que des événements pour interagir avec la vue.

2.1 Présentation du framework

2.2 Dominos simples et avec gommettes

Nous avons commencé par implémenter le jeu des dominos simples à l'aide d'une ArrayList, les pièces étaient ajoutées aux extrémités de la liste. Une méthode vérifiait si une pièce pouvait se poser à gauche ou à droite d'une autre pièce. Lorsque nous avons implémenté le placement des pièces pour le jeu du saboteur, nous avons fait évoluer notre modèle en introduisant les types `Tile<S extends Side>` pour les pièces que l'on pose sur un plateau et `Side` pour les côtés des pièces, ce dernier type servant à regrouper les variables qui définissent un côté. Plusieurs évolutions successives de notre framework ont finalement cassé le code qui devait être repris.

Nous avons en revanche constaté que le jeu des dominos simples peut être vu comme un sous-ensemble des dominos gommettes, dans le sens où les dominos simples s'assemblent pour former une seule chaîne contrairement aux dominos gommettes avec lesquels on peut en plus former des branches.

C'est pourquoi nous avons implémenté le jeu des dominos simples et dominos gommettes en étendant une classe plus générale qui permet de placer côte à côte des pièces rectangulaires sur un plateau si leurs côtés se correspondent.

2.3 Saboteur et puzzle

Après avoir codé le jeu des dominos simples nous avons réalisé qu'il serait plus intéressant d'implémenter le jeu du saboteur car son implémentation représentait l'implémentation demandant le plus de réflexion et donc de temps de part sa complexité de jeu (vérifier quatre côtés d'une carte ainsi que la connexion de ces chemins, plusieurs types de cartes permettant de bloquer/débloquer un autre joueur). Une fois écrit nous avons pu réutiliser beaucoup de code. C'est au cours du développement de ce jeu que nous avons développé une base de code commune réutilisable et facilement extensible pour chaque jeu. Les autres jeux étant toujours des jeux de plateaux, mais en une version simplifiée.

2.4 Pattern MVC

L'architecture modèle-vue-contrôleur a été implémentée pour les dominos simples et le jeu du saboteur. C'est cette architecture qui nous a paru la plus judicieuse et la plus efficace, comme vue en cours puisqu'il permet une implémentation plus simple, séparée et épurée. Le code est d'autant plus compréhensible et lisible.

3 Difficultés rencontrées

Lors du développement nous avons réécrit à plusieurs reprises certaines méthodes ainsi que certains aspects de notre modélisation, ce qui introduisait quelques fois de nouveaux bugs. C'est pourquoi nous avons écrit des tests unitaires à l'aide de la librairie jUnit que l'on lance à chaque modification pour identifier toute régression dans le code.

L'implémentation du jeu du saboteur nous a demandé un certain temps avant de parvenir au résultat exigé. L'utilisation d'Enums pour les cartes Path ainsi que l'utilisation de classes abstraites, d'interfaces et d'héritages, nous a finalement permis de surmonter ces quelques (mais légers) obstacles.

Cependant, nous sommes d'accord pour admettre que la tâche qui nous a paru la plus compliquée a été de mettre en place une interface graphique. En effet, malgré que nous soyons habitués à utiliser les concepts d'héritage, d'interfaces, de classes abstraites, de programmation orientée objet, d'exceptions ou encore de généricité, nous n'avons que peu de connaissances en Interfaces graphiques. Finalement, la documentation Java et de nombreuses heures de tests successifs et de discussions dans des Forums, nous ont permis de vous présenter cette interface-ci, qui peut être largement améliorée.

Malgré tout, il est évident que ces difficultés rencontrées n'ont été que bénéfiques pour nous, nous ont permis d'en apprendre davantage sur l'utilisation de l'intégralité des concepts étudiés ce semestre en cours de Programmation Orientée Objet (et bien plus encore), et surtout sur l'implémentation d'une interface Graphique.

4 Pistes d'amélioration

Pour ce qui est de l'interface graphique, nous avons essayé de permettre aux joueurs d'effectuer n'importe quelle action, en les laissant dans un premier temps, cliquer sur la carte de leur main à jouer, puis dans un second, à cliquer sur la case ou joueur où effectuer l'action, ces derniers remarqués par un changement de bordures (rouge). Tout ceci évidemment géré par des ActionListeners. Malheureusement, après plusieurs essais et tirages de cheveux pour bien limiter à 2 clics (le premier dans la main et le second ailleurs), ainsi que de récupérer la carte au clic associé, nous avons préféré une solution "plus simple" qui est de récupérer les valeurs à partir de pop-up, lancées par des fonctions display(), grâce à l'utilisation de JOptionPane. Il est évident que cet échec est une grande frustration pour nous, et nous savons qu'avec un peu plus de temps, nous aurions trouvées le bon moyen d'y parvenir. Nous comptons chercher ce moyen après la date de rendu de ce projet pour notre apprentissage personnel.

5 Pistes supplémentaires

Nous aurions apprécié implémenter bien d'autres options supplémentaires comme par exemple , permettre à l'utilisateur d'effectuer des sauvegardes de parties puis de les charger, ou encore mettre en place des petites animations dans un background (gifs) mais faute de temps, nous nous sommes vus obligés de nous concentrer principalement, sur ce qui a été exigé.

Nous avons pu tout de même ajouter une piste audio (en boucle) lors de l'ouverture de l'interface graphique.

6 Conclusion

Nous avons développé le projet itérativement en travaillant à plusieurs niveaux d'abstraction : sur chaque jeu et le framework. Le développement du framework était guidé par les besoins de chaque jeu et réciproquement le développement d'un jeu dépendait directement de l'organisation du framework. Ainsi, si une extension du code était compliquée à écrire, nous revoyions le modèle. Cet aspect du projet a été de loin le plus intéressant et enrichissant puisqu'il nous a permis remettre en question nos choix, de les rectifier et persévérer.

Il était de plus très plaisant de remarquer que beaucoup de concepts se retrouvaient dans les jeux à implémenter, ce qui nous a conduit à introduire de nouveaux types qui ont grandement facilité le développement des jeux.

Nous avons également apprécié la liberté qui nous a été donnée au cours de ce projet quant aux choix de l'implémentation, les consignes laissant parfois beaucoup de choix.

Nous avons pris ce projet très à cœur et y avons consacré de nombreux jours et malgré que nous savons qu'il est critiquable et imparfait en certains points, nous sommes fier de vous présenter le fruit de notre travail.