Année : 2016-2017

RAPPORT PROJET POOIG 2016 : JEUX D'ALIGNEMENT

Réalisé par : BAMBA Ibrahim
ROCHA Da Silva Matheus
Groupe : Info 1

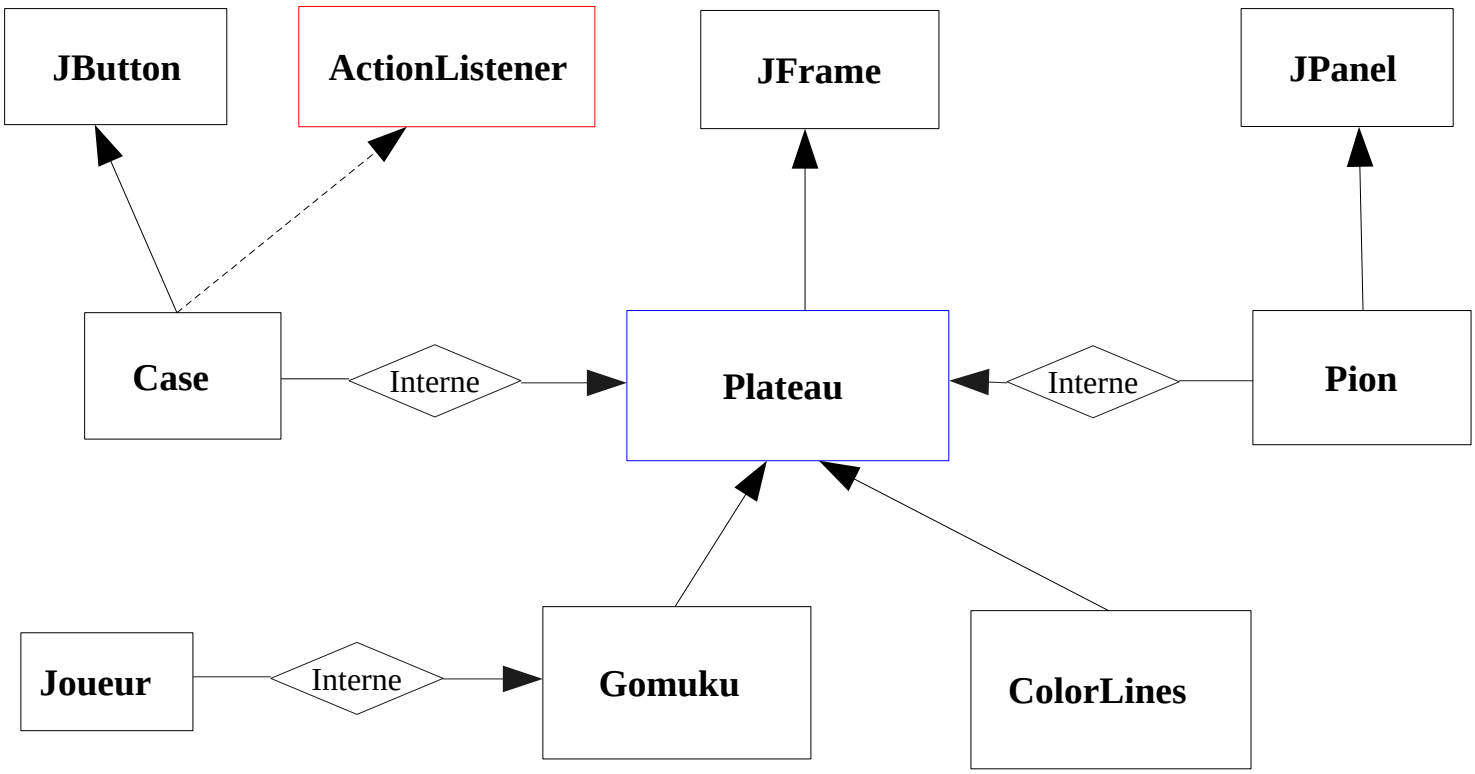
INTRODUCTION

Pour ce projet, nous devons implémenter deux jeux d'alignements de sociétés assez connus : Gomuku et Color Lines. Nous avons été deux à travailler sur projet. Ces deux jeux ayant beaucoup de similarités, l'objectif était de mettre en application tous les notions et chapitres vus en cours et/ou en TDs tout au long de ce chapitre. Certes, cela n'a pas été facile pour nous mais nous y sommes arrivés. Dans la suite de ce rapport qui a pour but de vous clarifier notre travail, nous étaleront en premier lieu notre projet en intégralité c'est-à-dire nous expliqueront notre travail et nos différents choix. En second lieu, nous aborderons les difficultés que nous avons rencontrés dans la réalisation de notre travail et comment sommes nous pris pour les surmonter.

I. EXPLICATION DU PROJET

Dans ce projet, nous avons implémenter deux jeux ayant des similarités. Le but n'était donc pas d'implémenter deux jeux distincts mais deux jeux qui auront des entités en commun telles que l'extension, l'implémentation... Tous ces jeux sont programmés avec un interface graphique en swing et dans ces deux jeux, la taille du plateau est paramétrable avant le début du jeu.

1. Diagramme de classes



Légende : ----▶ : Implements (Interface)
——▶ : Extends (Sous-classe)
——▷ : Est interne (Classe interne)

▭ : Interface
▭ : Classe normale
▭ : Classe abstraite

- ActionListener, JFrame, Jpanel : Classes et interface interne prédéfinies dans le package JAVA, librairie AWT.

- Classe Plateau : Classe abstraite, étend JFrame. C'est la classe des jeux dans la globalité. Cette classe initialise le plateau d'une partie de jeu. Elle contient aussi les classes internes Case et Pion. Un plateau est composé de plusieurs cases, chaque case pouvant contenir un pion. Cette classe implémente la méthode jeuFini qui vérifie si la partie est terminée. Elle déclare aussi les méthodes robotStupid (qui joue les coups de l'ordinateur), miseAJour(qui apporte les modifications nécessaires au plateau après chaque coup), alignement (vérifie si un alignement de 5 pions ou plus a été réalisé et agit en conséquence) et jeuGagne (met fin au jeu en affichant un message si le jeu est fini) qui devront être toutes redéfinies par chaque classe qui hérite de plateau.

- Classe Case : Cette classe est interne à Plateau et est un JButton. Le plateau est constitué de cases cliquables (implémente ActionListener pour entendre les actions) pouvant être occupées par un pion ou vides. La classe Case définit les méthodes estVide(teste si une case est vide), removePion(supprime un pion de la case), fill(qui place un pion sur la case), et les méthodes d'alignements (vertical, horizontal, diagonal) qui renvoient le nombre de cases successives contenant des pions de même couleur que celle de la case à l'horizontal, la verticale et à la diagonale.

- Classe Pion : Elle est aussi interne à la classe Plateau et étend JPanel. Un pion peut être mis sur une case du plateau et selon le jeu, pourra être déplacé ou supprimé du plateau.

- ColorLines : Cette classe implémente le jeu ColorLines en lui-même. Elle dérive de la classe Plateau donc est une JFrame. Elle redéfinit les méthodes abstraites déclarées dans Plateau et définit également la méthode deplacementValide (teste si un déplacement proposé par le joueur est valide) qui lui est particulière.

- Gomuku : Comme la classe ColorLines, elle dérive de Plateau et redéfinit également les méthodes abstraites déclarées dans cette dernière. Elle définit aussi sa propre méthode changeJoueur qui change le joueur d'une

partie. Cette classe contient la classe interne Joueur. Un joueur est caractérisé par une couleur et son score.

2. Gomuku

Le gomuku est un jeu de plateau chinois où il est nommé "Wŭzi qí". Ce jeu se joue à deux sur un plateau avec des pions. Un joueur possède des pions d'une couleur et l'autre d'une autre (dans notre cas, les deux couleurs utilisées sont le bleu et le rouge). A chaque tour, un joueur pose un seul pion sur une case vide le but étant d'avoir 5 alignements de pions de sa couleur. Le gagnant est celui qui a réussi à faire le plus d'alignement de 5 pions de sa couleur. L'alignement se fait horizontalement, verticalement ou en diagonal. Pour ce jeu, nous avons implémenté un joueur "robot stupide" qui, après le tour du joueur, choisi au hasard une case vide sur le plateau et y place le pion. Ce n'est donc pas une intelligence artificielle pour qu'il ait un vrai duel. Donc, dans notre version de Gomuku, le jeu ne se joue pas à deux, on y joue à un contre un robot stupide.

Lorsque le jeu commence, le plateau est intégralement vide. Le joueur est le premier à placer son pion. Il possède les pions de couleurs bleus et le robot stupide ceux de couleurs rouges. Lorsque un joueur a réussi à aligner 5 pions ou plus, son score augmente du nombre de pion aligné puis ces pions sont rendu invalides, on ne peut plus les utiliser pour d'autres alignements (c.f Partie II, Difficultés rencontrées et solution trouvée). Aussi, il existe des bonus lorsque un joueur réussi à réaliser des alignements simultanés (ex : un alignement vertical et un alignement horizontal). Dans ce cas, le nombre de points qu'il gagnerait en réalisant ces alignements séparément est multiplié par le nombre d'alignements simultanés réalisé. *Explication : Dans notre exemple, le joueur a réalisé un alignement horizontal de 6 pions et un alignement vertical de 5 pions en même temps. Avant cet alignement, il avait 10 points. Dans ce cas, séparément, il aurait eu d'abord 6 points puis 5 points en plus. Mais grâce au bonus, puisqu'il a réalisé deux alignements simultanés, il gagne donc (6+5 = 11) * 2 = 22 points supplémentaires donc il aura après ces alignements 10+22 = 32 points.* Le jeu se termine lorsque tous les cases sont occupés. Dans ce cas, aucun joueur ne peut plus jouer. Le vainqueur est celui qui a le plus haut score. A la fin du jeu, le plateau disparaît et on affiche un message de victoire ou de défaite du joueur avec les différents points dans la fenêtre.

3. Color Lines

Color Lines est un jeu de puzzle d'ordinateur, inventé par Oleg Demin et introduite comme un jeu vidéo par la compagnie russe Gamos (en russe: Геймос) en 1992. On y joue à un contre l'ordinateur sur un plateau. Le but de l'ordinateur est de

remplir le plateau pendant que celui du joueur est d'empêcher cela en alignant au moins 5 pions de même couleur sur le plateau et ainsi les supprimer. Il doit réaliser le plus d'alignement de pions sur le plateau. Les pions utilisés sont de différentes couleurs. Le nombre de couleur est choisi par le joueur avant le début du jeu. En plus des couleurs normaux, nous avons une couleur spéciale; c'est la couleur d'un type de pion qui est lui aussi spécial : le pion arc-en-ciel. C'est un petit carré de couleur jaune. Ce pion se combine avec toutes les couleurs. Il peut rentrer dans l'alignement de pion vert et en même temps celui de couleur rouge par exemple. Étant donné que nous n'avons pas pu générer une couleur arc-en-ciel, nous avons décidé de considérer la couleur jaune comme la couleur spéciale (c.f : Partie Difficultés rencontrées et résolution) ainsi, les pions normaux ne peuvent avoir cette couleur.

Lorsque le jeu commence, l'ordinateur place 6 pions de couleurs sélectionnées aléatoirement sur le plateau au hasard. Après cela, le joueur peut déplacer un pion. Dans ce jeu, nous avons modélisé une variante : *"Les cases du plateau sont colorées en damier noir et blanc. Les pions ne peuvent maintenant plus être déplacés que vers une case de même couleur. Si par le plus grand des hasards toutes les cases noires sont occupées et aucune des blanches ne l'est (ou vice-versa), le joueur ne plus déplacer de pion: le jeu est alors fini".* Le joueur ne peut donc déplacer un pion vers une case d'arrivée que si cette case est de la même couleur que celle de départ. Après avoir déplacé un pion, l'ordinateur place à nouveau trois pions avec le même système et ainsi de suite. Lorsque le joueur réussi à aligner 5 pions ou plus, ils sont supprimés du plateau. Le système d'attribution de points est pareil à celui de Gomuku énoncé plus haut. Le jeu termine lorsque le plateau est rempli. A ce moment, on affiche le score final sur la fenêtre.

II. DIFFICULTÉS RENCONTRÉES ET RÉSOLUTION

Comme signifié tout au long de ce rapport, la conception de ce projet n'a pas été sans coup férir. Non, bien au contraire. Nous avons rencontrés plusieurs obstacles qui ont été difficiles à remonter et certains même dont nous n'avons pas trouvé de solution et donc avons trouvé une autre alternative. Dans cette partie, nous exposons les principales difficultés rencontrées et les solutions apportées.

Tout d'abord, dans le jeu Gomuku, il a été demandé que les pions qui ont servi à un alignement, peuvent resservir à un autre. Mais problème; si un joueur réussi à faire un alignement de 5 pions horizontalement lors d'un tour, puisque les pions déjà comptabilisés ne sont pas éliminés du plateau, lors du prochain tour, il peut placer un autre pion à la suite des 5 précédents et gagner à nouveau des pions alors que cela n'est plus un alignement dans la normal. Pour éviter cela, on a décidé que les pions qui ont servi à un alignement ne pourront plus resservir à un autre. C'est pour cela chaque case à un attribut "boolean invalide" qui dit si la case a déjà été comptabilisée ou non. Si c'est le cas, on ne l'utilise plus pour un alignement.

A côté de cela, dans le jeu Color Lines, pour les pions arc-en-ciel, nous avions décidé de choisir une couleur spéciale du genre arc-en-ciel. Mais après plusieurs recherches, il a été fort dommage de constater que nous ne pouvions pas générer de couleur arc-en-ciel. Pour palier ce problème, nous avons arbitrairement choisi la couleur jaune comme la couleur spéciale, la couleur des pions arc-en-ciel. Ainsi, les autres pions normaux n'ont plus le droit d'avoir la couleur jaune. Pour se faire, nous avons veillé à ce que, lorsque nous choisissons les couleurs au hasard, elles ne soient pas bleues. Nous vérifions d'abord que la couleur choisie ne soit pas bleu avant de la valider.

Pour finir, hormis ces deux principales difficultés, voici ci-après nos deux témoignages sur le projet, nos impressions et nos difficultés en générales :

<<Dans la globalité, je n'ai pas trouvé le projet assez compliqué. J'avoue que dans les débuts, j'avais un peu peur car je voyais mal comment on pourrait finir le projet à temps. Je trouvais le temps très serré. Mais on a fini par réussir. Le plus difficile était de commencer. Cependant, j'ai rencontré pas mal de difficultés. D'abord, je suis arrivé à Paris Diderot cette année donc je n'ai pas fait de JAVA l'année dernière. Du coup j'avais du retard sur les autres et sur les cours. Mais avec le temps, je me suis rattrapé. Aussi, on a mis du temps à former notre groupe mon binôme et moi. Ce n'est seulement que le 1 décembre que nous nous sommes associés et avons commencé à travailler ensemble ce qui nous laissait donc très peu de temps. A côté de cela, dans les débuts, je voyais mal comment on allais réussir ce projet puisqu'il nous a été laissé aucun guide. On devait faire nos propres classes nous-même. Cela a été particulièrement pénible pour nous de savoir choisir les classes et les méthodes à implémenter surtout à trouver des héritages et/ou des communs entre les deux jeux. Pour finir, le plus pénible dans ce travail a été le fait qu'on devait faire sans cesse des recherches sur internet pour trouver des explications sur certaines méthodes de la librairie AWT.>> **BAMBA Ibrahim**

<<Ce travail a nécessité beaucoup de temps, bien que je n'aie eu que quelques difficultés. Ma plus grande difficultés a été l'utilisation de la plateforme swing, que je n'avais jamais utilisé auparavant. Ainsi, j'ai mis beaucoup de temps à faire des recherches et à apprendre les méthodes mais j'ai surtout eu des difficultés à mettre en application et exécuter ce que j'avais prévu de faire. J'ai aussi eu des difficultés à faire en sorte que les méthodes reçoivent les actions des utilisateurs et puissent les exécuter. Faire la méthode robotStupide de Color Lines a été un grand défis. La plus grande difficulté a été de faire en sorte que la méthode robotStupide mette trois pièces seulement après le déplacement d'une pièce. Sur ce même jeu, cela a été dur d'appliquer la méthode d'effacer les pièces et compter un point lorsque le joueur complète une ligne de cinq pièces. J'ai aussi passé beaucoup de temps a travailler sur le temps de réponse de l'ordinateur suite a la réalisation de l'action du jouer, qui était trop long.>> **ROCHA Da Silva**

CONCLUSION

Somme toute, ce projet que nous avons réussi à traiter en groupe, n'a pas été facile pour nous. Face aux difficultés rencontrées, seules le courage, les connaissances acquises aux cours et aux TDs et certaines recherches internet nous ont permis de continuer et de l'achever. Ce projet avait pour but de mettre en application nos acquis du semestre mais nous en avons tiré beaucoup plus. Grâce à ce projet, nous avons appris beaucoup de choses sur la programmation objet et désormais, nous aimons encore plus programmer.