Loann BARRAUD ISEP 2016

Projet d’algorithmique et programmation

Document de conception

# 1. Rappel du Sujet

Description du jeu - rappel du sujet

L’objectif était de réaliser le jeu des six couleurs en Java, comme présenté ci-dessous :

« Le jeu des six couleurs est un jeu de stratégie se déroulant sur un plateau découpé en cases de 6 couleurs différentes (rouge, orange, jaune, vert, bleu, ou violet). Le but du jeu est de contrôler plus de cases que les adversaires à la fin de la partie. Les joueurs (de 2 à 4) commencent la partie en contrôlant chacun une case de la grille, ces cases doivent être de couleurs différentes. Les joueurs jouent chacun leur tour. À son tour, un joueur choisit une couleur différente de celle qu'il a actuellement, et non utilisée par ses adversaires.

- Toutes les cases contrôlées par le joueur deviennent alors de la couleur choisie.

- Toutes les cases de la couleur choisie et qui touchent une case contrôlée par le joueur passent sous son contrôle.

La partie s'arrête lorsqu'il n'y a plus de case non contrôlée, ou lorsqu'un joueur contrôle plus de la moitié des cases. Le joueur gagnant est celui qui contrôle le plus de cases. »

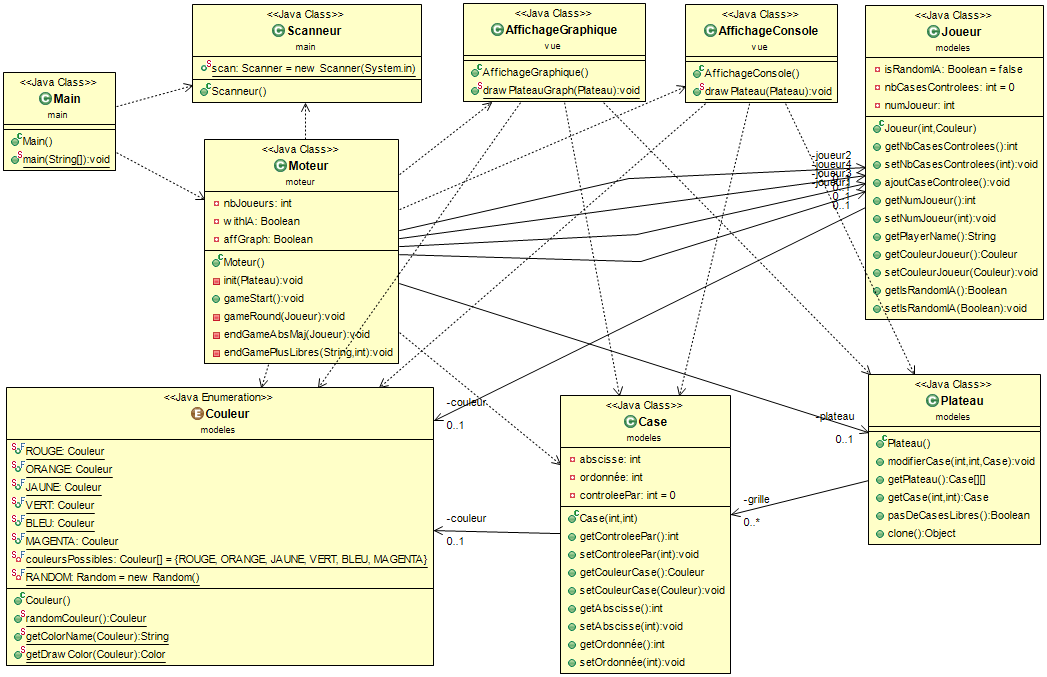
La version minimale attendue était avec un jeu qui « est entièrement en mode console, uniquement 2 joueurs, une grille carrée de 13 cases de côté. Le joueur 1 commence en haut à gauche, le joueur 2 en bas à droite. Les couleurs sont représentées par leurs initiales en minuscule si la case n’est pas contrôlée, en majuscule sinon :

* rouge : r/R,
* orange : o/O,
* jaune : j/J
* vert : v/V
* bleu : b/B,
* violet (indigo) : i/I  » (NOTE : en raison de l'absence d’indigo dans StdDraw, cette couleur a été remplacée dans mon jeu par le magenta : m/M)

A cette version minimale on pouvait ajouter au choix de nombreuses extensions.

# 2. Modélisation (description des classes)

Voici la structure du code orienté objet de mon jeu des 6 couleurs :



**Package main :**

*Main :* classe principale, qui crée l’instance du moteur du jeu puis lance le jeu

*Scanneur :* ouvre le Scanner qui permet de collecter les entrées de l’utilisateur

**Package modeles :**

*Case :* classe dont chaque instance est une case du tableau

*Couleur :* énumère les six couleurs et permet d’en choisir une au hasard

*Joueur :* classe dont chaque instance est un joueur (IA ou non)

*Plateau :* classe dont une instance est le plateau de jeu qui crée et permet d’accéder à chaque case

**Package moteur :**

*Moteur :* là où se déroule le cœur du fonctionnement du jeu : initialisation, gestion du tour de jeu, demande d’affichage de la grille et fin de partie, avec communication avec l’utilisateur

**Package vue :**

*AffichageConsole :* permet d’afficher le tableau de jeu en console lorsque demandé

*AffichageGraphique :* permet d’afficher le tableau de jeu en graphique avec StdDraw lorsque demandé

# 3. Choix et description des algorithmes utilisés.

**Fonctions et algorithmes utilisés**

* Le choix d’une couleur aléatoire se fait grâce à la création d’une instance Random() associée à l’énumération Couleur
* La fonction clone() que j’ai ajouté sur ma classe Plateau devait permettre d’effectuer une copie en profondeur du plateau de jeu. Cela devait permettre de modifier une *deepcopy* du tableau lors du tour de jeu et pouvoir la comparer avec le tableau d’origine à la fin. Ceci serait surtout utile pour arrêter la boucle d’analyse des cases du jeu (nécessaire à cause de la propagation de l’appartenance du joueur sur plusieurs cases non contrôlées) lorsque le plateau après application de toutes les modifications reste identique au plateau d’origine. Malheureusement je n’ai pas réussi à faire fonctionner cela car plateau.clone().equals(plateau) retourne false. L’utilisation de clone() dans mon jeu est donc superflue dans cet état.
* L’affichage graphique se fait grâce à StdDraw

**Extensions :**

En plus de la version minimale demandée, j’ai ajouté quelques options :

→ 2 à 4 joueurs.

→ Possibilité d’un affichage graphique de la grille avec StdDraw.

→ IA qui choisit une couleur de façon aléatoire à chaque tour. Il faut cependant choisir manuellement leur couleur de départ.

# 4. Conclusion

Le jeu fonctionne avec tout ce qui est demandé au minimum, et en option on peut jouer avec 2 à 4 joueurs, afficher la grille de jeu graphiquement et jouer avec des IA au choix aléatoire.

Ce projet m’a de plus permis de mieux me familiariser avec Java et ses particularités par rapport aux autres langages de programmation. J’ai pu appliquer en pratique ce que j’ai appris en cours, et cela a permis de solidifier mes connaissances.

Lien vers mon github : utilisateur lbarraud, repository Algorithmie\_A1\_S2, folder projet Java