Rapport de projet POO-IG

Participants :

YAZICI Servan
PARIS Albin
141

Partie du cahier des charges traités

Notre jeu respect bien le cahier des charges minimal, en effet l'utilisateur a à sa disposition une page d'accueil qui lui permet de procéder au paramétrage du jeu :

- Il peut choisir le nombre de joueur bot ou humain qu'on a limité à 4.
- Il peut ensuite démarrer soit le jeu de Domino ou le jeu de Carcassonne.
- Une version jouable du jeu de Domino sur le shell est disponible ainsi qu'une version graphique développé avec les bibliothèques awt et swing.

Nous avons implémenté toutes les règles du domino, en effet on compte les points lorsqu'on pose un domino à côté lorsque les côtés conjoints correspondent. Des bots autonomes sont capables de jouer.

Pour le jeu de Carcassonne, nous avons fait une implémentation partielle du jeu, le joueur humain et le bot peuvent jouer en posant une pièce à côté de pièces dont les côtés dont les côtés correspondent et ceux jusqu'à ce que le sac soit vide.

Page d'accueil :

Voici la page d'accueil qui permet à l'utilisateur de définir les joueurs (humain ou robot), de définir le nombre de joueur 4 maximum, puis d'ajouter un nom au joueur Humain.

Différents bouton sont présents pour le paramétrage du jeu par l'utilisateur.

Le rectangle permettant de créer un joueur est un objet CreationJoueurView, cet objet étend JPanel. Le bouton ajouter Joueur va créer un objet de type CreationJoueurView qui va se placer à côté du précédent. Cette manœuvre permet également d'ajouter un nombre non limité de joueur.

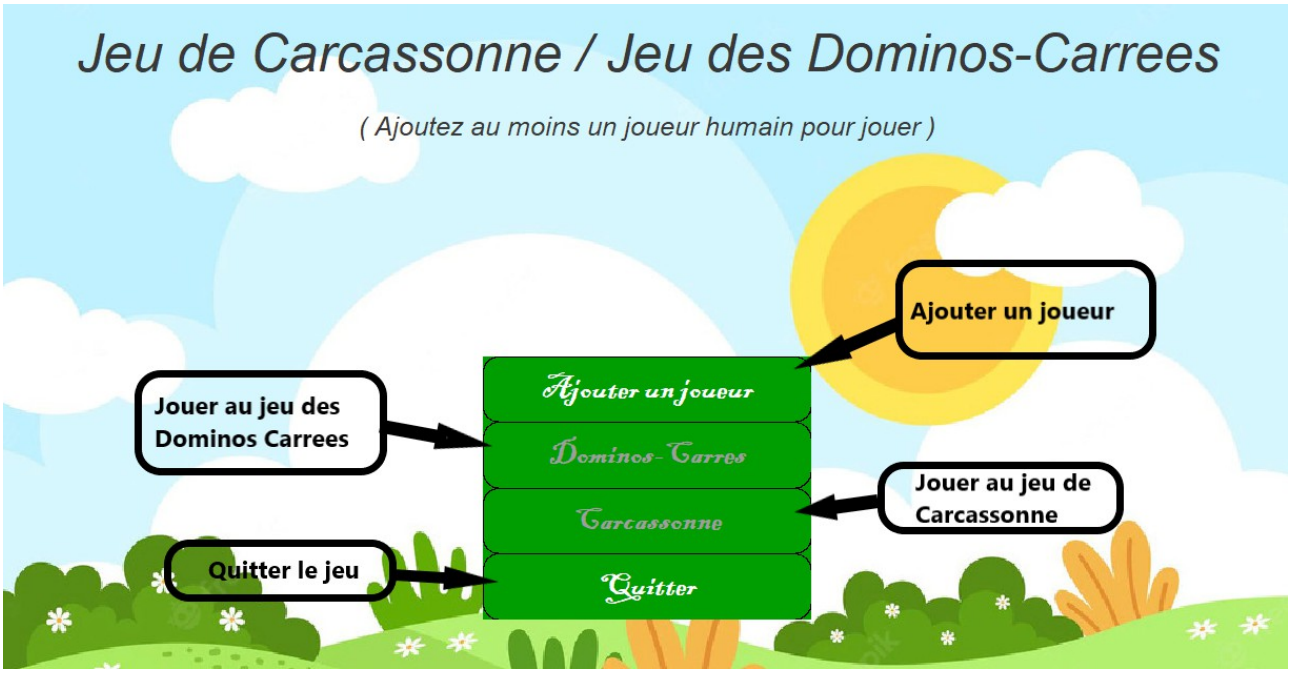
Pour pouvoir jouer, l'utilisateur est obligé d'ajouter un joueur humain à la partie.

Grâce aux différents boutons, l'utilisateur peut sélectionner le jeu qu'il souhaite, ajouter des joueurs et quitter le jeu.

Dès lors que le joueur clique sur Valider, le programme va créer un nouveau joueur, et le placer dans la liste de Joueur pour le jeu

Lorsqu'il clique sur la case Robot, le programme va créer un Ordinateur et va le placer dans la liste de joueurs.

Cette action est possible car Ordinateur est sous-classe de Joueur.



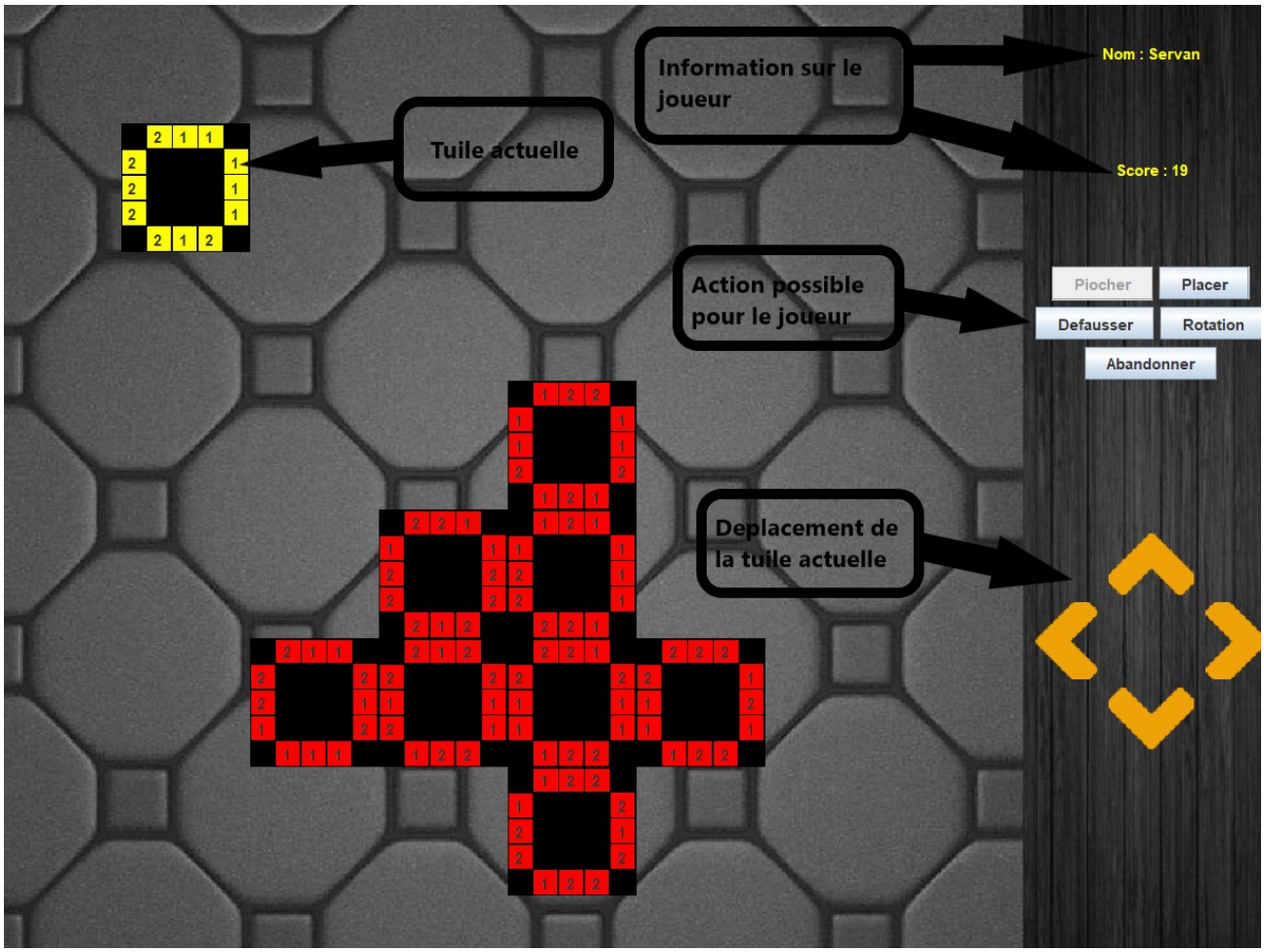
Jeu des dominos-Carres :

Pour le jeu de Dominos-Carres, nous avons implémentés toutes le règles, en effet les joueurs peut abandonner la partie à tout moment, peut placer une tuile si ses côtés correspondent, il peut défausser une tuile et la faire tourner.

Une tuile au hasard est placé au début de la tuile sur le plateau et le jeu se construit autour de celui-ci.

Le score est mis à jour en additionnant le score du score du joueur avec la somme des chiffres des côtés qui se touchent.

Pour ce jeu, le joueur n'a pas la possibilité de placer un pièce qui ne touche aucune autre pièce déjà présente sur le plateau.



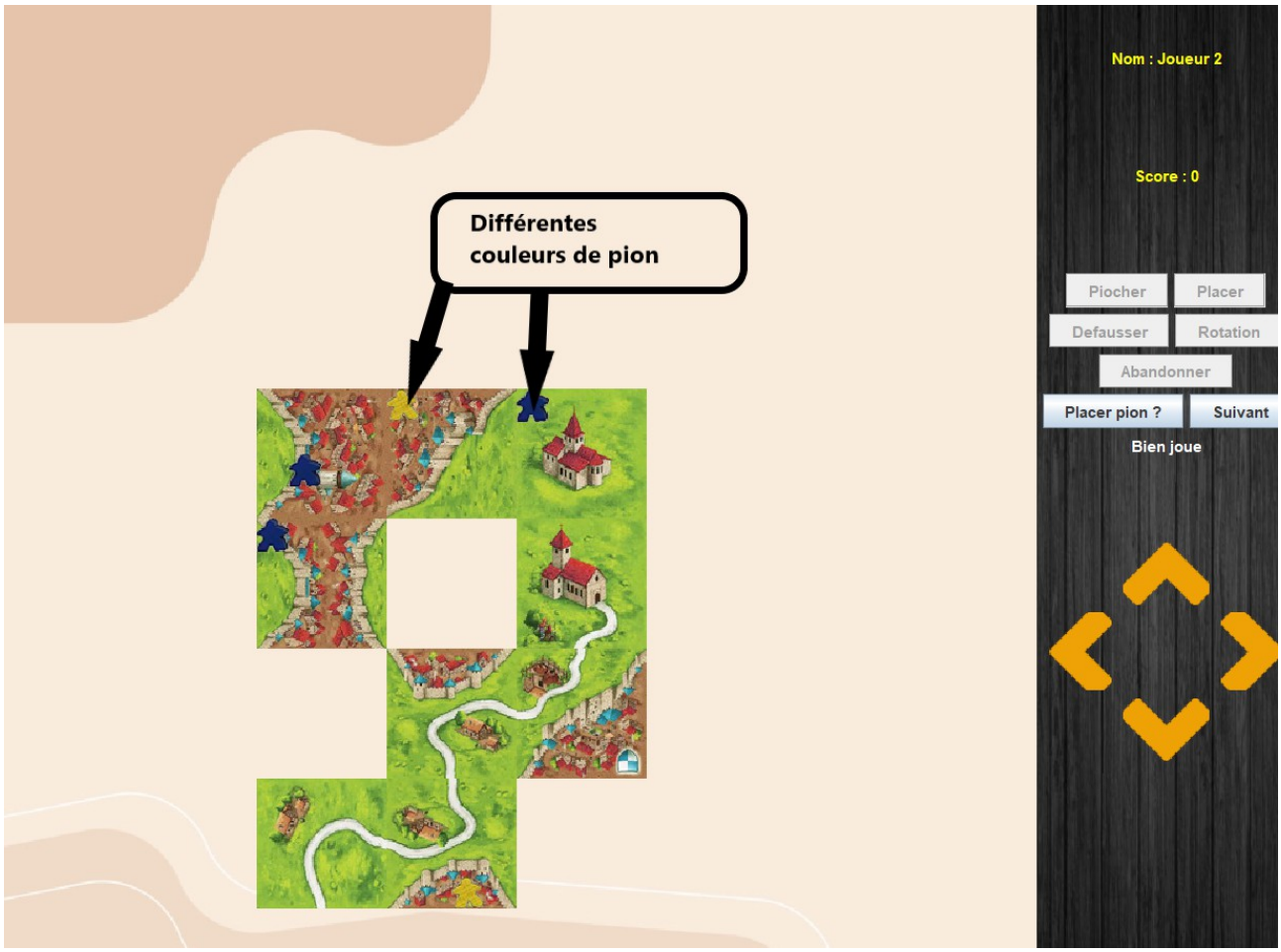
Jeu de Carcassonne :

Pour le jeu de Carcassonne, on avons aussi respecté le cahier des charges minimales, l'utilisateur a à sa disponibilité les mêmes fonctions pour une tuile que dans le jeu des Dominos-Carres. De plus, il a la possibilité de poser un pion sur la tuile qu'il vient de poser. Pour différencier les pions des joueurs, chaque joueur a une couleur de pion différente.

Pour pouvoir déplacer le pion, le joueur doit cliquer sur Bouton « Placer pion » et manipuler le déplacement de celui-ci via les flèches de déplacement.

Lorsque le joueur n'est pas entrain de déplacer une tuile, il a la possibilité de déplacer le plateau étant donné que celui-ci est infini, cette action est aussi possible pour le jeu des Dominos-Carres.

Les tuiles qui composent le jeu sont au nombre de 27, elles ont été crée à la main et reproduit, grâce à des boucles, le nombre de fois qu'il faut car certaines pièces doivent être 2 ou 3 fois dans le sac



Langage de programmation

Le projet POO-IG a été entièrement programmer en Java avec l'utilisation des bibliothèques standards tels que Math.Random, ainsi que les bibliothèques graphiques tels que awt et swing.

Pour ce projet, ce langage est très adapté en effet son aspect orienté objet est très utile ici, car les deux jeux sont assez similaires.

Des classes communes peuvent alors être crées pour servir aux deux jeux.

Implémentation du jeu

Différentes classes ont été crée pour subvenir aux besoins du jeu de Carcassonne et du jeu de Domino.

Le premier besoin évident est le plateau dans lequel on va poser les tuiles.

Nous avons passé beaucoup de temps, et avons essayé beaucoup de choses pour trouver le moyen, dans un premier temps d'avoir un plateau infini, ce qui n'était pas possible avec un tableau de tableau de tuile. On a donc décider d'implémenter le plateau avec un dictionnaire. Nous avons crée un objet Coordonnée qui représente les coordonnées x et y du plateau. Le dictionnaire associe un objet Coordonnée à une tuile.

L'avantage de cette implémentation est le fait que le plateau aura une taille infinie, et de plus la méthode pour vérifier qu'une tuile est bien placée est plus simple à se représenter et à implémenter.

Lors de la création des tuiles pour le jeu des Dominos-Carres, nous avons décidé de mettre dans un sac tous les dominos possibles continués des chiffre 1, 2 ou 3 sur les côtés.

Les tuiles du jeu de Carcassonne ont été faite à la main pour représenter correctement l'image sur la tuile. Pour celui, nous avons créée un objet Paysage qui contient des sous classes Route Mur Pre Ville pour différencier les différents paysages possibles.

La classe Jeu et toutes ses sous-classes sont en charge du bon fonctionnement de la partie, il s'assure que le joueur courant change à chaque tour, d'ajouter les joueurs à la partie, et de l'abandon d'un joueur. La classe Jeu est également chargé de la création de la pioche au commencement de la partie.

La classe Plateau et ses sous-classes sont en charge du bon suivi des règles du jeu, ils ont la responsabilité de confirmer si un joueur peut poser une tuile à une place précise. Cette classe recueille aussi les endroit où l'IA peut poser une tuile.

La classe Tuile représente une pièce du jeu qui peut être un domino ou un paysage du jeu de Carcassonne.
Elle est composé de 4 attributs qui sont des objets Coté qui représentent les 4 côtés du cette pièce.

La classe Cote est une classe abstraite qui permet de modéliser un cote d'une tuile, il possède un seul attribut qui est un tableau d'Objet de taille 3.

Toutes ces classes sont des classes abstract, elles sont utilisés pour le jeu de Carcassonne et le jeu des Dominos-Carres. Elles forment donc la base des deux jeux.

Pour gérer les mauvaises manipulations du joueur, nous avons crée de nouvelles exception tels que : ActionImpossibleException, CasePleineException et TitulaireAbsentException. Leur nom explicite leur rôle dans ce jeu.

Pour vérifier si une tuile peut être placé à une coordonnée, cette tâche est décomposée pour rendre le travail plus simple.

Dans un premier temps, la classe Jeu va prendre la tuile à placer et les coordonnées x et y de placement, puis va laisser le plateau gérer le reste.
Le plateau va vérifier si les cotés de la tuiles sont conformes avec les tuiles situé en (x,y+1), (x,y-1), (x-1,y) et (x+1,y). Si aucune de ses tuiles existent alors il ne place rien sinon il délègue la tâche à la tuile qui va elle même déléguer la tâche à la classe côte qui va juste se charge de faire 3 comparaisons.
Cette méthode de comparaison est identique pour le jeu de Dominos-Carres et du jeu de Carcassonne.

Tout les autres classes pour le fonctionnement du jeu dérivent des ces classes abstraites.

Organisation du projet

Pour commencer, on avons décidé de l'organisation qui semblait la meilleure pour aborder ces deux jeux. Tout d'abord, on a commencé à coder les différentes méthodes et classes pour pouvoir jouer au jeu des Dominos-Carres dans le shell.

Après cette phase de développement, nous nous sommes penchés sur le graphiques en commençant par coder le jeu des Dominos-Carres sur le graphiques. A l'aide d'une sous-classe TuileDCG situé dans PlateauDCG, celle-ci permet de modéliser graphiquement une tuile de Dominos-Carres.

Pour pouvoir contrôler le placement des pièces, nous avons mis en place un contrôleur qui entièrement gérer les actions graphiques du plateau.

Ce contrôleur est légèrement différent pour le jeu de Carcassonne, il a été modifié pour pouvoir aussi contrôler l'action de poser un pion sur la tuile actuelle.

Fonctionnalités avancées

Les stratégies des IA :

Différentes stratégies pour le bot ont été crée :

La première stratégie, l'IA regarde à l'endroit où il peut poser sa tuile et il la pose sinon il passe son tour

La deuxième stratégie, l'IA regarde à l'endroit où il peut poser sa tuile si il peut pas il fait une rotation et il re-vérifie jusqu'à avoir vérifier les 4 côtés puis il passe à un autre endroit où il peut poser.

Le deuxième stratégie est plus intelligente car statistiquement elle a moins de chance de passer sont tour.

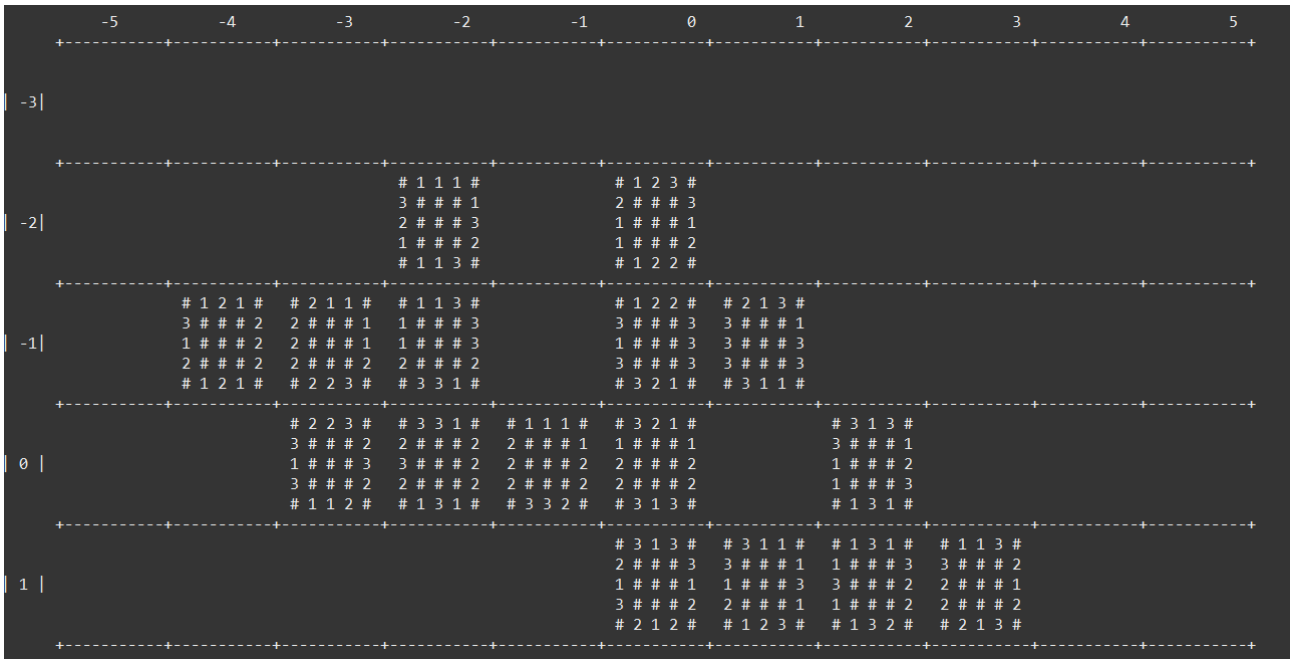
Nous avons également penser à une autre stratégie, mais nous l'avons pas implémenter. Celle-ci

serait en combinaison avec la deuxième. Donc cette stratégie va parcourir toute la liste des possibilités de placement entièrement sans s'arrêter directement sur le premier emplacement possible mais va retenir le nombre de points que celui-ci lui apporterait et son indice dans la tableau. Lorsqu'il a fini de parcourir toutes la liste, il va se placer sur l'emplacement d'indice i qui lui rapporte le plus de points.

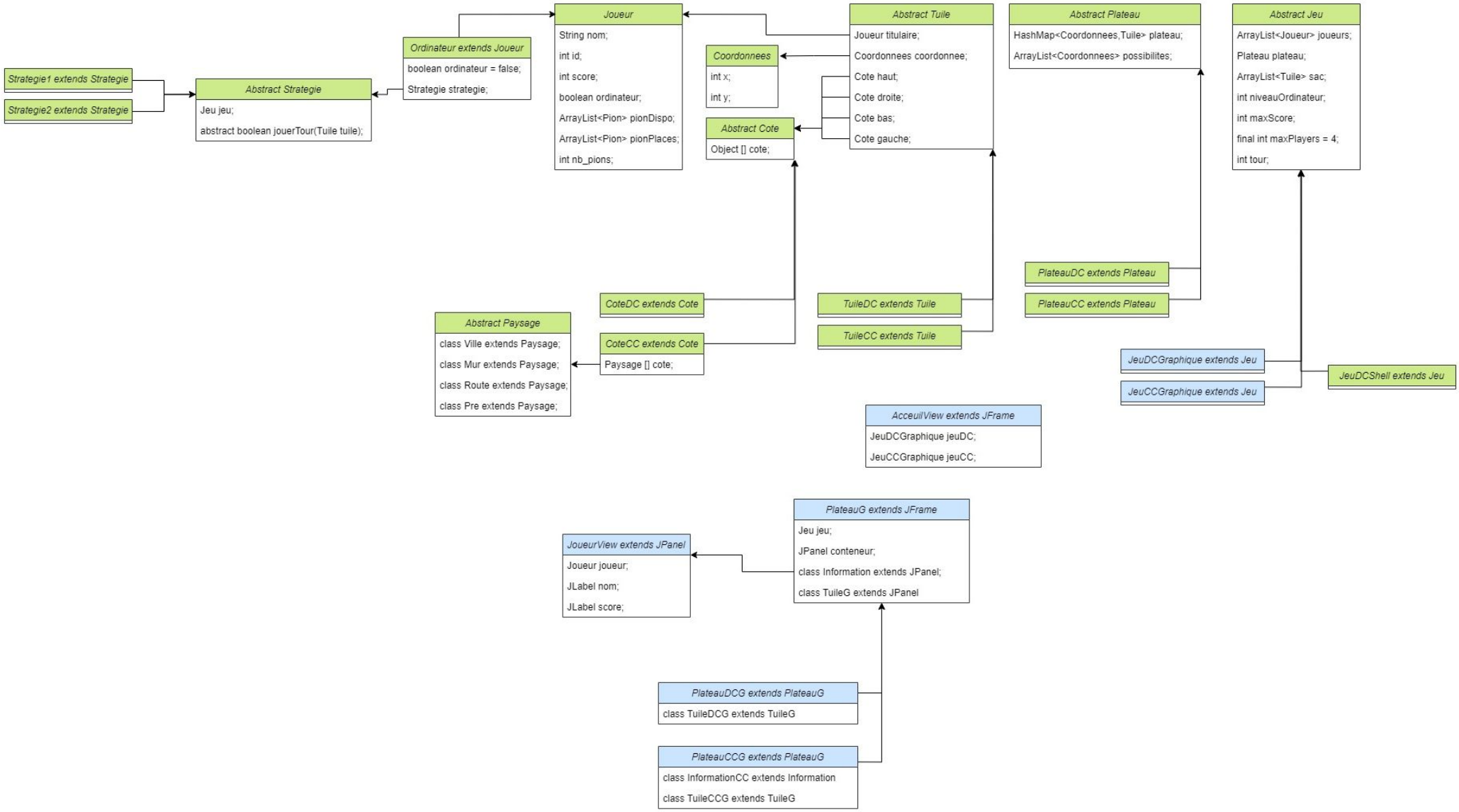
Difficultés

- Ne pas se mélanger les pinceaux avec les classes, en effet on pouvait se retrouver facilement avec une classe qui manipule l'entièreté du jeu.
- Maximiser la réutilisation des classes qui peuvent servir pour différentes manipulations
- Pour le graphique, il fallait bien comprendre le fonctionnement des différents Layout tel que le gridLayout qui est très souvent utilisé, et le flowLayout qui est utilisé pour la page d'accueil.
- Une grosse difficulté a été l'affichage dans le shell ; On ne pouvait pas faire un affichage de chaque tuile les uns à la suite des autres, en effet on ne pouvait pas remonter pour afficher la prochaine tuile à côté de la précédente. Il a fallu chercher les tuiles avec le y le plus petit, on a fait un affichage de tous les haut de chaque tuile puis revenir à la ligne et un affiche de la ligne du dessous.

  Regardons le schéma ci dessous :



Voici le plan des classes


Strategie1 extends Strategie
Strategie2 extends Strategie
Abstract Strategie
Jeu jeu;
abstract boolean jouerTour(Tuile tuile);
Ordinateur extends Joueur
boolean ordinateur = false;
Strategie strategie;
Joueur
String nom;
int id;
int score;
boolean ordinateur;
ArrayList<Pion> pionDispo;
ArrayList<Pion> pionPlaces;
int nb_pions;
Coordonnees
int x;
int y;
Abstract Cote
Object [] cote;
Abstract Tuile
Joueur titulaire;
Coordonnees coordonnee;
Cote haut;
Cote droite;
Cote bas;
Cote gauche;
Abstract Plateau
HashMap<Coordonnees,Tuile> plateau;
ArrayList<Coordonnees> possibilites;
Abstract Jeu
ArrayList<Joueur> joueurs;
Plateau plateau;
ArrayList<Tuile> sac;
int niveauOrdinateur;
int maxScore;
final int maxPlayers = 4;
int tour;
CoteDC extends Cote
CoteCC extends Cote
Paysage [] cote;
Abstract Paysage
class Ville extends Paysage;
class Mur extends Paysage;
class Route extends Paysage;
class Pre extends Paysage;
TuileDC extends Tuile
TuileCC extends Tuile
PlateauDC extends Plateau
PlateauCC extends Plateau
JeuDCGraphique extends Jeu
JeuCCGraphique extends Jeu
JeuDCShell extends Jeu
AcceuilView extends JFrame
JeuDCGraphique jeuDC;
JeuCCGraphique jeuCC;
PlateauG extends JFrame
Jeu jeu;
JPanel conteneur;
class Information extends JPanel;
class TuileG extends JPanel
JoueurView extends JPanel
Joueur joueur;
JLabel nom;
JLabel score;
PlateauDCG extends PlateauG
class TuileDCG extends TuileG
PlateauCCG extends PlateauG
class InformationCC extends Information
class TuileCCG extends TuileG