RAPPORT DE PROJET – POOIG

PROJET PROGRAMMATION ORIENTÉE OBJET

Projet réalisé par

Yacine TAMDRARI

Tsiory Irina Mendritiana RARISON RABENASOLO

Année 2022-2023

Université Paris Cité

1) Cahier des charges :

1.1Environnement de jeu :

Le package de menu du jeu est menu. Pour la création de notre menu de jeu, nous avons fait le choix de gérer en premier lieu une classe HomeController. Celle-ci hérite de la classe JFrame et elle va nous permettre de gérer tout le contrôle sur quel jeu on va choisir et les paramètres de lancement. Dans cette classe on retrouve un attribut homeView qui est une instance de la classe HomeView qui s'occupe d'afficher les éléments à choisir. On y a placé des JButtons, des JSliders, des JLabels et une classe ImagePanel héritant de JPanel pour montrer les choix disponibles et avoir un bon aperçu graphique.

Nos sliders vont nous permettre de paramétrer le nombre de joueurs réel ou artificiel.

Puis notre HomeController va se charger d'assigner les informations nécessaires avec les bons actionListeners pour le lancement d'une partie de domino ou de Carcassonne.

« Menu d'accueil »

1.2 Implémentation de toutes les règles du jeu de Domino avec le mode graphique

Pour l'implémentation de dominos nous avons 4 packages principaux :

* Le premier package concerne le modèle de nos objets de jeu qui sont la tuile, le sac et le plateau(package components).

Pour modéliser les cotés de notre jeu nous avons une classe abstraite Cote implémentant les interfaces Suitable (possède l'unique méthode correspond pour tester la correpondance des côtés) et Serializable. Pour ensuite pouvoir modéliser les côtés de Dominos nous avons la classe CoteDominos.

La classe Tuile représentera un tableau de Cote. On aura ensuite une nouvelle classe TuileDominos qui hérite de Tuile pour pouvoir rajouter les fonctions nécessaires à la représentation d'une tuile de Domino.

La TuileDominos va être représenté sous la forme d'un tableau de CoteDominos de taille 4 pour les 4 cotes de la tuile. Pour limiter les possibilités de création des tuiles, on a limité les valeurs possibles à 1 et 2 (générées aléatoirement), et c'est facile de changer les valeurs possibles en modifiant la constante MAX_VALUE dans la classe TuileDominos pour les cases du tableau de la TuileDominos. Cette classe va s'occuper de faire tourner la tuile, compter le nombre de point et de faire des vérifications par rapport aux placements sur un plateau.

La classe sac est notre stockage pour les tuiles, on y aura un attribut tableau de tuile pour représenter cela. La classe SacDominos est une classe fille de sac et va nous permettre par la suite de gérer toute la création des TuileDominos pour le jeu de Dominos.

La classe abstraite Plateau va représenter notre terrain de jeu pour positionner les tuiles. La classe PlateauDominos hérite de cette classe qui a un constructeur qui permet de créer un plateau de dominos avec un nombre de ligne et de colonne passé comme arguments de constructeurs et va par la suite définir la fonction d'affichage du plateau dans le terminal.

Toutes ces classes du package components implémente Serializable pour nous permettre la gestion du flux d'information.

* Le second package primordial est le package pour créer soit un joueur simple soit un joueur artificiel. La classe Joueur possède les attributs assez basiques pour pouvoir le présenter, et des méthodes pour jouer comme le fait de piocher ou abandonner... Le joueur Artificiel est une hérite (extends) de joueur et s'occupe en plus de pouvoir jouer selon la possibilité d'une tuile présente

sur le plateau par ces propres choix. Celle-ci va avoir la méthode de vérification coupPossible et se chargera de créer la meilleur possibilité d’action du joueur artificiel.

* Le troisième package controller va s'occuper de gérer la gestion d'une partie en graphique ou sur le terminal.

Pour la partie sur le terminal, nous avons la classe TerminalController qui va s'occuper de demander toutes les informations aux utilisateurs qui vont jouer et de lancer la partie.

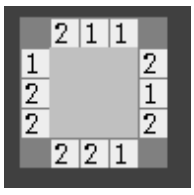
Dans ce cas il y aura la gestion de notre sac et un affichage défini via une méthode dans la classe Plateau. La gestion de la partie dans ce mode se fera en grande partie avec un scanner.

Pour la partie en mode graphique nous avons un attribut principal pour l'affichage du jeu qui sera mieux expliquer dans le dernier package concernant la vue du jeu. Pour le cas de la classe abstraite GameController il suffit de connaître qu'elle a cette attribut gameView et d'autre attribut comme le plateau, le sac, et les joueurs pour la gestion de la partie dans ce mode. Nous avons dans ce mode une gestion du placement des tuiles par des méthodes et de MouseListener car GameController hérite de JFrame et implémentes ces 2 interfaces cités précédemment. GameControllerDominos hérite ensuite de GameController et se charge de générer la bonne vue.

* Le quatrième package concernant la vue va nous permettre d'avoir une meilleure représentation pour l'affichage.

La classe JGameInfo va nous permettre de représenter toutes les informations concernant le score des joueurs et les informations pour le joueurs courant entrain de jouer. Cette classe hérite de JComponents. Les informations seront principalement représenter par des JLabels et des JPanels.

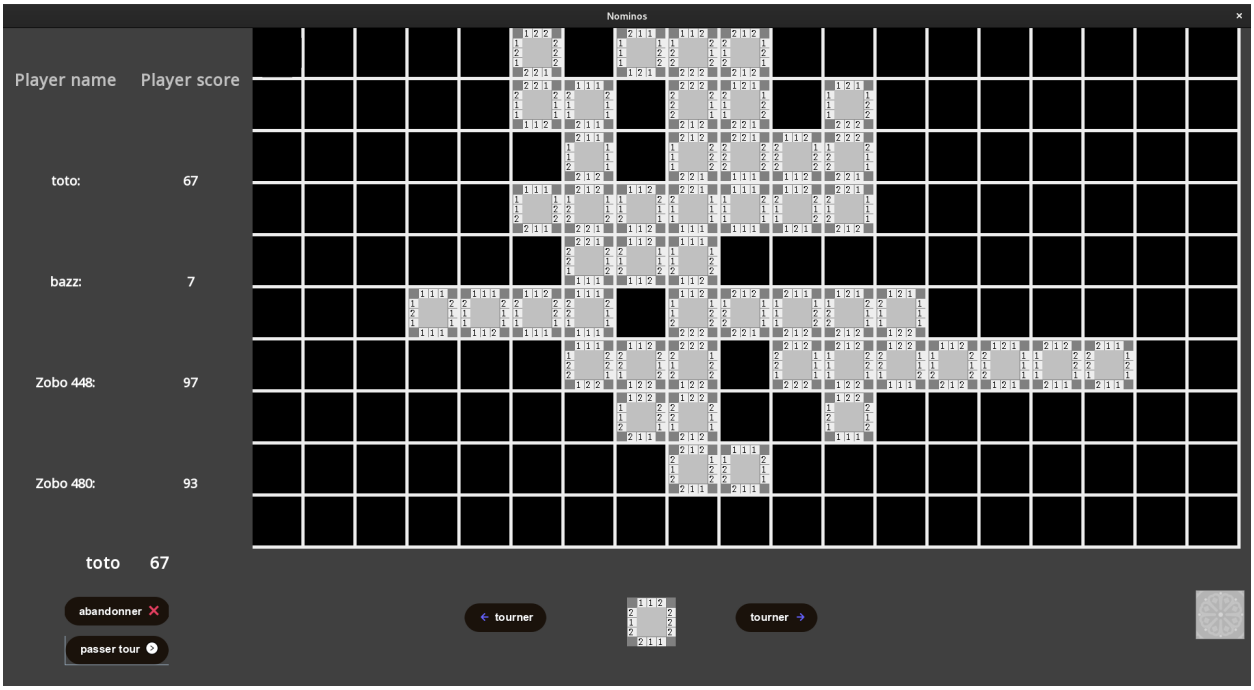
La classe abstraite JTuile représentera la vue des tuiles, celle ci hérite de JPanel. La classe JTuileDominos qui hérite de JTuile s'occuper de gérer toute la transformation pour représenter la tuileDomino, cela par diverses méthodes de création extrêmement précises avec les méthodes de Graphics2D.



« Représentation graphique d’une tuile de Dominos »

La classe abstraite JPlateau sera une manière de représenter la vue de notre plateau , cette classe hérite de JPanel. Pour ce faire on aura un attribut tableau de tableau de JTuile et un attribut plateau (model) et le plateau sera affiché en utilisant la méthode paintComponent de JTuile qui permet de dessiner une tuile et chaque méthode d'affichage de chaque plateau que ce soit pour le jeu Carcassonne ou pour le jeu de Dominos sera après définie dans les classes concrète JPlateauDominos et JPlateauCarcassonne.

La classe GameView est un ensemble de toutes les classe de ce package vu précédemment pour avoir tous ces éléments en un seul. En effet la classe GameView hérite de JPanel et va nous donner un affichage clair et précis.La classe GameViewDominos hérite ensuite de GameView pour pouvoir styliser l'affichage avec un style plus cohérent avec le jeu Domino.



« Représentation graphique d'une partie de Dominos »

1.3 Implémentation des règles du jeu de Carcassonne avec un aperçu graphique

Pour gérer la partie du mode Carcassonne , nous avons gardés les idées qui fonctionnaient bien

pour domino et nous les avons adapter pour jouer à Carcassonne.

Dans components on a la classe TuileCarcassonne qui va nous permettre de faire une composition des tuiles. En effet la classe abstraite Paysage sera celle qui nous donnera le modèle pour le côté des tuiles. On aura ensuite les classes Pres,Ville, et Chemin qui héritent de Paysage.

Pour représenter la tuile de jeu de Carcassonne, on modélise TuileCarcassonne heritant de Tuile et ayant comme côté des CoteCarcassonne. On y retrouve aussi un attribut partisan, avec d'autres attribut boolean pour vérifier la possibilité pour avoir un bouclier,une abbaye ou un carrefour. Cette tuile pourra être tourner avec la méthode définie dans la classe abstraite Tuile sans besoin de la redéfinir, et va avoir une redéfinition de toString pour simplifier les paramètres de recherches que l'on verra plus loin.

La classe Partisan représentera le placement partisan avec un String pour définir sa couleur et un int pour le positionnement.

La classe PlateauCarcassonne hérite de la classe abstraite Plateau et définie juste un constructeur pour initialiser les cases de plateau avec des TuilesCarassonnes et elle utilise toutes les méthodes de la classe abstraite Plateau partagés entre PlateauDominos et PlateauCarcassonne.

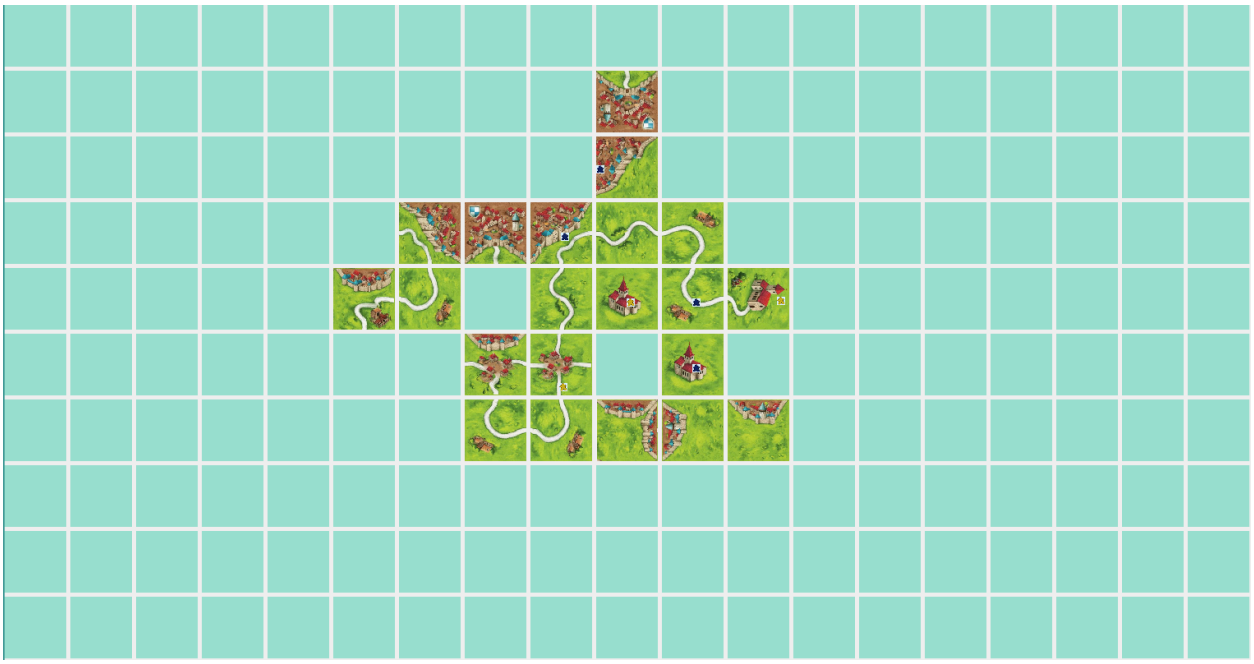
La classe SacCarcassonne celle qui se chargera de créer toute les tuiles disponibles dans le jeu de carcassonne. Le tableau de TuileCarcassonne se chargera de garder toutes les possibilités de création de tuile d'après les règles de jeu Carcassonne. On pourra par la suite retirer une des tuilesCarcassonne depuis notre sacCarcassonne. Le nombre tuile dans notre sacCarcassonne sera de 72 comme dans un vrai jeu de Carcassonne.

La représentation de JTuileCarcassonne qui hérite de JTuile et l'ajout de la classe JPartisan est le seul changement essentiel pour la représentation des tuiles de Carcassonne.Les changements apportés à JTuileCarcassonne est la manière dont on pourra montrer la tuile, on va pour cela charger une image qui sera dans notre dossier Ressource en fonction de la description de la tuile dans cette classe.On pourra repeindre l'image en fonction de la tuile que l'on affecter (changement de paintComponent).

La nouvelle classe Partisan se chargera de représenter un partisan de manière graphique via une taille défini et une image que l'on chargera sachant que cette classe hérite de JPannel.

GameControllerCarcassonne héritant de GameController se chargera ensuite d'afficher tous les éléments que l'on a défini dans la GameViewCarcassonne et assurera par la suite le bon fonctionnement du jeu de Carcassonne. Pour le placement de tuile elle se fait comme dans

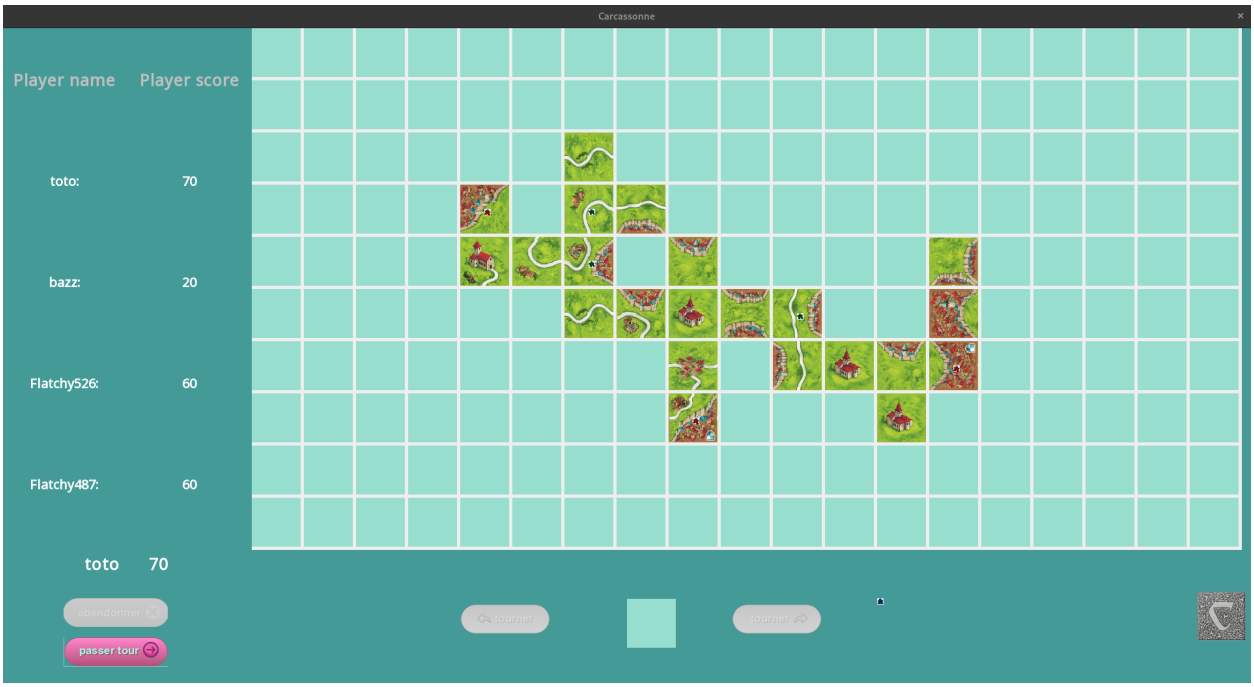
domino , et pour le placement de partisan sachant que GameViewCarcassonne hérite de GameView qui est la classe général qui se charge d'afficher le JPlateau, la JTuile le JGameInfo et les boutons nécessaires pour le contrôle du jeu qui sont les même entre Dominos et Carcassonne



« Représentation graphique d'un plateau de Carcassonne »

Pour l'implémentation des sauvegardes du jeu nous avons dans GameControllerDomino et GameControllerCarcassonne des méthodes qui se chargeront de lancer les fichiers de sauvegarde saveCarcassonne.ser ou saveDomino.ser.

La méthode save dans GameController créera un nouveau fichier en vérifiant l'instance courante quand on fermera le fenêtre du jeu(sauvegarde automatique après la fermeture de la fenêtre du jeu). Ensuite au menu quand on utilisera le bouton load on fera appel à la méthode load de selon quel jeu est lancé et on affectera l'instance courant du jeu au fichier qui devrait lui être associé.



« Représentation graphique d'une partie de Carcassonne »

2) Problèmes connus(liste non exhaustive) :

L'un des premiers problèmes rencontrés est la bonne structure de donnés à choisir pour représenter un plateau de jeu que ce soit pour Carcassonne ou pour Dominos et après avoir chercher on a trouvé que la plus part des parties de jeux de Carcassonne on ne dépasse pas 10 * 10 donc on décider d'utiliser une taille fixe pour le plateau mais on profitant de toute la largeur et la hauteur de l'écran sur lequel le jeu sera lancé.

Au cours du projet on a eu un problème de comment pouvoir afficher les tuiles sur une fenêtre au départ on a utilisé un gridlayout avec une taille de 5*5 mais on avait pas un bon rendu et on étais pas satisfait du résultat, après on a appris qu'on peut redéfinir la méthode paintComponent mais on a utilisé une méthode paintComponents avec un « s » à la fin (mauvaise signature en plus on a mit que notre classe JTuile hérite de JComponent) et il n'y a rien qui s'affiche sur le Jframe et en changeant le nom vers paint la tuile s'affiche, mais on a lue aussi que c'est une mauvaise pratique de redéfinir la méthode paint et il faut redéfinir paintComponent et au final on a réussi l'affichage des tuiles et tout l'affichage est venu tout seul après sans aucun problème.

Au départ on a implémenté chaque jeu indépendamment mais en développant Carcassonne on s'est rendu compte qu'on fait que des copier coller et qu'on change quelque méthodes surtout

pour le contrôle et la vue donc on a essayé de factoriser le code en commun pour les deux jeux mais chaque jeu à son modèle spécifiques des tuiles donc on a réglé ça en définissant une interface Suitable et des Cote pour représenter les cotés des tuiles et on a réussi à factoriser le code pour les tuiles que se soit pour Dominos ou Carcassonne et tout le reste est devenu facile à factoriser.

Sans oublier aussi qu'au cours de développement de notre projet on aurait du repérer et régler plusieurs bugs en jouant à plusieurs parties en étant paranoïaque et testé plusieurs cas limites que les utilisateurs peuvent générées s'ils veulent pas suivre le flux du jeu et essayer de trouver une faille dans le jeu ou un « hack » pour gagner à toutes les parties…...

3) Extensions possibles :

Une extension possible des autres modes possibles du jeu de Carcassonne en utilisant les mêmes classes qu'on a déjà définie comme ajouter les autres tuiles d'extension dans le jeu.

On n'a pas implémenter le calcul de score pour le jeu Carcassonne mais ceci est aussi facile car tous les composants du jeu Carcassonne sont indépendants et communiquent entre eux d'une manière facile à comprendre.