2019-2020

Quoridor

fotso guiffo

umons

2019-2020

Table des matières

Quoridor

Introduction

L'objectif de ce projet était de réaliser le jeu Quoridor, un jeu de société qui peut se jouer soit à deux ou à quatre sur un tableau de taille 9x9. O*ù* chaque joueur a pour but t'attendre le camp adversaire tout en empêchant l'autre d'attendre son camp et poussant des murs sur le tableau. Dans ce jeu chaque jouer possèdent 10 murs à sa disposition, et le jeu ce joue tour par tour. Si un jouer venais a ce déplacer ou à pousser un mur c'est donc à l'autre joueur de jouer. Dans les paragraphes suivant je vais décris le projet et sont contenue en profondeur.

Répartition des tâches au sein du groupe :

Le travail devait se faire par groupe de deux et mon partenaire et moi avions décidés de repartir le travail et crée des comptes « github » pour qu'on puisse voir l'avancer de chacun dans sa partir du projet mais malheureusement mon partenaire ne faisais rien du coup j'ai tout faire seul.

Contenu du projet :

- Le package board
  - Board
  - Tile
  - Wall
  - Alliance
  - BoardUtils
  - Pawn

- RedPawn
- YellowPawn
- Move

Le package GraphicInterface

- Game
- GameHistory
- GameSetUp
- Main
- NumWallPanel

Le package PGN

- Utilities

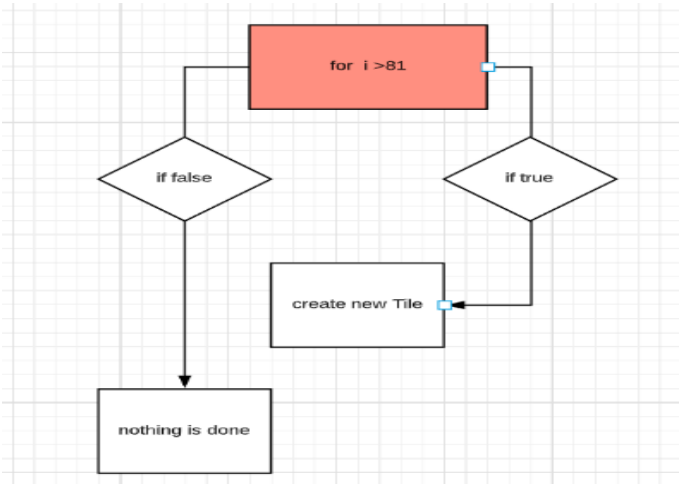
Le Package player

- Player
- RedPlayer
- YellowPlayer
- MoveStatus
- MoveStrategy
- MoveTransition
- Computer

Description des différentes class et class interne :

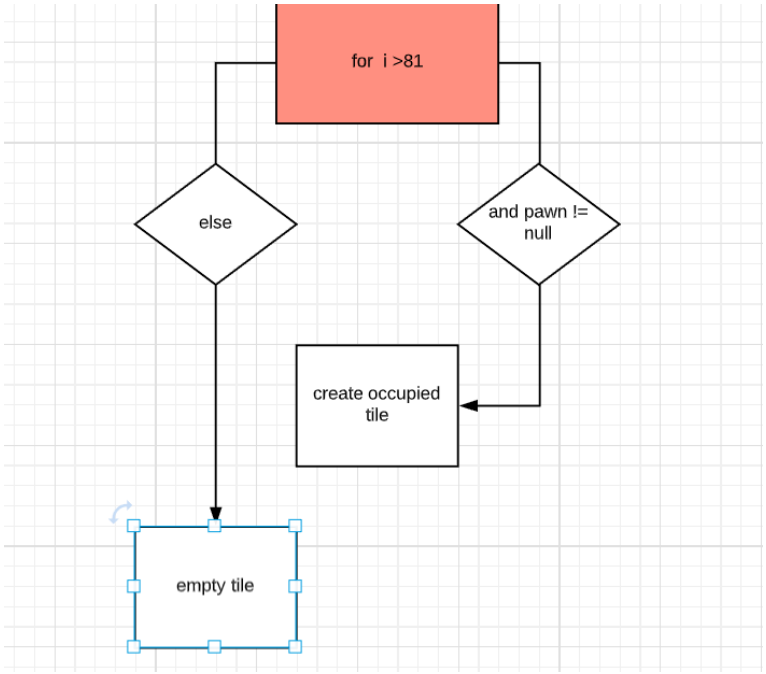
- **Board** :

Cette class représente le tableau sur le quelle la partie vas se dérouler. Elle contient une class interne Builder qui permet de construire le tableau « build », placer les pions sur le tableau « setPawn » et changer les tours de jeux « SetMoveMaker ». Elle contient aussi des méthodes importantes comme « defaultBaord » le positionnement du tableau par défaut et la méthode « MakeBoard » qui construire le tableau constituants des Objets de type Title.



- Tile :

Cette class permet de représenter les casse sur le tableau. Cette classe a coordinate » et éventuellement pawn si le casse est occupée par un pion. Elle a des méthodes comme « createTile » qui permet de créer le Tile



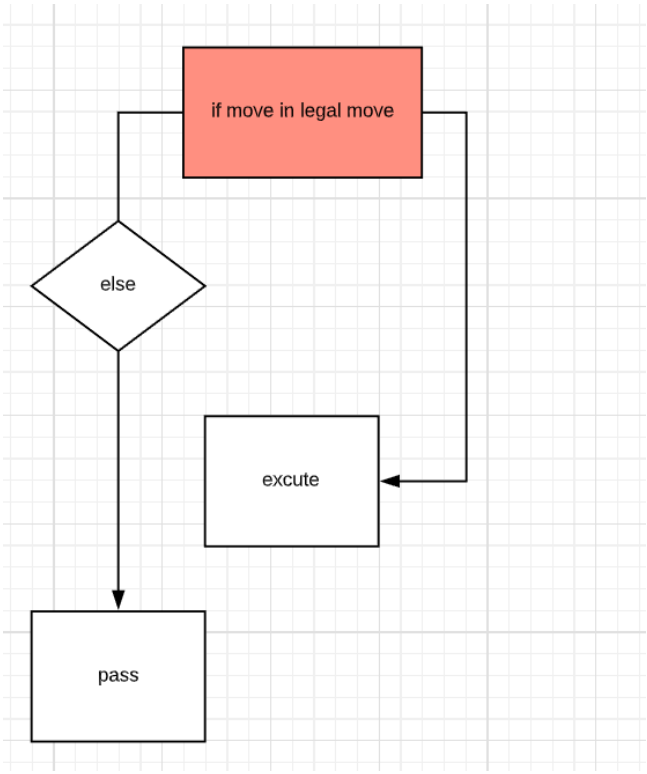
Et aussi une méthode comme « createAllPossibleTiles » qui crée tous les tile possible.

- Wall :
  Cette représente les murs du jeu elle nous permet d'empêcher l'adversaire d'avancer. Elle a un variable direction qui le permet de connaitre la direction dans laquelle elle doit être placée « horizontal ou vertical ».
- Alliance :
  Elle nous permet de représenter les différente équipes dans ce cas rouge et jaune.
- BoardUtils :
  Cette méthode fais office de caisse elle contient les méthodes assez nécessaire pour le jeu, comme la « isGameOver » qui test si le jeu est fini.

- Pawn :
  Cette class est la représentation des pions dans jeux. Elle contient une méthode assez importante pour le déplacement des pions dans le jeu « CalculateLegalMove » qui calcule les déplacements que chaque pion peut effectuer. Une méthode comme « getAlliance » qui retourne à quelles équipes le pion appartient. Elle deux enfants RedPawn et YellowPawn qui redéfinies une méthode comme getAlliance.

- Move :
  Elle est l'une des class les plus importante du jeu car sans elle les déplacements seront impossible. Elle une classe interne « MainMove »

Et une méthode comme exécute qui permet d'effectuer le déplacement d'une classe vers une autre. Toute en vérifiant si le déplacement peut être effectué sans problème.



- Game :

  La classe game gère tout ce qui est interface graphique, comme les différents objets logique vont être représentés graphiquement. Comme Tableau logique, les casses logiques etc. le tableau logique est représenté par la classe interne « BoardPanel », les cases par la classe interne «TilePanel » et enfin les murs par les classes internent « horizontalWallPanel » et « verticalWallPanel » qui devient jaune quand la souris passe sur eux et reste jaune quand le joueur ou AI place un mur sur l'un d'eux.

- GameHistory :

Elle nous permet de sauvegarder l'historique du déplacement des pions elle crée aussi un panel qui est à droite du panel principal du jeu. Elle complété par la class move historie dans la classe Game .c'est dans classe que le différent déplacement sont ajouter dans l'historique de déplacement et qui est mis à jour quand un joueur ce déplace sur le tableau

- Game Setup :
  Elle nous permet de créer un setup pour le jeu qui nous permet de faire un partie joueur contre joueur ou joueur contre Ai.
- Main :
  Elle contient simplement une méthode qui nous permet de lancer le jeu.

- NumWallPanel :
  Ce panel devait contenir les nombre de chaque joueur.

- Utilities :
  Cette classe joue un rôle dans la sauvegarde du jeu.

- Player :
  Elle la représentation des jouer et contient des méthodes comme « getActivePawns » qui retourne les jouer d'un certain jouer, « getAlliance » qui permet de connaitre l'appartenance du joueur et enfin une méthode comme « getOppenent » qui retourne l'adversaire de chaque joueur. Elle trois en « Redplayer » le jouer de l'équipe rouge, « yellowPlayer » le joueur de l'équipe jaune et enfin « computer » qui contient des méthodes comme « RandomMove » et « RandomPlacedWall » qui effectue des déplacements et place des murs aléatoire respectivement.

Points forts du projet :

Le point fort du projet c'est l'historique de déplacement qui peut nous permettre de facilement debugger certain problème

Points faibles du projet:

Les point faible du projet le faite que j'ai juste une intelligence artificielle, je fais que les sauvegarde sont incomplète.et ils j'ai bug très rare dans le positionnement des murs. C'est souvent vue quand c'est Ai qui joue

Les apports positifs et/ou négatifs de ce projet

Le projet ma de mieux comprendre la programmation et savoir me débrouiller tout seul et aussi mieux faire mes rechercher.

Guide utilisateur

Quand le jeu est lancer on peut directement jouer joueur contre joueur ou si on veut jouer contre Ai on peut cliquer sur setup puis sélectionner computer pour jouer avec Ai. Et si on veut afficher les différent déplacement on peut cliquer sur préférences et clocher la casse highlight.

Biographie :

https://github.com/amir650/BlackWidow-Chess/tree/master/src

https://www.ultraboardgames.com/quoridor/game-rules.php

https://github.com/sbox/quoridor/blob/master/src/quoridor/GameImpl.java

https://github.com/tetrapus/Quoridor/blob/master/src/quoridor/Board.java