isep
École d'ingénieurs du numérique

Domi'Nations

Bouchama Sofiane - Caruel Rémi - Gauthier Étienne

Vue d'ensemble

Ce projet consistait à créer un jeu jouable sous java. Ce fût l'occasion de tester notre connaissance de java mais aussi de l'améliorer.

Objectifs

1. Développer une interface graphique fonctionnelle.
2. Lire un fichier CSV pour y récupérer les dominos stockés.
3. Implémentation d'une IA.
4. Implémentation du jeu King Domino en java.

Introduction

Java est un langage de programmation très utilisé dans le monde professionnel. Sa particularité est sa force en programmation orientée objet. Programmer un jeu était alors un bon moyen de faire comprendre toute l'importance de cette programmation orientée objet. Faire ce jeu a été un défis pour plusieurs raisons. La première est qu'il a fallu partir d'une feuille vide et commencer à coder. Il a donc fallu tout d'abord réfléchir à l'architecture de notre projet avant de commencer à coder. La seule chose connue étant les règles du jeu et la manière de jouer. La seconde fût le travail en groupe. On a du apprendre à s'organiser, à se répartir le code à faire. il ne fallait pas que l'on travaille tous sur les même fonctions en même temps pour des raisons d'efficacité. La dernière fût le défi du temps. Il fallait rendre le projet à une certaine date qu'il soit fini ou non. Ceci est semblable au domaine de l'entreprise où un client fait une commande pour une certaine date. Dépasser cette date coûte alors de l'argent c'est pourquoi il est primordial de réussir à respecter les délais. Il était donc fourni un livret et un fichier CSV. Ce fichier contenait tous les dominos utiles pour le jeu. Il était composé de 5 colonnes : 2 colonnes pour les types des moitiés, 2 colonnes pour le nombre de couronnes sur chaque moitiée et enfin l'index de la tuile. C'est ce fichier qui est lu dans le programme afin de connaître la composition de la pioche du jeu. Ensuite le livrable décrivait le jeu qu'il fallait développer. Il y était précisé les règles du jeu, les fonctionnalités à implémenter mais aussi d'autres fonctionnalités que l'on pouvait implémenter mais de manière facultative.

Grandes étapes

I. étude du projet.

1. Définition de certaines règles.
2. Création du diagramme UML.

II. Réalisation du projet

1. Découpage du travail en plusieurs sous-parties.
2. Développement du projet
3. Réalisation du calcul du score
    a. Calcul du score
    b. Gestion en cas d'égalité au score

III. Manuel d'utilisation du projet

I. Etude du projet.

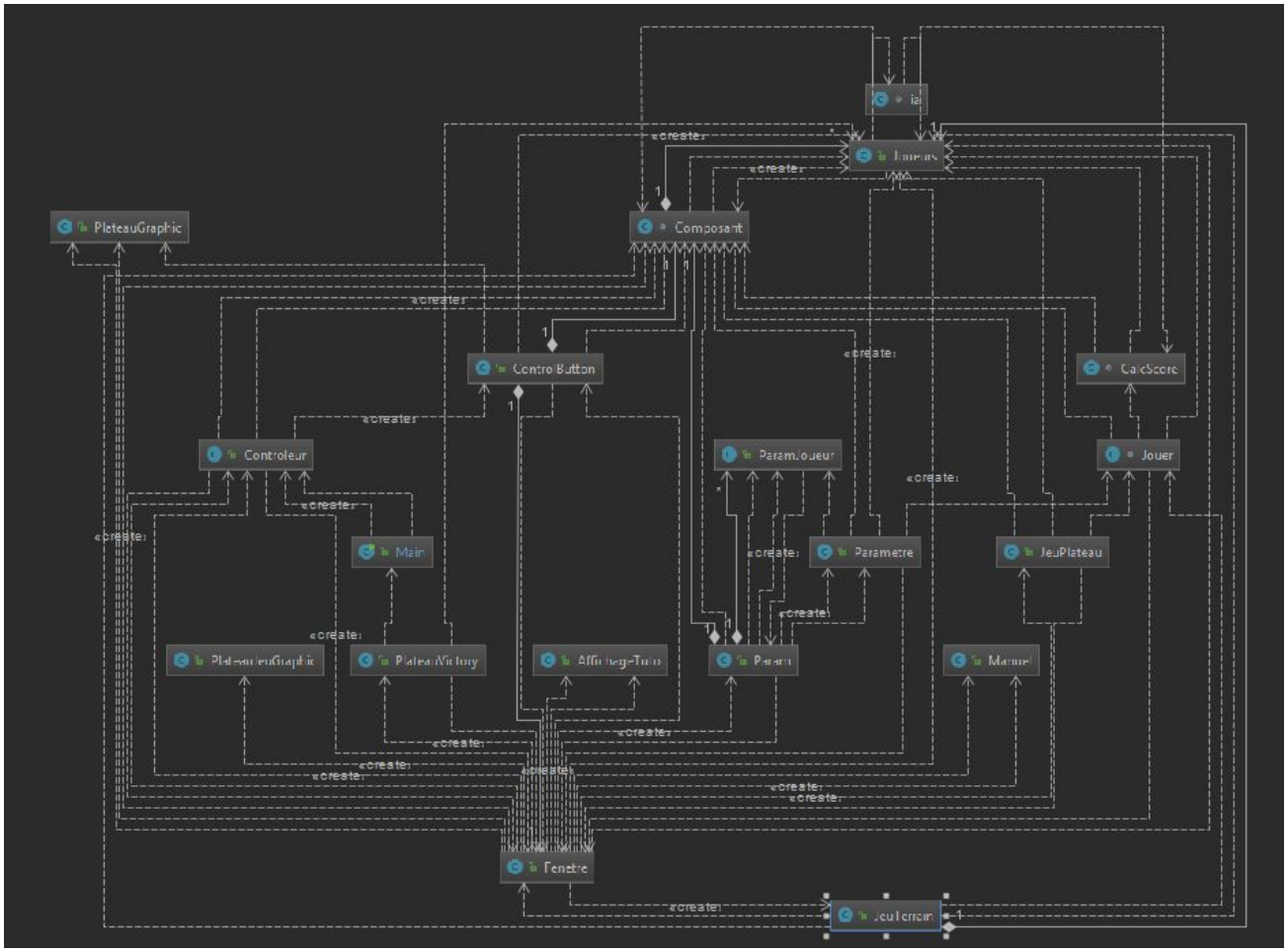
1. Définition de certaines règles.

Voici notre version des régle de Domi'Nation :

- Lors du premier tour 4 dominos seront tirés au hasard dans la pioche puis ils seront placés par ordre croissant puis retournés. Ensuite chaque roi seront tirés aussi au hasard. Le premier roi tiré peut choisir en premier la carte qu'il souhaite prendre, ensuite le 2ème roi peut choisir et ainsi de suite. Une fois que tous les joueurs ont choisi leur carte, le deuxième tour peut commencer.
- A partir du deuxième tour, tous les tours vont se dérouler de la même manière jusqu'à épuisement des cartes dans la pioche. De nouveaux dominos vont être tirées de la pioche puis placés dans l'ordre croissant et retournés. Ensuite le roi sur la carte avec le plus petit indice commence par choisir une nouvelle carte., puis chaque roi en montant dans les indices. Une fois que vous placez votre roi, prenez le domino sur lequel était placé votre roi et placez le sur votre terrain. Le terrain est de taille 5x5 dominos et comporte en son centre un château. Tous les dominos doivent être placés de sorte à ce qu'il soit en contacte avec au moins un autre dominos (ou avec le château) sans rien recouvrir et en étant totalement dans le terrain.
- Une fois que tous les dominos ont été placés, le jeu va calculer votre score en fonction de : (cf partie II.3 pour plus de détaille)
    - du nombre de couronnes sur un bloc de terrain d'un type.
    - de la taille d'un bloc de terrain du même type.

Le joueur avec le score le plus élevé gagne la partie.

2. Création du diagramme UML.



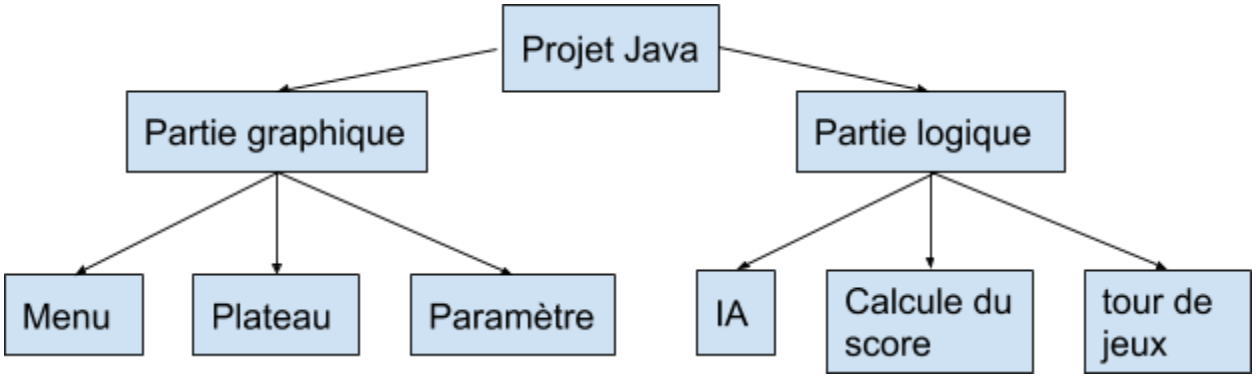
Voir diagramme UML joint dans le dossier JavaProject nommé MainPackage.uml.

II. Réalisation du projet

1. Découpage du travail en plusieurs sous-parties.

Nous avons découpé le projet en plusieurs partie afin de faciliter la répartition du travail, de se donner plus facilement des objectifs et pour permettre de rendre le code plus lisible.

Deux grosses parties formées de sous parties se dégage, la partie graphique d'un côté et la partie logique de l'autre. Voici un shéma pour clarifier notre découpage ;



Descriptif complet des classes du projet:

Partie graphique :

- AffichageTuto : affiche une fenêtre expliquant comment jouer
- ControlButton : contrôle les boutons de l'interface graphique
- Domino : défini l'aspect d'un domino, un domino a sur une face 2 images et l'autre face 2 numéros
- Fenetre : gère ce qui est affiché sur la fenêtre du jeux
- Parametre : demande aux joueurs de régler les paramètres de la partie
- Manuel : affiche le manuel d'utilisation du projet
- PlateauGraphic : menu du Jeu
- ControlButton : qui va attribuer les actions de la partie graphique une fois que l'on aura cliqué sur le bouton "Commencer"
- JeuPlateau : on crée un plateau avec des soclePion où seront déposés les dominos avec un écouteur de souris qui permettra des déposer les dominos dans le plateau en maintenant la souris en mode "pressed" sur l'image puis en relachant dans une case
- JeuTerrain : permet de placer le domino sur le terrain

- Param, ParamJoueur et Parametre : permettent de stocker les paramètres des joueurs tel que le statut du joueur ainsi que la page de lacement avec les rois et les dominos où l'on place le r

Partie logique :

- CalcScore : fonctions qui calculent le score d'un joueur en fin de partie
- Composant : classe de types database qui sauvegarde les données de la partie (nombre de joueurs, nombre de dominos…)
- ia : fonctions qui font jouer un ia au jeux. A chaque tour, l'ia, calcule le score que lui donne chaque domino qu'elle peut choisir et elle choisi le domino qui lui rapport le plus de point
- Jouer : fonctions qui régisse les tours de jeux et l'initialisation de la partie
- Joueur : créé une database par joueur contenant les informations sur le joueur (pseudo, couleur…)
-

2. Développement du projet

Nous avons réalisé le projet en nous aidant de github afin de pouvoir coder efficacement en même temps sur un même projet java.

Le développement de la partie logique s'est passé sans trop de difficulté car nous connaissions déjà tout le vocabulaire nécessaire et notre jeux a très vite marché dans la console. La partie graphique, elle, demande des connaissances que nous n'avions pas du tout, il a donc fallu aller chercher sur internet et nous pensons qu'un tp orienté graphique nous aurait beaucoup aidé car il existe des centaines de librairies et méthodes sur internet et nous nous sommes un peu perdu dans tout ce flot d'information. Cette partie nous a vraiment pris beaucoup plus de temps.

3. Réalisation du calcul du score

1. Calcul du score

On a découpé le calcul du score en 5 fonctions. Trois d'entre elles sont utilisées pour calculer le score des joueurs.

La première fonction effectue une copie du tableau qui contient les données relatives au terrain du joueur. Ceci est important car durant le calcul du score, les

données présentes dans le tableau vont être modifiées. Une fois cela fait, il est possible de calculer le score du joueur sans modifier son terrain.

Ensuite la fonction calcul (la deuxième) appelle la fonction calcScore (la troisième) à partir de la première case du tableau. De là, cette fonction va s'y ballader en récoltant à chaque fois les informations relatives au nombre de couronnes sur chaque case du même type que la première. Afin de ne pas se balader à l'infini, la fonction va remplacer la valeur de cette case par une autre dont on sait qu'elle ne pourra jamais se trouver sur le terrain sauf placée par cette fonction. calcScore renvoie à la fonction de calcul du score 2 valeurs, le nombre de tuiles en groupes du même type et le nombre de couronnes présentes sur ces terrains.

Ensuite, la fonction calcul va à nouveau faire appel à la fonction calcScore à partir de la prochaine tuile non déjà visitée. Elle va renvoyer le score du joueur.

2. Gestion en cas d'égalité au score

D'après le livret, il fallait gérer différents cas d'égalité. Si il y avait égalité du score, on devrait alors prendre celui qui avait le plus grand terrain, et enfin, s'il y avait toujours égalité, on devait prendre pour gagnant celui qui avait le plus grand nombre de couronnes présentes sur son terrain. Pour éviter d'utiliser 3 conditions, on a choisit d'appliquer des coefficients et d'utiliser un score de comparaison. Deux choses importantes : le nombre de tuiles est au maximum de 25 et le nombre de couronnes est au maximum de 75. Donc si on multiplie le score du joueur par 100 et qu'on y ajoute le nombre de tuiles, puis qu'on multiplie ce résultat par 100 afin d'y ajouter le nombre de couronnes présentes sur son terrain. Ainsi, si un joueur a un score de 62 avec 24 tuiles et 12 couronnes, alors son score de comparaison serait 622412. On garde toujours l'information exacte sur le score du joueur, son nombre de tuiles, et son nombre de couronnes.

III. Manuel d'utilisation du projet

Pour une utilisation console lancez le Main.java et suivez les instruction de la console.

Pour une utilisation graphique lancez le Main_bis.java puis réglez les paramétres de la partie en appuyant sur "commencer la partie". Une fois les paramètres remplis cliquez sur "lancer la partie". Pour déplacer une tuile placez votre souris sur la tuile et

gardez le clic gauche de votre souris enfoncé, relâchez le pour déposer votre tuile. Cliquez sur “ok” quand vous avez terminé votre tour.