Rapport de projet de POO

Thibault Poux et Albert Tomasi, groupe 11

Introduction :

Ce rapport vise à rendre compte du travail rendu dans le cadre du projet développé dans l'UE Programmation Orientée Objet 3. Ce projet consistait à créer deux jeux basés sur un fonctionnement similaire : le jeu de Carcassonne et le jeu de Dominos ; en modularisant un maximum ce qui pouvait l'être.

Pour répondre à l'objectif donné, notre binôme a travaillé avec le logiciel de gestion de versions GIT, dont le dépôt en ligne est disponible à cette adresse : https://gaufre.informatique.univ-paris-diderot.fr/poux/projet-de-pooig

Le développement du projet a nécessité jusqu'à présent la création de 9 branches différentes et plus de 200 commits. Il a permis d'arriver à un stade où tous les objectifs essentiels et majeurs ont été atteints.

Cahier des charges :

Tous les éléments du cahier des charges minimal ont été traités dans notre projet. Ce dernier contient donc :

- Un **environnement de jeu** réalisant l'accueil de l'utilisateur. Cet environnement permet le choix du jeu (Dominos ou Carcassonne), le choix du nombre de joueurs ainsi que l'entrée de leur nom, et la sélection du paramètre « IA » pour chaque joueur. Il est également possible depuis le menu affiché avant l'interface de jeu de charger une partie précédemment sauvegardée.
- Une **implémentation totale des règles du Domino**. Il est possible de jouer une partie de Domino du début jusqu'à la fin avec toutes les règles qu'elle comprend.
- Une **implémentation partielle du jeu de Carcassonne**. Seules les règles de comptage des points et de contraintes du plaçage des pions n'ont pas été implémentées.

Des fonctionnalités avancées ont été implémentées, parmi lesquelles :

- **Sauvegardes.** Il est possible à tout moment d'une partie en cours de la sauvegarder grâce à un bouton « save » placé en haut à droite de l'interface graphique. Cette sauvegarde prendra le nom qu'on avait donné à la partie en cours, sera retrouvable et reproduira à l'identique la partie au moment de la sauvegarde si elle est chargée depuis le menu. Une fermeture de la fenêtre de jeu ou un retour au menu prématurés (avant fin de la partie) entraînent une sauvegarde automatique.
- **HAL 9000.** L'IA fait le choix du placement de sa tuile en fonction de la case qui lui rapporte le plus de points. Le comptage de points n'ayant été implémenté pour le moment que pour le jeu Dominos, l'IA est meilleure que le hasard uniquement dans ce jeu. Cependant, son intelligence sera aussi effective dans le jeu Carcassonne dès lors que toutes ses règles auront été implémentées, car le fonctionnement de l'IA est le même pour les deux jeux.
- **Aide au placement.** Lorsqu'un utilisateur déplace sa souris sur les cases de la grille de jeu, ces dernières voient leur bordure devenir rouges ou vertes suivant la validité du placement de la tuile courante sur ces cases.

Des ajouts mineurs ont été implémentés dans l'interface graphique, parmi lesquels :

- **Tuiles restantes et infos.** L'interface de jeu affiche au fur et à mesure de la partie le nombre de tuiles restantes dans la pioche et un bouton d'informations faisant apparaître un lien vers les règles du jeu en cours.
- **Quitter et retour.** Dans toute interface de l'environnement de jeu se trouvent un bouton « quitter » représenté par une croix et un bouton « retour » représenté par une flèche, qui permettent respectivement de fermer la fenêtre de jeu et de retourner à l'interface précédente. Utilisés depuis l'interface de jeu, les boutons « quitter » et « retour » entraînent respectivement une fermeture de la fenêtre du jeu et un retour au menu, et entraînent tous deux une sauvegarde automatique de la partie en cours si elle n'est pas finie.

Problèmes connus :

Notre binôme a rencontré de nombreux problèmes durant le développement du projet, parmi lesquels un mauvais comportement de l'interface graphique à de nombreuses reprises, une difficulté à sauvegarder les parties, un comportement destructeur et incohérent de l'IA, des difficultés de constitution et de sauvegarde de la pioche du jeu de Carcassonne (tuiles avec images), des difficultés à ajuster la taille de la grille de jeu en fonction du nombre de tuiles posées et de leurs emplacements, etc. Ces problèmes ont été aujourd'hui résolus et l'environnement de jeu est a priori raisonnablement robuste aux bugs et aux erreurs.

Le problème principal rencontré à l'heure actuelle de développement du projet est un temps de chargement long du jeu, particulièrement pour le jeu de Carcassonne, qui charge de nombreuses images.

Pistes d'extensions :

La première piste d'extension que notre binôme peut suivre est l'implémentation de l'intégralité des règles du jeu de Carcassonne, ce qui permettrait de jouer à une partie de ce jeu jusqu'à la fin en profitant de la fonctionnalité de l'interface graphique.

Une autre piste d'extension peut être l'ajout de nouvelles tuiles dans la pioche initiale du jeu de Carcassonne, ce qui ne pose pas de problème technique ; il suffirait pour cela de disposer de nouveaux modèles de tuiles et d'images correspondantes.

Une autre amélioration serait d'optimiser le temps de chargement du jeu, peut-être en utilisant la fonction SwingUtilities.invokeLater sur les lignes de codes qui génèrent beaucoup des calculs, ce qui n'a pas été étudié pour l'instant.

D'autres pistes d'extensions mineures ont été laissées en commentaires dans le code du projet sous forme de TODOS, comme la possibilité de déplacer la tuile courante avec la souris jusqu'à un emplacement sur la grille (et pas uniquement de cliquer sur la case d'arrivée pour pouvoir la placer) ou l'ajout de spécificités aux vues des deux jeux, qui ont pour l'instant des vues presque identiques.

Architecture du projet :

Le projet contient en tout 21 classes et est divisé en trois répertoires :

- Un premier répertoire modulaire intitulé JeuTuilesGenerique qui implémente le fonctionnement d'un jeu de tuile général le plus précisément possible, et dont les classes sont étendues dans les deux autres répertoires. JeuTuilesGenerique contient deux sous-répertoires : Modele (6 classes) et Vue (4 classes).
- Un deuxième répertoire supportant le jeu de Carcassonne intitulé JeuCarcassonne (5 classes).
- Un troisième répertoire supportant le jeu de Dominos intitulé JeuDominos (5 + 1 classe pour l'affichage du jeu sur le terminal).

Voir en annexe le schéma représentatif des classes du projet et de leurs relations pour plus de précisions.

Détails des classes de JeuTuilesGenerique :

Le répertoire JeuTuilesGenerique contient les classes les plus essentielles du projet, qui sont ensuite étendues dans JeuDominos et JeuCarcassonne, et qui seraient étendues les répertoires correspondants à d'autres jeux à tuiles s'il était décidé d'en créer d'autres. Voici le détail des classes qu'il contient :

- Modele :
    - Tuile.java : Modélise une tuile. Une tuile peut réagir aux actions de la souris dessus, donc cette classe implémente MouseInputListener. Contient notamment la fonction rotate qui permet de faire tourner une tuile dans un sens ou dans l'autre.
    - Bord.java : Modélise un bord d'une tuile, et contient notamment la fonction « estCompatibleAvec » qui permet de savoir si deux bords sont compatibles.
    - Plateau.java : Modélise le plateau de jeu, représenté par un tableau de tableaux de tuiles. Ce dernier est en permanence mis à jour au fur et à mesure de la partie en fonction de la nécessité d'agrandir le plateau.
    - Pioche.java : Modélise la pioche du jeu, c'est-à-dire l'ensemble des tuiles qui peuvent être posées au cours de la partie. La partie s'arrête lorsque la pioche ne contient plus aucune tuile.
    - Joueurs.java : Regroupe l'ensemble des joueurs qui participent à la partie. Contient la classe interne Joueur qui modélise un joueur.
    - Partie.java : Classe centrale, connecte toutes les classes précédentes, et contient les fonctions qui représentent le cœur du fonctionnement d'un jeu, comme « check » qui permet de savoir si une tuile est plaçable à un certain endroit, « jouer » qui permet jouer une tuile, ou encore « recursiveIA » qui permet d'obtenir un coup de l'IA.
- Vue :
    - ButtonImage.java : Modélise un bouton qui contient une image.
    - Menu.java : Représente tout le menu avec les différentes interfaces qu'il contient et les nombreux boutons et entrées de textes qui s'y trouvent.
    - GameView.java : Représente la vue de l'interface de jeu, ce qui comprend la grille sur laquelle on pose les tuiles, les panneaux consacrés aux informations des joueurs, un panneau consacré à la tuile courante, etc.
    - Launcher.java : Classe comportant la fonction main, et qui permet de tout lancer : soit l'ensemble de l'environnement de jeu avec le menu, soit directement une partie Carcassonne, une partie Dominos ou une partie sauvegardée.

Architecture Projet