PROJET DE POOIG

Les Colons de Catane

CAHIER DES CHARGES

- Nous avons commencé par la spécification du programme en nous demandant quelles sont les classes que nous avons besoin pour jouer une partie en mode textuelle. Nous nous sommes donc mis d'accord sur l'héritage des classes comme par exemple PremierTour et TourNormal qui étendent la classe abstraite Tour. C'est ainsi que nous avons entamé la phase de conception architecturale de notre projet. (Voir schéma dans Schéma.pdf)
- Nous avons donc décidé d'adapter le plateau Catan en plateau de différentes cases et de leurs usages.
- Nous avons ensuite enchainé avec les algorithmes des méthodes de chaque classe en s'aidant des diagrammes de « flowchart » dans un premier temps avant d'implémenter le code en JAVA.
- Après avoir créé le jeu de base jouable pour des humains nous nous sommes mis à créer les différentes options de jeux : échange avec port, voleurs, jouer des cartes développements …
- Une foi le jeu en mode textuelle fini et testé pour les humains nous avons décidé d'ajouter la possibilité de jouer avec l'intelligence artificiel. En adaptant la fonction de départ.
- Nous avons ensuite enchainé avec la création de la base de l'interface graphique en y ajoutant le jeu de base.
- Puis tout comme la conception du jeu en mode textuelle, après l'avoir testé, nous avons ajouté au fur et à mesure les options de jeux, en améliorant la « vue » pour un meilleur affichage.
- Une foi celle-ci fonctionnelle et testé nous nous lançâmes dans la conception de l'intelligence artificielle.

PROBLEMES CONNUS

1. Les bots pouvaient se bloquer dans une boucle infinie en voulant placer une colonie ou une route quand c'était infaisable et donc, nous avons rajouté la condition qu'une colonie puisse être placée au premier tour uniquement à condition qu'une route pourra être placée juste après. Ensuite, pendant un tour normal, on a rajouté un nombre d'essaies limité pour les bots qui veulent rajouter une colonie/route alors que c'est impossible.
2. L'algorithme servant à trouver le joueur ayant la route la plus longue contenait souvent des bugs et nous avons dû l'améliorer à maintes reprises tout en faisant des tests avant de pouvoir obtenir une version finale.
3. Lors de l'implémentation de l'interface graphique, nous avions remarqué que l'utilisateur ne pouvait plus interagir avec l'interface graphique tant qu'il y avait du code qui s'exécutait dans d'autres méthodes et donc on ne pouvait pas enchainer des tours dans des boucles « while » ou « for ». La solution a été de créer de nouvelles méthodes qui termine leur exécution dès qu'on attende une interaction entre l'utilisateur et l'interface graphique avant de pouvoir, par exemple, placer une colonie. Nous avons rajouté un booléen indiquant qu'un joueur a terminé son tour avant d'enchainer avec le tour d'un autre.
4. Les boutons des cases ressources étaient grisés quand ils étaient désactivés et cela rendait l'affichage moins beau. La solution était de définir la même image pour le bouton ressource quand ce dernier est désactivé et cela nous a donné l'idée de griser la case ressource qui contient le voleur.

EXTENSIONS NON-IMPLEMENTEES

1. Choisir la taille du plateau.
2. Négoce.
3. Ressource mine d'or : Choisir la ressource de votre choix si vous devez toucher une mine d'or.
4. Possibilité de transformer ville en mégapole (toucher 3 ressources).
5. Possibilité de créer un village avec 3 ressources qui sera transformable en colonie en utilisant 3 ressources. Un village touche les ressources uniquement s'il est situé à côté d'une case ressource numéroté 2, 3, 4, 10, 11, 12 (Nombres sortant moins fréquemment aux dés).
6. Ajout d'une carte de développement qui donne la possibilité de s'attribuer un port choisi au hasard.
7. Possibilité de choisir les cartes ressources à défausser sur l'interface graphique quand il y a un 7 aux dés.
8. Créer des bots utilisant des bonnes stratégies.

TESTS

- Lors de l'implémentation de notre code, nous réalisions des tests unitaires pour chaque nouvelle méthode ajoutée et ensuite des tests d'intégration afin de s'assurer que la méthode en question ne génère pas de bugs en interagissant avec le reste du code déjà présent.
- En faisant jouer seuls les bots plusieurs erreurs ont été révélées, nous indiquant toutes les failles de notre code et à force de les corriger au fur et à mesure, il n'y avait plus de bugs apparents.
- Pendant l'implémentation de l'interface graphique nous réalisions des mini parties afin de déterminer tout ce qui pouvait faire bugguer notre logiciel. Par exemple, pour le premier tour, nous testions la pose de colonie suivie de la pose de route et c'est ainsi que nous avons remarqué que le blocage en posant une colonie comme cité précédemment.
- Nous avons ensuite enchainé avec des tests de boites blanches et noires avec des piles de tests pour savoir si le logiciel fonctionne correctement dans l'ensemble.

COMMENT FONCTIONNE LE CODE ?

Tout d'abord, quand l'utilisateur exécute Jouer.java, il devra faire un choix dépendant de s'il veut jouer en mode textuel ou en interface graphique.

Dès que la partie est lancée, une nouvelle partie est créée et l'utilisateur doit alors paramétrer 3 ou 4 Joueur.

Un Joueur est composé d'un tableau d'entiers qui correspond respectivement au nombre de ressources qu'il possède pour chaque ressource : [ARGILE, BOIS, CHAMPS, MOUTON, PIERRE]. Il contient aussi un ArrayList de CarteDev, un ArrayList de String pour ses ports, …

Un plateau de taille 9 x 9 est créé tout en générant des cases vides pour les futures cases colonies (transformables en villes à travers un booléen) et routes ainsi que les cases paysages contenant un booléen définissant si elle contient le voleur. Les cases paysages sont divisées en 2 : la case du désert et les cases ressources. Les cases ressources contiennent un numéro attribué au hasard parmi une liste biaisée de numéro ainsi qu'un string indiquant la ressource qu'elle contient. Les cases routes et colonies possèdent respectivement des méthodes qui leur permettent de poser des routes et des colonies en affectant false au booléen estVide et en s'attribuant le joueur qui possède la case.

Alors se lance une suite de PremierTour dans lesquels chaque joueur pourra choisir une case valide de son choix grâce au Scanner s'il est en mode textuel ou en cliquant sur un bouton actif en mode IG où les fonctions ajouterColonie et ajouterRoute (surchargées dépendant de textuel, IG ou IA) poseront une colonie ou route du joueur s'il le peut. Dans le cas contraire, il devra réessayer.

Ensuite se lance une succession de TourNormal (tant que l'entier point de chaque Joueur ne dépasse pas 10) dans lesquelles chaque joueur pourra réaliser les différentes options possibles de son choix.

Dans Tour, qui englobe PremierTour et TourNormal, est stocké le joueur courant, le plateau courant, les 25 CarteDev en tant qu'ArrayList et d'autres information qui évoluent pendant la partie (voir les champs dans la classe Tour).

TourNormal contient lui de nombreuses méthodes (variant selon le mode textuel ou IG ainsi que le joueur étant IA ou humain) permettant de réaliser les différentes actions possibles d'un Joueur pendant son TourNormal comme ajouterVille qui affecte true à estVille dans la CaseColonie choisie si le Joueur possède la CaseColonie et dispose des ressources nécessaires. En mode Textuel, le 'c' sera remplacé par 'v' et sur l'IG, l'ImageIcon du bouton change.