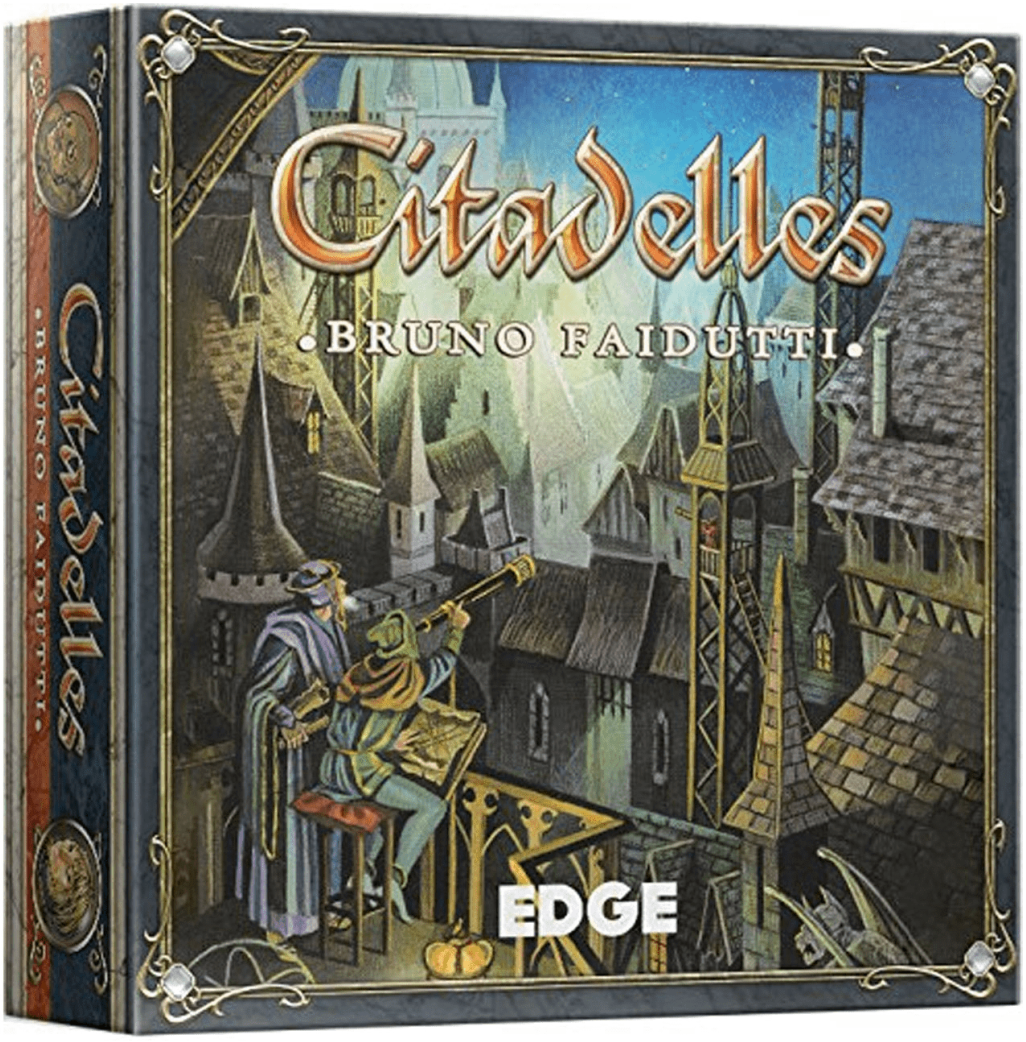
RAPPORT DE PROJET :
Citadelle



Présenté par l'équipe **JAVABIEN**

BABOURI Sofiane - DOMINGUEZ Lucas - FRANCIS Anas - MADRIAS Julien - POULAIN Timothée

I - Synthèse du Projet

Structure du projet

La gestion des parties est effectuée à l'aide de la classe **GameLoader,** elle permet de lancer une ou plusieurs parties avec les mêmes robots et de récupérer les statistiques de victoire de ces robots. Si une seule partie est lancée alors un **logger** affichera le déroulement de la partie.

La classe **GameEngine** gère le déroulement d'une partie, il permet au les différentes actions possibles au cours d'un tour (piocher, prendre de l'argent, …)

Les bâtiments sont stockés dans un fichier json, ainsi il est simple d'implémenter de nouveaux bâtiments (sans capacités spéciales) ou encore de changer le set de cartes pour le remplacer, par exemple, par celui d'une extension.

Déroulement d'une partie

Au début de la partie, on définit combien de **cartes personnages** doivent être retirées lors du choix du personnage (ce nombre dépend du nombre de joueurs) ensuite le premier tour commence.

Chaque tour se déroule de la même manière. On mélange les personnages puis on retire le nombre préétabli de ces personnages. Le joueur qui a la **couronne** choisie en premier (si il n'y a pas de couronne c'est le joueur qui est le premier dans la liste qui commence) puis les autres choisissent chacun leur tour dans la liste des personnages restant. Ensuite le **GameEngine** appel chaque joueur selon l'ordre de priorité de son rôle (l'assassin en premier, puis le voleur, …).

Le joueur appelé devient le joueur actif et peut pendant son tour utilisé le pouvoir de son personnage (s'il en a un), piocher de l'argent ou des cartes et construire un bâtiment.

La partie se termine quand le premier tour où l'un des joueurs construit huit bâtiments s'achève.

Le gagnant est celui qui a le plus de points à la fin de la partie.

Eléments de jeu implémentés

Tout d'abord, notre projet couvre toute les fonctionnalités du jeu de base sans extension, à savoir :

-Les huits personnages et leurs pouvoirs

-Tous les bâtiments du jeu de bases et les pouvoirs des merveilles

-La couronne est la priorité qu'elle donne à celui qui la possède

Les robots

Nous avons conçu au total 4 robots (le robot basique, le robot jaloux, le robot intelligent et le robot capitaliste)

Le robot basique : (ou le **BasicBot**) ses actions sont très basiques pour le choix des personnages il va choisir le 1er personnage qui trouve dans la pioche des personnages.

Le robot jaloux : (ou **JealousBot**) Ce robot a pour stratégie d'ennuyer les autres joueurs. Il va essayer de prendre des personnages qui ont des pouvoirs sur les autres joueurs (comme Condottiere, Assassin et Magicien) en priorité.

Le robot intelligent : (ou le **MentalistBot**): Ce robot détermine une cible parmi ses adversaires, en choisissant en priorité le joueur avec le plus de bâtiments construits. En cas d'égalité, il choisit celui qui a le plus de pièces d'or. Ensuite, pour prendre des décisions, il agit selon 8 scénarios différents en fonction de la différence de cartes posées, de pièces d'or et du nombre de cartes dans la main du mentaliste et de sa cible.

Le robot capitaliste : (ou **UncleSamBot**) Ce robot a pour stratégie de maximiser ses gains. Il va toujours essayer de prendre des personnages qui peuvent apporter (comme Roi, Marchand, etc.).

II - Qualité du code

Des valeurs constantes ont été implémentées dans le code afin de pouvoir modifier, en cas de changement, depuis un seul endroit.

Les tests réalisés sur nos robots testent leur comportement dans plusieurs scénarios différents ainsi qu'à différents moments de la partie, ce qui les rend fiables.

Cependant, il reste de rares cas qui n'ont pas été testés, cela par manque de temps.

Les tests qui ont été réalisés sont suffisants pour assurer que les robots appliquent leurs stratégies, après avoir testés chacun d'entre eux et joué un grand nombre de parties, nous pouvons attester qu'ils suivent leurs stratégies, mais nous ne pouvons pas certifier à 100% que sur un nombre infini de partie, ceux-ci continueraient de se comporter comme prévu.

Le **MentalistBot** n'a pas été testé en totalité car implémenté en dernier, en effet, ce robot est le plus ambitieux parmi toutes les IAs que nous avons créé et nous n'avons malheureusement pas eu le temps de réaliser la totalité des tests possibles pour veiller au bon déroulement de son comportement.

III - Pourquoi notre projet est un bon projet

Les raisons qui nous poussent à dire que notre projet est un bon projet sont les suivantes. Le jeu bien que non complet à 100% répond aux exigences requises du cahier des charges. En ce qui concerne le découpage, notre équipe a su bien découper le projet en petites tâches chaque semaine, réalisables et utiles au bon fonctionnement

du jeu. De plus, lorsque nous pensions manquer de temps pour réaliser une milestone, notre équipe planifiait à nouveau les milestones suivantes pour rendre réaliste notre progression.

En ce qui concerne notre code, il a toujours été très bien commenté tout du long afin de permettre un suivi facile pour chacun d'entre nous, le but étant de pouvoir travailler à plusieurs sur les mêmes portions de code. Nous avons donc rendu nos méthodes et classes les plus courtes possible pour une meilleure relecture.

La démonstration de notre projet n'était au début que très peu intéressante, ne proposant aucunes statistiques et uniquement une succession de lignes indiquant les informations sur l'état de la partie, cependant, nous avons su tenir compte du feedback qui nous a été donné afin de donner un meilleur visuel à notre démo qui ajoute une plus-value à notre projet.

Enfin, en ce qui concerne nos livraisons, nous avons tenu à être les plus réguliers possibles afin de réussir à tenir les délais qui nous avaient été imposés, pour cela, il nous a fallu bien nous organiser et nous répartir les tâches que nous avions mis en lumière.

IV - Pourquoi notre projet est un mauvais projet

Il existe des limites à notre modélisation du jeu de Citadelles, certaines fonctionnalités n'ont pas été implémentées comme les règles à 3 ou 8 joueurs. Au niveau des bots, bien qu'ils aient des intelligences leur permettant d'avoir des stratégies évoluées, nous n'avons pas pu optimiser et développer une intelligence ayant une très grande capacité d'analyse du jeu pour battre les autres bots.

La structure du programme laisse des failles de sécurité, possiblement à cause des responsabilités et des niveaux de visibilité des méthodes. Cette structure, bien qu'adaptée au jeu, pourrait s'avérer coûteuse à modifier lors d'une future extension (si trop différente du jeu originel).

V - Conclusion

Pour conclure, dans notre projet, nous avons implémenté la majorité des éléments de jeu, ainsi que plusieurs bots aux stratégies différentes. Seules manquent les règles du jeu à 3 ou 8 joueurs. Nos tests ont une couverture de plus de 80%, et notre code bien organisé et commenté, ce qui nous a permis de tous nous y retrouver.

Cependant, la structure globale aurait gagné à être mieux pensée dès le départ, de manière à implémenter plus facilement certaines fonctionnalités et limiter les failles de sécurité. La diversité des stratégies chez les bots nous permet d'obtenir des statistiques réalistes.