# PROGRAMMATION
# Rapport - PROJET – JEU MARVEL SNAP

BOUDEFLA DOUNIA

FATOUMBI Afdol

## INTRODUCTION

Marvel Snap est un jeu de cartes rapide dans l'univers Marvel, où les joueurs s'affrontent pour des cubes cosmiques dans des batailles 1 contre 1. Une partie se compose de 6 tours , et à chaque tour, chaque joueur peut jouer simultanément une ou plusieurs cartes sur l'un des trois emplacements disponibles. Chaque carte a une valeur de puissance qui, lorsqu'elle est jouée, contribue à la puissance totale du joueur à l'emplacement où la carte est placée. Le gagnant est déterminé par le joueur qui a la puissance totale la plus élevée dans deux des trois emplacements 1. Le plateau de jeu est composé de trois emplacements; chacun a une capacité unique qui peut affecter considérablement la stratégie de chaque joueur et où les cartes sont jouées à chaque tour. Le jeu commence avec un seul emplacement révélé, le deuxième et le troisième emplacements sont révélés aux tours deux et trois respectivement. Cependant, même si un emplacement n'a pas été révélé, les joueurs peuvent toujours jouer des cartes sur ces emplacements. Un maximum de quatre cartes peut être joué à n'importe quel emplacement.

## SPECIFICITE DE NOTRE PROJET

Dans notre projet, nous nous sommes principalement basés sur UTerminal pour l'affichage du jeu. Ainsi, nous avons implémenté toutes les classes abstraites contenues dans AbstractCard dans une classe Card qui contient le nom de la carte, sa valeur, son coût ainsi que sa position. Dans la classe Game qui constitue le modèle du jeu, nous avons utilisé les LinkedList d'AbstractCard ainsi que leurs propriétés d'ajout (add) et de retrait (remove) pour implémenter les méthodes contenues dans AbstractGame. En plus des méthodes prédéfinies de AbstractGame, nous avons ajouté des méthodes comme pioche et ClearPlay qui permettent respectivement de piocher une carte dans le deck du joueur pour l'ajouter à sa main et de vider les listes des coups en préparation des joueurs pour qu'elles soient vides au tour suivant. Nous avons rajouter également des noms aux cartes des joueurs ainsi que des effets révélés et des effets Continu.

Effet "Révélés Score +2" : Si une carte dotée de cet effet est placée à l'emplacement du milieu, le score du joueur qui l'a posée est augmenté de 2.

Effet "Révélés +1 Cartes" : Si une carte est posée avec cet effet, le joueur qui l'a posée ajoute une carte spéciale à sa main.

Effet "Révélés Destroy+2" : Si une carte est posée avec cet effet, elle détruit toutes les cartes du joueur sur cet emplacement. De plus, le joueur qui a posé la carte gagne un point de score pour chaque carte détruite.

Effet "Continu +1 Adversaire" : Pour chaque carte adverse sur cet emplacement, score +1.

Effet "Continu +1" : Score +1 pour chaque carte placée sur le même emplacement

Une propriété importante de notre programme est que lors de la déclaration des LinkedList<AbstractCard> (emplacement, cout_prep et score), nous avons utilisé un tableau de taille 2 de LinkedList<AbstractCard> (LinkedList<AbstractCard>[2][3]) dont la première partie stocke les sides des joueurs et la deuxième partie représente les positions LEFT, MIDDLE et RIGHT qui correspondent respectivement aux valeurs 0, 1 et 2 dans notre LinkedList.

En général, notre programme démarre normalement une partie du jeu avec UTerminal en demandant aux joueurs leurs noms et en leur attribuant un side de façon aléatoire. Ensuite, le programme procède à l'initialisation du deck de chaque joueur et pioche trois cartes du deck dans la main de chaque joueur. Il demande ensuite aux joueurs d'effectuer un choix de carte dans leur main qu'il affiche auparavant et leur demande la position à laquelle ils souhaitent poser la carte dans cout_prep. Lorsque les conditions sont valides pour que la carte puisse être positionnée, le coup est validé et la carte est déposée sur l'un des emplacements du jeu, puis on passe au tour suivant.

Le jeu s'arrête avec la fonction endGame du Controller lorsqu'on atteint le sixième tour du jeu et affiche le gagnant du jeu qui est celui dont le score est le plus élevé sur les deux plateaux.

## DIFFICULTÉS

Nous avons cependant rencontré certaines difficultés lors du déroulement de ce projet. Après avoir implémenté en binôme les classes Card et Game, nous avons essayé d'implémenter individuellement la classe Controller avec pour objectif d'avoir deux classes Controller, l'une utilisant UITerminal et l'autre UIGraphic. Malgré les efforts fournis de chaque côté, nous avons décidé de retenir la classe Controller utilisant UITerminal à cause des nombreux bugs de la classe Controller utilisant UIGraphic. La principale difficulté dans l'implémentation de la classe Controller utilisant UIGraphic venait surtout de la mise a jour du score des joueurs et des placements des cartes après avoir lancer le tour. Nous avons obtenu une classe Controller qui utilisait UIGraphic, mais qui n'arrivait pas correctement à mettre à jour le placement des cartes sur le plateau de jeu au fur et à mesure du jeu. Nous continuons cependant l’implémentation afin de pouvoir le présenter pour la soutenance.