Rapport de projet Python
Marvel Game

Dan Calamia, Judicael Corpet, Baptiste Rouquette

Décembre 2024

1 Introduction

L'objectif de ce rapport est de présenter les fonctionnalités implémentées dans le jeu réalisé, ainsi que les choix de conception qui ont conduit au diagramme de classes UML associé.

2 Fonctionnalités de base implémentées

2.1 Unités

L'objectif du jeu est de pouvoir adopter une réelle stratégie en fonction des personnages sélectionnés parmi une liste. Au total, 14 personnages différents ont été créés, autour de l'univers Marvel, offrant un large choix de composition d'équipes. Chaque personnage est ainsi caractérisé par **un nom, un nombre de points de vie, une vitesse de déplacement** (symbolisée par les cases vertes)**, une défense, une liste de compétences et une puissance d'attaque.**

2.2 Carte

La carte du jeu offre un gameplay simple mais répondant aux critères fixés par le projet, à savoir au moins 4 types de cases différentes. Ici nous avons choisi les 4 types suivants : **une case qui bloque les déplacements de certains joueurs, une case qui permet de récupérer de la vie, une case qui permet de se téléporter et une case qui retire des points de vie.**

2.3 Compétences

Plusieurs types de compétences ont été implémentés, caractérisés par le **nom, la distance d'attaque, l'effet** (soigner, briser les défenses, retirer des points de vie) **leur portée (symbolisée par les cases rouges)**. Au total, chaque personnage est doté d'au moins 2 compétences, sur un total de 17 compétences différentes.

2.4 Calcul des dégâts

Les dégâts causés lors des attaques ont été calculés en fonction de plusieurs critères : **la puissance du personnage, la compétence utilisée, la capacité de défense du personnage ciblé et la défense.**

3 Fonctionnalités supplémentaires implémentées

En plus des fonctionnalités de base, il était demandé d'ajouter au moins 2 fonctionnalités supplémentaires. Nous avons choisi d'ajouter un menu complet, une mobilité spécifique pour les personnages volants (Iron-man, Thor et Torch) et intégration d'une précision dans les coups d'attaque. Enfin, une IA a été générée pour jouer en solo contre l'ordinateur.

3.1 Menu

Une vidéo de présentation ainsi qu'un menu ont été créés afin de donner à l'utilisateur un accueil convivial, permettant de le plonger directement dans l'ambiance du jeu vidéo. Ce menu permet, entre autre, de sélectionner un mode avec ou sans son, un mode de jeu en solo ou à 2, puis de sélectionner les personnages qui feront partie de son équipe. Enfin, c'est le point de départ pour lancer le jeu.

3.2 Mobilité spécifique

Plusieurs unités ont la capacité de voler. leur vitesse de déplacement est donc augmentée, le nombre de cases vertes est supérieur aux autres personnages, mais ils peuvent également survoler les obstacle, à l'exception des cases dégâts .

3.3 Intégration de la précision

Lors des attaques, un paramètre de précision, prenant en compte la chance générée avec un score aléatoire, et la distance entre la cible et l'attaquant, permet de générer un attaque avec des dégâts variés.

4 Choix de conception des diagramme de classes

Au total, **45 classes ont été créées.** Le choix des relations entre ces classes, expliqué dans cette partie, a permis une bonne répartition des tâches au sein de notre équipe et a facilité la construction progressive du jeu.

4.1 Relations d'héritage

Plusieurs classes ont fait l'objet de relations d'héritage afin d'éviter les redondances et les confusions. Par exemple, les classes des personnages ont hérité de la classe Unit pour reprendre notamment la position ou la taille.

4.2 Relation d'association simple

Chaque tour de joueur a fait appel à l'association simple pour notamment décrire les actions à mener en appelant des classes distinctes qui n'ont pas de relation particulière. Ces associations simple se retrouvent par exemple dans la méthode "handle player turn" de la classe "Game".

4.3 Relation d'agrégation

Cette relation a été utilisée dans la fonction "main" notamment pour créer les listes de joueurs ou d'ennemis une fois ceux-ci sélectionnés à partir du menu. En effet, les personnages existent même si cette liste n'est pas établie, ce qui définit bien la relation d'agrégation. Cette relation était nécessaire pour garantir la création d'une nouvelle liste à chaque nouvelle partie.

4.4 Relation de composition

Cette relation a été mise en place notamment lorsqu'il s'agissait de faire appel à une attaque précise, car nous devions la relier à un personnage. La relation de composition a donc été nécessaire notamment dans la méthode "handle player turn" par exemple, dans la classe "Game", pour identifier l'attaque utilisée.

5 Conclusion

Ce jeu réponds au cahier des charges imposé avec les fonctionnalités demandées implémentées, des fonctionnalités supplémentaires, une logique de construction et de lecture permettant d'améliorer le jeu. En bonus, un Menu complet a été élaboré, garantissant un immersion immédiate du joueur dans l'univers Marvel.