Conception agile de projets informatiques et génie logiciel

KARFA Hichem - KOLLI Lucas

27/11/2022

Image réaliser avec l'IA Midjounney

Contents

1 Présentation globale du projet **3**
1.1 Comment fonctionne le jeu 3
1.2 Comportement de l'IA 3
1.3 Partie future 3

2 Motivation et explications des choix **4**
2.1 Description rapide des classes 4
2.2 Choix d'architecture et diagrammes 5
2.3 Designs Patterns ? 6
2.3.1 Designs Patterns utilisés 6
2.3.2 Explication des designs patterns 6

1 Présentation globale du projet

1.1 Comment fonctionne le jeu

Ce projet java prend la forme d'un jeu de balle au prisonnier. Il se joue à 2 joueurs sur le même clavier. Chaque joueur est accompagné de 4 autres personnages contrôlés informatiquement (appelé Bots). Lorsqu'un joueur à la balle, il tir vers l'équipe adverse. S'il touche un adversaire, celui-ci perd un points de vie. Lorsqu'un joueur n'a plus de point de vie au cours de la partie, il perd et disparait. La partie s'arrête lorsqu'un des joueurs principaux est éliminé. Si le tir est raté, la balle revient au joueur principal adverse et la partie continue. Un joueur peut effectuer une passe en visant un de ses allié. Passivement, la balle va de plus en plus vite, il faut donc essayer d'éliminer rapidement les bots avant d'affronter le joueur principal en face.

1.2 Comportement de l'IA

Il existe plusieurs niveaux de difficulté du jeu. On peut par exemple changer le mode de déplacement des bots. Il existe pour l'instant 2 modes de déplacement. Un mode normal, ou chaque bots se déplace régulièrement de gauche à droite dans un espace donné. Dans ce mode de déplacement, il y a des feintes. Elles permetent aux bots de changer brusquement de direction. Il est possible de rendre le jeu plus compliqué en augmentant le taux de feinte (donc en diminuant la variable feinte initialisé à 50). L'autre mode de déplacement des bots permet aux deux joueurs de contrôler leurs bots. Chaque bot se déplace de la même manière que le joueur principal, on bouge donc toute l'équipe d'un coup à l'image d'un baby-foot. Les bots tirs lorsqu'ils ont la balle, avec un angle aléatoire entre 70° et 190°.

1.3 Partie future

Concernant les changements et amélioration que nous aurions aimé faire, il y a un système de parade. Le joueur devrait appuyer sur une touche au moment ou il se ferait toucher avec un timing bien précis, afin de ne pas perdre de point de vie. Cela nous donnerai un levier supplémentaire afin de gérer la difficulté du jeu en ajoutant aux bots cette fonctionnalité qui serait plus ou moins précise.

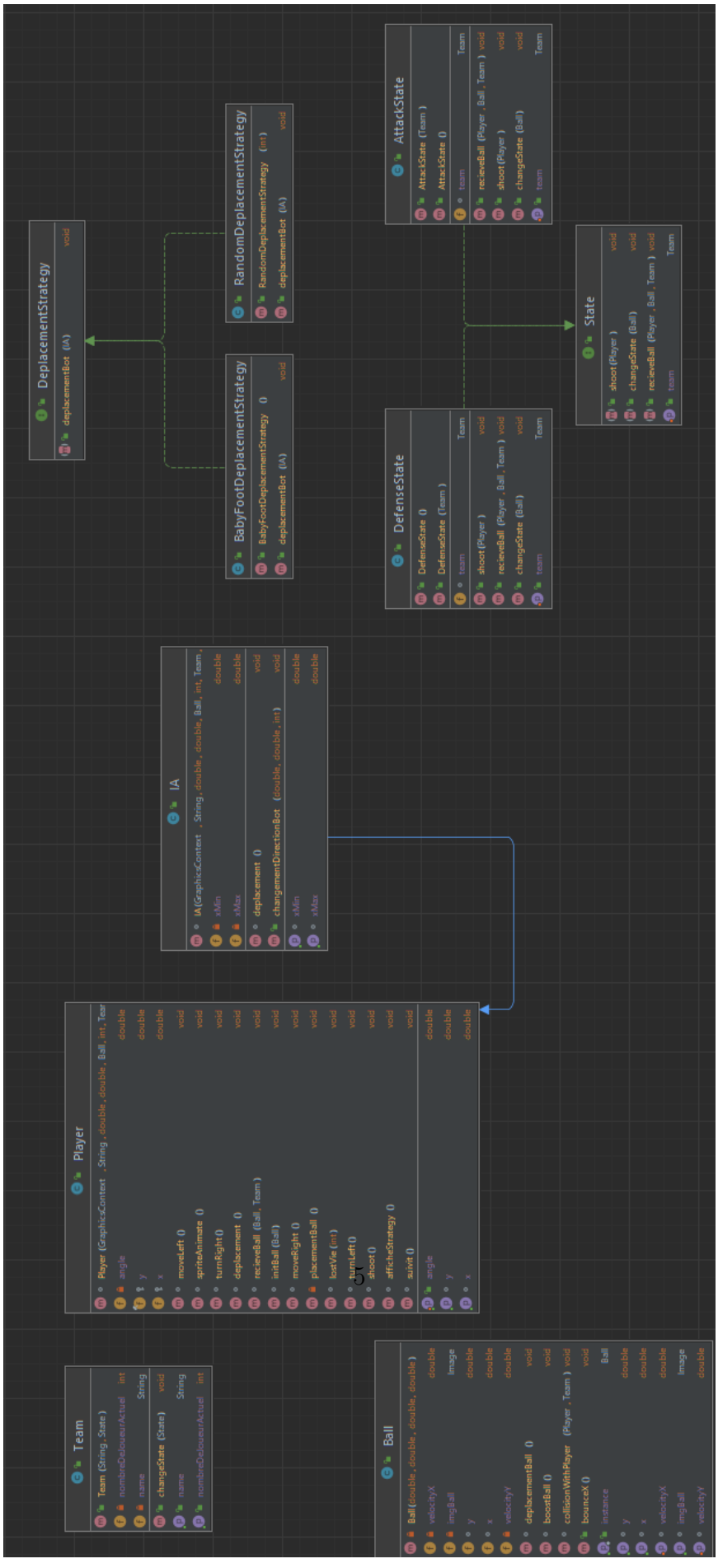
2 Motivation et explications des choix

2.1 Description rapide des classes

Nous allons tout d'abord décrire les classes :

- App : Créer la scène et lance l'application
- Field : S'occupe de gérer tout le jeu, création des joueurs etc...
- Display : Gère les affichages, des joueurs, des labels.
- LabelGestion : Construction des labels.
- Sprite : Animation des joueurs et de leur mort.
- Player : Caractéristiques des joueurs ainsi que les actions liées aux joueurs.
- IA : hérite de Player, afin de faire le même travail mais avec les bots.
- Ball : Classe qui a pour objectif de s'occuper à part entière de la balle.
- Team : Utilise les joueurs et les bots pour former les équipes.
- DéplacementStrategy : L'IA de déplacement, il y a une classe qui hérite de celle-ci par mode de déplacement (2 ici).
- State : État des joueurs, afin de décider les actions à faire. Deux classes qui hérite de celle-ci pour l'état attaque et l'état défense.

2.2 Choix d'architecture et diagrammes



A cause d'une mauvaise gestion de notre temps, nous n'avons pas pu utiliser le patron de conception MVC (Modèle, Vue, Contrôleur).Cela aurait pu nous être très bénéfique. En effet, répartir nos classe avec un but bien précis permetterait une bien meilleur structuration de nos classes. Comme vu au dessus, l'architecture de notre application est assez complexe. Il y a énormément de dépendance entre les classes. Grace au MVC, nous aurions pu réduire grandement ces dépendances. Les classes que nous avons créer fonctionnenerait bien dans un MCV, car le travail de ces classes ressemble a ce patron de conception. Premierement, les classes d'affichage pourrait être dans les Vues. La classe Field seraît le contrôleur et les classes de création (Player, IA, Ball etc) le modèle. C'est un patron de conception qui s'adapte bien à ce genre de projet et qui aurait été un gain de temps. En revanche, d'autre patron de conception nous on aidé à structurer notre application.

2.3 Designs Patterns ?

2.3.1 Designs Patterns utilisés

Afin de structurer l'application, nous avons utilisé le patron de conception Singleton. Pour ce qui est de la stratégie utiliser, notamment avec les déplacements des IA, nous avons utilisé le Design pattern Strategy. Enfin, pour les divers états des joueurs (attaque, défense, en possession ou non de la balle, en train de tirer etc), le design pattern utilisé est State.

2.3.2 Explication des designs patterns