2I013 : Projet Foot

Rapport

02 Mai 2017

Kevin SINGH
Samy ALI AHMED MAGID

Groupe 1
M Nicolas BASKIOTIS

2I013 : Projet Foot

Rapport

Table des matières

RAPPORT

Introduction

Cette unité d'enseignement nous a introduit à l'apprentissage statistique et l'intelligence artificielle. Nous avons tout d'abord découvert la plateforme de simulation du jeu avant de développer les différentes stratégies dans le but de créer la meilleure IA possible pour les joueurs.

Stratégies

Selon les différentes possibilités de jeu (1 vs 1, 2 vs 2, 4 vs 4) nous avons adopté une stratégie spécifique.

1 vs 1

Pour le 1 vs 1, nous avons codé un joueur qui était un « fonceur ». Cette stratégie permet au joueur d'avancer vers le ballon et de tirer vers les buts. Au cours des séances, nous avons pensé à la possibilité de l'améliorer le plus possible et nous sommes donc arrivés à coder un joueur qui pouvait « dribbler », c'est à dire un joueur qui pouvait avancer avec le ballon et tirer vers les buts à partir d'une certaine distance que nous avons choisie. Pour cela, il suffisait de le faire tirer des « mini-shoot » qui lui permettait d'avancer avec le ballon et déclencher un tir à partir de la distance optimale que nous avions obtenue grâce à l'observer. (cf. la partie Optimisation des paramètres)

2 vs 2

Pour le 2 vs 2, nous avons choisi un gardien et un attaquant.

Le gardien est placé assez proche de ses buts et a une zone d'action limitée. En effet, lorsque le ballon passait la moitié de terrain, notre gardien commençait à se placer en fonction de la position de la balle sur le demi-terrain et s'alignait sur la ligne horizontale où il était de base. Nous avions coupé le demi-terrain en 3 parties : haut, milieu, bas.

Lorsqu'il interceptait la balle à l'attaquant, nous avions au début pensé à le « transformer » en attaquant pour qu'il aille marquer lui-même. Mais nous avons vite abandonné cette idée car trop compliquée à réaliser.

Nous nous sommes donc contentés du fait que le gardien tire le plus loin possible ; mais cette solution n'était pas la bonne car le ballon arrivait rarement à l'attaquant. Nous avons donc amélioré la fonction passe() grâce à l'Observer.

4 vs 4

Pour le 4 vs 4, nous avons opté pour une équipe composée d'un gardien, d'un milieu et deux attaquants.

Le gardien et les attaquants sont inchangés par rapport au 2 vs 2 exceptés leurs positions sur le terrain. Le gardien est placé un peu plus bas sur le terrain et les attaquants sont un peu plus haut.

Le milieu est quant à lui un « gardien » qui est placé au quart de terrain, constituant ainsi le premier rideau défensif. Nous avons amélioré la fonction passe() qui cette fois-ci permet de trier la position des tous les coéquipiers du plus éloigné au plus proche et de faire la passe au joueur le plus éloigné.

Organisation du code

Au cours des deux premières séances, nous avions un fichier unique qui contenait toutes les fonctions. Nous nous sommes vite rendus compte qu'il s'agissait là d'une grande erreur et avons réorganisé notre code de façon à avoir toutes les stratégies dans un fichier (strategy.py), toutes les fonctions qui permettent de jouer avec ses stratégies dans une toolbox (test_bis.py). Cette toolbox étant divisée en 2 classes: la classe Position qui définit la position de tous les objets du jeu (joueurs, placement des joueurs, buts) et une classe Action qui contient des fonctions qui sont des briques de base ; qui en s'additionnant ont permis de créer nos stratégies. Nous avions 5 fichiers principaux :

- Test_bis.py qui était notre toolbox
- Strategy.py qui contenait toutes les stratégies créées
- Simulation_attaquant et simulation_gardien qui étaient utilisés en tant qu'observer
- Arbre.py qui avec ses dérivés permettait de créer l'arbre.

Difficultés et résultats

Nous avons eu des difficultés au début du projet. En effet, ayant oublié les fonctionnalités du langage Pyhton, nous avons dû nous remettre à niveau en suivant des cours et nous inspirant de l'unité d'enseignement que nous avions suivi en L1. De plus, nous avons ajouté à ces cours, nos connaissances en langage objet acquises durant l'unité d'enseignement en Java que nous avions le semestre précédent. Une fois ce bagage de connaissances de nouveau acquis, nous étions plus à l'aise avec la plateforme de simulation et nous avons mieux cerné ce qu'on attendait de nous. C'est à ce moment-là que nous avons codé notre toolbox et défini les stratégies de base qui nous ont permis d'avancer rapidement. Enfin, nous nous sommes inspirés du Design Pattern Decorator pour notre code mais malgré le fait que nous avions pensé à de bonnes stratégies, il y avait un écart entre la théorie et la pratique. De plus, les résultats n'étaient pas ceux attendus, il fallait donc trouver une façon d'améliorer notre code.

Améliorations

Une fois nos stratégies de base faites, le vrai travail commence alors. L'amélioration et l'optimisation de celles-ci.

Améliorations générales

Dans un premier temps, seules deux actions étaient réalisées par nos joueurs aller vers le ballon et tirer en permanence vers le but adverse. En plus de ne pas être optimal, le niveau de joueur était médiocre. Pour remédier à cela nous avons enrichi notre code de multiples fonctions qui agissent sur le joueur ou non.

Par exemple :

- **zone_tir()** qui renvoie un booléen vrai si le joueur peut tirer ou faux sinon ; ceci pour ne pas être pénalisés par les 10 pas d'attente entre chaque tir.

-**ball_trajectoire()** qui prédit la position où sera le ballon dans 20 pas pour éviter au joueur d'avoir un déplacement courbé et aussi pour ne pas qu'il réagisse au dernier moment.

-**get_coequipier()** et **get_ennemi()** qui renvoient respectivement la position des coéquipiers et des ennemis sous forme d'une liste contenant la position de ces joueurs du plus proche au plus éloigné.

Améliorations spécifiques

En ce qui concerne les stratégies plus spécifiquement, nous avons ajouté, dans la stratégie en 1 vs 1, une fonction pour faire en sorte que le joueur se déplace à une vitesse peu élevée tout en se dirigeant vers le but adverse le tout en gardant le ballon a proximité, contrairement au code de base dans lequel le joueur tirait tout le temps vers le but.

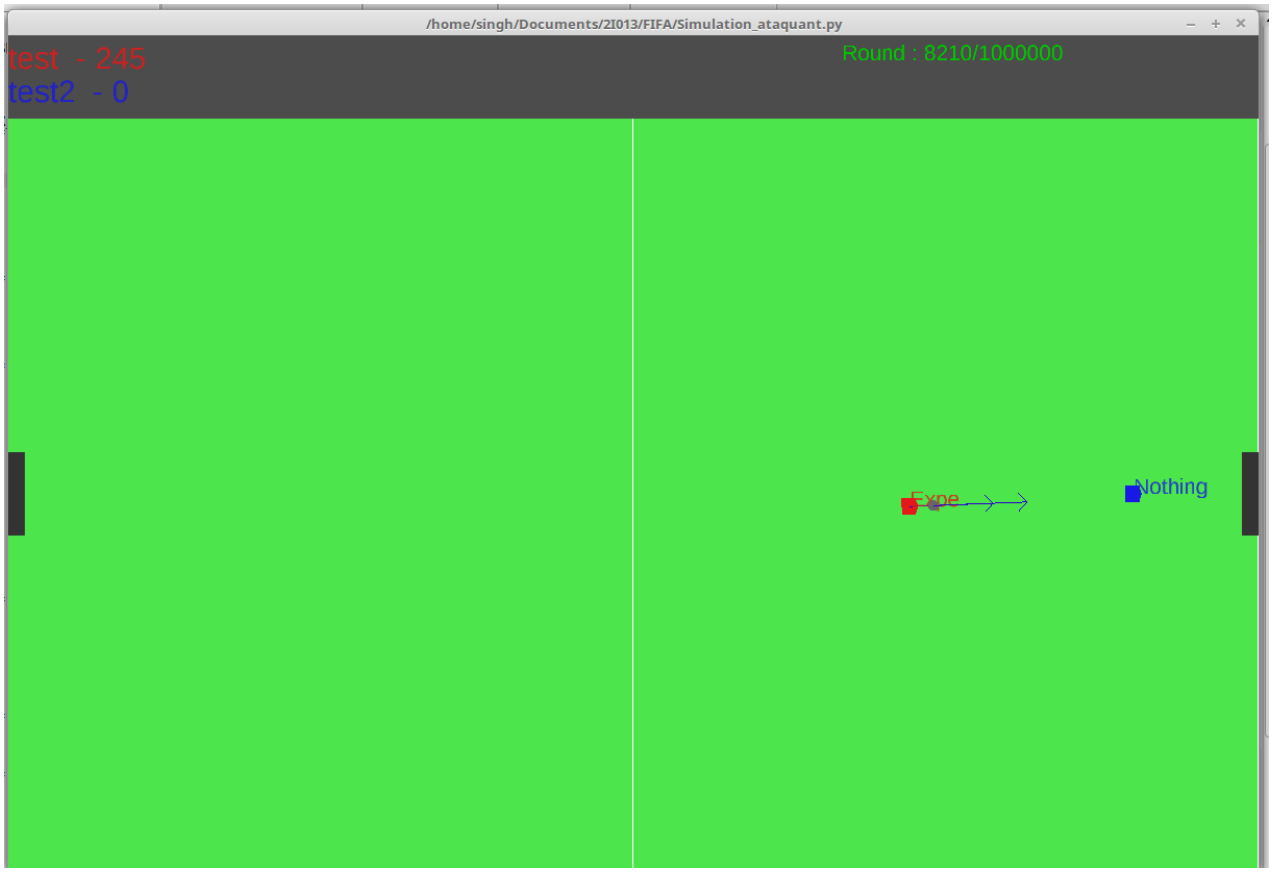
Pour le 2 vs 2, nous avons modifié le comportement de l'attaquant. Cette fois ci, nous avons amélioré son placement; il ne se déplace plus sur tout le terrain mais se replace à une position voulue chaque fois que le ballon est dans sa propre moitié de terrain et laisse le travail défensif au gardien. Le gardien aussi a eu droit à son lot d'amélioration. Au premier abord, il se contentait d'aller vers le ballon dès que ce dernier est dans sa moitié de terrain et de dégager. Nous avons fini par faire en sorte qu'il se place à 3 positions différentes en fonction de la position du ballon et nous avons implémenter une fonction passe() à l'aide de la fonction get_coequipier() vu auparavant.

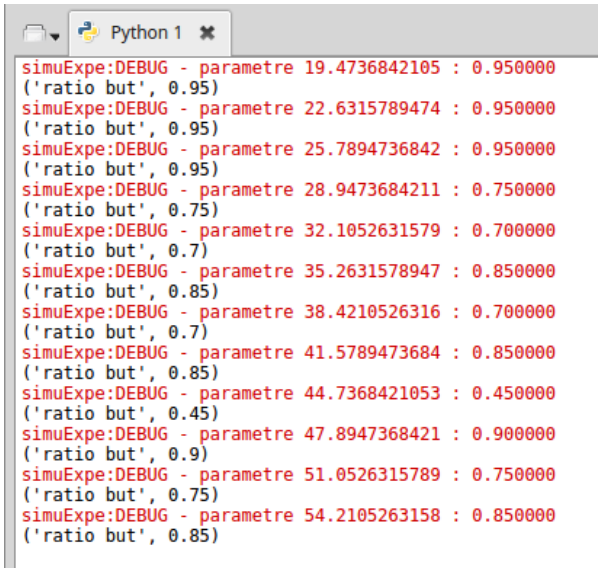
Venons en au 4vs4, nous avons jugé qu'il fallait de nouveau améliorer le positionnement du gardien et de l'attaquant étant donné que deux joueurs s'ajoutent au jeu. La zone du gardien a été réduite ; il sort moins la zone devant étant couvert par un défenseur.

L'attaquant lui s'est vu créer une variation avec un placement différent, le terrain est séparé en zone haute et basse chaque joueur occupe une zone.

Optimisation des paramètres

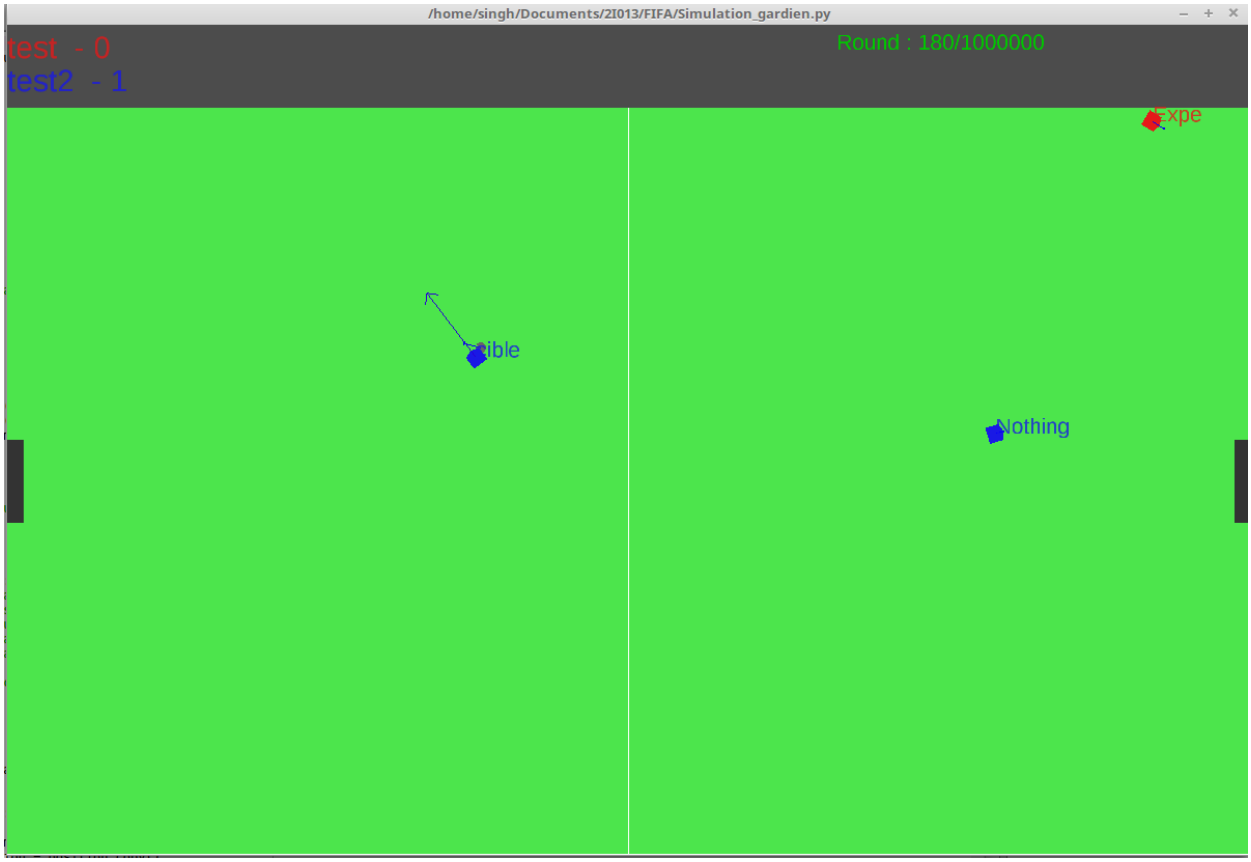
Le code étant amélioré, il fallait encore l'optimiser. Par exemple, il fallait choisir une distance à partir de laquelle l'attaquant serait suffisamment efficace et le faire tirer seulement une fois dans cette zone. C'est ainsi comme évoqué plus tôt nous avons utilisé l'observer fourni par le professeur pour optimiser au mieux notre joueur.





En fonction des paramètres, nous avons obtenu grâce à l'observer le meilleur ratio but et avons déduit la distance optimale à laquelle doit tirer l'attaquant.

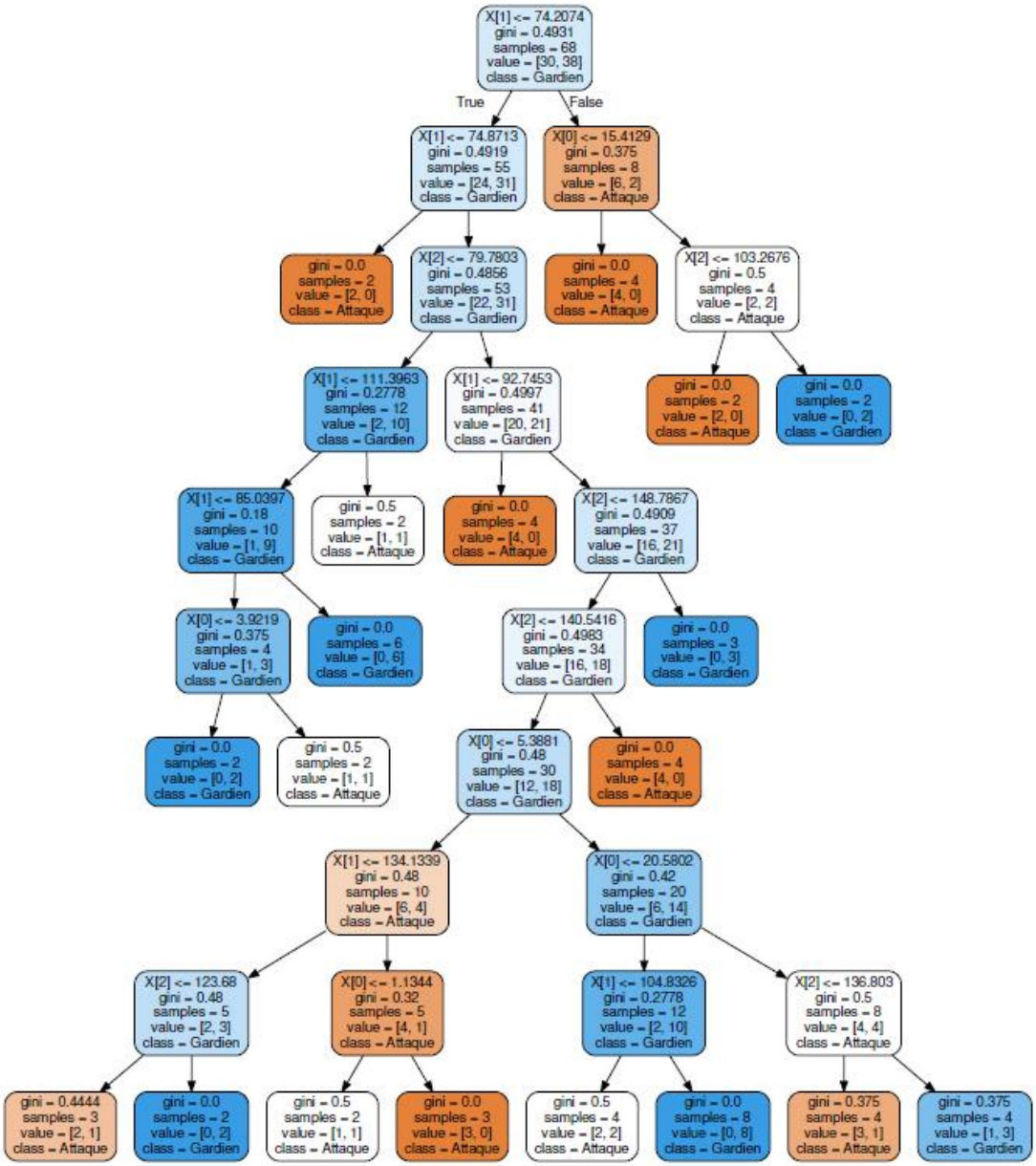
Pour le 2 vs 2 et pour le 4 vs 4, nous avons développé une fonction passe() qui permettait directement au gardien de passer le ballon à l'attaquant qui se retrouvait dans le milieu de terrain adverse et qui pouvait lancer la contre-attaque. Il était difficile de code cette fonction car la passe n'était pas très précise et que l'attaquant réagissait trop tard. Nous avons, toujours grâce à l'observer, déterminé la force nécessaire de la passe du gardien, le positionnement de l'attaquant lors de la phase de défense et l'anticipation nécessaire à l'attaquant avant de recevoir le ballon.



Apprentissage supervisé (Arbre de décisions)

Enfin, nous avons utilisé l'apprentissage supervisé pour faire en sorte que le joueur choisisse l'une des stratégies en fonction de différents paramètres. Pour cela, nous avons manipulé nous même les joueurs et leur avons « appris » que faire à quel moment. Par exemple, pour le gardien , nous avons

Après avoir simulé plusieurs cas possibles nous nous sommes retrouvés avec un arbre de décision tel que le suivant :



Conclusion

Cette unité d'enseignement nous a permis de mener un projet dans son intégralité. Ce que nous pensions être facile au début s'est finalement avéré être compliqué au fur et à mesure de l'avancement du projet. Nous avons appris à organiser notre code et une méthode particulière pour être efficace. Cette découverte de l'intelligence artificielle fut une bonne expérience. Github nous a permis de travailler chacun de notre côté tout en gardant une trace écrite des anciennes versions de notre code.