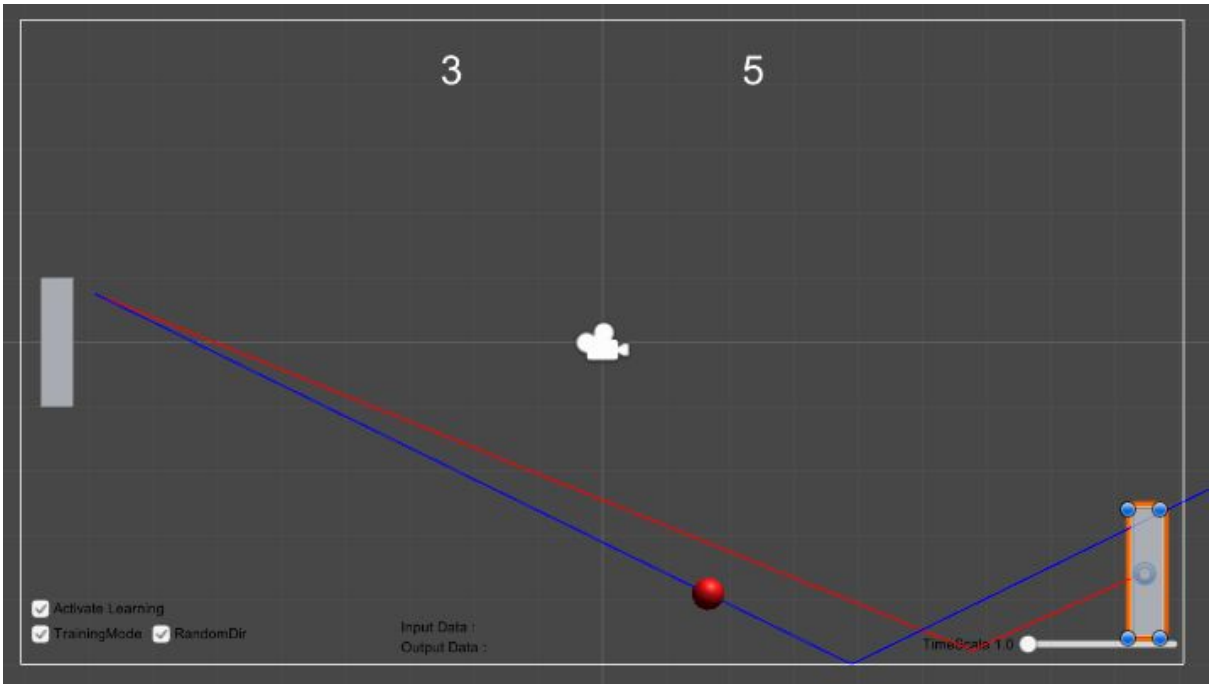
Projet Pong ANN

1. Objectifs réalisés

- Implémentation d'un réseau de neurones dans le jeu
- Trouver et implémenter l'algorithme qui permet à l'IA d'anticiper la trajectoire de la balle.
- Sauvegarde des données de l'entraînement de l'IA dans le JSON pour pouvoir les réutiliser par la suite.

2. Procédure du réseau de neurones

Dans le projet présenté ici, l'IA tente de calculer constamment la trajectoire que va effectuer la balle à chaque fois qu'elle se dirige vers lui.



Selon le schéma ci-dessus :

- La ligne bleue représente la trajectoire réelle de la balle
- La ligne rouge représente la trajectoire de la balle estimée par l'IA après calcul via le réseau de neurones.

● Input Layer

Les inputs sont envoyés au réseau de neurones au moment où le joueur lance la balle, ou bien au moment où ce dernier la renvoie vers le camp adverse.

Dans le projet présent, l'IA reçoit 4 valeurs d'input :

- Les valeurs X et Y de la position de départ. Elles prennent pour valeur la position de la balle au moment du lancement, ou bien la position dont l'IA estime qu'un rebond contre le mur va avoir lieu (cf Output Layer)
- Les valeurs X et Y de la direction de la balle.

● Hidden Layer

Ce réseau fonctionne avec un seul layer contenant 15 neurones.

● Output Layer

Après le calcul, l'IA renvoie 3 valeurs d'outputs :

- Les valeurs X et Y qui représentent le point d'arrivée de la balle estimée avant le rebond.
- Une valeur "finishedForesight" de 0 ou 1. L'IA estime à travers cette valeur si la trajectoire qu'elle a calculé va droit dans le mur (0) ou dans son camp (1), auquel cas il déplacerait ainsi sa raquette vers la coordonnée Y de l'output.

● Etapes du calcul

En reprenant l'exemple du schéma précédent, voici l'étape entière du calcul :

1. Au moment du lancement de la balle ou de sa collision avec la raquette du joueur, on envoie au réseau de neurones la position en X et Y de la balle ainsi que la direction vers laquelle elle a été envoyée.
2. L'IA s'aperçoit que la balle va arriver dans le mur inférieur. Elle renvoie en output la position de la collision avec ce mur. Le finishedForesight sera à 0.
3. Si finishedForesight = 0, alors on prend les 2 valeurs de l'output, puis on fait refléter la direction de la balle et on les envoie en tant qu'input dans le réseau pour que l'IA calcule la suite du chemin. Une telle procédure présentait un risque de boucle infinie si l'IA estime sans cesse cet output à 0, alors une limite de calculs à effectuer a été définie.
4. Si finishedForesight = 1, alors l'IA déplace la raquette vers la position X et Y de l'output.
5. La backpropagation se lance à la fin de ces calculs.

3. Pourquoi utiliser cette méthode ?

L'objectif de cette implémentation était de pouvoir limiter les variations du poids des neurones. Avant d'y recourir, mes tests ont montré que l'IA ne parvenait pas à trouver le poids idéal des neurones à cause de la direction qui variait beaucoup et des rebonds sur les murs. L'IA tentait d'estimer le point d'arrivée de la balle malgré tous ces facteurs, et générait des valeurs d'output qui fluctuaient beaucoup.

En utilisant cette méthode, on pousse l'IA à procéder étape par étape en découpant la trajectoire en fonction du nombre de rebonds sur les murs. On peut limiter ainsi l'aspect aléatoire des facteurs cités précédemment.

L'inconvénient, en revanche, c'est que plus il y a de rebonds, moins les valeurs d'outputs sont précises. Le problème étant que des outputs, donc des valeurs parfois erronées, sont utilisées en tant que valeur d'input pour l'étape suivante. Cela rend la backpropagation moins efficace.