Lien de la vidéo : <https://youtu.be/1sGDfrJeK4s>

# Rapport de Projet

## Tic-Tac-Toe with Minimax

Eric Saad (202175) – Joseph Raji (223496) – Joe Youssef (200376)

### Introduction :

Voici notre implémentation du jeu Tic-Tac-Toe utilisant l’algorithme **minimax, qui en théorie des jeux, est une méthode qui vise à minimiser la perte attendue. Pour ce faire, le joueur suppose que la décision prise par son adversaire sera défavorable. C'est-à-dire que le pire scénario est attendu avant le mouvement de l'adversaire.**

### Outils utilisés :

#### Librairies python utilisées:

* **TURTLE** : Un module de Python qui permet la réalisation de formes graphiques.
* **PYGAME** : Un module qui offre des outils permettant de créer des jeux.
* **RANDOM**: Un module qui offre le moyen d’implémenter DUMMY AI

### Implémentation :

#### Les fonctions utilisées:

* **play() :** Sert à construire l’interface graphique ainsi que le dictionnaire qui sert de grille.
* **spaceIsFree(position) :** Vérifie si une case définie est vide ou non.
* **insertLetter(letter, position) :** Insert une lettre (X ou O) à la position voulue si celle-ci est free.
* **click\_joueur() :** Détecte la position (x, y) de la souris lors d’un click
* **playerMove() :** Qui permet de faire le lien entre le choix du joueur, la validité du placement et l’appel de insertLetter.
* **compMove() :** Simule un coup d’ordinateur grâce à l’algorithme Minimax
* **fin\_de\_partie(letter) :** Affiche le résultat du jeu à l’aide de deux fonctions :
  + **checkForWin()**
  + **checkDraw()**

### L’algorithme MiniMax :

Pour l’algorithme minimax on a commencé par définir 3 conditions d’arrêt (victoire de l’AI, du joueur, ou une égalité).

Ensuite, on regarde si isMaximizing est true ou false (représentant un pion ordinateur ou joueur)

• Si True :

o On fixe le best score comme négatif.

o On teste toutes les cases vide du dictionnaire à tour de rôle.

o On simule un placement de pion par l’ordinateur.

o On effectue un placement de pion par l’humain (isMaximizing = False) grâce à un appel de minimax.

o On continue ainsi de manière itérative en plaçant un pion ordinateur puis un pion joueur itérativement jusqu’à la fin de la partie.

• Si False :

o On fixe le best score comme positif.

o On teste toutes les cases vide du dictionnaire à tour de rôle.

o On simule un placement de pion par le joueur.

o On effectue un placement de pion par l’ordinateur (isMaximizing = True) grâce à un appel itératif de minimax.

o On continue ainsi de manière itérative en plaçant un pion joueur puis un pion ordinateur itérativement jusqu’à la fin de la partie.

**Why not Alpha-Beta-Pruning ?**

Les deux algorithmes donnent le même résultat. Cependant, leur principale différence est que alpha-beta-pruning n’explore pas tous les chemins, comme le fait minimax : il élargit seulement les chemins garantis optimaux pour le joueur (donc les chemins qui peuvent être maximaux pour l’AI). Dans notre cas, l’algorithme minimax est déjà très rapide car les possibilités dans un jeu de tic tac toe sont très limitées, ainsi implémenter une solution Alpha-Beta-Pruning serait de l'over-engineering qui ajouterai plus de complexité au code.

Ça aurait été utile pour un jeu plus conséquent tel que puissance 4, les échecs, etc.