Université Libanaise Faculté des sciences 1

Project I407 F

Intelligence Artificielle

Ultimate tic-tac-toe

Préparé par:
Hassan Abdallah

Présenté à :
Dr. Rami Baida

Dr. Kifah Tout

Année académique

2020-2021

Table of Contents

Chapitre 1: Introduction **3**
1.1. Objectifs 3
Chapitre 2: théorie du jeu **4**
2.1. Tic-Tac-Toe 4
2.2. Ultimate tic-tac-toe 4
Chapitre 3 : Agent **7**
1. Agent 7
2. Description de l'environnement de l'agent 7
Chapitre 4 : Détails d'implémentation **8**
Chapitre 5: Analyse des résultats **13**
Chapitre 6: Algorithme utilisé **14**
1. Minimax Algorithme 14
2. Alpha-Beta Pruning 14
3. Monte Carlo Search Tree (MCTS) 14
Chapitre 7 : Conclusion **15**

Chapitre 1: Introduction

Le jeu Tic-tac-toe est l'un des jeux les plus connus. Ce jeu ne permet pas de gagner tout le temps, et une part importante des parties jouées se soldent par un match nul. Ainsi, le mieux qu'un joueur puisse espérer est de ne pas perdre la partie.

Cette étude vise à faire évoluer un certain nombre de stratégies et d'algorithmes sans perte et à les comparer aux méthodologies existantes.

Pour faire évoluer efficacement des stratégies sans perte, nous avons développé des moyens innovants de représenter et d'évaluer une solution.

Fait intéressant, notre implémentation d'algorithme est capable de trouver plus d'un millier de stratégies sans perte pour jouer au jeu.

Sur la base de cette expérience, nous avons développé des stratégies efficaces spécialisées ayant un rapport win-to-draw élevé. L'étude et ses résultats sont intéressants et peuvent être encourageants pour les techniques à appliquer à d'autres jeux « board games » pour trouver des stratégies efficaces.

1.1. Objectifs

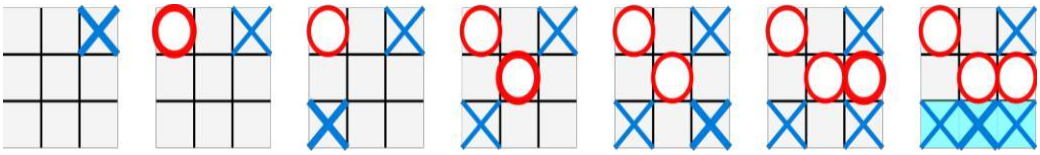
1. Développer un jeu tic-tac-toe en Unity avec deux modes : Deux joueurs (Deux joueurs humains qui jouent l'un contre l'autre, pas d'IA), Un joueur (humain contre l'ordinateur avec IA) en mettant en œuvre un algorithme minimax avec un concept de recherche contradictoire.
2. Utiliser l'algorithme minimax avec l’algorithme alpha-beta pruning dans ce jeu et étudier la complexité de ces algorithmes et encore étudier la vitesse de recherche du choix optimal dans le jeu et la mémoire complexité.

Chapitre 2: théorie du jeu

2.1. Tic-Tac-Toe

Tic-tac-toe (également connu sous le nom de Xs et Os) est un jeu de papier et crayon pour deux joueurs, X et O, qui à tour de rôle marquent les espaces dans une grille 3×3. Le joueur qui réussit à placer trois de ses marques sur une ligne horizontale, verticale ou diagonale gagne la

partie. L'exemple de jeu suivant est remporté par le premier joueur, X :



Les joueurs découvrent bientôt que le meilleur jeu des deux parties mène à un match nul. Par conséquent, le tic-tac-toe est le plus souvent joué par de jeunes enfants.

En raison de la simplicité du tic-tac-toe, il est souvent utilisé comme un outil pédagogique pour enseigner les concepts de bon esprit sportif et la branche de l'intelligence artificielle qui traite de la recherche d'arbres de jeu (Searching of game trees).

Il est simple d'écrire un programme informatique pour jouer parfaitement au tic-tac-toe ou pour énumérer les 765 positions essentiellement différentes (the state space complexity) ou les 26 830 jeux possibles jusqu'aux rotations et réflexions (the game tree complexity) sur cet espace.

Si le tic-tac-toe est joué correctement, le jeu se terminera par un match nul, ce qui en fera un jeu futile.

2.2. Ultimate tic-tac-toe

Ultimate tic-tac-toe est un "board game" compose de neuf tic-tac-toe boards disposées dans un grille de 3 par 3. Les joueurs jouent à tour de rôle dans les plus petits boards de tic-tac-toe jusqu'à ce que l'un d'eux gagne dans le plus grand board de tic-tac-toe. Par rapport au tic-tac-toe traditionnel, la stratégie dans ce jeu est conceptuellement plus difficile et s'est avérée plus difficile pour les ordinateurs.

Chaque petit tic-tac-toe board 3-en-3 est appelé local board, et le plus grand board 3-en-3 est appelé global board.

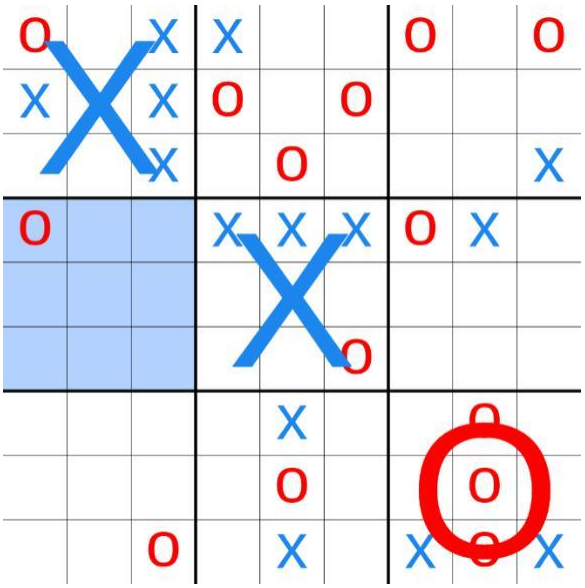
Le jeu commence avec O jouant où il veut dans l'un des 81 emplacements vides. Ce mouvement "envoie" son adversaire (Opponent) à son emplacement relatif.

Par exemple, si O a joué dans la case en haut à droite de son local board, alors X doit jouer ensuite dans le local board en haut à droite du global board. X peut alors jouer dans l'un des neuf emplacements disponibles sur ce local board, chaque coup envoyant O vers un local board différent.

Si une move est jouée pour gagner un local board selon les règles du tic-tac-toe normal, alors tout le local board est marqué comme une victoire pour le joueur sur le global board.

Une fois que le résultat d'un local board est décidé (gagnant ou nul), plus aucun move ne peut être joué sur ce board. Si un joueur est envoyé sur un tel board, alors ce joueur peut jouer sur n'importe quel autre board.

Le jeu se termine lorsqu'un joueur gagne le global board ou qu'il ne reste plus de moves légaux, auquel cas le jeu est un match nul.



Le tableau incomplet d'un jeu de « Ultimate tic-tac-toe » (les grands "X" et "O" représentant les jeux de local boards qui ont été gagnés par ce joueur). Le mouvement le plus récent était O jouant dans la case médiane gauche de la grille médiane supérieure (in the middle-left square of the top-middle grid), forçant X à jouer son prochain mouvement dans la grille médiane gauche (middle-left grid).

Le « Ultimate tic-tac-toe » est nettement plus complexe que la plupart des autres variantes du tic-tac-toe, car il n'y a pas de stratégie claire pour jouer. C'est à cause de la ramification compliquée du jeu dans ce jeu. Même si chaque mouvement doit être joué dans un local board, équivalent à un board de tic-tac-toe normal, chaque mouvement doit prendre en compte le global board de plusieurs manières :

1. Anticiper le prochain mouvement : Chaque mouvement joué dans un local board détermine où le prochain mouvement de l'adversaire peut être joué. Cela peut rendre

viables des mouvements qui peuvent être considérés comme mauvais dans un tic-tac-toe normal, car l'adversaire est envoyé sur un autre local board et peut être incapable d'y répondre immédiatement. Par conséquent, les joueurs sont obligés de considérer le plus grand board de jeu au lieu de se concentrer simplement sur le local board.

2. Visualiser l'arbre du jeu : Visualiser les futures branches de l'arbre du jeu est plus difficile qu'un tic-tac-toe à une seule board. Chaque mouvement détermine le mouvement suivant et, par conséquent, la lecture anticipée- la prévision des mouvements futurs - suit un chemin beaucoup moins linéaire. Les positions futures du conseil d'administration ne sont plus interchangeables, chaque mouvement menant à des positions futures possibles très différentes. Cela rend l'arbre du jeu difficile à visualiser, laissant éventuellement de nombreux chemins possibles négligés.

3. Gagner la partie : En raison des règles du « Ultimate tic-tac-toe », le global board n'est jamais directement affecté. Il n'est régi que par les actions qui se produisent dans les locales boards. Cela signifie que chaque mouvement local joué n'est pas destiné à gagner le local board, mais à gagner le global board. Les gains locaux n'ont pas de valeur s'ils ne peuvent pas être utilisés pour gagner le global board. En fait, il peut être stratégique de sacrifier un local board à votre adversaire afin de gagner vous-même un local board plus important. Cette couche supplémentaire de complexité rend plus difficile pour les humains d'analyser l'importance relative et la signification des mouvements, et par conséquent plus difficile de bien jouer.

Chapitre 3 : Agent

1. Agent

Son comportement est décrit par sa fonction qui fait correspondre le précepte à l'action. Dans le tic-tac-toe, il y a deux agents, le système informatique (computer system) et humain. Dans le jeu de tic-tac-toe basé sur l'AI, la fonction est mappée en utilisant l'algorithme minimax aux côtés de alpha-beta pruning. L'agent peut être décrit plus en détail par les facteurs suivants :

Mesure de performance : nombre de gains ou de nuls.

Environnement : Une grille 3x3 avec 9 cases respectivement, avec adversaire (agent humain). La cellule autrefois occupée par une marque ne peut plus être réutilisée. L'environnement des tâches est déterministe, entièrement observable, statique, multi-agent, discret, séquentiel.

De plus, il s'agit d'un agent basé sur l'utilité (utility) car il utilise des concepts "heuristics and pruning " comme fonction d'utilité qui mesure ses préférences parmi les états et choisit l'action qui conduit à l'utilité la plus attendue, c'est-à-dire la condition de gagner ou d'égalité pour l'agent du système informatique.

2. Description de l'environnement de l'agent

L'environnement de base pour l'agent de jeu tic-tac-toe est une grille 3x3 avec 9 cellules respectivement avec adversaire en tant qu'agent humain. La cellule autrefois occupée par une marque ne peut plus être réutilisée. L'agent a accès à l'état complet de l'environnement à tout moment et peut détecter tous les aspects pertinents pour le choix de l'action, rendant l'environnement entièrement observable. Aussi, le prochain état de l'environnement peut être complètement déterminé par l'état actuel et l'action exécutée, ce qui signifie que l'environnement est déterministe. C'est un jeu à deux joueurs avec l'adversaire en tant qu'agent humain, c'est donc un environnement multi-agents. De plus, l'environnement de jeu a un nombre fini d'états, de perceptions et d'actions, rendre l'environnement statique et discret. De plus, l'environnement est séquentiel, puisque le choix actuel de la cellule pourrait affecter toutes les décisions futures. Ainsi, l'environnement de tâche pour l'agent du système de tic-tac-toe est séquentiel, déterministe, entièrement observable, statique, discret et multi-agent.

Chapitre 4 : Détails d'implémentation

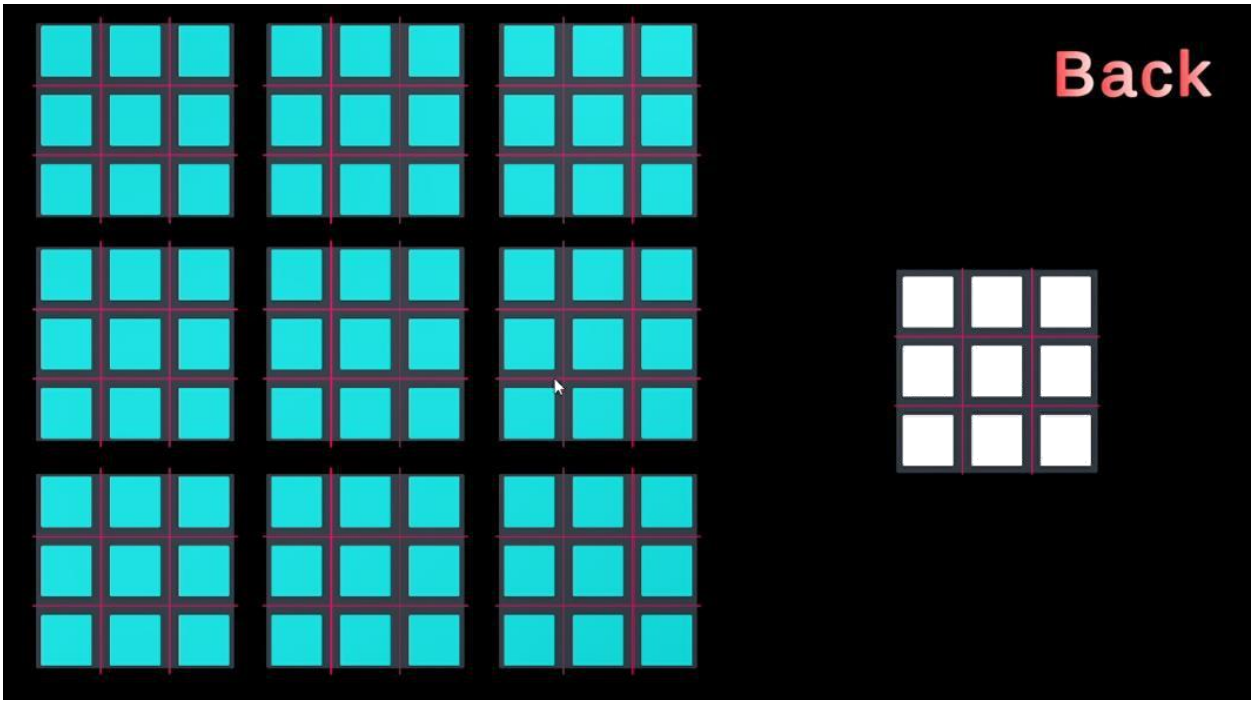
Dans notre code :

1- La génération du l'arbre minimax ce fait en utilisons l'algorithme minimax avec Alpha-Beta Pruning, dans la méthode '`GenerateStates`'.
2- Dans la méthode '`AIMove`', l'ordinateur prends une décision intelligente pour maximiser sa chance de gagner avec une prédiction pour prendre en considération l'avenir de cette décision.
3- La méthode d'utilité prend en considération la longueur du 'depth' de cette nœud, Par exemple pour le joueur o qu'il doit minimiser la valeur, '`result = -1000 + node.depth * 100`', le code complet et dans la méthode '`Utility`'.
4- La Méthode '`evaluationFunction' aide l'ordinateur à savoir quelle est la bonne décision dans des cas précise par l'évaluation de l'avenir.`

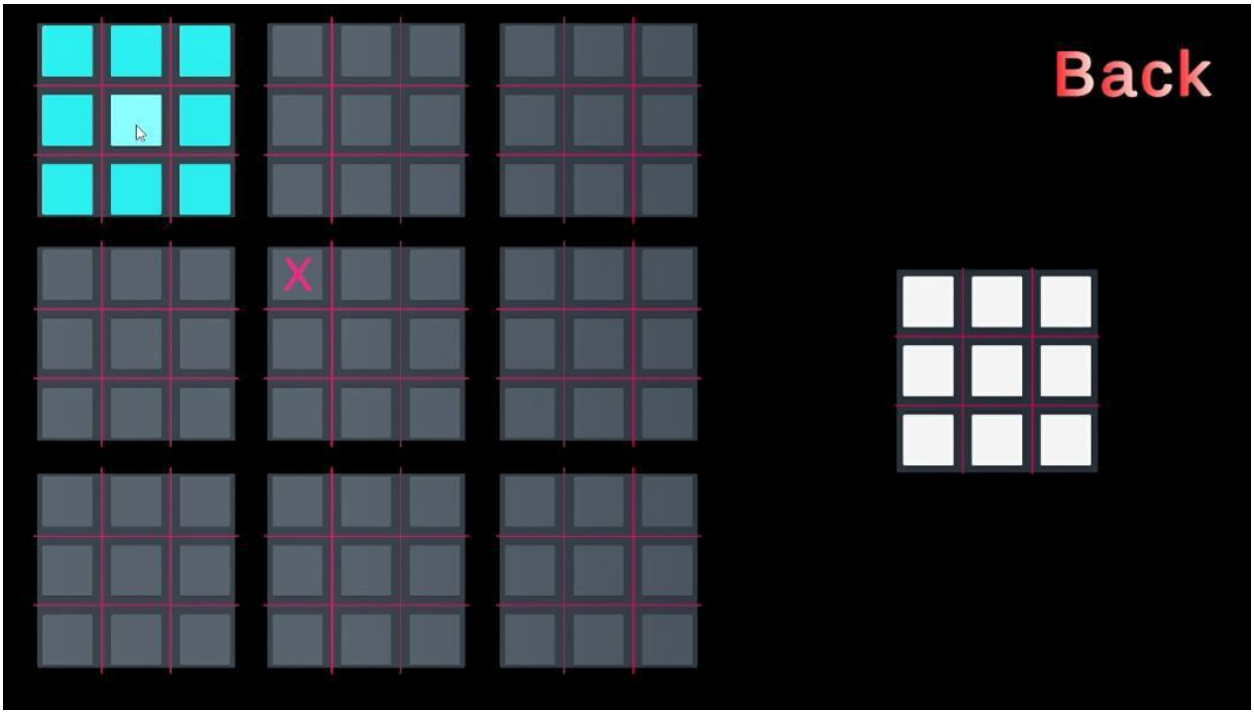
Il y a plusieurs d'autre méthode mais ses sont les principaux, dans notre code nous avons prendre ~ toutes les possibilités pour que la décision de l'ordinateur soit fiable, nous avons essayé beaucoup de fois pour gagner sur l'ordinateur mais cela n'a pas marché, alors notre jeu a une performance élevé, les détails sont dans le chapitre 'Analyse des résultats'.

Deux joueurs (Deux joueurs humains qui jouent l'un contre l'autre, pas d'IA) :

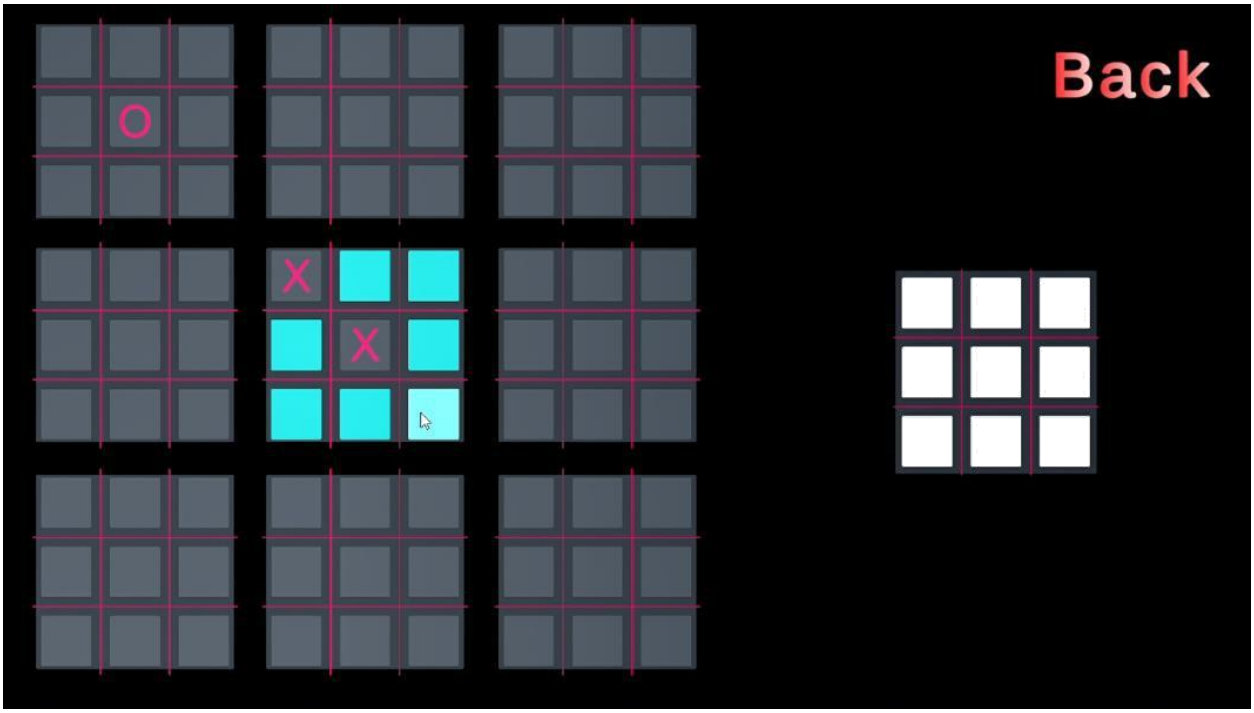
Board initial :



Première joueur joue :

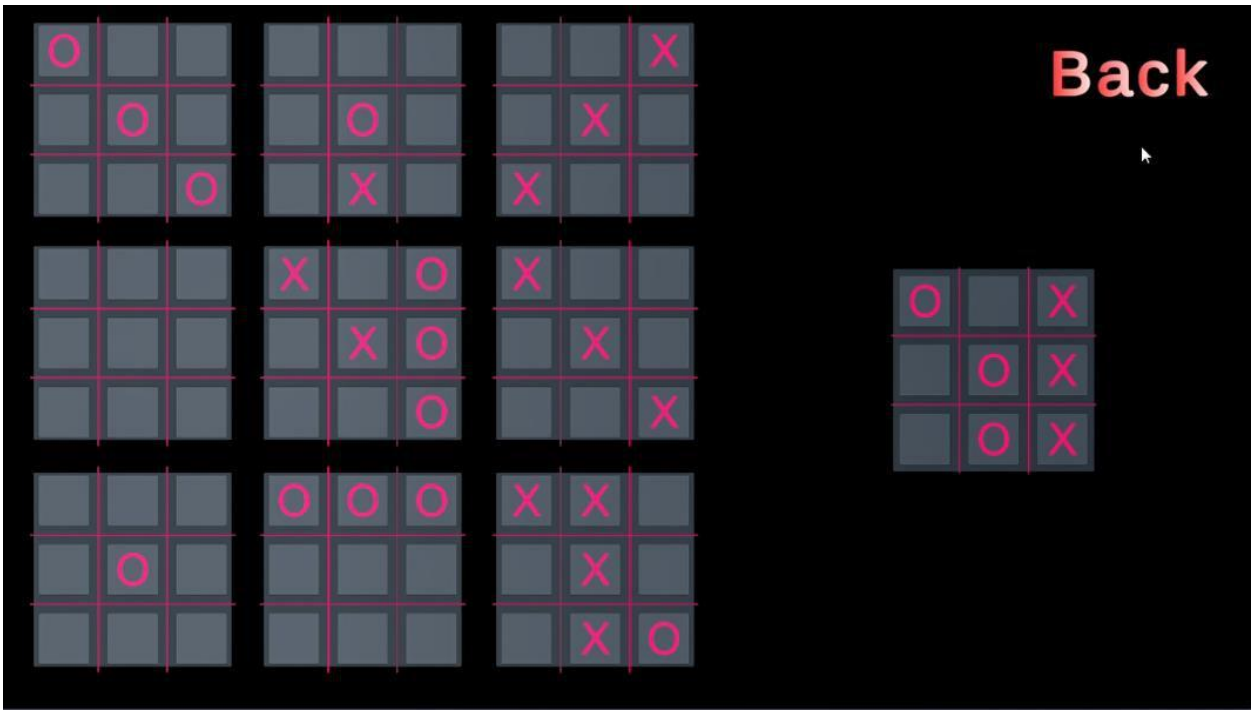


L’autre joueur joue : (O)



Apres quelque essai :

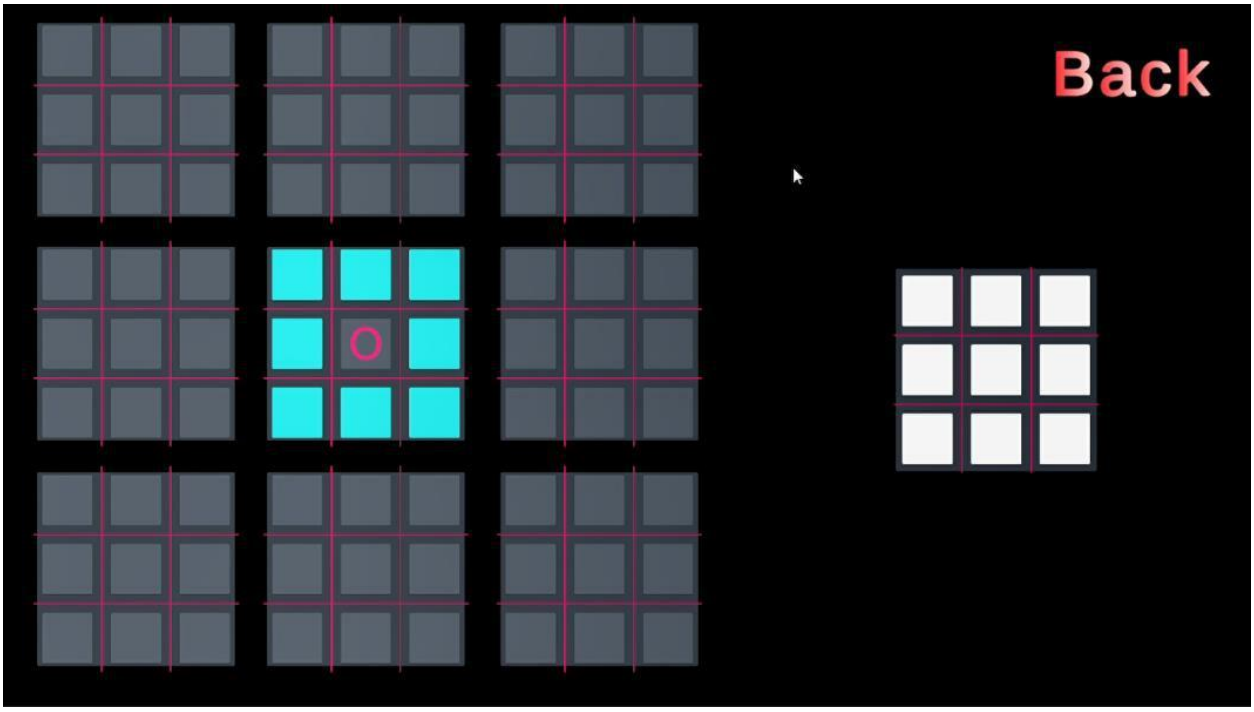
Board finale : le joueur X gagne



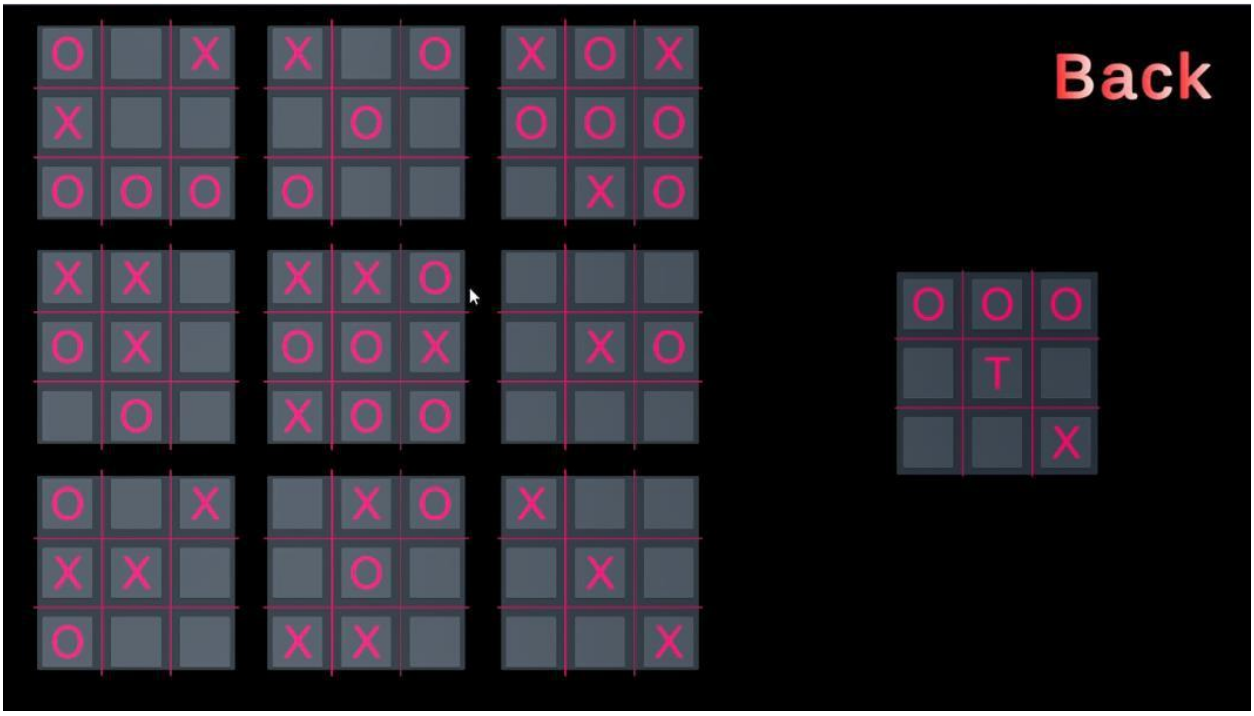
Un joueur (humain contre l'ordinateur avec IA) :

L’humain ne peut pas gagner, seulement nul ou l’AI gagne.

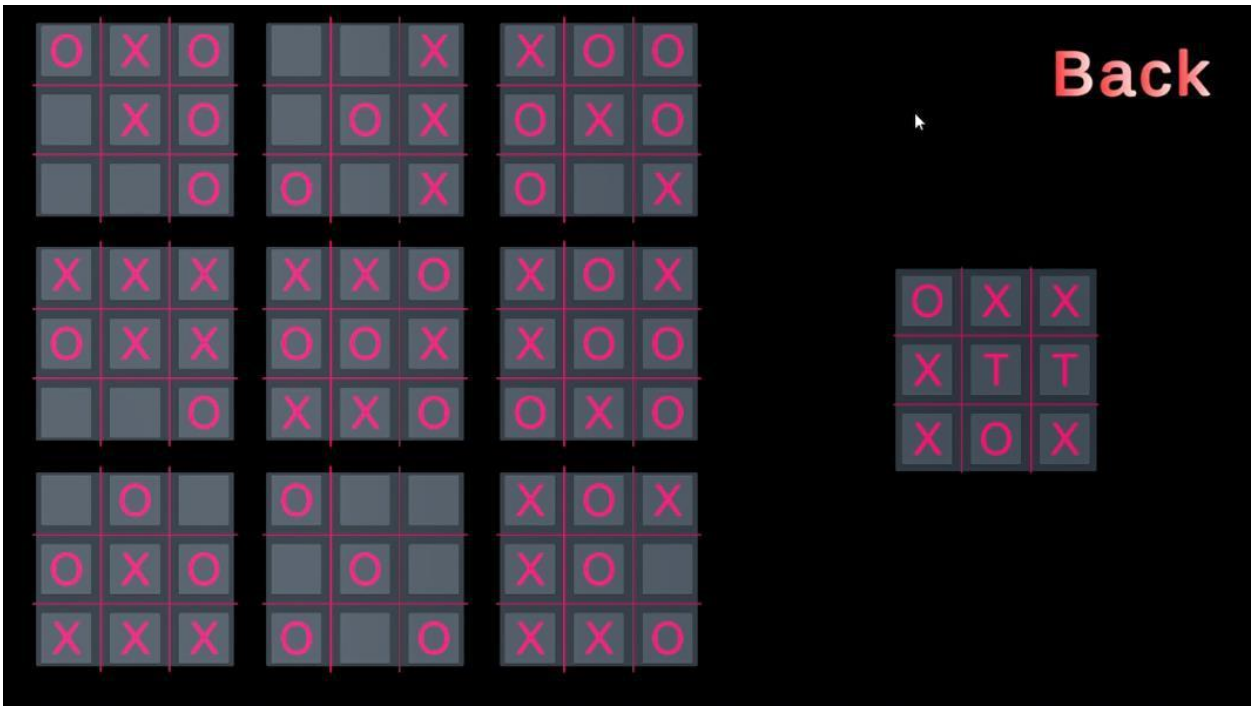
Board initial : l’AI met O au centre toujours.



Board finale : l’AI gagne.



Autre essai : Nul entre humain et ai


Back

Chapitre 5: Analyse des résultats

Nous avons envoyé le jeu sous forme d'un fichier exécutable aux 15 personnes, le nombre total des tests sur ceux personnes est 61 test, dont 22 l'ordinateur a gagné, et dans les autres tests la résulta était en égalité entre le joueur et l'ordinateur (Draw).

Alors notre algorithme a une haute performance.

Chapitre 6: Algorithme utilisé

1. Minimax Algorithme

2. Alpha-Beta Pruning

3. Monte Carlo Search Tree

.

Chapitre 7 : Conclusion

Par le développement de l'Ultimate Tic-tac-toe, nous apprenons de nouvelles connaissances sur le platform Unity (https://unity.com) et comprenons mieux certaines connaissances que nous avons apprises auparavant. En plus de ceux-ci, nous apprenons des concepts importants de l'intelligence artificielle.

Sur la base de l'algorithme minimax pour l'analyse mathématique, en accélérant le calcul par le concept alpha-beta pruning et en optimisant la fonction d'utilité à l'aide de la fonction heuristique, le jeu de tic-tac-toe 3x3 dans 3x3 a été développé. Nous avons exploré qu'un Ultimate tic-tac-toe 3x3 dans 3x3, un jeu de technique de recherche d'adversaires en intelligence artificielle, peut être développé à l'aide de ces techniques. Augmenter la taille de ce jeu créerait un énorme problème de complexité temporelle avec le même algorithme et les mêmes techniques, pour lesquels d'autres logiques doivent être étudiées plus avant.