Nous nous sommes inspirés de travaux existants d'IA pour Ultimate Tic-Tac-Toe, notamment ceux basés sur Minimax + heuristique (le projet AdvancedMinimaxPlayer) ces approches combinent : victoire locale quand possible, blocage des menaces adverses, et évaluation heuristique des positions restantes (importance du centre, des coins, de l'envoi de l'adversaire vers des mini-grilles « défavorables »)

 # Résumé des IA — Ultimate Tic-Tac-Toe

Description du fonctionnement des deux IA présentes dans notre projet.

IA Aléatoire (IAJoueur) ----------
-Rôle : choisir un coup valide au hasard.
- Fonctionnement : calcule la liste des coups valides (respect de la grille ciblée si définie) puis retourne `choice(coups_valides)`.
- Avantages : très rapide, consommation CPU minimale.
- Limites : aucun raisonnement tactique — ne bloque pas les menaces et ne cherche pas à maximiser les gains.

IA Intelligente (IAIntelligente) —----------
- Rôle : heuristique rapide visant à jouer de manière raisonnable sans recourir à un Minimax complet.
L'IA suit les principes suivants :

1. Jouer une victoire locale (sur une petite grille) dès que possible.
2. Bloquer immédiatement toute menace directe de l'adversaire.
3. Éviter d'envoyer l'adversaire dans une grille libre ou gagnante (ce qui lui donnerait du choix).
4. Favoriser les positions fortes : centre > coins > côtés.
5. Attribuer un score à chaque coup pour choisir la meilleure action si aucun coup immédiat n'est obligatoire.
6. Détecter les menaces globales : éviter de créer une victoire simple pour l'adversaire au tour suivant.

Ordre de décision :

1. **Victoire locale** : jouer un coup qui gagne immédiatement une petite grille si disponible.
2. **Blocage réel** : si l'adversaire a une menace de gagner dans une petite grille, ne choisir que les coups qui empêchent réellement cette victoire (simulation légère pour vérifier le blocage).
3. **Heuristique de scoring**: évaluer chaque coup par un score pondéré (préférer centre/corners, pénaliser l'envoi de l'adversaire vers des grilles problématiques, pénalité si l'adversaire obtient une menace immédiate après notre coup, bonus si le coup mène potentiellement à une victoire globale).
4. **Choix final** : prendre le coup avec le meilleur score (sinon fallback aléatoire).

Composants additionnels :
- Simulation locale pour vérifier si un coup bloque vraiment l'adversaire.
- Simulation d'un coup pour détecter s'il provoque une victoire globale.
- Vérification si l'adversaire disposera d'une menace immédiate après la simulation de notre coup.
- Poids paramétrables (dans `self.weights`) pour calibrer les pénalités et bonus

Avantages : bloque correctement les menaces immédiates, évite des coups qui donnent trop de liberté à l'adversaire, rapide.

Limites : heuristique (poids fixes), pas de recherche exhaustive sur des situations profondes, un Minimax limité serait plus robuste.

Comparaison rapide —----------

- Robustesse : `IAIntelligente` >> `IAJoueur`.
- Performance : `IAJoueur` (instantané) > `IAIntelligente` (toujours très rapide mais exécute quelques simulations locales).
- Facilité d'ajustement : l'intelligente expose `self.weights` pour réglages manuels.

Références & inspirations —----------

Voici l'ensemble des sources utilisées, avec lien + ce que j'en ai retenu précisément pour mon IA.

1. AdvancedMinimaxPlayer — Ultimate Tic Tac Toe (GitHub) le code est en java mais je me suis servie des idées principales
Lien : https://github.com/kvombatkere/UltimateTicTacToe-AI

Ce que nous avons utilisé :

- L'idée qu'une IA pour Ultimate Tic-Tac-Toe doit prioriser les victoires locales et blocages directs avant toute heuristique globale.
- Le principe de scoring heuristique simple par grille, utilisé ici pour créer mon propre système de pondérations (center_bonus, corner_bonus, etc.).

La distinction micro-jeu (mini-grille) / macro-jeu (grille 3×3), essentielle à notre architecture.

2. Ultimate Tic Tac Toe AI — Minimax + heuristique
Lien : https://jatin7gupta.github.io/Ultimate-tic-tac-toe/

Ce que j'ai utilisé :

- L'importance de l'ordre de priorité : Gagner>Bloquer>Choisir une bonne position
- L'idée de favoriser centre > coin > côté, que j'ai reprise dans mes pondérations :

'center_bonus': 12,
'corner_bonus': 6,
'side_bonus': 2

- L'approche consistant à simuler un coup pour mesurer son impact futur, reprise dans la fonction _opponent_has_immediate_threat.

3. A Heuristic for Ultimate Tic Tac Toe — Analyse stratégique

Lien
https://bleedingedgemachine.blogspot.com/2013/08/a-heuristic-for-ultimate-tic-tac-toe.html

Ce que j'ai utilisé :

- L'idée centrale : une victoire locale n'est pas toujours le meilleur coup, car elle peut parfois envoyer l'adversaire dans une mini-grille dangereuse.
- Cela a mené à mon heuristique : 'penalize_full_target': 200 qui évite d'offrir du « choix libre » à l'adversaire.
- L'analyse selon laquelle le macro-jeu prime sur le micro-jeu à haut niveau, reflété dans : 'global_win_bonus': 10000