

**Puissance 4**

**Projet JAVA**



**Réalisé par :**

**Mariel Evha ABIR**

**Encadré par :**

**Pr Rémi SYNAVE**

**Année Universitaire**

**2019-2020**

# Puissance 4

# Introduction

Encore appelé « 4 alignés », puissance 4 est un jeu de stratégie très connu qui se joue à deux. Le but du jeu est d'aligner le premier 4 pions de sa propre couleur. Les 4 pions peuvent être alignés horizontalement, verticalement ou encore en diagonale. C’est dans cette optique qu’il nous a été demandé dans le cadre de notre projet java de développer une puissance 4.

# Moteur de jeu

Il s’agit ici de la partie centrale et la plus importante du jeu. En effet c’est à ce niveau que toute la structure jeu est mise en place c’est-à-dire, la construction du plateau de jeu ainsi que les lignes et case de celui-ci

## Classe Case

Il s’agit de la classe permettant de stocker l’état d’une case du plateau de jeu. La case peut se trouver sous 3 états, l’état 0 si la case est vide et l’état 1 ou 2 si lase est occupée.

## Classe Line

Il s’agit de la classe permettant de construire une ligne de Case. Cette classe fournit l’ensemble des méthodes assurant la manipulation de ses données dont les plus importantes sont SetLine et Win.

* SetLine : Il s’agit de la méthode de permettant de mettre à jour les données d’une ligne
* Wiin : Une des méthodes les plus importantes du jeu. Celle-ci permet de terminer la partie en vérifiant s’il y’a un alignement d’au moins 4 pions sur la ligne.

## Classe Plateau

Cette classe permet de stocker une grille complète de jeu composé d’un tableau de ligne. Cette dernière est la classe autour de laquelle s’articule la majeure partie des actions du jeu.

* **FullColumn** : Il s’agit ici de la méthode qui vérifie la disponibilité d’une colonne.
* **AddPoint** : Cette méthode permet d’ajouter un pion dans le plateau
* CheckVertical, CheckHorizontal, CheckDiagonal et CheckReverseDiagonal : L’ensemble de ces méthodes constituent celles qui composent les ligne verticale, horizontale et les diagonales afin de vérifier s’il un y’a au moins un alignement de 4 pions.
* **NoCheck** : Cette méthode parcourt le plateau et vérifie s’il est plein et qu’il n’y a pas de gagnant

## Classe Game

Cette classe fournit un ensemble de méthode permettant de créer une partie et d’y jouer en mode console en invitant les deux (2) joueur à tour de rôle de préciser la colonne où il souhaite déposer le pion.

# Graphique

Comme son nom l’indique, cette partie met à disposition du joueur un ensemble d’interface graphique afin d’y jouer avec un visuel plus adapté. Les quelques interfaces fournies sont présentées ci-dessous :

* L’interface de choix du mode de jeu : 2 modes de jeux sont disponibles, le mode « Real VS IA » permettant à un joueur physique de jouer avec une intelligence artificielle et le mode « Real VS Real » pour 2 joueurs physiques.
* L’interface de sélection du niveau de l’IA.
* L’interface du plateau de jeu.

# Intelligence Artificielle

Cette partie se consacre sur la stratégie implémentée pour l’intelligence artificielle. Le choix de la stratégie s’est porté sur une IA évoluant à plusieurs (5) niveaux du plus facile au plus simple.

**Niveau 1 (LevelOneMove)** : Mode de jeu aléatoire

* ***L’IA joue sur une colonne aléatoire du plateau si elle est disponible.*** *L’algorithme consiste à générer un nombre n’excédant pas les dimensions du plateau puis vérifie si la valeur générée correspond à une colonne disponible et la retourne.*

**Niveau 2 (LevelTwoMove)** : Amélioration du niveau 1

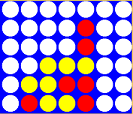
* ***Soit l’IA joue un coup gagnant s’il existe.*** *L'IA parcours le plateau en simulant un coup, vérifie si le coup simulé lui permet de gagner et retourne la colonne concernée.*
* ***Soit l’IA joue sur une colonne aléatoire*** (comme décrit précédemment au niveau 1).

**Niveau 3 (LevelThreeMove)** : Ce niveau offre quatre (4) possibilités jouer à l’IA

* ***Soit l’IA joue un coup gagnant s’il existe***. *L'IA parcours le plateau en simulant un coup, vérifie si le coup simulé lui permet de gagner et joue sur la colonne concernée.*
* ***Soit l’IA joue un coup lui permettant de bloquer un coup gagnant de l’adversaire***. *L'IA parcours le plateau en simulant un coup de l’adversaire, vérifie si le coup simulé permet à l’adversaire de gagner et joue à la colonne concernée afin d’empêcher une victoire de l’adversaire.*
* ***Soit l’IA joue un coup pouvant lui donner un alignement de trois (3) pions***. *L'IA parcours le plateau en simulant un coup, vérifie si le coup simulé lui permet d’aligner trois pions et joue la colonne concernée.*
* ***Soit l’IA joue sur une colonne aléatoire*** *(voir niveau 1)*.

**Niveau 4 (LevelFourMove)** : 5 possibilités de jouer un coup

* ***Soit l’IA joue un coup gagnant s’il existe*** *(expliquez au niveau 3)*.
* *S****oit l’IA joue un coup lui permettant de bloquer un coup gagnant de l’adversaire*** *(expliquez au niveau 3)*.
* ***Soit l’IA joue un coup lui permettant d’empêcher l’adversaire d’aligner horizontalement trois (3) pions avec deux (2) coups gagnant à chaque extrémité***.  *La solution implémentée à ce niveau consiste dans un premièrement à parcourir le plateau et à vérifier si l’adversaire à un alignement de 2 pions, deuxièmement à compter le nombre de cases vides consécutives à gauche et à droite de l’alignement, troisièmement test si à gauche il y’a au moins 2 cases vide et au moins une case vide à droite (vise versa) et joue la colonne à gauche de cet alignement tout assurant qu’au niveau de cette colonne la ligne en dessous soit occupée.*
* ***Soit l’IA bloque à nouveau une option à 2 coups gagnants de l’adversaires.*** *Le traitement appliqué ici a été de parcourir le plateau à partir du bas et de vérifier tout d’abord s’il y’a un alignement vertical de 2 cases entre les colonnes 2, 3 et 4, ensuite vérifier qu’au niveau de la ligne supérieure à l’alignement à gauche ou à droite (voir figure ci-dessous).*

**

* ***Soit l’IA joue un coup par rapport à la case disponible ayant la plus grande***. *En initialise une variable max à 0, puis l’on parcourt le plateau dès lors que l’on a une place disponible (c’est-à-dire que la case en dessous n’es), on récupère la valeur de la matrice correspondant aux coordonnées si celle-ci est supérieur à max, max est remplacé par cette valeur. A la fin du parcours on l'IA joue selon la colonne de la plus grande valeur disponible. Ci-après la matrice contenant le nombre d’alignements possibles de chacune des cases.*

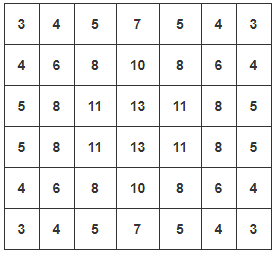


Figure 1 : Heuristique

**Niveau 5 (LevelFiveMove)**: Basé sur l’algorithme « Min Max ».

* ***Soit l’IA joue un coup gagnant s’il existe*** *(expliquez au niveau 3)*.
* S**oit l’IA joue un coup lui permettant de bloquer un coup gagnant de l’adversaire** *(expliquez au niveau 3) et met à jour la copie du plateau de jeu utilisé par Min Max*.
* ***Soit l’IA joue le coup le plus optimal après simulation de plusieurs coups en évaluant à chaque fois pour chacun des joueurs le plateau en fonction des alignements des pions sur le plateau****.*

# Conclusion

Au sortir de ce travail, nous mettons à disposition un livrable fonctionnel en plus d’une source assez lisible et suffisamment commentée.

**Sources**

<https://www.javatpoint.com/mini-max-algorithm-in-ai>

<https://www.baeldung.com/java-minimax-algorithm>

<https://www.geeksforgeeks.org/minimax-algorithm-in-game-theory-set-3-tic-tac-toe-ai-finding-optimal-move/?ref=lbp>