كلية العلوم والتقنيات بطنجة
Faculté des Sciences et Techniques de Tanger
جامعة عبد الملك السعدي
Université Abdelmalek Essaadi

**Département Génie Informatique**

Logiciels et systèmes informatiques

1ére année (LSI)

# Ultimate Tic-Tac-Toe

Java smart Game

**Auteur :**

M. Mouad CHAOUKI

**Encadrant :**

Pr. **AIT KBIR**

# Remerciements

Il était agréable de mon acquitter d'une dette de reconnaissance envers tous ceux, dont la contribution au cours de ce projet, a favorisé son aboutissement.

Ainsi, je tiens vivement à remercier notre encadrant **Pr. Ait KBIR** pour son encadrement précieux et pour le soutien qu'il m'a donné.

Que le corps professoral et administratif de la FSTT trouve ici mes vifs remerciements.

Finalement, Je veux remercier toute personne qui a contribuée de près ou de loin à l'élaboration de ce rapport.

# Table des matières

Table des matières

# Introduction

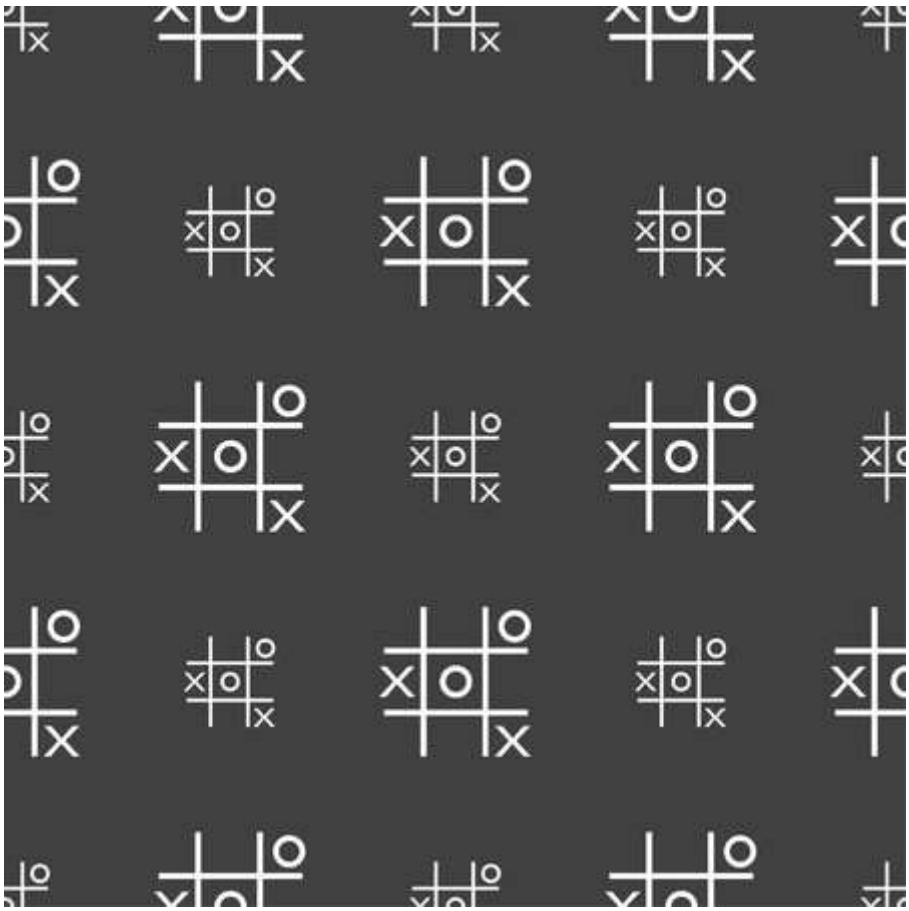


Figure 1 Konigsberg-7-ponts au format JPEG.

## 1.1 Description

Ultimate Tic-Tac-Toe est comme le Tic-Tac-Toe normal mais à plus grande échelle et avec une touche intéressante. Alors que le Tic-Tac-Toe normal se traduira presque toujours par un match nul (pour deux joueurs raisonnablement bons), Ultimate Tic-Tac-Toe nécessite beaucoup plus de compétences pour maîtriser.

## 1.2 Règles

Ultimate Tic-Tac-Toe, c'est 9 parties normales de Tic-Tac-Toe jouées simultanément. Le plateau est composé de 9 tuiles, chacune contenant 9 cases. Votre coup dictera cependant où votre adversaire peut jouer ; Donc, si vous jouez votre pièce dans le carré supérieur droit d'une tuile, votre adversaire doit jouer sa prochaine pièce dans la tuile supérieure droite.

Il existe deux variantes de ce jeu :

- Victoires de la première tuile - Le premier joueur à gagner une seule tuile.
- Tuiles d'affilée (plus difficile) - Le premier joueur à gagner 3 tuiles d'affilée gagne.

Le premier joueur doit placer sa pièce dans n'importe quelle case du plateau. Après cela, vous êtes limité à la tuile vers laquelle votre adversaire vous envoie. Il existe cependant deux exceptions.

Si l'une de ces situations se produit :

- Votre adversaire vous envoie sur une tuile qui a déjà été gagnée (par vous ou par eux).
- Votre adversaire vous envoie sur une tuile pleine.

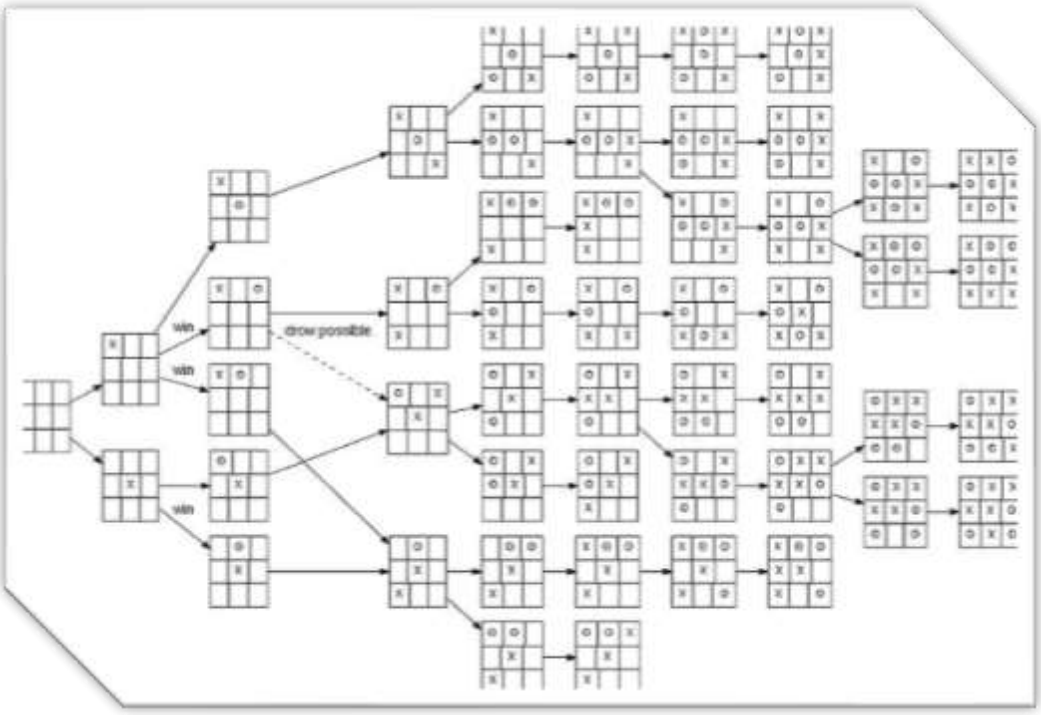
Alors, vous obtenez un tour ouvert et pouvez placer votre pièce sur n'importe quelle tuile de votre choix.

## 1.3 But

Développer une application Java du jeu "Ultime Tic-Tac-Toe" et implémenter les stratégies de recherche adversiales vues dans le cours.

# Chapitre 1

# Intervention des méthodes de l'intelligence artificielle




## 1.1 Représentation d'un problème

### 1.1.1 Définition

**Représentation par un graphe d'états :** Structuration de l'ensemble des états par un graphe orienté où chaque nœud est un état du problème.

**Le système de production :** Description de l'ensemble des

mécanismes permettant de passer d'un état du problème à un autre.

**La stratégie de résolution :** donnera comment choisir parmi les états accessibles celle menant à

### 1.1.2 Caractéristiques

- État initial (départ)
- État ou ensemble d'états à atteindre (but).
- Les conditions sur les états et les actions à entreprendre pour construire progressivement une solution.
- Examiner les différentes séquences d'actions menant à un état but et choisir la meilleure (Recherche).
- Accomplir la séquence d'actions sélectionnée.

## 1.2 Recherche adversiale

La recherche adversiale correspond aux techniques de recherche pour les jeux à deux personnes tels que les échecs, les dames, Tic-Tac-Toe, etc …

Chaque joueur dispose de toutes les informations sur la partie, ce qui n'est pas le cas par exemple pour le "scrabble" ou le "monopoly", où les joueurs cachent leurs cartes.

# Chapitre 2

# Analyse et conception

Dans ce chapitre, je présente l'analyse et la conception de l'application. Elles ont été faites en utilisant la notation UML.

## 2.1 UML

UML est un langage graphique de modélisation des données et des traitements. C'est une formalisation très réussie de la modélisation objet utilisée en génie logiciel.

Il est l'accomplissement de la fusion des précédents langages de modélisation objet Booch, OMT Et OOSE.

### 2.1.1 Le formalisme d'UML

Le formalisme UML est composé de 13 types de diagrammes (9 en UML 1.3) [11]. UML n'étant pas une méthode, leur utilisation est laissée à l'appréciation de chacun, même si le diagramme des cas d'utilisation est généralement considéré comme l'élément central d'UML. De même, on peut se contenter de modéliser seulement partiellement un système, par exemple certaines parties critiques.
UML se décompose en plusieurs sous-ensembles

***Les vues :***
Les vues sont les observables du système. Elles décrivent le système d'un point de vue donné, qui peut être organisationnel, dynamique, temporel, architectural, géographique, logique, etc. En combinant toutes ces vues il est possible de définir (ou retrouver) le système complet.

***Les diagrammes*** :
Les diagrammes sont des éléments graphiques. Ceux-ci décrivent le contenu des vues, qui sont des notions abstraites. Les diagrammes peuvent faire partie de plusieurs vues.

***Les diagrammes de cas d'utilisation :***
Utilisés pour donner une vision globale du comportement

fonctionnel d'un système logiciel. Les deux composants principaux des diagrammes de cas d'utilisation sont les acteurs et les cas d'utilisation.

***Le diagramme de classes :***

C'est un schéma utilisé en génie logiciel pour représenter les classes et les interfaces d'un système ainsi que les différentes relations entre celles-ci.

## 2.2 Analyse

Le but de cette application est de créer le fameux jeu Ultimate Tic-Tac-Toe. En permettant l'utilisation des algorithmes qui permet de jouer conte l'intelligence artificielles.

Dans un premier temps, on va créer l'interface pour permet la manipulation graphique des components du jeu (Les panneaux, les boutons…), et en arrière-plan, on va appliquer les algorithmes qui permet d'adapter les algorithme d'intelligence artificielles.

Pour commencer une partie, il faut obligatoirement se connecter ou bien créer une compte.

## 2.3 Le diagramme des classes

Ce diagramme représente les éléments de modélisation statique : les classes, leur contenu et leurs relations.

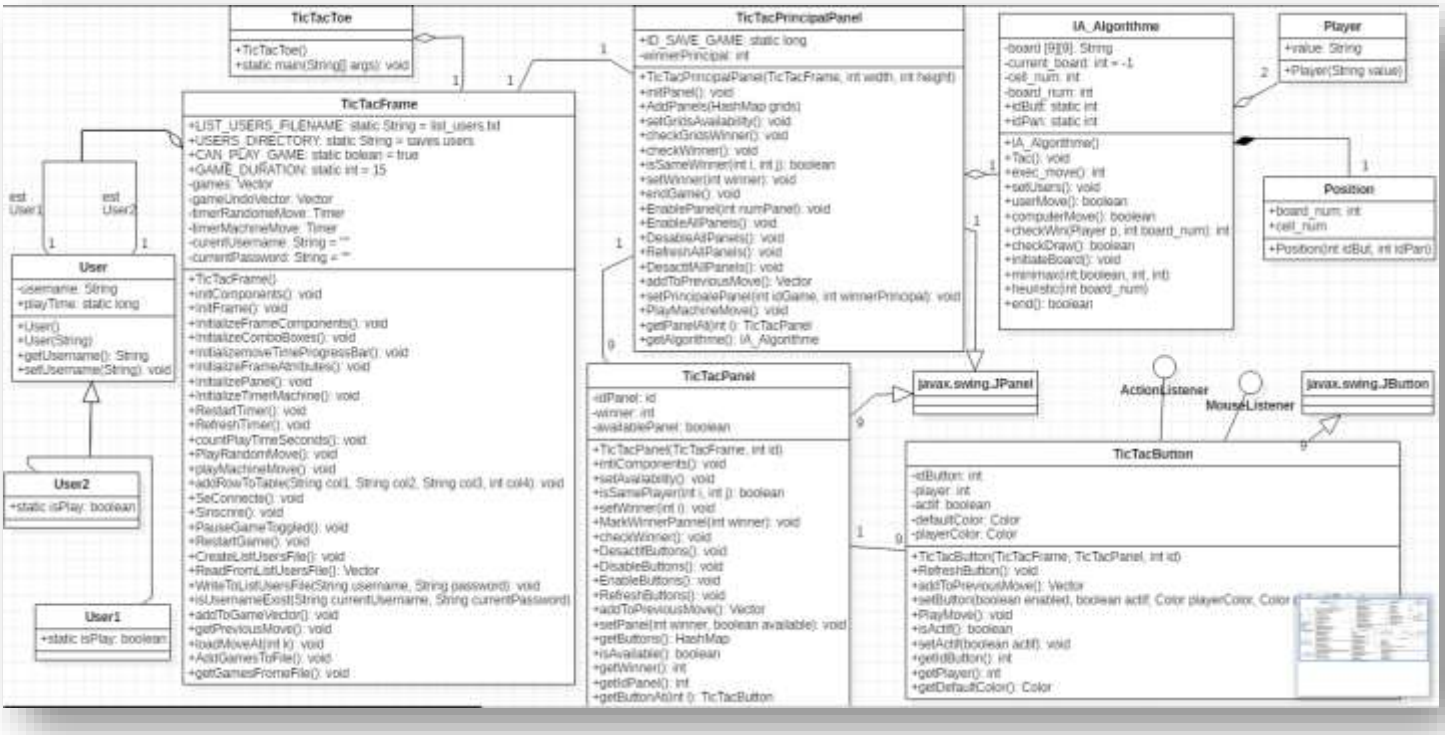



Figure 2.1 diagramme de cas d'utilisation.

### 2.3.1 Le diagramme des paquetages

Le diagramme des paquetages s'explique les différents paquetages utilisées dans l'application.

Dans notre cas, ces classes peut regroupées selon quatre principaux paquetages :

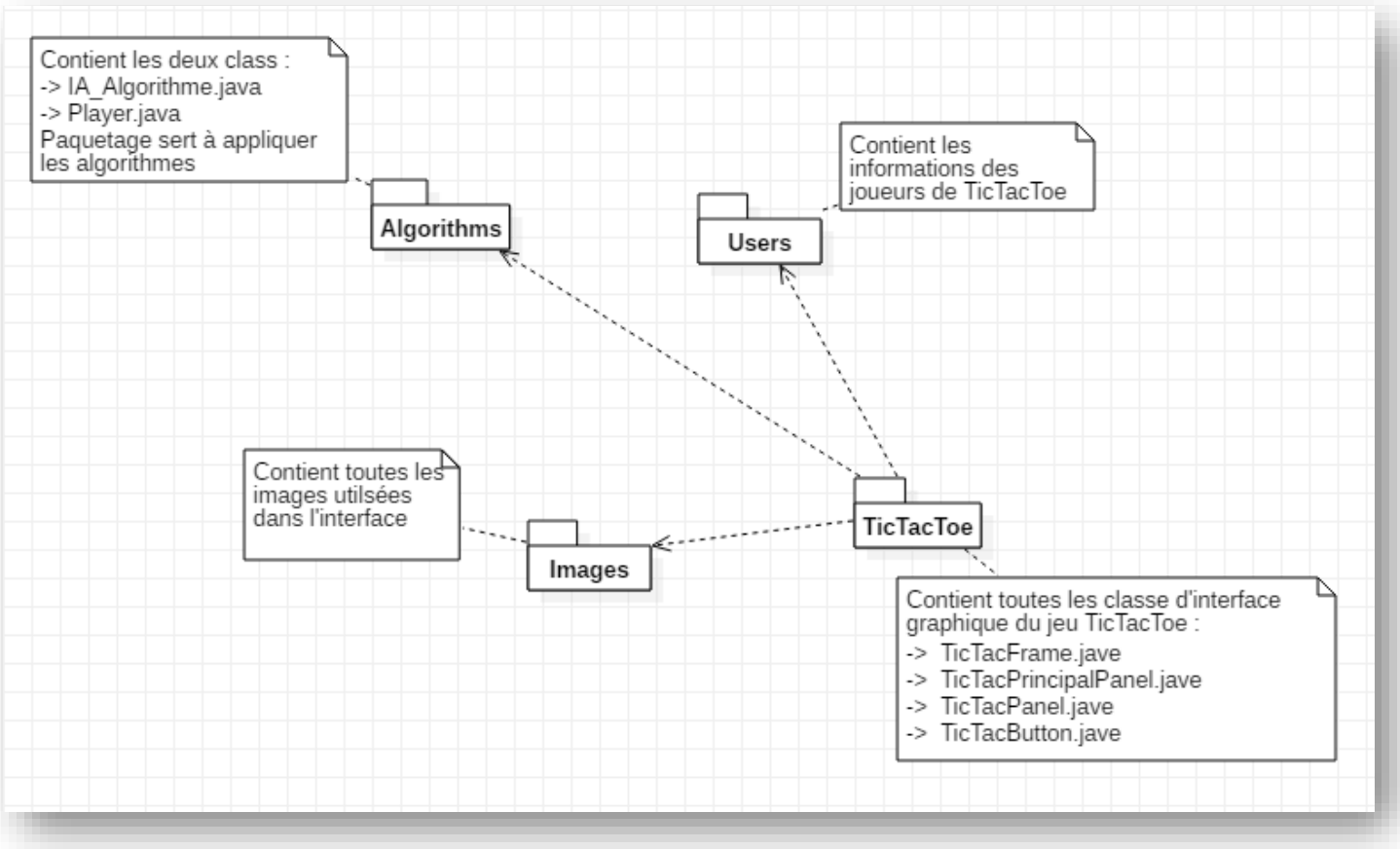



Figure 2.3 Le diagramme des paquetages.

Main paquetage : **TicTacToe**

Ce paquetage contient les classes principaux :

*TicTacToeFrame.java* qui le code principal de cette application, hérite du classe JFrame ; puis redéfinir ses fonctions qui permettent la manipulation de l'application, en utilisant les 3 classe *TicTacToePrincipalPanel.java, TicTacToePanel.java* et *TicTacToeButon.java* qui contient les caractéristique de nos actions.

**Algorithms** paquetage

Qui contient les différents algorithmes, comme MinMax, AlphaBeta, Heuristique, Helper... etc.

**Users** paquetage

Ce dernier paquetage contient toutes les informations des utilisateurs qui permettent le déroulement du jeu.

**Images** paquetage

Finalement le paquetage qui contient toutes les images de notre applications.

# Chapitre 3

# Interprétation des choix

## .1 Maquette de l'application

Dans ce qui suit, je vais présenter les interface de l'application pour bien comprendre ces mécanisme

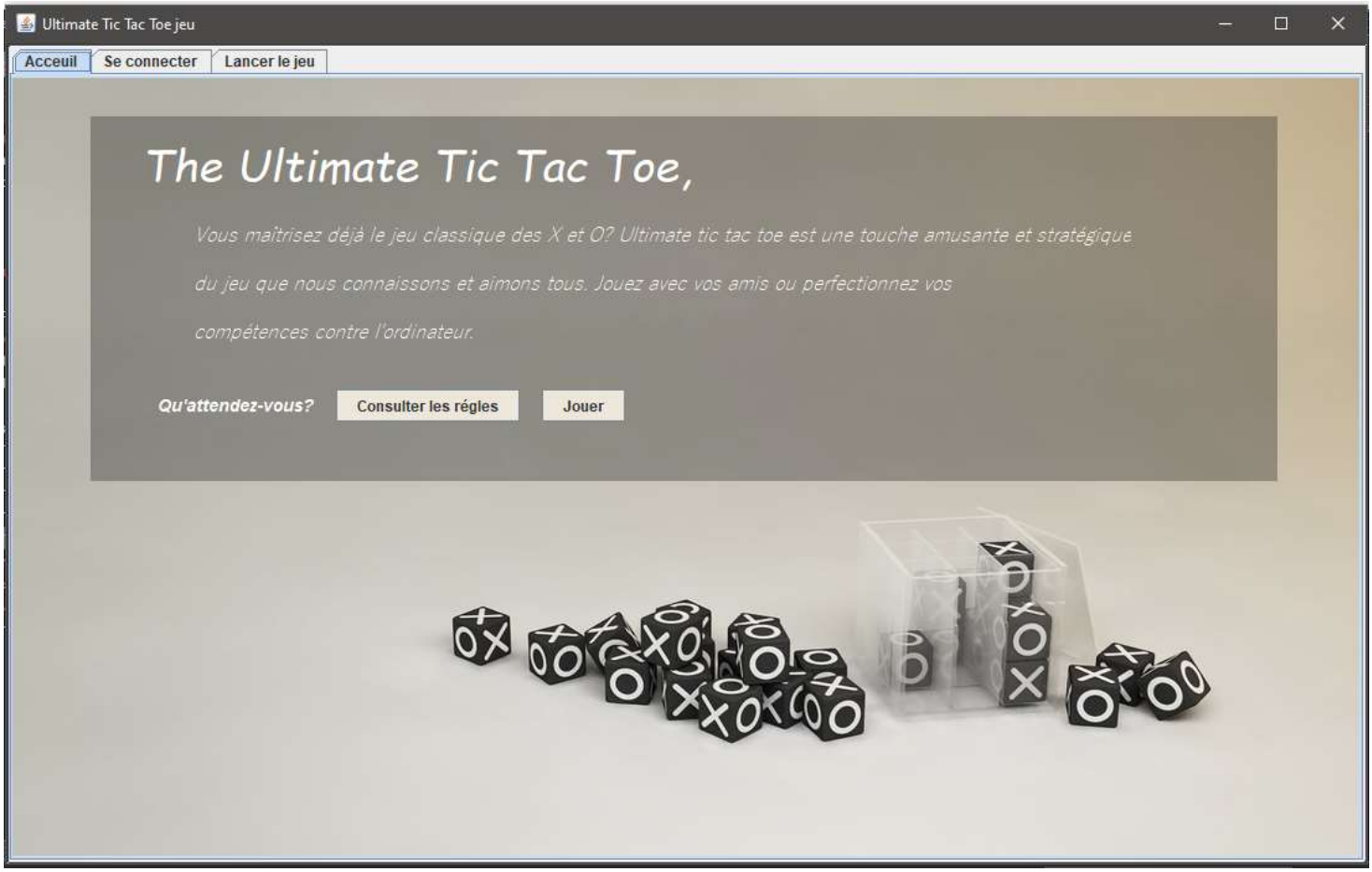



Figure 3.1 TicTacToe Main Page - Application.

### .1.1 Partie des Tabs

## *Se connecter Panel*

Cette interface permet l'authentification ou bien créer des comptes. Pour commencer à jouer.

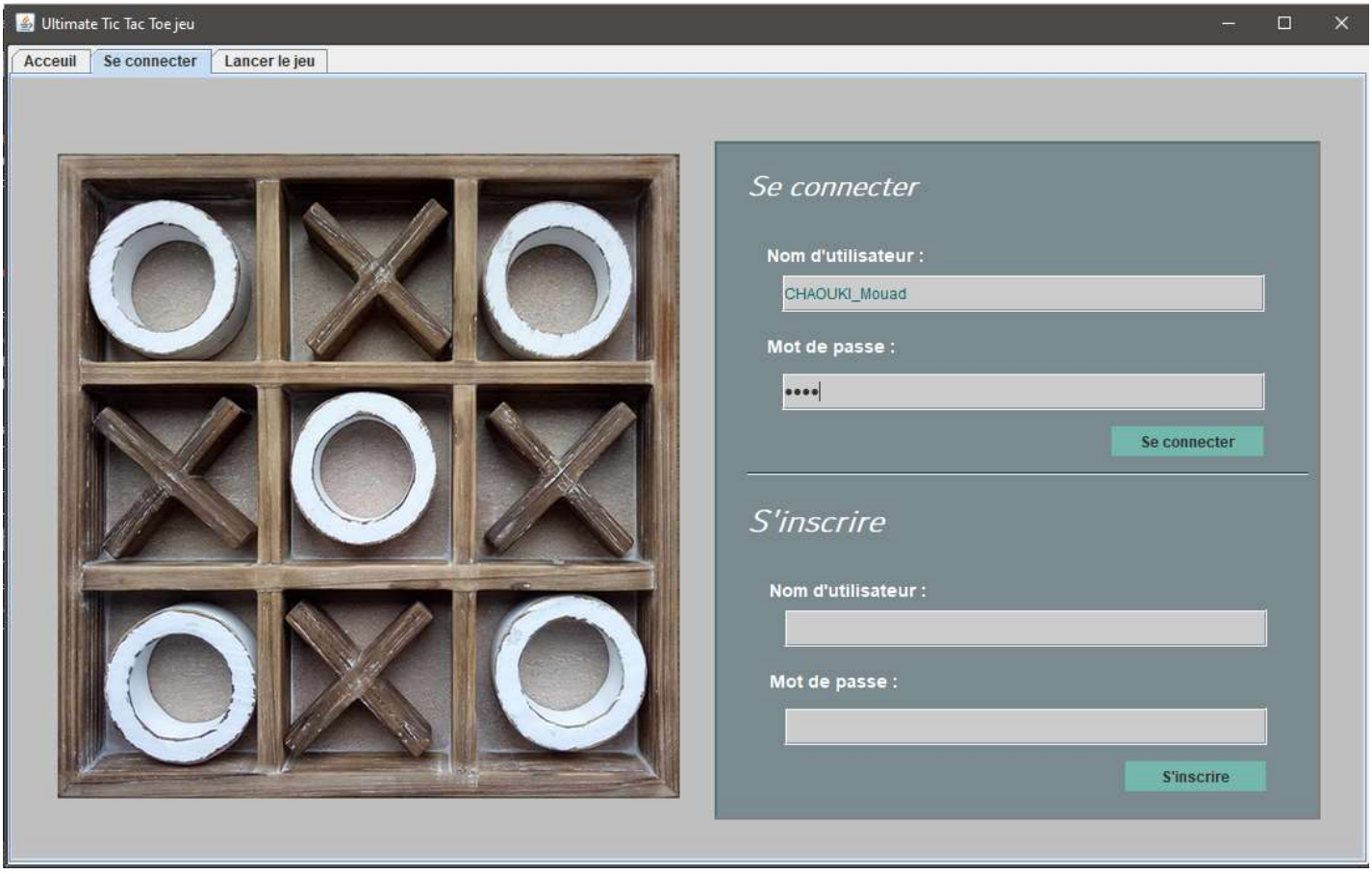



Figure 3.2 Se connecter Panel.

## **Lancer le jeu** Panel

Cette interface, nous permet de jouer, en spécifiant le :

- *Le mode de jeu (HommeVsHomme, HommeVsMachine).*
- *La contraint de victoire (Premier victoire, 2 successive victoire).*

- *Le minuteur pour jouer. En seconds.*

- *Le 2ème joueur.*

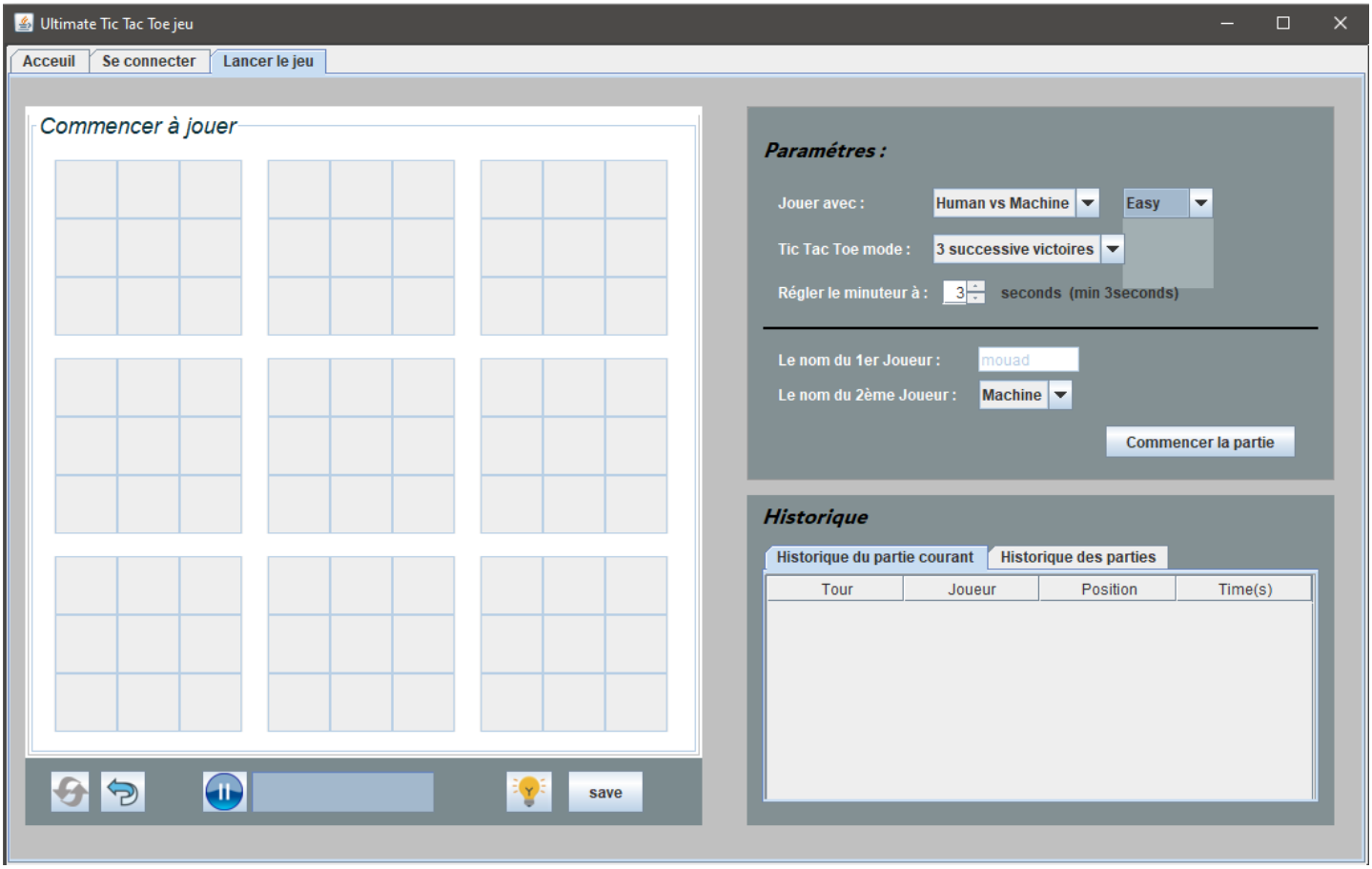



Figure 3.3 Lancer le jeu Panel.

## .1 Justification des choix

### .1.1 Algorithme MinMax

- L'Algorithme Minimax détermine la stratégie optimale pour Max.
- Générer l'arbre de recherche
- Appliquer la fonction utilité aux états terminaux

- Utiliser les valeurs de l'utilité de la couche courante pour calculer les valeurs utilité de chaque nœud de la couche ancêtre. Un nœud Max/Min calcule le maximum/minimum des valeurs utilité de ces enfants.
- Continuer jusqu'à ce que le nœud racine est atteint. Choisir l'action qui mène à l'enfant avec la plus haute valeur.
- La décision Minimax : maximise l'utilité pour Max en supposant que Min essaie de la minimiser.

```
public scoreStates minimax(int depth, boolean isMaximizingPlayer, int board_num, int alpha, int beta){

        int max = -99999;
        int min = 99999;
        int curr_val =0;
        int best_val;
        int best_cell = -1;
        List<Position> emptyStates = emptySpace(board_num);//gets list of all empty states

        if(depth == TicTacFrame.MACHINE_NIVEAU || emptyStates.isEmpty() || end()){
                best_val = heuristic(board_num, true);
                return new scoreStates(best_val, new Position(board_num, best_cell));

        }
        String move_value = (isMaximizingPlayer) ? computer.value : user.value;//gets the value of the move
                //Print statement for depth
                for(int i=0; i<depth; i++){
                        System.err.print("      ");
                }
                if(isMaximizingPlayer){
                        for(Position state : emptyStates){
                                board[state.board_num][state.cell_num] = move_value;
                                int original_val = value_array[board_num];
                                curr_val = heuristic(board_num, true) + minimax(depth+1, !isMaximizingPlayer, sta
```

## .1.2 Fonction Alpha-Beta

- Explorer l'arbre en profondeur d'abord
- Pour un nouveau nœud :
  - Si le nœud est une feuille, alors appliquer la fonction d'évaluation.
  - Si le n'est pas une feuille et n'a pas encore reçu de valeur, alors on lui donne -∞, s'il s'agit d'un nœud de la couche Max, ou +∞, s'il s'agit d'un nœud de la couche Min.
- Lorsqu'on donne une valeur à une feuille en utilisant la fonction d'évaluation, la valeur est

distribuée à ses ancêtres.

- Lorsqu'on visite un nœud, on décide de le tailler s'il n'y a pas de raison de l'étendre.

```
if(isMaximizingPlayer){
        for(Position state : emptyStates){
                board[state.board_num][state.cell_num] = move_value;
                int original_val = value_array[board_num];
                curr_val = heuristic(board_num, true) + minimax(depth+1, !isMaximizingPlayer, state.cell_num, alpha, beta).score;
                max = Math.max(curr_val, max);
                alpha = Math.max(curr_val, alpha);
                System.err.println("score is: " + curr_val + " on board "+ board_num+ " cell: " + state.cell_num);
                if (curr_val ==alpha){//if we took the curr_val, this is the best cell to take
                        alpha = curr_val;
                        best_cell = state.cell_num;

                }
                value_array[board_num]=original_val;
                board[board_num][state.cell_num]=" ";//undo move
                if (alpha >= beta) break;//pruning the tree

        }
}else{//if it is not the maximizing player
```

## .1.3 Fonction Heuristique et Helper

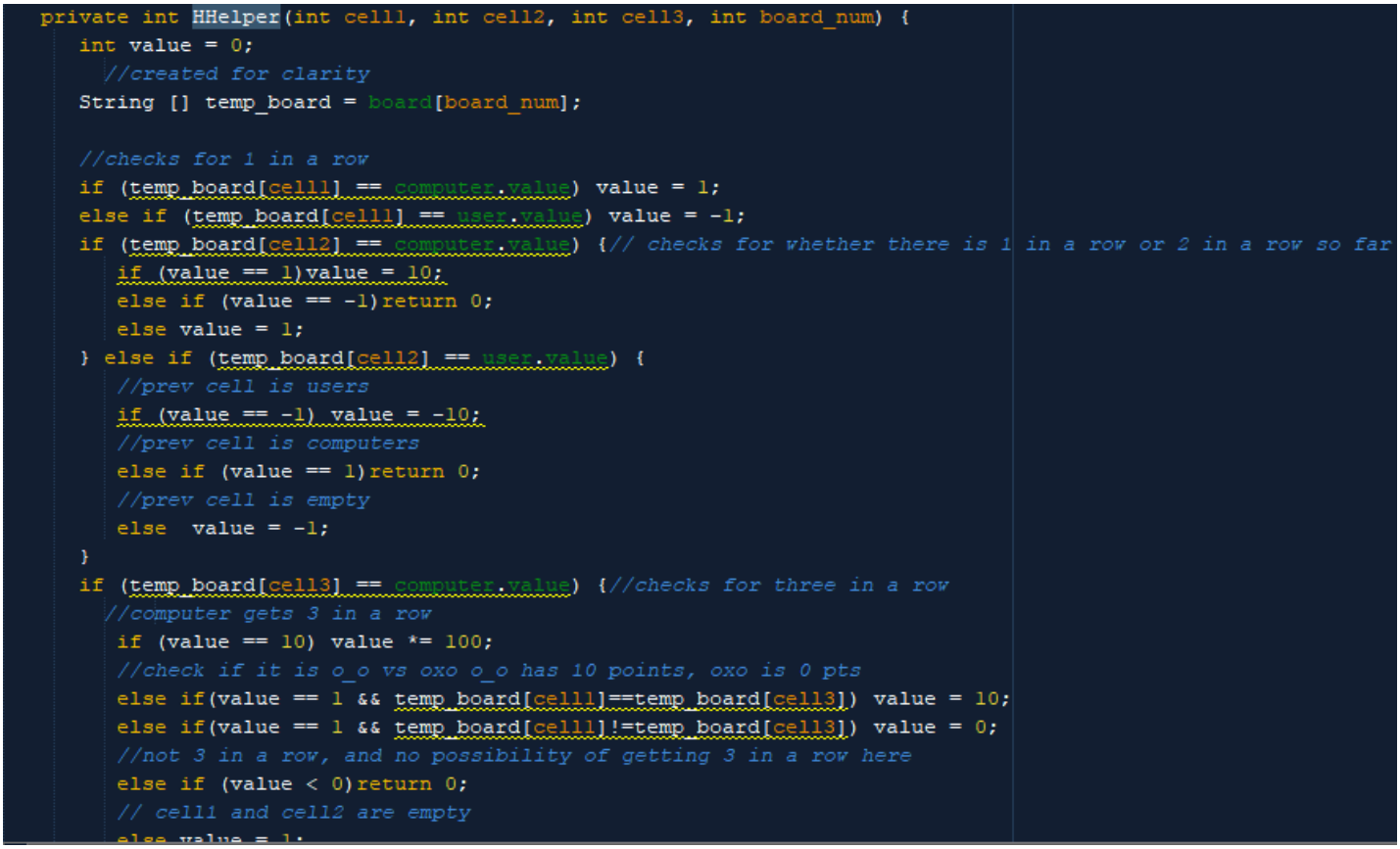
```
    private int heuristic(int board_num, boolean isMaximizingPlayer) {
        int value = 0;

        value += HHelper(0, 4, 8, board_num);// diagonal UL - BR
        value += HHelper(6, 4, 2, board_num);// diagonal BL - UR

        //Down
        value += HHelper(1, 4, 7, board_num);
        value += HHelper(0, 3, 6, board_num);
        value += HHelper(2, 5, 8, board_num);

        //Across
        value += HHelper(3, 4, 5, board_num);
        value += HHelper(6, 7, 8, board_num);
        value += HHelper(0, 1, 2, board_num );

        //update value
        int original_val = value_array[board_num];
        value_array[board_num] = value- original_val ;//gets the difference
        //returns positive if value is bigger, negative if value is smaller
        return value;
    }
```



```
private int HHelper(int cell1, int cell2, int cell3, int board_num) {
    int value = 0;
      //created for clarity
    String [] temp_board = board[board_num];

    //checks for 1 in a row
    if (temp_board[cell1] == computer.value) value = 1;
    else if (temp_board[cell1] == user.value) value = -1;
    if (temp_board[cell2] == computer.value) {// checks for whether there is 1 in a row or 2 in a row so far
       if (value == 1)value = 10;
       else if (value == -1)return 0;
       else value = 1;
    } else if (temp_board[cell2] == user.value) {
       //prev cell is users
       if (value == -1) value = -10;
       //prev cell is computers
       else if (value == 1)return 0;
       //prev cell is empty
       else  value = -1;
    }
    if (temp_board[cell3] == computer.value) {//checks for three in a row
      //computer gets 3 in a row
       if (value == 10) value *= 100;
       //check if it is o_o vs oxo o_o has 10 points, oxo is 0 pts
       else if(value == 1 && temp_board[cell1]==temp_board[cell3]) value = 10;
       else if(value == 1 && temp_board[cell1]!=temp_board[cell3]) value = 0;
       //not 3 in a row, and no possibility of getting 3 in a row here
       else if (value < 0)return 0;
       // cell1 and cell2 are empty
       else value = 1;
```

C’est deux fonctions permet d’améliorer le raisonnement du machine pour une meilleur action.

# Conclusion et perspectives

Le travail qui a été réalisé durant ce projet sort du commun. En effet, il s'agissait de travailler sur un sujet de haut niveau qui touche au fond toutes le domaine d'intelligence artificielle, à savoir effectuer des calculs pour manipuler les différents données.

Outils de développement, concepts mathématiques tout a été mêlé pour produire un livrable satisfaisant, c'est-à-dire un projet qui vous permettez de jouer contre la machine dans le jeux du TIC-TAC-TOE.

Le processus de réalisation n'a pas été dépourvu d'obstacles spécialement les taches de permettre à la machine de jouer. D'autre part, il a fallu beaucoup de patience et d'attention quand il s'agissait par exemple d'appliquer les algorithmes MIN-MAX et la fonction d'évaluation pour mesurer les heuristiques des meilleurs valeurs pour jouer.