

Rapport de conception du Jeu d’échecs

**Mark KPAMY & Fabien OGLI**

Table des matières

1. [Introduction 2](#_Toc483814346)
2. [Modèle de conception 2](#_Toc483814347)
3. [Algorithmes d’IA 2](#_Toc483814348)
4. [Conclusion 2](#_Toc483814349)

# Introduction

Ce rapport présente nos travaux sur la réalisation d’un jeu d’échecs doté d’une intelligence artificielle avec plusieurs niveaux de difficultés.

# Modèle de conception

# Algorithmes d’IA

Les algorithmes d’IA que nous avons sont principalement l’algorithme MIN-MAX combiné avec l’algorithme alpha-bêta. Ces deux algorithmes sont les plus utilisés pour la recherche du meilleur coup à jouer. Il est donc notamment très en vogue chez les développeurs de jeux de réflexion (échecs, dames, go, etc). En dépit de ses avantages et sa popularité, l’algorithme MIN-MAX tient son inconvénient dans la récursivité. Le temps de parcours peut s’avérer très long si la profondeur est élevée. Dans notre cas :

* Pour une profondeur de 2 coups, l’IA fait son choix en moins de 3 secondes en début de partie
* Pour une profondeur de 4 coups, l’IA fait son choix en environ 10 secondes en début de partie
* Pour une profondeur de 6 coups, on avoisine à peu près 20 secondes

Nous allons maintenant présenter notre classe IA.

## Présentation de la classe IA

Notre classe IA est le moteur d’intelligence artificielle de notre jeu d’échecs. Ses principales méthodes sont :

ia(int level);

void setLevel(int i);

int getLevel();

QVector<QPoint> jouer(Joueur \*joueur,Plateau \*plateau);

int eval(int couleur[8][8],char idPiece[8][8]);

QVector<QPoint> calc\_echec\_et\_mat(int idJoueur,QPoint pos\_rois\_joueur);

void initTableauTmp(Plateau \* plateau);

int max(int idJoueur,int profondeur,int alpha,int beta,int couleur[8][8],char idPiece[8][8]);

int min(int idJoueur,int profondeur,int alpha,int beta,int couleur[8][8],char idPiece[8][8]);

int gagnantEnCours(int idJoueur);

~ia();

Le detail des paramètres de ces fonctions pourra être retrouvé dans le code source.

## Fonction jouer()

Cette fonction constitue le point d’entrée de l’algorithme. C’est lui qui parcourt les pièces du joueur et élague les branches.

## Fonction min()

Cette fonction permet de simuler tous les coups possibles de l’adversaire.

## Fonction max()

Cette fonction permet de simuler tous les coups possibles de l’IA.

## Fonction gagnantEncours()

Cette fonction permet de savoir à un instant donné qui a l’avantage, c’est le gagnant de la situation.

## Fonction calc\_echec\_et\_mat()

Cette fonction permet de calculer la fin ou la continuation de la partie. Si le jeu n’est pas encore terminé car le joueur adverse ou l’IA ont encore des coups possibles, il retourne 0 sinon il retourne -1000 ou 1000 suivant que le joueur qui joue a perdu ou gagné.

## Fonction eval()

Cette fonction permet d’évaluer le plateau. On attribue à chaque pièce une valeur fixe (Pion = 1, Fou=3, T=5) . On n’ajoute pas le roi car il ne peut être pris. Ensuite, on calcule le score obtenu par chaque joueur à travers ses pièces, cela permet d’avoir une idée du sort pour chaque joueur et donc de comparer les situations pour en choisir la plus avantageuses.

# Interface graphique

L’interface graphique de notre jeu a été réalisée sous Qt Creator. Cette interface permet de visualiser et faire les déplacements, de lister les déplacements effectués, faire le choix du niveau de difficulté. Cette interface reste assez basique car elle n’était pas le but principal du projet. Nous avons tout de même apprécié utiliser ce logiciel car sa librairie graphique est très facile à prendre en main et la gestion des signaux est plus que jamais simplifiée.

# Conclusion

La réalisation de ce jeu nous a permis de mûrir nos réflexions sur le fonctionnement et l’implémentation d’une intelligence artificielle. Il fût aussi pour nous un challenge car nous avons décidé de le coder en C++ avec une interface graphique réalisé avec Qt Creator. Nous avons vite compris les défis auxquelles se heurtent les développeurs de jeu en termes de temps de réponse, d’efficacité et de complexité. Les problèmes auxquelles nous avons été confrontés sont principalement l’amélioration du temps de décision de l’IA. Pour résoudre cela, nous avons simplifier tout notre code et utiliser des variables assez légères comme des tableaux d’entiers et de caractères.