UniLaSalle Amiens

Projet I1 - 2025

PROJET ECHECS

ETUDIANTS: PORTOIS Loïc
GUILBERT Baptiste
MARQUET Abel

ENSEIGNANTS: RUSSO Marjorie

CONFÉRENCE DES GRANDES ECOLES — EXCELLENCE FOR A COMPLEX WORLD

ETABLISSEMENT HABILITE PAR LA COMMISSION DES TITRES D'INGENIEUR ET MEMBRE DE LA CONFERENCE DES GRANDES ECOLES

CTI — Commission des Titres d'Ingénieur

14 quai de la Somme – BP 10100
80082 AMIENS CEDEX 2
Tél. : 03.22.66.20.00 – Fax : 03.22.66.20.10
https://amiens.unilasalle.fr/

CCI AMIENS PICARDIE

amiens & MÉTROPOLE

Région Hauts-de-France

UniLaSalle Amiens

Sommaire :

1. Introduction

1.1. Contexte et Ambition du Projet

Ce projet est plus qu'un simple exercice de programmation ; il représente la concrétisation d'une ambition partagée : celle de bâtir une simulation d'échecs complète et performante à partir de zéro. Notre projet a été réalisé en langage C (ce qui faisait partie du cahier des charges), associé à la bibliothèque SDL2 pour la gestion de l'interface graphique et des interactions. Le projet a été pensé pour offrir une expérience immersive et gratifiante, s'adressant aussi bien aux stratèges chevronnés désireux de tester leurs limites contre l'ordinateur qu'aux joueurs occasionnels recherchant une partie conviviale.

1.2. Cahier des Charges et Objectifs Initiaux

Le projet a été initié avec un cahier des charges précis visant à développer un jeu d'échecs complet et fonctionnel. Les objectifs fondamentaux étaient les suivants :

- **Implémentation des Règles Complètes :** Assurer une application rigoureuse de toutes les règles officielles du jeu d'échecs, incluant les mouvements spéciaux.
- **Système de Sauvegarde et Chargement :** Permettre aux joueurs de suspendre une partie et de la reprendre ultérieurement sans perte de progression.
- **Interface Graphique Fonctionnelle :** Créer une interface utilisateur claire et intuitive en utilisant la bibliothèque SDL2, capable de présenter toutes les informations de jeu de manière lisible.
- **Intelligence Artificielle (IA) :** Développer une IA capable d'offrir un défi réel au joueur, au-delà d'un simple adversaire jouant des coups aléatoires.

2. Architecture Logicielle et Moteur de Jeu

2.1. Principes d'Architecture Modulaire

La conception du jeu repose sur une architecture modulaire rigoureuse, où chaque grande fonctionnalité est isolée dans une section de code dédiée ainsi que plusieurs fichiers .c lier par des fichiers .h. Cette approche garantit une meilleure organisation, facilite la maintenance et les évolutions futures. L'architecture s'articule autour des modules suivants :

- Un **cœur applicatif** qui orchestre le déroulement général, de l'initialisation à la fermeture.
- Un **moteur de règles d'échecs** garant de l'authenticité du jeu.
- Un **module graphique** dédié à l'affichage et à l'interaction visuelle via SDL2.
- Un **module d'intelligence artificielle** qui analyse l'échiquier et joue les coups.
- Un **système de persistance des données** pour les sauvegardes.

2.2. La Machine à États Principale

Le programme est piloté par une machine à états depuis la fonction (`main`). Cette machine gère les différents écrans et contextes du jeu (menu principal, partie en cours, menu de sauvegarde, etc.) et distribue les commandes aux modules spécialisés. Cette structure permet de gérer de manière claire les transitions entre les différentes phases du jeu.

Algorigramme: https://dlul.portois.fr/-nrQ4iiuKyQ/main_algo.png

(Tous les liens présent dans ce rapport mène à un serveur personnel permettant d'héberger des jeux des programmes des images ETC)

2.3. Le Moteur de Règles d'Échecs

2.3.1. Représentation des Données

Le cœur du moteur est une structure de données (`EtatEchiquier`) qui centralise l'état global du jeu. Elle contient une représentation interne de l'échiquier, la position et l'identification (type, couleur) de chaque pièce, ainsi que le suivi des informations contextuelles essentielles comme le joueur dont c'est le tour, les droits au roque, la possibilité de prise en passant et les compteurs pour les règles de nullité.

2.3.2. Validation des Coups

Un contrôle rigoureux des déplacements est appliqué pour garantir la légalité de chaque coup. Ce processus interdit formellement tout mouvement qui mettrait ou laisserait son propre roi en situation d'échec. La validation s'opère en vérifiant la géométrie du mouvement propre à chaque pièce, l'absence d'obstructions sur la trajectoire et la validité des captures.

2.3.3. Implémentation des Règles Spéciales

Toutes les subtilités du jeu d'échecs ont été implémentées avec précision pour une fidélité totale aux règles officielles.

- **Coups Spéciaux :** Le moteur gère le roque/grand roque (en vérifiant toutes ses conditions), la prise en passant et la promotion des pions, pour laquelle une interface de choix est présentée au joueur.

- **Conditions de Fin de Partie :** Le jeu détecte automatiquement les situations d'échec, d'échec et mat, et de pat. Les diverses conditions de partie nulle sont également gérées, incluant la règle des 50 coups, la triple répétition de position et le matériel insuffisant pour mater.

Algorigramme de la logique du jeu : https://dlul.portois.fr/-YJBSnnm3JS/logique_algo.png

3. Interface Utilisateur Graphique (GUI) avec SDL2

3.1. Initialisation et Gestion des Ressources

L'interface visuelle est entièrement gérée par la bibliothèque SDL2 et ses extensions. La phase d'initialisation met en place la fenêtre principale, le système de rendu optimisé et charge les modules complémentaires SDL_image (pour les textures des pièces) et SDL_ttf (pour les polices de caractères). Toutes les ressources graphiques sont chargées en mémoire au démarrage pour garantir des transitions fluides.

Algorigramme de sdl : https://dlul.portois.fr/-G9YgubT38Z/graphics_algo.png

3.2. Boucle de Rendu et Affichage Dynamique

L'écran est redessiné à chaque "frame" pour refléter l'état actuel du jeu. La boucle de rendu affiche l'échiquier, positionne chaque pièce, puis superpose des indicateurs visuels dynamiques essentiels à la clarté du jeu. Ces surlignages incluent la mise en valeur de la pièce sélectionnée, les cases de destination valides, le tracé du dernier coup joué et une alerte visuelle sur la case du roi en cas d'échec.

3.3. Gestion des Interactions Utilisateur

SDL2 capte toutes les actions du joueur (clics de souris, frappes au clavier) via une boucle d'événements. Ces "événements" sont interprétés pour permettre au joueur de sélectionner ses pièces, jouer ses coups en cliquant sur la case de destination, naviguer dans les menus ou saisir du texte pour nommer ses sauvegardes ou les joueurs.

3.4. Conception des Menus et Panneaux

L'expérience utilisateur est structurée autour de plusieurs écrans et panneaux.

- **Menus :** Un menu principal oriente l'utilisateur vers les différents modes de jeu, le chargement d'une partie ou la consultation des règles. Des menus dédiés gèrent la sauvegarde, le chargement et l'édition des parties.

- **Panneau d'Information :** Durant une partie, un panneau latéral affiche en permanence les noms des joueurs et le tour actuel, et contient les boutons d'accès rapide, notamment pour la sauvegarde.

4. Système de Persistance des Données (Sauvegarde et Chargement)

4.1. Structure des Fichiers de Sauvegarde

Pour ne jamais perdre une partie, le jeu intègre un système de sauvegarde performant. L'état complet du jeu est capturé dans une structure dédiée, incluant l'échiquier, le tour du joueur, les divers indicateurs de règles et les noms des joueurs. Cette structure est ensuite écrite directement dans un fichier binaire pour un chargement rapide et fiable.

4.2. Mécanisme et Fonctions

Le mécanisme repose sur des fonctions de lecture et d'écriture binaire (`read`/`fwritef`). Une validation de l'intégrité des fichiers est effectuée pour s'assurer que les données chargées ne sont pas corrompues.

Algorigramme de save/load : https://dlul.portois.fr/-L2AXquYqF4/saveload_algo.png

4.3. Interface de Gestion

Le joueur dispose d'une interface complète pour gérer ses parties. Le jeu propose 5 emplacements de sauvegarde. L'utilisateur peut lister les parties existantes, les charger, les renommer ou les supprimer directement depuis un menu dédié.

5. Intelligence Artificielle (IA)

5.1. Introduction à l'IA de "ChessGame"

L'IA a été conçue pour offrir un défi réel et stimulant. Elle n'effectue pas de coups aléatoires mais anticipe les actions du joueur sur plusieurs tours en analysant des milliers de positions possibles pour sélectionner le coup qu'elle juge optimal.

5.2. La Fonction d'Évaluation : Le Jugement Positionnel

5.2.1. Principes Fondamentaux

Le cœur de la décision de l'IA réside dans sa capacité à "noter" une position sur l'échiquier. Cette fonction d'évaluation attribue un score numérique qui représente l'avantage ou le désavantage d'un joueur, permettant à l'IA de comparer différentes suites de coups.

5.2.2. Composantes de l'Évaluation

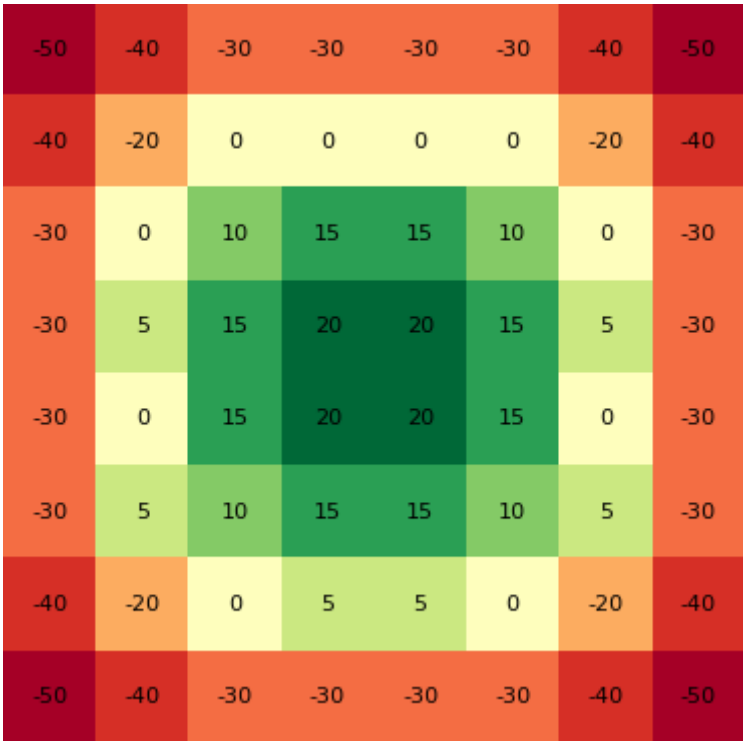
L'évaluation est une combinaison de plusieurs facteurs stratégiques :

- **Matériel :** La valeur brute des pièces restantes sur l'échiquier.

♙	1
♘	3
♗	3
♖	5
♕	9

- **Positionnement Stratégique :** L'influence d'une pièce varie selon sa position. L'IA utilise des tables de positionnement (PST - Piece-Square Tables) qui accordent des bonus ou des malus à chaque pièce en fonction de la case qu'elle occupe.

  Table des Cavaliers :

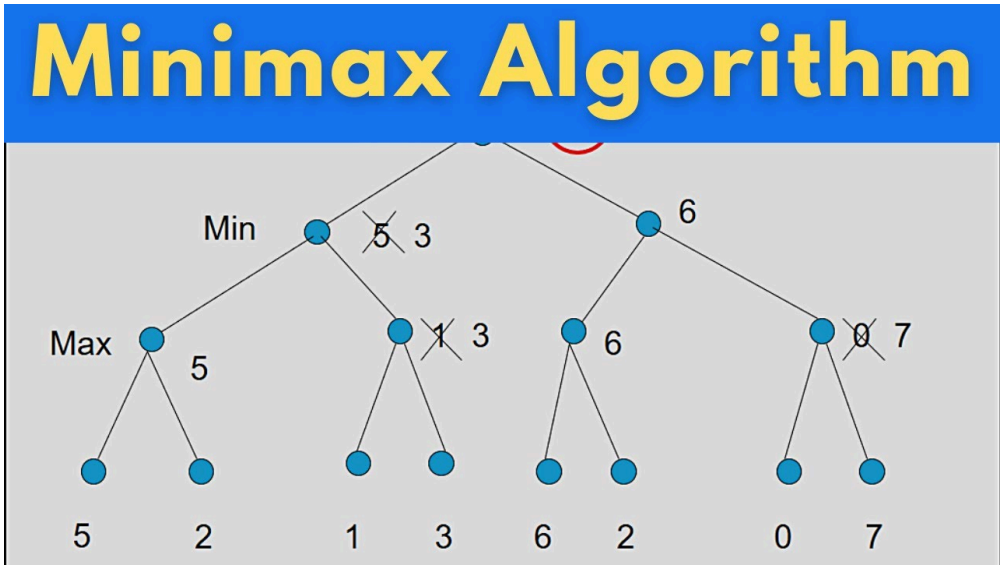


- **Bonus Spécifiques :** L'évaluation est affinée par des bonus pour des avantages positionnels clés comme la possession de la paire de fous, des tours actives sur des colonnes ouvertes, des pions passés et la sécurité du roi.
- **Adaptation à la phase de jeu :** La logique d'évaluation s'adapte pour la fin de partie, où, par exemple, la position du roi devient plus agressive.

5.3. Algorithme de Recherche de Coups

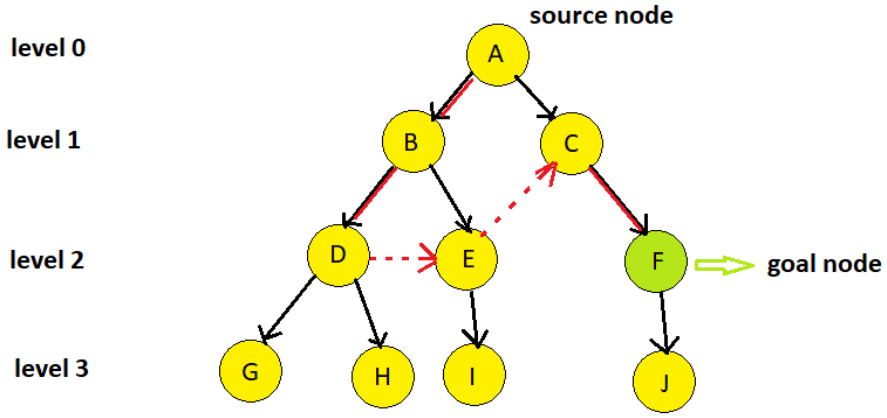
5.3.1. Exploration Stratégique : Minimax avec Élagage Alpha-Bêta

L'IA explore un "arbre" de possibilités sur plusieurs coups de profondeur. L'algorithme Minimax est utilisé pour cette exploration. Pour optimiser ce processus, une technique d'élagage Alpha-Bêta est implémentée. Elle permet d'ignorer des branches entières de l'arbre de recherche qui sont manifestement moins bonnes que des options déjà trouvées, réduisant de manière significative le nombre de positions à évaluer et permettant à l'IA de "voir" plus loin dans la partie.



5.3.2. Approfondissement Itératif (Iterative Deepening)

Plutôt que de lancer une seule recherche profonde, l'IA procède par itérations : elle cherche d'abord à une profondeur de 1 coup, puis 2, puis 3, et ainsi de suite. Cette approche permet de gérer le temps de réflexion alloué et garantit que l'IA a toujours un "meilleur coup" trouvé à une profondeur inférieure, même si elle est interrompue. Cela améliore également l'efficacité de l'élagage Alpha-Bêta pour les recherches plus profondes.



DFS: A->B->D->G->H->E->I->C->F
Depth Limit = 2
DLS: A->B->D->E->C->F

5.3.3. Analyse Tactique Approfondie (Recherche de Quiescence)

Dans les situations tactiques intenses (avec de nombreux échanges possibles), l'évaluation d'une position peut être trompeuse. Pour contrer " l'effet d'horizon ", l'IA effectue une recherche de quiescence : elle prolonge l'analyse au-delà de la profondeur fixée, mais en n'explorant que les coups "tactiques" (captures, promotions, échecs) jusqu'à ce que la position devienne "calme". C'est seulement alors que l'évaluation finale est effectuée, garantissant une meilleure stabilité tactique.

5.3.4. Mémoire à Long Terme (Tables de Transposition)

Pour éviter de recalculer des millions de fois les mêmes positions, l'IA utilise une grande "mémoire" appelée table de transposition. Chaque position analysée reçoit une "signature numérique" unique (via une technique de hachage Zobrist) et son évaluation est stockée. Si l'IA rencontre à nouveau cette position dans son exploration, elle réutilise l'information stockée, gagnant ainsi un temps de calcul considérable.

Algorigramme de l'ia : https://dlul.portois.fr/-oUe6GyCCtf/ia_algo.png

6. Projets Complémentaires et "Bonus"

6.1. Version Web en JavaScript

En parallèle du projet principal, une adaptation du jeu en JavaScript, HTML et CSS a été développée comme une initiative personnelle. L'objectif était de rendre le jeu directement accessible dans un navigateur. Cette version est fonctionnelle et disponible en ligne, démontrant la portabilité des concepts logiques du jeu.

La difficulté est réglable en choisissant quelle sera la profondeur de recherche maximale.

https://chess.portois.fr/

6.2. Exploration en Apprentissage par Renforcement

Un mini-projet exploratoire a été mené pour appliquer les concepts d'IA à l'apprentissage par renforcement.

- **Niveau 1 (Terminé) :** Implémentation du jeu de Morpion en Python et entraînement d'une IA avec la bibliothèque Tensorflow.

  vous pouvez tester l'IA a se lien :

  https://www.portois.fr/Tictactoe/

- **Niveau 2 (En cours) :** Application de la même logique au jeu d'échecs.

6.3. Portabilité sur Console de Jeu

Pour donner une concrétisation physique au projet, le jeu a été porté sur une console portable, et une version est téléchargeable pour tous.

La console a été réalisée avec un raspberry pi 5.

7. Environnement Technique

7.1. Outils de Développement

- IDE : Visual Studio Code
- **Compilateur :** GCC (GNU Compiler Collection)

7.2. Langages et Bibliothèques

- **Langages** : C, HTML, CSS, JavaScript, Mermaid (pour les algorigrammes)
- **Bibliothèques :** SDL2, SDL_image, SDL_ttf

8. Sources et Documentation

Le développement s'est appuyé sur une variété de sources :

- La documentation officielle de SDL2 et les références du langage C.
- Des forums et communautés en ligne comme Stack Overflow.
- Des communautés dédiées à la programmation d'échecs et d'IA.
- Des tutoriels vidéo et des articles sur des concepts spécifiques comme les algorithmes de jeu (Minimax, Alpha-Bêta) et le Zobrist Hashing.
- Youtube pour les tutoriels et explications.

9. Conclusion

9.1. Bilan du Projet et Atteinte des Objectifs

Notre projet a atteint tous ses objectifs initiaux. Il en résulte un jeu d'échecs complet et fonctionnel, doté d'une interface graphique agréable, d'un système de sauvegarde robuste et d'une intelligence artificielle très forte.

9.2. Apports Pédagogiques et Esprit Critique

Ce projet a été une expérience d'apprentissage extrêmement riche, permettant de renforcer la maîtrise du langage C, de la bibliothèque SDL2, des algorithmes d'IA complexes et de la gestion de projet en équipe. Les principaux défis ont résidé dans la complexité de l'IA.

9.3. Perspectives d'Amélioration

Plusieurs axes d'amélioration sont envisagés pour le futur :

- Amélioration des performances et de l'évaluation de l'IA.
- Implémentation d'un mode multijoueur en réseau.
- Améliorations de l'interface utilisateur et de l'expérience de jeu (UI/UX), comme l'ajout de thèmes ou de sons.
- Intégration de l'IA entraînée par apprentissage par renforcement une fois celle-ci finalisée.

>Lien de téléchargement du jeu :

https://dlul.portois.fr/-gzBsGNj6kz/ChessV13.zip