RAPPORT DU PROJET C

Variante des Echec : Capablanca

ISEN
ALL IS DIGITAL!
LILLE

yncréa

Introduction

Dans la cadre de notre 1ère année en cycle ingénieur en apprentissage à Isen Lille, nous avons comme projet de développer une variante des jeux d'échec, le Capablanca.

Les échecs Capablanca sont un jeu se jouant sur un échiquier 10x8. Le jeu fut nommé d'après son inventeur, l'ancien champion du monde d'échecs, José Raúl Capablanca (1888-1942). En plus des pièces standard, chaque joueur dispose de deux pièces féeriques, à savoir :

Une Impératrice (ou Chancelier), qui combine les mouvements de la tour et du cavalier ;
Une Princesse (ou Centaure), qui combine les mouvements du cavalier et du fou.

Le projet devra permettre de jouer une partie d'échecs Capablanca, selon les contraintes suivantes :
— Joueur contre Joueur
— Joueur contre IA
— IA contre IA

L'IA devra être basée ou bien sur une implémentation d'une IA existante, ou bien sur un modèle de jeu que vous implémenterez nous-même.

Technologie & Compétence :

Le projet devra être codé en C (éventuellement C++ pour la partie graphique le code doit être de qualité et maintenable. Cela passe par la mise en place de commentaires, par une charte de nommage claire, mais aussi par des solutions dédiées à améliorer la lisibilité de votre code.

De Massary Aymeric
Marchand Tom
Dragon Gulian

Année 2019
Encadré par Mr Cros

Sommaire :

Partie 1 : Analyse du jeu

1.Les cas d'utilisations

Le programme se présente comme un .exe à lancer.
Dès le lancement, un menu s'affiche permettant au joueur de choisir un type de partie entre :

- Joueur vs IA
- Joueur vs Joueur
- IA contre IA

Après sélection du type de jeu, la partie se lance en suivant les règles basiques des échecs.

2.Les règles du jeu

Le joueur ayant les pièces blanches commencent.

Sur la sélection d'une pièce, les cases à sa portée sont en verts.

Toutes les pièces classiques, les 2 pièces supplémentaires par joueur ainsi que tous leurs déplacements sont implémentées.

Chaque joueur, s'il est dans les bonnes conditions peut effectuer un roque (grand ou petit).

Un pion arrivé en bout d'échiquier peut procéder à une promotion.

Partie II : Le guide de l'utilisateur

1.Présentation Générale

Ici le manuel qui permet au joueur de profiter pleinement de notre programme de capablanca.

Le programme prend en compte le déplacement du curseur.

Pour sélectionner dans le menu, il suffit de cliquer sur l'option désirée.

La sélection d'une pièce se fait sur clique sur la pièce désirée, lorsque la pièce est sélectionnée, s'affiche alors les cases à sa portée.

Sur fermeture de la fenêtre, le programme se stop et s'arrête proprement.

1.Déroulement du jeu

Après passage au menu et sélection du joueur, l'échiquier ainsi que les pièces sont affichés, et la partie est lancée.

Le programme se déroule jusqu'à la fin d'une partie : Jusqu'à ce qu'un joueur mette en échec le roi adverse en respectant les règles et les déplacements des pièces.

Plusieurs informations sont disponibles à la fin d'une partie :

- Le joueur vainqueur

3.Exécution du jeu

Le jeu a été développer sur linux à l'aide de SDL 1.2. Pour construire l'exécutable et lancer le jeu, il suffit de respecter les consignes suivantes :

- Se trouver dans le dossier "capablanca/Chess"
- Lancer la commande "make"
- Lancer la commande "./main"

Partie III : Le guide du programmeur

1. Structure principale et objets

Pour développer le programme, nous avons commencé par choisir les différents objets que nous allons créer ainsi que les informations qu'ils comporteraient. Ensuite, nous avons décrit étape par étape le déroulement d'une partie entière en détaillant les fonctions dont nous aurons besoin.

A. Objets

Chaque objet a été créé comme étant un pointeur sur une structure.

L'objet grille est l'objet contenant toutes les informations sur la partie, il comporte la grille (implémentée comme étant un tableau de pièces à une seule dimension de 8x10 cases) qui contient les toutes pièces en vie, le joueur dont c'est le tour de jouer, un tableau contenant les pièces ayant été tuées et le nombre total de morts.

L'objet pièce contient le type de la pièce, à quel joueur elle appartient, ses coordonnées et ses déplacements possibles.

L'objet déplacement est un objet contenu par chacune des pièces et qui contient un tableau de toutes les coordonnées où la pièce peut se déplacer et le nombre de fois où elle a été déplacée. Cet objet est rechargé à chaque tour afin de toujours avoir les déplacements de chaque pièce disponible ce qui sera très pratique pour l'intelligence artificielle ainsi que pour les calculs d'échec et échec et mat.

Chacune des cases du tableau est initialisé avec une pièce, une case vide est symbolisée par une pièce ayant un numéro d'équipe égal à 0.

B. Structure du programme

Le programme principal se déroule de telle manière :

-initialisation de la grille et de toutes les pièces
Début d'un tour
-demander au joueur la pièce à déplacer

-proposer une liste de déplacements possible pour cette pièce
-demander au joueur le déplacement à effectuer
-déplacer la pièce
Retour au début du tour s'il n'y a pas échec et mat

À cette structure basique s'ajoutent de nombreuses fonctions intermédiaires comme de calcul de tous les déplacements possibles pour chaque pièce, la vérification que la pièce sélectionnée existe et lui appartienne, la vérification que le mouvement est possible, vérifier si le joueur est en échec pour lui indiques, etc...

C. Coordonnées

Pour développer ce jeu, nous avons donc décidé de faire un tableau à une seule dimension, ainsi, les coordonnées sont enregistrées dans une structure Coord contenant une valeur x et y. Celle-ci sera convertie quand nécessaire en un indice de 0 à 79. Nous aurions pu tout faire directement avec un seul indice, mais nous avons gardé ce système de x et y pour nous faciliter le calcul des déplacements et la visualisation de la place d'une pièce dans la grille avec un indice en abscisse et en ordonné.

2. Fonctions

Parmi les fonctions principales qui font fonctionner le jeu, nous avons donc, la fonction qui calcule tous les mouvements possibles pour chaque pièce, la prise en compte du choix de mouvement par le joueur et la fonction de calcul de l'échec et mat.

A. Calcul des mouvements possibles (setDeplacement)

À l'initialisation de la grille ainsi qu'à chaque fin de tour, tous les objets déplacement des pièces sur la grille sont vidés et rechargés. Pour se faire, dans la structure déplacement, il y a un entier qui stocke le nombre d'éléments stockés dans le tableau de coordonnés et qui permet donc de le parcourir ainsi que d'ajouter des éléments dedans. Afin de "vider" ce tableau, cet entier est mis à 0, ce qui évite de faire un free suivi d'un malloc de la même taille que précédemment. Puisque de toute façon l'espace mémoire est alloué, autant le pas utiliser des ressources et simplement changer la valeur d'un entier.

Ensuite, chaque pièce étant envoyée dans cette fonction, elle regarde tout d'abord le type de pièce qui arrive et selon le résultat renvoie la pièce vers une fonction lui étant adapté (mouvRoi, mouvFou, mouvTour, mouvPion...).

Ensuite dans chacune de ces fonctions le programme regarde si dans les cases où la pièce pourrait aller (en faisant attention de ne pas regarder hors du tableau) il y a un allié, un ennemi ou si la case est vide.

Chaque coordonnée possible est ensuite ajoutée au tableau de déplacements possibles.

Certaines pièces ont quelques particularités supplémentaires comme le pion qui peut avancer de deux cases à son premier coup, d'où l'intérêt de stocker le nombre de coups effectués par chaque pièce dans celle-ci. Ainsi que le roi qui peut effectuer un petit ou grand roque s'il respecte un certain nombre conditions. Tout ceci est pris en compte dans cette fonction.

B. Choix du mouvement par le joueur

Fonction cruciale du programme, cette fonction vérifie un certain nombre de critères avant de valider le choix du joueur et d'effectuer le mouvement.

Tout d'abord, la fonction vérifie dans le tableau de déplacements de la pièce sélectionnée si le déplacement choisi est bien disponible. Si non, l'utilisateur est invité à faire un autre choix.

Ensuite, la fonction teste le mouvement sur une grille temporairement créée grâce à une fonction qui créé une copie parfaite de la grille actuelle avec toutes les pièces s'y situant, ainsi que chacun de leurs déplacements.

Une fois le mouvement effectué sur cette grille, le programme vérifie si ce mouvement ne met pas en échec le joueur ce qui ferait que ce mouvement est interdit et l'utilisateur serait ainsi invité à choisir un autre coup.

Si le coup est autorisé, alors la pièce est déplacée à la case demandée. Si une pièce adverse s'y trouvait, elle est déplacée dans le tableau contenant toutes les pièces mortes dans la grille.

C. Echec et Echec et mat

Pour vérifier si un joueur est en échec, le programme récupère la position actuelle du roi du joueur et regarde ensuite, pour toutes les pièces du joueur adverse, si ces coordonnées se trouvent dans leur tableau de déplacements possibles. Si oui, le joueur est en échec.

Pour la fonction échec et mat, la fonction teste chacun des coups possibles pour chacune des pièces du joueur actuel et regarde s'il est en échec dans toutes ces positions. Si oui, le joueur est en échec.

3. Intelligence artificielle

L'intelligence artificielle est entièrement autonome, une fois que son tour est venu de jouer, un simple appel à sa fonction principale lui permet de faire tous ses calculs et d'effectuer celui sélectionné.

Une fois lancée, tous ses mouvements possibles sont calculés et stockés dans des objets de type Move contenant : l'indice de la pièce de départ, celui d'arrivé, la valeur de la pièce déplacée et enfin le plus important, la "note" attribuée à ce mouvement. Tous ces mouvements sont stockés dans un tableau de Move. L'objectif de cette action est de pouvoir accéder aux données plus rapidement avec moins de calculs.

Chaque pièce a un degré d'importance, une valeur, attribuée de la sorte :

- Pion : 1
- Cavalier : 3
- Fou : 3
- Tour : 5
- Impératrice : 7
- Princesse : 7
- Dame : 9
- Roi : 10

Ensuite, le but est d'attribuer une note à chaque mouvement. Plus la note est élevée, plus le mouvement est intéressant à faire. Il faut donc ajouter le plus de critères possibles permettant d'affiner la note au maximum.

Pour ce faire, plusieurs calculs sont effectués :

a. Le mouvement permet de tuer une pièce adverse : note + valeur de la pièce adverse
b. X=valeur de la plus grosse pièce alliée pouvant se faire tuer avant le mouvement
   Y= valeur de la plus grosse pièce alliée pouvant se faire tuer une fois le mouvement effectué
   Note = X-Y
   Si Y=10 (que notre roi peut se faire tuer une fois le mouvement effectué), Note= -100

Ainsi, si tous les mouvements ont une note de –100, le joueur est en échec et mat.

Sinon, le mouvement ayant la meilleure note est effectué.

4. Graphique

Le développement de la partie graphique a été la partie du développement la plus compliquée et la plus chronophage. Dans un premier temps il nous a été difficile d'initialiser notre poste de travail. Le développement SDL sous windows nous posant trop de problème nous avons décidé de développer notre projet sous linux et il nous a donc fallut configurer nos machines virtuelles.

Une fois nos environnements de travail prêt nous avons pu commencer le développement. Les parties ayant posés problèmes furent dans un premier temps le déplacement et l'actualisation des pièces.

Cependant il reste un souci que nous n'avons pas réussis à résoudre. L'affichage des images s'empilent, de ce fait, la place que prend le programme en mémoire également.

Après quelques tests avec différentes configurations, une mémoire RAM de 8go permet une utilisation optimale pour voir une partie entre 2 IA sans subir de latences.

Comme précisé nous avons opter pour l'utilisation de la librairie SDL et ses composantes (SDL_image et SDL_ttf)) pour toute la partie graphique.

SDL utilise une logique de surface rectangulaire pour sa gestion d'affichage. Chaque bloc est donc une surface utilisable, qui possède des caractéristiques modifiables par le développeur.

La logique est simple, la fenêtre de base est une surface de base sur laquelle toutes nos opérations sont effectuées. Chaque nouvelle image, une nouvelle surface est créée et lui est liée. La logique est la même pour le texte.

Ainsi chaque image est une surface indépendante des autres, et la modification de ses caractéristiques (comme le déplacement) devient possible. La partie graphique est alors liée à la « partie donnée » développée en amont, et la logique de grille de donnée nous a permis d'épouser parfaitement au graphisme d'un jeu d'échec.

Chaque pièce est représentée par une surface de la taille d'une case, complété par une image. A l'appel des fonctions associés, la sélection, le déplacement et le reste des fonction d'un jeu d'échec se déroule en back, et est directement vu en graphique avec quasi-aucune latence.

Conclusion :

Voici des images de notre projet d'échec achevé :

Menu :

Echiquier en jeu :

Après plusieurs semaines de réflexion et de développement, ce jeu d'échec est fonctionnel. Il regroupe une grande partie des notions vu durant cette année de cours d'Informatique Théorique en langage C :

- Pointeur
- Allocation de mémoire
- Header
- Appel de librairies