MAZENQ Chanelle
HOLAY Jérémy
1ère année | TP 1.1

Projet tutoré 1A :
Petits Chevaux

Sommaire

1. Introduction

Dans le cadre du projet tutoré il a été demandé aux élèves de concevoir un jeu de petits chevaux en utilisant le langage C.

2. Fonctionnalités

Le programme comporte, en plus du jeu d'origine:

- L'affichage du plateau de la partie
- Un moyen de sauvegarde
- Un mode de jeu "solo" permettant de jouer avec une ou plusieurs intelligences artificielles.

3. Organisation du programme

1. Les fonctions utilisées dans le programme

case

void InitCaseExt(coordonnees plateauExt[]);
void InitEcurie(coordonnees ecurie[]);
void InitCentre(coordonnees centre[]);

deplacement

void SortieEcurie(Pion jeu[], int joueurDuTour,int pionDuTour, const char couleur[][6]);
int DeplacementPlateau(Pion jeu[], int pionDuTour, int scoreDes, const char couleur[][6], int joueurDuTour);
int EntreeCentre(Pion jeu[], int pionDuTour, int scoreDes, int joueurDuTour);
int DeplacementCentre(Pion jeu[], int pionDuTour, int scoreDes, int joueurDuTour);

ia

void InitIA(int *nombreJoueur, int whitelist[], Pion jeu[], int joueurDuTour, int scoreDes, int joueurIA[]);
int ChoixPionIA(int valide[], Pion jeu[], int scoreDes);

jeu

int PremierJoueur(int nombreJoueur);
int LancerdeDes();
int ConditionFin(Pion jeu[],int joueurDuTour);
int Mouvements(int joueurDuTour, int scoreDes, Pion jeu[], int *action, int *pionDuTour, int whitelist[], int joueurIA[]);
int DeplacementPion(int joueurDuTour, int scoreDes, int action, int pionDuTour,const char couleur[][6],Pion jeu[]);
void TestDeplacement(Pion jeu[], int joueurDuTour, int scoreDes, int valide[]);

joueur

int ChoixCouleur(int nombreJoueur, int whitelist[], int i, Pion jeu[], int couleurPick[],
int joueurDuTour, int scoreDes, int joueurIA[]);
void InitJoueur(int *nombreJoueur,int whitelist[], Pion jeu[], int joueurDuTour, int scoreDes, int joueurIA[]);

misc

int TestPresence(int tab[], int n, int valTest);
int PresencePlateau(Pion jeu[], int pionDuTour);
int VerifEcurie(Pion jeu[], int pionTest);
int JoueurSuivant(int whitelist[],int joueurDuTour);
int DetectionBlocage(Pion jeu[], int pionDuTour, int scoreDes);
void Manger(Pion jeu[], int pionDuTour, int scoreDes, int joueurDuTour, int CaseDepart, char const couleur[][6]);
int CaseLibre(Pion jeu[], int scoreDes, int PionTest, int CaseDepart);
int LireEntier(int whitelist[], Pion jeu[], int save, int joueurDuTour, int scoreDes, int joueurIA[]);
int TestPresenceChar(char tab[], int n, char valTest);

pion

void InitPion(Pion jeu[], coordonnees ecurie[]);
void AfficherPion(Pion jeu[],const char couleur[][6],const char partiePlateau[][10],int whitelist[]);
void UpdateCoord(Pion jeu[], coordonnees ecurie[], coordonnees plateauExt[], coordonnees centre[]);

plateau

void AfficherPlateau(Pion jeu[]);
int CheckCoord(int *x, int y, Pion jeu[]);

sauvegarde

void EcrireSauvegarde(int whitelist[], Pion jeu[], int joueurDuTour, int scoreDes, int joueurIA[]);
void ChargerSauvegarde(int whitelist[], Pion jeu[], int *joueurDuTour, int *scoreDes, int joueurIA[]);

2. Les fonctions principales

`InitCaseExt(coordonnees plateauExt[]);`

Initialise les coordonnées des cases extérieures du plateau.

`InitEcurie(coordonnees ecurie[]);`

Initialise les coordonnées des cases d'écuries du plateau.

`InitCentre(coordonnees centre[]);`

Initialise les coordonées des cases centrales du plateau.

`LireEntier(int whitelist[], Pion jeu[], int save, int joueurDuTour, int scoreDes, int joueurIA[]);`

Fonction permettant de lire un entier (ou le caractère 's' pour une sauvegarde). Elle est utilisée à chaque demande de participation de l'utilisateur.

`InitPion(Pion jeu[], coordonnees ecurie[]);`

Initialise tous les pions.

`PremierJoueur(int nombreJoueur);`

Choisi le premier joueur de la partie au hasard.

`ConditionFin(Pion jeu[],int joueurDuTour);`

Fonction qui teste la condition de fin et retourne 0 tant qu'elle n'est pas remplie.

`JoueurSuivant(int whitelist[],int joueurDuTour);`

Permet de passer au joueur suivant.

`LancerdeDes();`

Lance un dés de 6

.

`UpdateCoord(Pion jeu[], coordonnees ecurie[], coordonnees plateauExt[], coordonnees centre[]);`

Pour chaque pion, on lui associe les coordonnées à au n° de case et au plateau correspondant.

`AfficherPlateau(Pion jeu[]);`

Fonction qui affiche le plateau

.

`Mouvements(int joueurDuTour, int scoreDes, Pion jeu[], int *action, int *pionDuTour, int whitelist[], int joueurIA[]);`

Fonction qui permet de déterminer les mouvements possible pour chaque pion, et qui permet la sélection du pion.

`DeplacementPion(int joueurDuTour, int scoreDes, int action, int pionDuTour,const char couleur[][6],Pion jeu[]);`

Fonction qui effectue le mouvement indiqué par **`Mouvements()`**.

3. Explication de la démarche

Le premier objectif a été d'avoir un jeu sous forme textuel qui fonctionnait sans affichage du plateau.

Une fois cela accompli on d'abord essayé de faire un affichage avec la SDL, mais n'ayant pas réussi à le faire fonctionner sur les machines de l'IUT on a repris le programme pour avoir un affichage en console.

Par la suite nous avons complété avec l'implémentation de la sauvegarde puis des IAs .

On a essayé de faire un programme modulaire. Par exemple, le déroulement du jeu n'est pas impacté par l'affichage du plateau et le comportement des IAs peut-être modifié sans aucun problème de compatibilité avec le reste.

Les entités du programme ont été pensées dès le début afin de pouvoir anticiper les situations particulières et l'ordre dans lequel les fonctions ont été conçues est celui du déroulement "réel" d'une partie de petits chevaux.

4. Répartition des tâches

Nous avons d'abord décidé d'une base de travail commune en se mettant d'accord sur comment modéliser chaque entité (Joueur, Pion et Plateau).

Suite à cela, Chanelle c'est occupée de gérer les paramètres de la partie avec l'initialisation de chaque entité, de l'affichage du plateau et du découpage final en différents fichiers header avec la création du makefile.

Pour sa part, Jérémy a géré le déroulement de la partie ainsi que la conception des intelligences artificielles et le système de sauvegarde.

5. Bilan qualitatif

Chanelle :

Nous avions commencé à écrire tout le programme dans un même .c, avec un unique fichier header associé. Nous avons vite compris qu'il était plus judicieux de tout découper selon les fonctions, et j'ai ainsi tout restructuré.

En reprenant l'échec de Jérémy avec la bibliothèque SDL, j'ai conservé le système de coordonnées pour chaque case et j'ai pu sans trop de problèmes créer le plateau. La difficulté majeure a été de gérer lorsque deux pions possédant un numéro à deux chiffres devaient s'afficher l'un à côté de l'autre.

Jérémy :

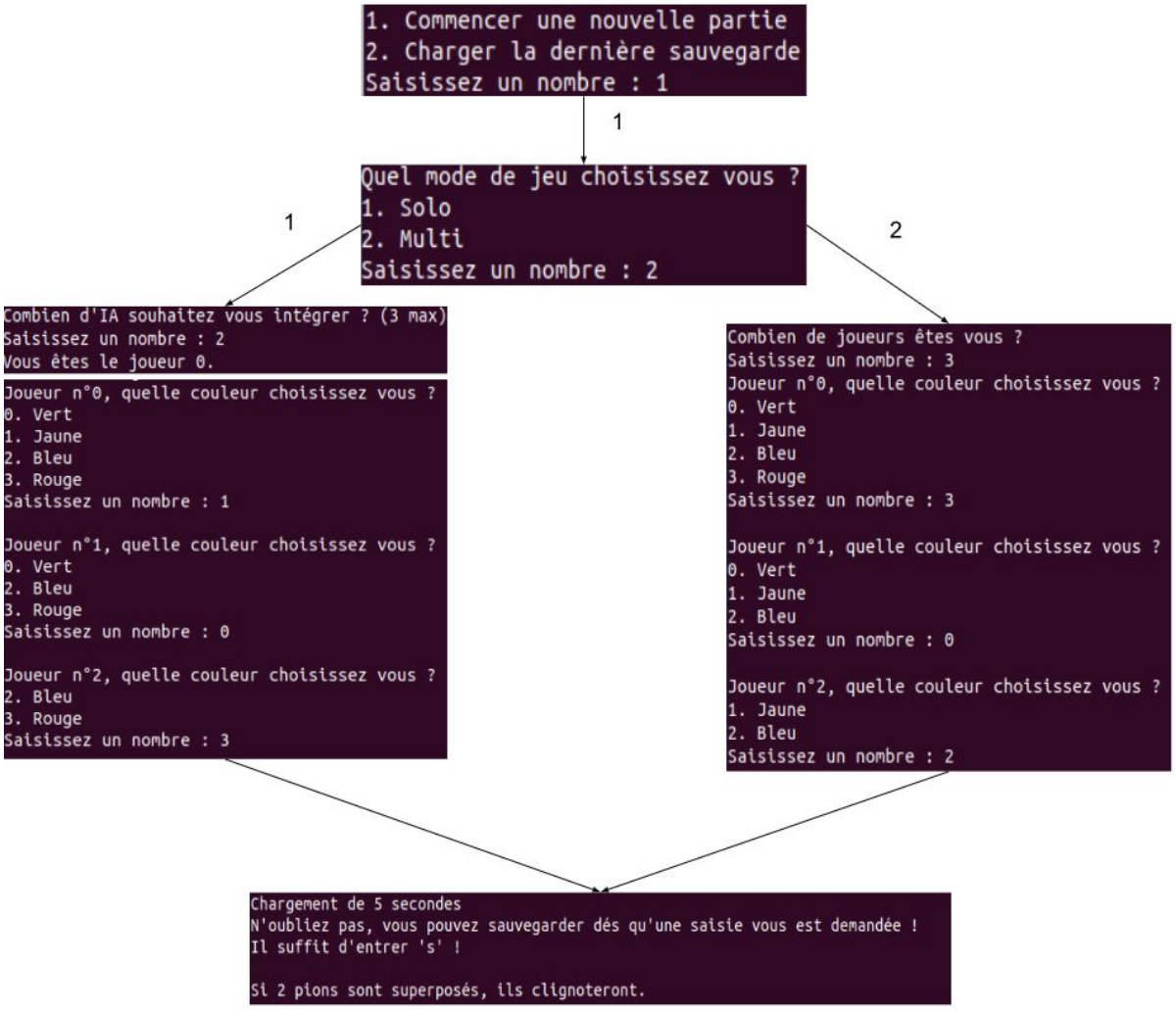
Au niveau de la programmation j'ai dû apprendre à réfléchir aux différents cas "particuliers" tels que le chevauchement des pions d'une même couleur ou encore les pions qui peuvent se manger.

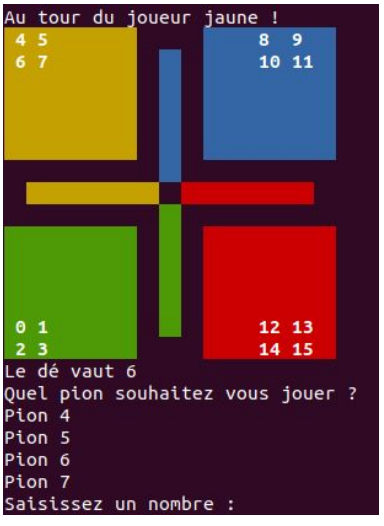
De plus, j'ai bien aimé concevoir les IAs en implémentant leur intelligence petit à petit et j'ai pu constater très vite que lorsque plusieurs facteurs sont pris en compte, il devient difficile de prévoir le comportement de la machine.

Globalement, je suis content du rendu de l'application.

6. Mode d'emploi

Le programme donne des indications à l'utilisateur, lui proposant plusieurs options selon les paramètres de jeu désirés. Il est guidé tout au long de la partie, chaque fois qu'un choix se présente à lui.





7. Conclusion

Pour conclure, la création du programme à été intéressante pour nous deux, aussi bien pour le travail d'équipe que pour les recherches effectuées afin de résoudre les problèmes rencontrés.