Projet Othello

Soupilas Dimitrios,Pashinin Aleksei

13.05.2018

1 L'architicture de projet :

Pour effectuer notre jeux, nous avons decouper notre programme en certains modules :

— `game.c/game.h`
— `coupe.c/coupe.h`
— `1v1.c/1v1.h`
— `arbre.c/arbre.h`
— `level1.c/level1.h`
— `level2.c/level2.h`
— `menu.c` - Le module menu dirige tout l'affichage du menu du jeu. Il initialise "intro" de le programme, et aussi il affiche le menu principale, dirige la fonctionalite de tous les boutons dans le jeu.
— `main.c/main.h` - Le module principal du programme `main` dirige tous les modules et fait tourner notre programme.

2 menu.c :

Le module `menu.c` contient les fonctions d'administration des menus et boutons du programme.

2.1 Fonctions

— `void newgame()` - L'implementation du menu newgame qui se situe au dessous du menu principale. Dans ce menu on peut choisir nombre de joueurs.
— `void newgame2()` - L'implementation du menu "1 player" qui se situe au dessous du menu newgame. Dans ce menu on peut choisir le niveau du jeu.
— `void newgame3()` - L'implementation du menu "1 player->Level2" qui se situe au dessous du menu newgame. Dans ce menu on peut choisir la prfondeur du jeu.
— `void about()` - L'implementation de la partie de menu about qui se situe au dessous du menu principale. Dans cette partie on peut lire les regles du jeu.
— `void button()` - La fonction qui dirige tous les 'clicks' des boutons par utilisateur.
— `void nav()` - La fonction qui dirige le changement du couleur des boutons de noir au bleau au cause de la position de la courseur de souris.
— `void blic_test(int i)` - La fonction pour distinguer les clics de souris et changement des coordonees du courseur.
— `void menu_princ()` - L'implementation du menu principale de jeu.

2.2 Les difficultes

Dans cette fonction nous avons trouver des difficultes a distinguer les boutons avec les memes coordonees. Nous avons decider d'ajouter un troisieme parametre a notre fonction `button` :
— `int z` - Le numero de menu.
— `int x1` - Le valeur de X de souris.
— `int y1` - Le valeur de Y de souris.
Ou z est :
— `z=0` - Le menu principale.
— `z=1` - Le menu "newgame".
— `z=2` - Le menu "1player".
— `z=3` - Le menu "1player"->"Level2".
— `z=4 ou z=5` - Le menu "settings" et "about"
Apres d'inclusion de ce parametre nous avons resolus ce probleme.

3 game.c / game.h

Le module game rassemble des fonction basiques sur l'implementation du jeu.

3.1 Typedefs

```
typedef struct {
  int score;
  char name[20];
}player;

typedef struct {
  int board[8][8];
  player p1;
  player p2;
  int tour;
  int debutant;
}game;

typedef struct {
  int pos_x;
  int pos_y;
}coup;
```

La stucture `player`, est constitue d'un entier qui represente le score, et d'une chaine de character qui supposement contient le nom des joueurs (que l'on a pas utiliser).
La structure `game`, est constitue d'un tableau d'enter 8 * 8 qui represente la partie en cours, deux `player`, un entier tour qui contient le tour du joueur et un entier debutant qui contient le joueur qui a commencer.
La structure `coup`, est constitue de deux entier qui representent des positions dans le `board` du `game`.
Il est important de noter que toutes les fonctions sont implementes pour fonctioner sur toutes les definitions d'ecrants.

3.2 Fonctions

- `game init_game();`
- `void affiche_board(game *g);`
- `void affiche_possibilite(int tablu[8][8]);`
- `int adverse(game *g);`
- `void calc_affiche_score_MLV(game *g);`
- `void affiche_score_fin_MLV(game *g, MLV_Image *board, MLV_Image *white, MLV_Image *black);`
- `void affichage_MLV(game *g, MLV_Image *board, MLV_Image *white, MLV_Image *black);`
- `game copy_game(game* g);`
- `int est_fini(game *g);`

Vous pouvez retrouver toutes les descriptions sur ce que font et comment elles fonctionnent dans le fichier `game.h`.
On a pas affronter de difficultes particulieres en creant ce module. Mais le calcul des coordones exactes a ete assez particulier affin de le faire fonctionner sur toutes les machines. Voir le fichier `sizes` pour les detailles.

4 coup.c / coup.h

Le module coup rassemble les fonction sur les coups dans le jeux. Jouer des coups, verifier si un coup est possible, calculer et afficher les coups possibles etc..
La deffinition du type `coup` a ete faite dans le module game.

4.1 Fonctions

- `int exist_dev(coup *cp, game *g);`
- `int exist_der(coup *cp, game *g);`
- `int exist_bas(coup *cp, game *g);`
- `int exist_haut(coup *cp, game *g);`
- `int exist_dev_haut(coup *cp,game *g);`
- `int exist_dev_bas(coup *cp, game *g);`
- `int exist_der_haut(coup *cp, game *g);`
- `int exist_der_bas(coup *cp, game *g);`
- `int jouer(coup *cp, game *g);`
- `void possibilite(game *g,int tablu[8][8],int *tablu_i,int *tablu_j);`
- `int existe_affiche_coup(game *g,MLV_Image *options);`
- `int existe_coup(game *g);`
- `coup lire_coup();`

Vous pouvez retrouver toutes les descriptions sur ce que font et comment elles fonctionnent dans le fichier `coup.h`.

Les difficultes retrouves dans ce modules sont de bien gere les coup joues, bien definir les limites dans le `board` pour toutes les fonctions `exist_X` ainsi que bien gere que lors d'un coup tout les cas soient verifies afin de transformer correctement le `board`.
De plus dans la fonction `void possibilite` bien gerer le tableaux dinamiques.

5 1v1

Ce module contient qu'une seul fonction :
`game mode_1v1(MLV_Image *board, MLV_Image *black, MLV_Image *white, MLV_Image *options)`
Elle implemente le 1v1, aucune difficulte.

6 level1.c /level1.h

Ce module implemente le mode de jeux contre une IA qui choisie a chaque fois le coup qui va lui raporter le plus grand score au prochain coup.

6.1 Fonctions

- `void position_gagnate(game *g,int *tablu_i,int *tablu_j);`
- `void ia_next_step_max(game *g,int taille_pos);`
- `game mode_1(MLV_Image *board, MLV_Image *black, MLV_Image *white, MLV_Image *options);`

Vous pouvez retrouver toutes les descriptions sur ce que font et comment elles fonctionnent dans le fichier `level1.h`.

La grande difficulte ete de bien calculer la coup joue pour la position gagnate.

7 arbre.c / arbre.h

Ce module implemente l'utilisation d'arbres, ansi que la creation d'arbre recursif et l'evaluation du plateau.

7.1 Typedefs

```
  typedef struct noeud{
    game g;
    coup cp;
    int nb_fils;
    struct noeud **fils;
}noeud;

typedef struct {
    int nb_choises;
    int *best;
}bestplay;
```

7.2 Fonctions

- `arbre arbre_vide();`
- `bestplay init_bp(game *g);`
- `int evaluation_plateu ( game * g);`
- `int est_vide(arbre a);`
- `arbre creer_arbre(game *g,coup *cp);`
- `arbre inserer_fils_n(arbre a, arbre fils, int num_fils);`
- `arbre creer_arbre_avec_prof3(game *g, coup *cp, int prof_act, int prof_lim, bestplay *bp, int *count);`
- `void free_arbre(arbre a);`

Vous pouvez retrouver toutes les descriptions sur ce que font et comment elles fonctionnent dans le fichier `arbre.h`.

La diificulte ultime creer un arbre recursivement avec n_fils gerer toutes les allocations, les creations etc.. cette fonction nous a manger deux jourss car nous avions (en plus de quelques erreurs d'allocation) definis dans la structure textttnoeud les `game` comme de pointeurs (`game *g`) ce qui nous sabotage le retour.

8 level2.c level2.h

Ce module implemente le mode de jeux contre une ia qui cree un arbre de profondeur 2.

8.1 Fonction

- `void ia_2(game *g);`
- `game mode_2(MLV_Image *board, MLV_Image *black, MLV_Image *white, MLV_Image *options);`

Apres la creation de l'arbre il n'y a aucune difficulte.

9 leve3.c level2.h

Ce Ce module implemente le mode de jeux contre une ia qui cree un arbre de profondeur N. Identique que level2.

- `void ia_3(game *g,int p);`
- `game mode_3(MLV_Image *board, MLV_Image *black, MLV_Image *white, MLV_Image *options,int prof);`

10 Repartition du travail

- PASHININ Aleksei :
  - Menu
- SOUPILAS Dimitrios :
  - Implementation du jeux
  - 1v1
- Ensemble :
  - Arbres
  - IAs