Phillippine Rousseau, Baptiste Oberbach

CNAM 1A Promotion 2016-2019

Algorithme et structure de donnée

Projet Jeu de Go

Explication des choix d'algorithme et de structure de donnée

Enumérations et structures

Les couleurs

```
enum Couleur
{
   VIDE, BLANC, NOIR,
};
```

Nous avons créé une structure couleur représentant les trois états que peux prendre une intersection, c'est-à-dire vide, si aucun pion n'est posé dessus, blanc ou noir suivant la couleur du pion posé.

Les coordonnées

```
typedef struct Coordonnees {
 int x;
 int y;
}Coord;
```

Cette structure représente les coordonnée d'une intersection du goban.

Le jeu

```
typedef struct JeuDeGo {
 Coord * lastCoordBlanc;
 Coord * lastCoordNoir;
 Pion ** plateau;
 int taille;
 Couleur joueurCourant;
 bool isFinish;
}Jeu;
```

Nous avons choisi de créer une structure jeu qui contient :

- Les coordonnées du dernier coup joué par les blancs et par les noirs, afin de vérifier les répétitions de coup et d'appliqué la règle « Un joueur, en posant une pierre, ne doit pas redonner au goban un état identique à l'un de ceux qu'il lui avait déjà donné. »
- Un pointeur sur un tableau de Pion qui représente le plateau
- La taille du plateau (9, 13 ou 19)
- Une couleur indiquant le joueur courant, c'est-à-dire le joueur devant jouer (et non pas celui venant de jouer).
- Un booléen indiquant si la partie est finis ou non

Les pions

```
struct PionDuPlateau {
 Couleur couleur;
 Liste * chaineLie;
 Coord * coord;
};
```

Cette structure représente un pion du plateau. Un pion est obligatoirement d'une couleur (blanc ou noir), lié à une chaine (dans lequel il peut être seul) et à des coordonnées.

Les chaines

Nous avons utilisé des chaines doublements chainées. Notons que des chaines simplement chainée aurait suffis pour notre utilisation, mais nous avons préférés être prévoyant. Chaque nœud représente un pion. Ces chaines permettent de calculer les degré de libertés.

La structure déroulement de la partie

```
typedef struct DeroulementDeLaPartie
{
        int taillePlateau;
        char* nomJoueurBlanc;
        char* nomJoueurNoir;
        Liste* listePionNoir; //pion noir pausé dans l'ordre
        Liste* listePionBlanc; //pion blanc dans l'ordre
        int nbRound; //nombre de round
        bool isFinish; //sert a savoir si la partie est fini ou pas
        char* result; //sert a stocker dans le fichier le resultat final
}DeroulementPartie;
```

Cette structure permet de sauvegarder une partie jouée.

Algorithme

L'ensemble des fonctions gérant le jeu de go sont stocké dans le fichier go.c

Dessiner le Goban

```
void draw_win(void);
```

La fonction draw_win permet de dessiner le goban. Etant donné que cette fonction ne prends pas de paramètre il a fallu utilisé une variable global appelé jeu.

Placer une pierre et passer son tour

```
void mouse_clicked(int bouton, int x, int y) ;
```

La fonction mouse_clicked permet de traiter le passage de tours mais aussi le posage de pierre avec l'appelle aux fonctions isAuthorizedMoove qui vérifie si un mouvement est autorisé et playMoove qui joue ce mouvement

Vérifier qu'un coup est autoriser

```
bool isAuthorizedMoove(Jeu * jeu, Coord futurMoove);
```

Cette fonction vérifie si un coup est autorisé, plusieurs conditions sont à remplir :

- L'emplacement où l'on veut placer le pion doit être vide.
- Si cette condition est remplie on place un pion temporaire de la couleur du joueur courant.
- Chacune des cases adjacentes (en haut, à droite, à gauche et en bas ne doit pas être un bord du plateau et correspondre au critère vérifié dans la fonction isAuthorizedSense

```
bool isAuthorizedSense(Jeu * jeu, Pion * adjacentPion) ;
```

Cette fonction vérifie si un sens est jouable. Si le pion adjacent est vide alors il est possible de jouer dans ce sens. On calcule ensuite le degré de liberté de la chaine du pion adjacent (il faut tenir compte du fait qu'on a déjà placé un pion de la couleur du joueur). Si le pion est de la même couleur que le joueur courant il faut que la chaine ait au moins un de degré de liberté. A l'inverse si le pion est de couleur adverse il faut que la chaine n'ait pas de degré de liberté (qu'elle soit capturable).

Joueur un coup

```
void playMoove(Jeu * jeu, Coord * coord, Couleur couleur)
```

Lorsque que l'on joue un coup on place un pion sur le plateau. Au même moment on initialise la chaine liée au pion en une chaine ne contenant que le pion. On vérifie chacun des pions adjacents, s'ils sont de la même couleur que le joueur courant on fusionne la chaine du pion courant avec la chaine du pion adjacent, si c'est un pion de couleur adverse on vérifie si sa chaine n'a pas de liberté, si c'est le cas on capture la chaine.

Lancer le jeu

```
void game(int argc, char *argv[]) ;
```

Cette function initialise le jeu en fonction des paramètres present dans argv.

L'IA

```
void playMooveIA(Jeu * jeu)
```

L'IA n'est pas élaboré et se contente de jouer à des endroits aléatoires. Elle vérifie si un pion peut être placé à la position sélectionnée, si c'est le cas elle le place sinon elle retente. Si elle n'arrive pas à placer un pion au bout de 10 essais elle passe son tours.

Sauvegarder une partie

```
void writePartyData(char* fileName, DeroulementPartie* deroulementPartie)
```

Cette fonction lit les différents coups d'un déroulement de partie et les enregistre dans le fichier fileName. Notons que durant le jeu chaque coup joué est enregistré dans un déroulement de partie et qu'il suffit donc d'interprété les informations contenues dans la structure et de les écrire au format voulu lors de la sauvegarde.

Charger une partie

```
void loadPartyData(char* fileName, Jeu** jeu, DeroulementPartie** deroulementPartie)
```

Cette fonction remplis les structures déroulementPartie et jeu à partir des informations contenues dans le fichier fileName. Pour cela elle rejoue le jeu coup par coup. Etant donné qu'un fichier sgf contient à la fois des informations relatives au jeu (la position actuelle, les chaines de pions etc..) et au déroulement de la partie (l'enchainement des coups) il a fallu passé le jeu et le déroulement de partie par référence pour pouvoir les modifier.