# Jeu de dames

Version Finale

Généré par Doxygen 1.7.3

Mon May 9 2011 00 :36 :50

# Table des matières

# Chapitre 1

# Index des structures de données

## 1.1 Structures de données

Liste des structures de données avec une brève description :

# Chapitre 2

# Index des fichiers

## 2.1 Liste des fichiers

Liste de tous les fichiers documentés avec une brève description :

# Chapitre 3

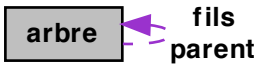
# Documentation des structures de données

## 3.1 Référence de la structure arbre

Un arbre à une valeur (la racine) et un certain nombre de fils.

`#include <arbre.h>`

Graphe de collaboration de arbre :




### Champs de données

- struct arbre * **parent**
- int **valeur**
- int **profondeur**
- int nb_fils
- struct arbre * fils [80]

### 3.1.1 Description détaillée

Un arbre à une valeur (la racine) et un certain nombre de fils.

Définition à la ligne 13 du fichier arbre.h.

### 3.1.2 Documentation des champs

#### 3.1.2.1 struct arbre* fils[80]

Un noeud n'aura jamais plus de 80 fils car on a, pour un demi-coup, 20 pions * 4 déplacements possibles = 80 coups différents.

Définition à la ligne 18 du fichier arbre.h.

#### 3.1.2.2 int nb_fils

Le nombre de fils de ce noeud

Définition à la ligne 17 du fichier arbre.h.

La documentation de cette structure a été générée à partir du fichier suivant :
– arbre.h

## 3.2 Référence de la structure Carre_clair

`#include <constantes.h>`

### Champs de données

– int **est_libre**
– int **numero_case**
– SDL_Rect **position**
– SDL_Surface * **surface**

### 3.2.1 Description détaillée

Structure definissant un carre clair

Définition à la ligne 37 du fichier constantes.h.

La documentation de cette structure a été générée à partir du fichier suivant :
– constantes.h

## 3.3 Référence de la structure Carre_fonce

`#include <constantes.h>`

### Champs de données

– int **est_libre**
– int **numero_case**
– SDL_Rect **position**
– SDL_Surface * **surface**

### 3.3.1 Description détaillée

Structure definissant un carre fonce

Définition à la ligne 29 du fichier constantes.h.

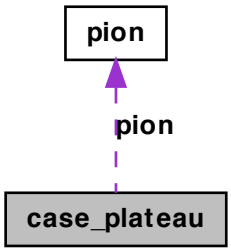
La documentation de cette structure a été générée à partir du fichier suivant :
– constantes.h

## 3.4 Référence de la structure case_plateau

Objet case du plateau de jeu. Une case est définie par une couleur et, si elle est noire,elle dispose d'une abscisse (x) et une ordonnée (y) sur le plateau. Une case peut être libre ou non, si elle n'est pas libre elle contient un pion.

`#include <plateau.h>`

Graphe de collaboration de case_plateau :




### Champs de données

– couleur_pion couleur
– int est_libre
– pion **pion**
– int **x**
– int **y**
– int notation_officielle
– int en_surbrillance

### 3.4.1 Description détaillée

Objet case du plateau de jeu. Une case est définie par une couleur et, si elle est noire,elle dispose d'une abscisse (x) et une ordonnée (y) sur le plateau. Une case peut être libre ou non, si elle n'est pas libre elle contient un pion.

Définition à la ligne 45 du fichier plateau.h.

### 3.4.2 Documentation des champs

#### 3.4.2.1 couleur_pion couleur

La couleur de la case.

Définition à la ligne 46 du fichier plateau.h.

#### 3.4.2.2 int en_surbrillance

Si la case doit être affichée en surbrillance.

Définition à la ligne 52 du fichier plateau.h.

#### 3.4.2.3 int est_libre

Vrai si la case est vide

Définition à la ligne 47 du fichier plateau.h.

#### 3.4.2.4 int notation_officielle

Le numéro de la case selon la notation officielle.

Définition à la ligne 51 du fichier plateau.h.

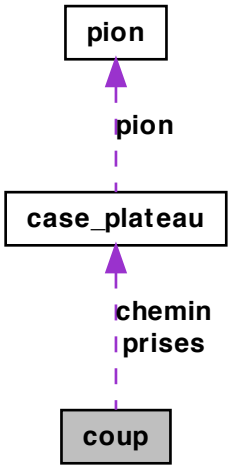
La documentation de cette structure a été générée à partir du fichier suivant :
– plateau.h

## 3.5 Référence de la structure coup

Objet coup. Un coup est défini par un numéro de case de départ, un numéro de case d'arrivée, un type (déplacement ou prise) et optinellement un commentaire.

`#include <plateau.h>`

Graphe de collaboration de coup :




### Champs de données

- int **old_case**
- int **new_case**
- type_coup tc
- int **nombre_prises**
- case_plateau prises [20]
- case_plateau chemin [20]
- char * commentaire

### 3.5.1 Description détaillée

Objet coup. Un coup est défini par un numéro de case de départ, un numéro de case d'arrivée, un type (déplacement ou prise) et optinellement un commentaire.

Définition à la ligne 74 du fichier plateau.h.

### 3.5.2 Documentation des champs

#### 3.5.2.1 case_plateau chemin[20]

Liste de cases sur lesquelles le pion s'arrête lors d'une rafle (pour repérer la fin de liste, la dernière case sera une case blanche).

Définition à la ligne 80 du fichier plateau.h.

#### 3.5.2.2 char* commentaire

Les commentaires sont :

– ! pour indiquer un coup fort ou bien joué
– ! ! pour indiquer un coup très fort
– ? pour indiquer un coup faible ou mal joué
– ? ? pour indiquer un coup très faible ou une gaffe
– ! ? pour indiquer un coup paraissant fort, mais qui en réalité se révèle faible
– ? ! pour indiquer un coup paraissant faible, mais qui en réalité se révèle fort
– * pour indiquer un coup forcé, tout autre mouvement entraînant une perte immédiate
– + pour indiquer le gain de la partie
– = pour indiquer un jeu égal
– +1 pour indiquer le gain d'un pion
– +n pour indiquer le gain de n pion
– +- pour indiquer un avantage aux blancs
– -+ pour indiquer un avantage aux noirs

Définition à la ligne 82 du fichier plateau.h.

#### 3.5.2.3 case_plateau prises[20]

Les cases sur lesquelles les jetons ont été pris.

Définition à la ligne 79 du fichier plateau.h.

#### 3.5.2.4 type_coup tc

Si le coup est un déplacement ou une prise.

Définition à la ligne 77 du fichier plateau.h.

La documentation de cette structure a été générée à partir du fichier suivant :
– plateau.h

## 3.6 Référence de la structure joueur

Objet joueur. Un joueur est caractérisé par la couleur qu'il joue et sa nature (humain ou intelligence artificielle).

`#include <plateau.h>`

### Champs de données

– int est_humain
– couleur_pion couleur

### 3.6.1 Description détaillée

Objet joueur. Un joueur est caractérisé par la couleur qu'il joue et sa nature (humain ou intelligence artificielle).

Définition à la ligne 62 du fichier plateau.h.

### 3.6.2 Documentation des champs

#### 3.6.2.1 couleur_pion couleur

La couleur avec laquelle joue ce joueur.

Définition à la ligne 64 du fichier plateau.h.

#### 3.6.2.2 int est_humain

Détermine si le joueur est une IA ou un joueur réel.

Définition à la ligne 63 du fichier plateau.h.

La documentation de cette structure a été générée à partir du fichier suivant :
– plateau.h

## 3.7 Référence de la structure pion

Objet pion. Un pion est déterminé par une couleur et si il est une dame ou non.

```
#include <plateau.h>
```

### Champs de données

– couleur_pion **couleur**
– int est_dame
– int en_surbrillance

### 3.7.1 Description détaillée

Objet pion. Un pion est déterminé par une couleur et si il est une dame ou non.

Définition à la ligne 33 du fichier plateau.h.

### 3.7.2 Documentation des champs

#### 3.7.2.1 int en_surbrillance

Si le pion doit être affichée en surbrillance.

Définition à la ligne 36 du fichier plateau.h.

#### 3.7.2.2 int est_dame

Détermine si on a un pion normal ou une dame.

Définition à la ligne 35 du fichier plateau.h.

La documentation de cette structure a été générée à partir du fichier suivant :
– plateau.h

## 3.8 Référence de la structure Pion_clair

`#include <constantes.h>`

### Champs de données

– int **est_dame**
– SDL_Surface * **surface**
– SDL_Rect **position**

### 3.8.1 Description détaillée

Structure definissant un pion clair

Définition à la ligne 45 du fichier constantes.h.

La documentation de cette structure a été générée à partir du fichier suivant :
– constantes.h

## 3.9 Référence de la structure Pion_fonce

`#include <constantes.h>`

### Champs de données

– int **est_dame**
– SDL_Surface * **surface**
– SDL_Rect **position**

### 3.9.1 Description détaillée

Structure definissant un pion fonce

Définition à la ligne 52 du fichier constantes.h.

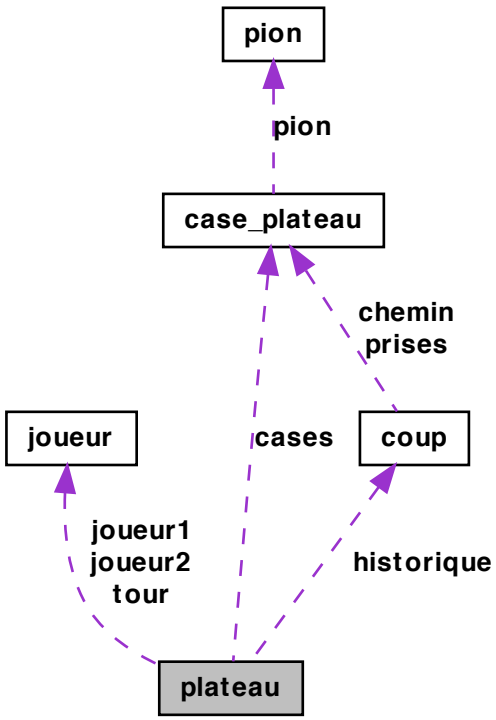
La documentation de cette structure a été générée à partir du fichier suivant :
– constantes.h

## 3.10 Référence de la structure plateau

Objet plateau. Le plateau est composé de 50 cases, numérotées de 1 à 50. Il comporte un historique des coups. Il sait quel joueur doit jouer le prochain coup.

`#include <plateau.h>`

Graphe de collaboration de plateau :




### Champs de données

- case_plateau cases [51]
- coup historique [500]
- int i
- joueur **joueur1**
- joueur **joueur2**
- joueur tour

### 3.10.1 Description détaillée

Objet plateau. Le plateau est composé de 50 cases, numérotées de 1 à 50. Il comporte un historique des coups. Il sait quel joueur doit jouer le prochain coup.

Définition à la ligne 107 du fichier plateau.h.

### 3.10.2 Documentation des champs

#### 3.10.2.1 case_plateau cases[51]

Un plateau est composé de 50 cases, numérotées de 01 à 50. (la case 0 stocke la case blanche).

Définition à la ligne 108 du fichier plateau.h.

#### 3.10.2.2 coup historique[500]

On peut enregistrer jusqu'à 500 coups.

Définition à la ligne 109 du fichier plateau.h.

#### 3.10.2.3 int i

Non utilisé : l'indice du prochain coup à entrer.

Définition à la ligne 110 du fichier plateau.h.

#### 3.10.2.4 joueur tour

Le joueur qui doit jouer le prochain coup.

Définition à la ligne 113 du fichier plateau.h.

La documentation de cette structure a été générée à partir du fichier suivant :
– plateau.h

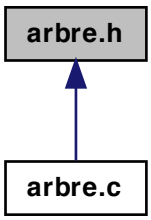
# Chapitre 4

# Documentation des fichiers

## 4.1 Référence du fichier arbre.h

Implémantation des arbres.

Ce graphe montre quels fichiers incluent directement ou indirectement ce fichier :




### Structures de données

– struct arbre
*Un arbre à une valeur (la racine) et un certain nombre de fils.*

### Définition de type

– typedef struct arbre **arbre**

### Fonctions

– int arbre_est_feuille (arbre t)

*Renvoie vrai si l'arbre est une feuille.*

– void print_arbre (arbre t)

*Affiche l'arbre en parcours préfixe "à la Scheme", c.à.d. de la forme (racine fils1 fils2 ...).*

### 4.1.1 Description détaillée

Implémantation des arbres.

**Auteur**

Bastien Auda

Définition dans le fichier arbre.h.

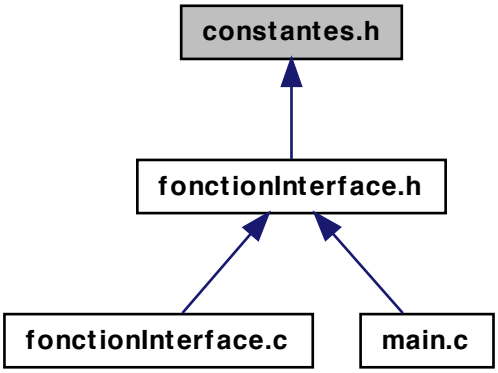
## 4.2 arbre.h

```
00001
00013 typedef struct arbre {
00014         struct arbre * parent;
00015         int valeur;
00016         int profondeur;
00017         int nb_fils;
00018         struct arbre *fils[80];
00019 } arbre;
00020
00021
00026 int arbre_est_feuille(arbre t);
00027
00032 void print_arbre(arbre t);
```

## 4.3 Référence du fichier constantes.h

Constantes utiles pour l'interface.

Ce graphe montre quels fichiers incluent directement ou indirectement ce fichier :




### Structures de données

– struct Carre_fonce
– struct Carre_clair
– struct Pion_clair
– struct Pion_fonce

### Macros

– #define **CONSTANTES_H_**
– #define **LARGEUR** 800
– #define **LONGUEUR** 800
– #define **TAILLECARRE** 80

### 4.3.1 Description détaillée

Constantes utiles pour l'interface.

**Auteur**

Mrah Mehdi

Définition dans le fichier constantes.h.

## 4.4 constantes.h

```
00001
00009 #if defined(linux) || defined(__linux) || defined(__linux__)
```

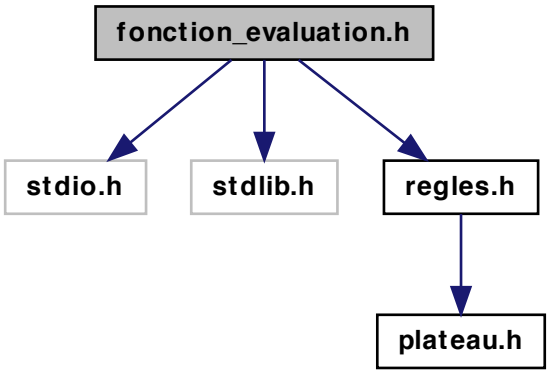
```
00010 #include <SDL/SDL.h>
00011 #endif
00012 #if defined(__APPLE__)
00013 #include <SDL/SDL.h>
00014 #endif
00015 #if defined(WIN32) || defined(_WIN32) || defined(WIN64) || defined(_WIN64)
00016 #include <SDL\SDL.h>
00017 #endif
00018
00019
00020 #ifndef CONSTANTES_H_
00021 #define CONSTANTES_H_
00022
00023
00024 #define LARGEUR 800
00025 #define LONGUEUR 800
00026 #define TAILLECARRE 80
00027
00029 typedef struct {
00030         int est_libre;
00031         int numero_case;
00032         SDL_Rect position;
00033         SDL_Surface *surface;
00034 } Carre_fonce;
00035
00037 typedef struct {
00038         int est_libre;
00039         int numero_case;
00040         SDL_Rect position;
00041         SDL_Surface *surface;
00042 } Carre_clair;
00043
00045 typedef struct{
00046         int est_dame;
00047         SDL_Surface *surface;
00048         SDL_Rect position;
00049 } Pion_clair;
00050
00052 typedef struct{
00053         int est_dame;
00054         SDL_Surface *surface;
00055         SDL_Rect position;
00056 } Pion_fonce;
00057
00058 #endif
```

## 4.5 Référence du fichier fonction_evaluation.h

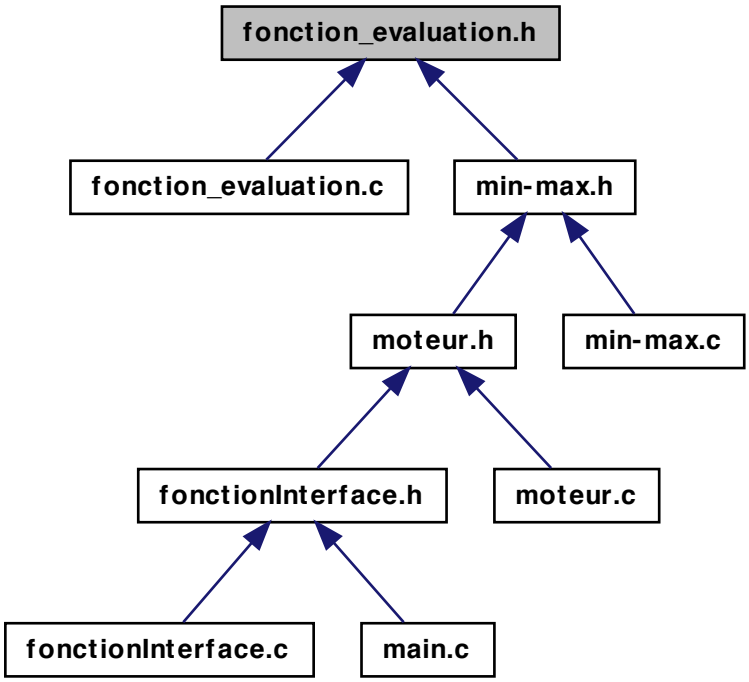
La fonction d'evaluation de l'algorithme MinMax. Renvoie un double compris entre -1 et 1 caracterisant un etat du plateau plus ou moins favorable au joueur courant.

```
#include <stdio.h>

#include <stdlib.h>

#include "regles.h"
```

Graphe des dépendances par inclusion de fonction_evaluation.h :



Ce graphe montre quels fichiers incluent directement ou indirectement ce fichier :




### Fonctions

– double **fct_eval** (const plateau ∗p)
– int **valeur_case** (const plateau ∗p, const case_plateau ∗c)
– int rang (const case_plateau ∗c)

*Renvoie, pour une case occupee, son rang ie le numero de sa ligne. (depend du camp du pion)*

– int est_isole (const plateau ∗p, const case_plateau ∗c, int rang)

*un pion isole n'est pas sur un bord et n'a aucun pion de son camp present dans une case adjacente.*

### 4.5.1 Description détaillée

La fonction d'evaluation de l'algorithme MinMax. Renvoie un double compris entre -1 et 1 caracterisant un etat du plateau plus ou moins favorable au joueur courant.

**Auteur**

Kyann Valai

Définition dans le fichier fonction_evaluation.h.

## 4.6 fonction_evaluation.h

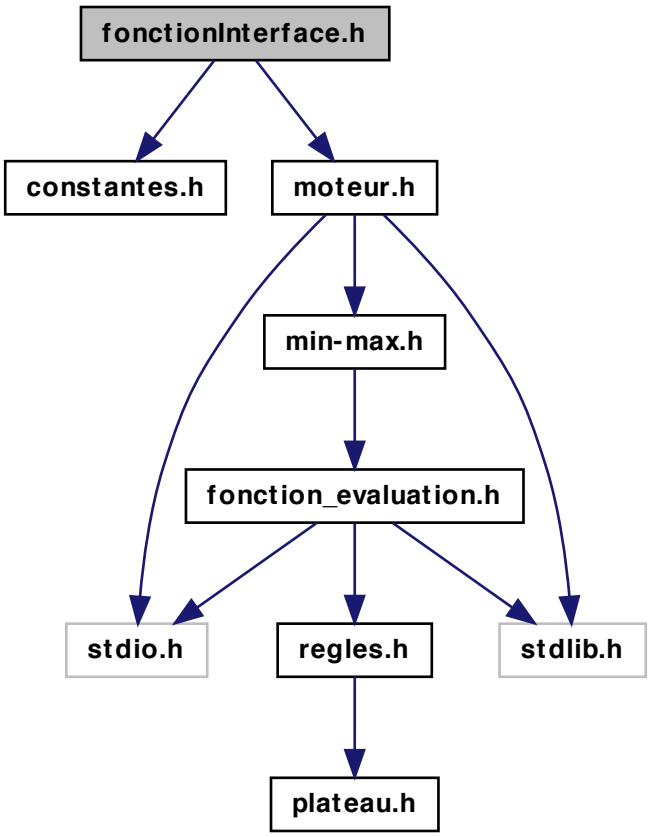
```
00001
00008 #include <stdio.h>
00009 #include <stdlib.h>
00010 #include "regles.h"
00011
00018 double fct_eval(const plateau * p);
00019
00024 int valeur_case(const plateau * p, const case_plateau * c);
00025
00030 int rang(const case_plateau * c);
00031
00036 int est_isole(const plateau * p, const case_plateau * c, int rang);
```

## 4.7 Référence du fichier fonctionInterface.h

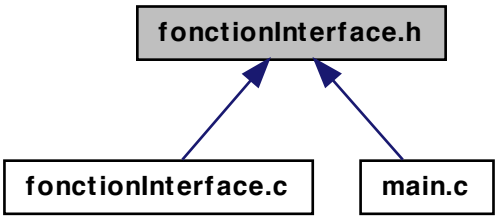
Implémentation de fonctionnalités pour l'interface.

```
#include "constantes.h"
#include "moteur.h"
```

Graphe des dépendances par inclusion de fonctionInterface.h :



Ce graphe montre quels fichiers incluent directement ou indirectement ce fichier :




## Fonctions

– void rafraichir_plateau ()
*Rafraichit le plateau aprés une action.*

– void afficher_ecran_depart_neutre (int choix)
*Affiche le menu de depart.*

– void **afficher_ecran_pause** (int choix)
– void afficher_ecran_choix_joueur ()
*Affiche le menu de choix IA/Joueur.*

– void afficher_ecran_noirs_gagnent ()
*Affiche l'ecran de fin de partie, les noirs gagnent.*

– void afficher_ecran_blancs_gagnent ()
*Affiche l'ecran de fin de partie, les blancs gagnent.*

– int clique_souris_choix_joueur (SDL_Event evenement)
*Retourne la position de la souris après un clique dans le menu choix IA/Joueur.*

– int ∗ clique_souris (SDL_Event evenement)
*Retourne la position de la souris après un clique.*

– void initialisation_cases_blanches ()
*Initialise les cases blanches.*

– int position_souris (SDL_Event evenement)
*Calcule la position de la souris a chaque mouvement.*

– int **position_souris_pause** (SDL_Event evenement)
– int **clique_souris_menu** (SDL_Event evenement)
– int **clique_souris_pause** (SDL_Event evenement)
– void **control_manger_pion** (case_plateau oldPosition, case_plateau newPosition)
– case_plateau **control_surbrillance** (int ∗tab)

– case_plateau control_premier_click (SDL_Event event, int ∗tab, plateau ∗p, case_-plateau ∗oldCase)
*Gere le premier click qui s'occupe uniquement de la selection du pion a jouer.*

– case_plateau **control_deuxieme_click** (SDL_Event event, int ∗tab, plateau ∗p, case_-plateau ∗newCase, case_plateau ∗oldCase)

### Variables

– Carre_fonce carre_fonce
– Carre_clair carre_clair
– Carre_clair carre_surbrillance
– Pion_clair pion_clair
– Pion_fonce pion_fonce
– SDL_Surface ∗ screen
– int **tableauChoix** [4]

### 4.7.1 Description détaillée

Implémentation de fonctionnalités pour l'interface.

**Auteur**

Mrah Mehdi

Définition dans le fichier fonctionInterface.h.

### 4.7.2 Documentation des fonctions

#### 4.7.2.1 void afficher_ecran_depart_neutre ( int *choix* )

Affiche le menu de depart.

Affiche le menu pause.

**Paramètres**

| | |
|---|---|
| *choix* | Represente le choix dans le menu depart. |
| *choix* | Represente le choix dans le menu pause. |

Définition à la ligne 118 du fichier fonctionInterface.c.

#### 4.7.2.2 int∗ clique_souris ( SDL_Event *evenement* )

Retourne la position de la souris après un clique.

**Renvoie**

int∗ Un tableau contenant la position x et y du clique

Définition à la ligne 263 du fichier fonctionInterface.c.

#### 4.7.2.3 int clique_souris_choix_joueur ( SDL_Event *evenement* )

Retourne la position de la souris après un clique dans le menu choix IA/Joueur.

**Renvoie**

int Un entier representant le choix du joueur.

Définition à la ligne 301 du fichier fonctionInterface.c.

#### 4.7.2.4 case_plateau control_premier_click ( SDL_Event *event*, int * *tab*, plateau * *p*, case_plateau * *oldCase* )

Gere le premier click qui s'occupe uniquement de la selection du pion a jouer.

**Paramètres**

| | |
|---|---|
| *tab* | Tableau contenant les coordonnees du click |
| *oldCase* | la case de depart |

**Renvoie**

case_plateau La case contenant le pion selectionne

**Auteur**

Mehdi M'rah

Définition à la ligne 515 du fichier fonctionInterface.c.

#### 4.7.2.5 int position_souris ( SDL_Event *evenement* )

Calcule la position de la souris a chaque mouvement.

Calcule la position de la souris a chaque clique dans le menu principal.

**Renvoie**

int Un entier representant une position dans le menu.
int Un entier representant une position dans le menu pause.
int Un entier representant un choix dans le menu principal.
int Un entier representant un choix dans le menu pause.

Définition à la ligne 279 du fichier fonctionInterface.c.

### 4.7.3 Documentation des variables

#### 4.7.3.1 Carre_clair carre_clair

Surface : marbre blanc

Définition à la ligne 19 du fichier fonctionInterface.h.

#### 4.7.3.2 Carre_fonce carre_fonce

Surface : marbre noir

Définition à la ligne 16 du fichier fonctionInterface.h.

#### 4.7.3.3 Carre_clair carre_surbrillance

Surface : marbre noir en surbrillance

Définition à la ligne 22 du fichier fonctionInterface.h.

#### 4.7.3.4 Pion_clair pion_clair

Pion : blanc

Définition à la ligne 25 du fichier fonctionInterface.h.

#### 4.7.3.5 Pion_fonce pion_fonce

Pion : noir

Définition à la ligne 28 du fichier fonctionInterface.h.

#### 4.7.3.6 SDL_Surface* screen

Surface principal

Définition à la ligne 31 du fichier fonctionInterface.h.

## 4.8 fonctionInterface.h

```
00001
00009 #include "constantes.h"
00010 #include "moteur.h"
00011
00012 #ifndef FONCTIONINTERFACE_H_
00013 #define FONCTIONINTERFACE_H_
00014
00016 Carre_fonce carre_fonce;
00017
00019 Carre_clair carre_clair;
00020
00022 Carre_clair carre_surbrillance;
00023
00025 Pion_clair pion_clair;
00026
00028 Pion_fonce pion_fonce;
00029
00031 SDL_Surface *screen;
00032
00033 int tableauChoix[4];
```

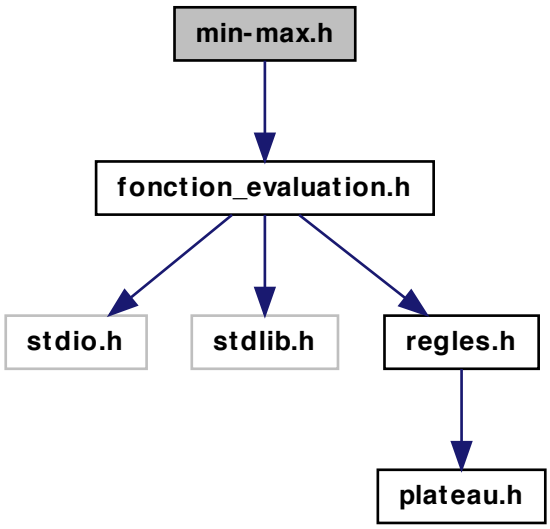
```
00038 void rafraichir_plateau();
00039
00045 void afficher_ecran_depart_neutre(int choix);
00046
00052 void afficher_ecran_pause(int choix);
00053
00058 void afficher_ecran_choix_joueur();
00059
00064 void afficher_ecran_noirs_gagnent();
00065
00070 void afficher_ecran_blancs_gagnent();
00071
00077 int clique_souris_choix_joueur(SDL_Event evenement);
00078
00084 int* clique_souris(SDL_Event evenement);
00085
00090 void initialisation_cases_blanches();
00091
00097 int position_souris(SDL_Event evenement);
00103 int position_souris_pause(SDL_Event evenement);
00109 int clique_souris_menu(SDL_Event evenement);
00115 int clique_souris_pause(SDL_Event evenement);
00116
00117
00118 void control_manger_pion(case_plateau oldPosition, case_plateau newPosition);
00119
00128 case_plateau control_surbrillance(int *tab);
00129
00138 case_plateau control_premier_click(SDL_Event event, int *tab, plateau *p,
      case_plateau *oldCase);
00139
00150 case_plateau control_deuxieme_click(SDL_Event event, int *tab, plateau *p,
      case_plateau *newCase, case_plateau *oldCase);
00151
00152 #endif /* FONCTIONINTERFACE_H_ */
```

## 4.9 Référence du fichier min-max.h

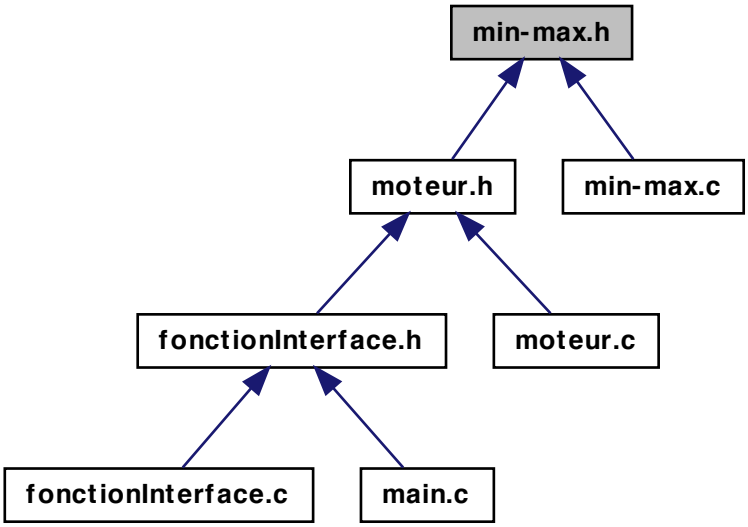
Implémentation du min-max.

```
#include "fonction_evaluation.h"
```

Graphe des dépendances par inclusion de min-max.h :



Ce graphe montre quels fichiers incluent directement ou indirectement ce fichier :




## Macros

– #define **MINIMUM** -1
– #define **MAXIMUM** 1

## Fonctions

– coup jouerIA (const plateau p, int profondeur)
*Renvoie le meilleur coup que peut jouer le joueur courant.*

### 4.9.1 Description détaillée

Implémentation du min-max.

**Auteur**

Bastien Auda

Définition dans le fichier min-max.h.

### 4.9.2 Documentation des fonctions

#### 4.9.2.1 coup jouerIA ( const plateau *p*, int *profondeur* )

Renvoie le meilleur coup que peut jouer le joueur courant.

**Paramètres**

| | |
|---|---|
| *profondeur* | La profondeur à laquelle on doit utiliser la fonction d'évaluation. (profondeur $> 0$) |

Définition à la ligne 19 du fichier min-max.c.

## 4.10 min-max.h

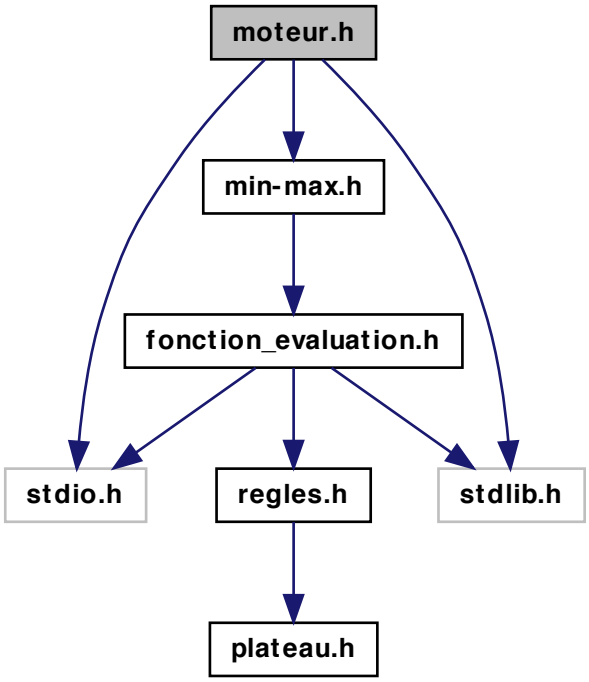
```
00001
00009 #define MINIMUM -1
00010 #define MAXIMUM 1
00011
00012 #include "fonction_evaluation.h"
00013
00020 coup jouerIA(const plateau p, int profondeur);
```

## 4.11 Référence du fichier moteur.h

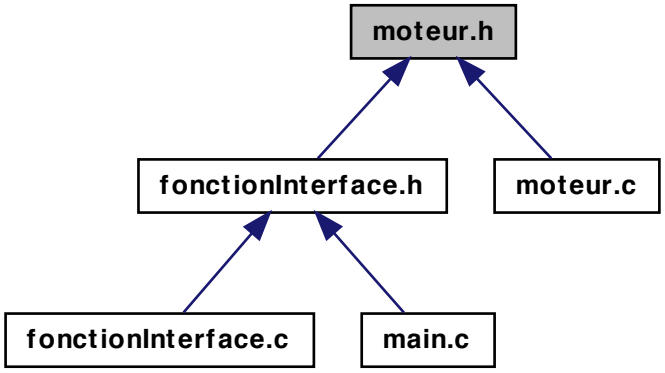
Le moteur de jeu. Il permet de gérer une partie, 1 ou 2 joueurs, fait jouer l'IA, fournit une aide au joueur.

```
#include <stdio.h>
#include <stdlib.h>
#include "min-max.h"
```

Graphe des dépendances par inclusion de moteur.h :



Ce graphe montre quels fichiers incluent directement ou indirectement ce fichier :




## Fonctions

– void initialiser_partie ()
*Initialise une nouvelle partie, par défaut une partie 2 joueurs, le joueur pourra ensuite être piloté par l'IA en changeant son type grâce à la fonction set_joueur_est_humain.*

– void set_joueur_est_humain (couleur_pion couleur, int boolean)
*Change le type humain ou IA du joueur.*

– plateau get_plateau ()
*Renvoie le plateau de la partie pour le consulter.*

– int sauvegarder_partie (char ∗filename)
*Sauvegarde l'état courant de la partie.*

– int charger_partie (char ∗filename)
*Charge une partie depuis le disque.*

– int jouer_coup (int depart, int arrivee)
*Joue le coup pour le joueur courant.*

– int **jouer_coup_xy** (int x1, int y1, int x2, int y2)
– void jouer_tour_ia ()
*Fait jouer un tour à l'IA.*

– void set_difficulte (int i)
*Joue sur la profondeur d'évaluation du min-max.*

– int commencer_tour ()
*Débute un nouveau tour de jeu, fait jouer l'IA si c'est à elle de jouer, attend un coup humain sinon.*

– void hint_pions_jouables ()
  *Met en surbrillance les pions qui permettent un déplacement valide pour ce tour.*

– int hint_deplacements_possibles (int c)
  *Met en surbrillance les déplacements possibles partant d'une case donnée.*

– int hint_deplacements_possibles_xy (int x, int y)
  *Identique à hint_deplacements_possibles(int c) mais prends les coordonnées (x,y) de la case.*

– void hint_meilleur_coup ()
  *Met en surbrillance le meilleur coup à jouer en utilisant l'IA.*

– int partie_terminee ()

### Variables

– plateau p_jeu

## 4.11.1 Description détaillée

Le moteur de jeu. Il permet de gérer une partie, 1 ou 2 joueurs, fait jouer l'IA, fournit une aide au joueur.

**Auteur**

Mehdi M'rah
Bastien Auda

Définition dans le fichier moteur.h.

## 4.11.2 Documentation des fonctions

### 4.11.2.1 int charger_partie ( char * *filename* )

Charge une partie depuis le disque.

**Paramètres**

| | |
|---|---|
| *filename* | Le chemin vers le fichier à charger. |

**Renvoie**

Vrai si le chargement s'est bien déroulé, faux sinon.

**Auteur**

Mehdi M'rah

Définition à la ligne 249 du fichier moteur.c.

#### 4.11.2.2 int commencer_tour ( )

Débute un nouveau tour de jeu, fait jouer l'IA si c'est à elle de jouer, attend un coup humain sinon.

**Renvoie**

Faux si on attend q'un humain joue, vrai si l'IA à joué et qu'on doit relancer immédiatement un nouveau tour.

**Auteur**

Bastien Auda

Définition à la ligne 107 du fichier moteur.c.

#### 4.11.2.3 plateau get_plateau ( )

Renvoie le plateau de la partie pour le consulter.

**Auteur**

Mehdi M'rah

Définition à la ligne 351 du fichier moteur.c.

#### 4.11.2.4 int hint_deplacements_possibles ( int *c* )

Met en surbrillance les déplacements possibles partant d'une case donnée.

**Paramètres**

| | |
|---|---|
| *c* | Notation officielle de la case de départ. |

**Renvoie**

Vrai si le pion séléctionné fait partie d'un coup authorisé.

**Auteur**

Bastien Auda

Définition à la ligne 154 du fichier moteur.c.

#### 4.11.2.5 void hint_meilleur_coup ( )

Met en surbrillance le meilleur coup à jouer en utilisant l'IA.

**Auteur**

Mehdi M'rah

#### 4.11.2.6 void hint_pions_jouables ( )

Met en surbrillance les pions qui permettent un déplacement valide pour ce tour.

**Auteur**

Bastien Auda

Définition à la ligne 142 du fichier moteur.c.

#### 4.11.2.7 void initialiser_partie ( )

Initialise une nouvelle partie, par défaut une partie 2 joueurs, le joueur pourra ensuite être piloté par l'IA en changeant son type grâce à la fonction set_joueur_est_humain.

void initialiser_partie() ;

**Auteur**

Mehdi M'rah

Définition à la ligne 325 du fichier moteur.c.

#### 4.11.2.8 int jouer_coup ( int *depart*, int *arrivee* )

Joue le coup pour le joueur courant.

**Paramètres**

| | |
|---|---|
| *depart* | La case de départ du coup. |
| *arrivee* | La case d'arrivée du coup. |

**Renvoie**

- 0 si le coup est invalide.
- 1 si le coup est valide et terminé.
- 2 si le mouvement fait partie d'un coup valide, on attend que le joueur termine son coup par un nouvel appel à cette fonction jouer_coup.

**Auteur**

Bastien Auda

Définition à la ligne 25 du fichier moteur.c.

#### 4.11.2.9 void jouer_tour_ia ( )

Fait jouer un tour à l'IA.

**Auteur**

Bastien Auda

Définition à la ligne 126 du fichier moteur.c.

#### 4.11.2.10 int partie_terminee ( )

**Renvoie**

Renvoie Faux si la partie n'est pas terminée, la couleur gagnante + 1 sinon.

Définition à la ligne 355 du fichier moteur.c.

#### 4.11.2.11 int sauvegarder_partie ( char * *filename* )

Sauvegarde l'état courant de la partie.

**Paramètres**

| | |
|---|---|
| *filename* | Le chemin vers le fichier de sauvegarde. |

**Renvoie**

Vrai si la sauvegarde s'est bien effectuée, faux si un problème est survenu.

**Auteur**

Mehdi M'rah

Définition à la ligne 185 du fichier moteur.c.

#### 4.11.2.12 void set_difficulte ( int *i* )

Joue sur la profondeur d'évaluation du min-max.

**Paramètres**

| | |
|---|---|
| *i* | La profondeur d'évaluation. |

**Auteur**

Bastien Auda

Définition à la ligne 138 du fichier moteur.c.

#### 4.11.2.13 void set_joueur_est_humain ( couleur_pion *couleur*, int *boolean* )

Change le type humain ou IA du joueur.

**Paramètres**

| | |
|---|---|
| *couleur* | La couleur du joueur pour lequel on doit changer le type. |
| *boolean* | Vrai si le joueur est humain, faux sinon. |

**Auteur**

Mehdi M'rah

Définition à la ligne 335 du fichier moteur.c.

### 4.11.3 Documentation des variables

#### 4.11.3.1 plateau p_jeu

Le plateau de jeu qui sera utilisé tout au long de la partie. (Ne pas utiliser en dehors de moteur.c, utilisez get_plateau())

Définition à la ligne 13 du fichier moteur.h.

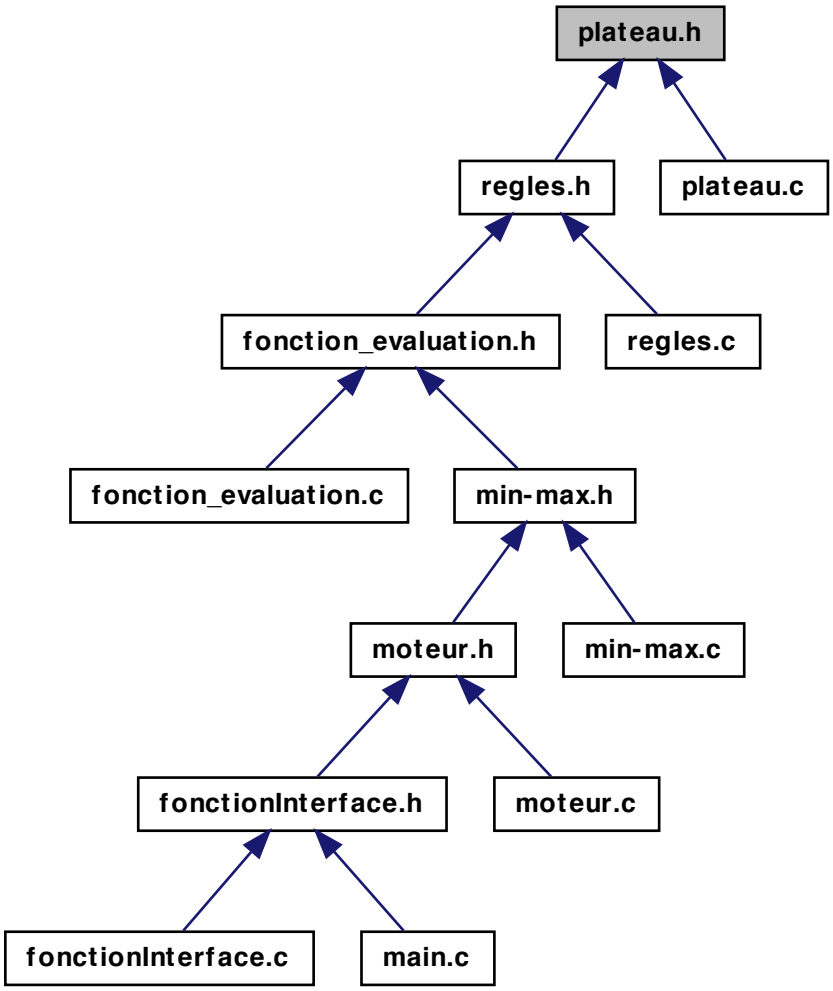
## 4.12 moteur.h

```
00001
00008 #include <stdio.h>
00009 #include <stdlib.h>
00010 #include "min-max.h"
00011
00013 plateau p_jeu;
00014
00020 void initialiser_partie();
00021
00029 void set_joueur_est_humain(couleur_pion couleur, int boolean);
00030
00031
00037 plateau get_plateau();
00038
00046 int sauvegarder_partie(char * filename);
00047
00055 int charger_partie(char * filename);
00056
00067 int jouer_coup(int depart, int arrivee);
00068
00073 int jouer_coup_xy(int x1, int y1, int x2, int y2);
00074
00080 void jouer_tour_ia();
00081
00088 void set_difficulte(int i);
00089
00096 int commencer_tour();
00097
00103 void hint_pions_jouables();
00104
00112 int hint_deplacements_possibles(int c);
00113
00118 int hint_deplacements_possibles_xy(int x,int y);
00119
00125 void hint_meilleur_coup();
00126
00131 int partie_terminee();
```

## 4.13 Référence du fichier plateau.h

Tout ce qui concerne le plateau de jeu (damier, pions et joueurs).

Ce graphe montre quels fichiers incluent directement ou indirectement ce fichier :




## Structures de données

– struct pion
*Objet pion. Un pion est déterminé par une couleur et si il est une dame ou non.*

– struct case_plateau
*Objet case du plateau de jeu. Une case est définie par une couleur et, si elle est noire,elle dispose d'une abscisse (x) et une ordonnée (y) sur le plateau. Une case peut être libre ou*

*non, si elle n'est pas libre elle contient un pion.*

- struct joueur
  *Objet joueur. Un joueur est caractérisé par la couleur qu'il joue et sa nature (humain ou intelligence artificielle).*

- struct coup
  *Objet coup. Un coup est défini par un numéro de case de départ, un numéro de case d'arrivée, un type (déplacement ou prise) et optinellement un commentaire.*

- struct plateau
  *Objet plateau. Le plateau est composé de 50 cases, numérotées de 1 à 50. Il comporte un historique des coups. Il sait quel joueur doit jouer le prochain coup.*

## Énumérations

- enum couleur_pion { **blanc**, **noir** }
  *La couleur d'un pion.*

- enum type_coup { x, t }
  *Le type de coup. Le "tiret" '-' est remplacé par un 't'.*

## Fonctions

- plateau nouveau_plateau (joueur j1, joueur j2)
  *Fonction de création d'un nouveau plateau.*

- int plateau_deplacer_pion (int old_position, int new_position, plateau *p)
  *Déplace un pion.*

- int plateau_prendre_pion (int position, plateau *p)
  *Prend le pion de la case "position".*

- void plateau_ajouter_coup (coup c, plateau *p)
  *Ajoute un coup dans l'historique du plateau.*

- void plateau_appliquer_coup (coup c, plateau *p)
  *Met à jour le plateau en jouant le coup donné et passe la main au joueur suivant.*

- int plateau_partie_finie (plateau p)
  *Teste si la partie est gagnée pour un des deux joueurs.*

- case_plateau get_case_plateau (int x, int y, plateau p)
  *Renvoie la case à la position (x,y).*

- void print_case (case_plateau c)
  *Affiche les informations sur la case (à des fins de test).*

- void print_plateau (plateau p)
  *Affiche le plateau sur stdout.*

- void set_case_en_surbrillance (int numero, plateau *p)
  *Met la case en surbrillance.*

– void set_pion_en_surbrillance (int numero, plateau ∗p)
  *Met un pion en surbrillance.*

– void reset_surbrillance (plateau ∗p)
  *Annule la mise en surbrillance de toutes les cases et tous les pions du plateau.*

– void printCoup (const coup c)
  *print les infos d'un coup*

– int est_prenable (int position, plateau ∗p)
  *predicat pour savoir si une position est prenable.*

– void print_liste_coups (coup ∗l)
  *affiche la liste des coups dans l.*

– int **coup_inclus** (case_plateau c, case_plateau ∗liste, int taille)
– case_plateau get_case_plateau_silent (int x, int y, plateau p)
  *retourne la case si elle existe, ou la case 0 sinon*

– int nombre_coups (coup ∗set)
  *retourne le nombre de coups dans set*

### 4.13.1 Description détaillée

Tout ce qui concerne le plateau de jeu (damier, pions et joueurs).

**Auteur**

Bastien Auda

Définition dans le fichier plateau.h.

### 4.13.2 Documentation du type de l'énumération

#### 4.13.2.1 enum type_coup

Le type de coup. Le "tiret" '-' est remplacé par un 't'.

**Valeurs énumérées :**

***x*** Prise.

***t*** Déplacement.

Définition à la ligne 22 du fichier plateau.h.

### 4.13.3 Documentation des fonctions

#### 4.13.3.1 int est_prenable ( int *position*, plateau ∗ *p* )

predicat pour savoir si une position est prenable.

**Paramètres**

| | |
|---|---|
| *position* | la position du pion que l'on veut prendre. |
| *p* | le plateau sur lequel on veut tester la position à prendre. |

Définition à la ligne 258 du fichier plateau.c.

#### 4.13.3.2 case_plateau get_case_plateau ( int *x*, int *y*, plateau *p* )

Renvoie la case à la position (x,y).

**Paramètres**

| | |
|---|---|
| *x* | Abscisse de la case (colonne) [1-10]. |
| *y* | Ordonnée de la case [1-10]. |

Définition à la ligne 101 du fichier plateau.c.

#### 4.13.3.3 case_plateau get_case_plateau_silent ( int *x*, int *y*, plateau *p* )

retourne la case si elle existe, ou la case 0 sinon

**Paramètres**

| | |
|---|---|
| *x* | la position horizontale sur le plateau |
| *y* | la position verticale sur le plateau |
| *p* | le plateau courant |

**Auteur**

Paraita Wohler

**Paramètres**

| | |
|---|---|
| *x* | la position en abscisse |
| *y* | la position en ordonnée |
| *p* | le plateau courant |

**Renvoie**

la case qui est en position x y dans p

Définition à la ligne 929 du fichier regles.c.

#### 4.13.3.4 int nombre_coups ( coup * *set* )

retourne le nombre de coups dans set

**Paramètres**

| | |
|---|---|
| *set* | le tableau des coups. |

**Auteur**

Paraita Wohler

Définition à la ligne 300 du fichier plateau.c.

#### 4.13.3.5 plateau nouveau_plateau ( joueur *j1*, joueur *j2* )

Fonction de création d'un nouveau plateau.

**Paramètres**

| | |
|---|---|
| *j1* | Le joueur blanc. |
| *j2* | Le joueur noir. |

**Renvoie**

plateau Un plateau initialisé avec les pions en position initiale et associé aux joueurs.

Définition à la ligne 11 du fichier plateau.c.

#### 4.13.3.6 void plateau_ajouter_coup ( coup *c*, plateau * *p* )

Ajoute un coup dans l'historique du plateau.

**Paramètres**

| | |
|---|---|
| *c* | Le coup à ajouter. |
| *p* | Le plateau auquel on ajoute le coup. En raison du risque improbable qu'une partie dure plus de 500 coups, l'historique au-delà est ignoré. |

Définition à la ligne 93 du fichier plateau.c.

#### 4.13.3.7 void plateau_appliquer_coup ( coup *c*, plateau * *p* )

Met à jour le plateau en jouant le coup donné et passe la main au joueur suivant.

**Paramètres**

| | |
|---|---|
| *c* | Le coup à jouer. |

Définition à la ligne 141 du fichier plateau.c.

#### 4.13.3.8 int plateau_deplacer_pion ( int *old_position*, int *new_position*, plateau * *p* )

Déplace un pion.

**Paramètres**

| | |
|---|---|
| *old_position* | La position de départ. |

| | |
|---|---|
| *new_-position* | La position d'arrivée. |

**Renvoie**

int Vrai si le déplacement est possible, faux si un pion occupe déjà la case ou si on sort du plateau.

Définition à la ligne 70 du fichier plateau.c.

#### 4.13.3.9 int plateau_partie_finie ( plateau *p* )

Teste si la partie est gagnée pour un des deux joueurs.

**Renvoie**

0 si la partie n'est pas finie, couleur + 1 sinon.

Définition à la ligne 175 du fichier plateau.c.

#### 4.13.3.10 int plateau_prendre_pion ( int *position*, plateau * *p* )

Prend le pion de la case "position".

**Paramètres**

| | |
|---|---|
| *position* | Position de la case sur laquelle se trouve le pion. |

**Renvoie**

int Vrai si la prise est effectuée, faux si la case est vide ou si on sort du plateau.

Définition à la ligne 84 du fichier plateau.c.

#### 4.13.3.11 void print_case ( case_plateau *c* )

Affiche les informations sur la case (à des fins de test).

**Paramètres**

| | |
|---|---|
| *c* | La case à afficher. |

Définition à la ligne 120 du fichier plateau.c.

#### 4.13.3.12 void print_liste_coups ( coup * *l* )

affiche la liste des coups dans l.

**Paramètres**

| | |
|---|---|
| *l* | la liste des coups que l'on veut afficher. |

**Auteur**

Paraita Wohler

Définition à la ligne 269 du fichier plateau.c.

#### 4.13.3.13 void printCoup ( const coup *c* )

print les infos d'un coup

**Paramètres**

| | |
|---|---|
| *c* | le coup dont on veut les informations. |

**Auteur**

Paraita Wohler

Définition à la ligne 234 du fichier plateau.c.

#### 4.13.3.14 void set_case_en_surbrillance ( int *numero*, plateau * *p* )

Met la case en surbrillance.

**Paramètres**

| | |
|---|---|
| *numero* | Numéro de la case à mettre en surbrillance (selon la notation officielle). |

Définition à la ligne 125 du fichier plateau.c.

#### 4.13.3.15 void set_pion_en_surbrillance ( int *numero*, plateau * *p* )

Met un pion en surbrillance.

**Paramètres**

| | |
|---|---|
| *numero* | Numéro de la case sur laquelle se trouve le pion à mettre en surbrillance (selon la notation officielle). |

Définition à la ligne 129 du fichier plateau.c.

## 4.14 plateau.h

```
00001
00012 typedef enum {
00013         blanc,
00014         noir
00015 } couleur_pion;
00016
00022 typedef enum {
00023         x,
00024         t
```

```
00025 } type_coup;
00026
00027
00033 typedef struct {
00034         couleur_pion couleur;
00035         int est_dame;
00036         int en_surbrillance;
00037 } pion;
00038
00045 typedef struct {
00046         couleur_pion couleur;
00047         int est_libre;
00048         pion pion;
00049         int x;
00050         int y;
00051         int notation_officielle;
00052         int en_surbrillance;
00053 } case_plateau;
00054
00055
00056
00062 typedef struct {
00063         int est_humain;
00064         couleur_pion couleur;
00065 } joueur;
00066
00067
00068
00074 typedef struct {
00075         int old_case;
00076         int new_case;
00077         type_coup tc;
00078         int nombre_prises;
00079         case_plateau prises[20];
00080         case_plateau chemin[20];
00081         char * commentaire;
00097 } coup;
00098
00099
00107 typedef struct {
00108         case_plateau cases[51];
00109         coup historique[500];
00110         int i;
00111         joueur joueur1;
00112         joueur joueur2;
00113         joueur tour;
00114 } plateau;
00115
00116 /* ********* Fin des définitions de types, début des définitions de fonctions ***
      **********************/
00117
00126 plateau nouveau_plateau(joueur j1, joueur j2);
00127
00128
00137 int plateau_deplacer_pion(int old_position,int new_position,plateau *p);
00138
00146 int plateau_prendre_pion(int position,plateau *p);
00147
00156 void plateau_ajouter_coup(coup c,plateau *p);
00157
00163 void plateau_appliquer_coup(coup c, plateau * p);
00164
```

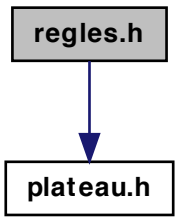
```
00170 int plateau_partie_finie(plateau p);
00171
00179 case_plateau get_case_plateau(int x, int y,plateau p);
00180
00187 void print_case(case_plateau c);
00188
00193 void print_plateau(plateau p);
00194
00201 void set_case_en_surbrillance(int numero,plateau *p);
00202
00209 void set_pion_en_surbrillance(int numero,plateau *p);
00210
00215 void reset_surbrillance(plateau *p);
00216
00217
00225 void printCoup(const coup c);
00226
00234 int est_prenable(int position, plateau *p);
00235
00236
00244 void print_liste_coups(coup *l);
00245
00246
00255 int coup_inclus(case_plateau c, case_plateau *liste, int taille);
00256
00266 case_plateau get_case_plateau_silent(int x, int y, plateau p);
00267
00275 int nombre_coups(coup *set);
```

## 4.15 Référence du fichier regles.h

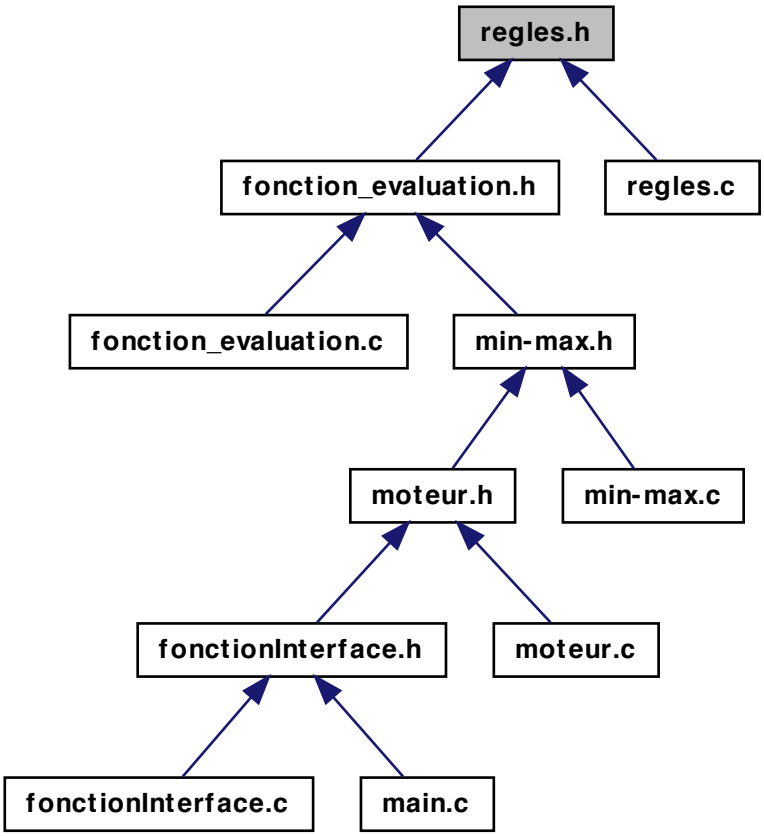
module des regles et d'enumeration des coups.

`#include "plateau.h"`

Graphe des dépendances par inclusion de regles.h :



Ce graphe montre quels fichiers incluent directement ou indirectement ce fichier :




## Fonctions

– coup ∗ coupsPossibles (const case_plateau c, const plateau p)
*retourne le tableau des coups possible pour un coup donné.*

– coup ∗ getCoups (const joueur j, const plateau p)
*retourne le tableau des coups possible pour le joueur donné.*

– coup ∗ **getCoupsMax** (const coup ∗cp)
– coup ∗ get_deplacements (const case_plateau c, const plateau p)
*retourne les deplacements possibles du pion de la case donné*

– coup ∗ completer_coup_dame (const coup c, int mvt, plateau p)
– int get_possible_case_pos (int c, int diag, plateau p)

– int compare_coups (coup ∗liste1, coup ∗liste2)
– case_plateau get_case_plateau_silent (int x, int y, plateau p)
*retourne la case si elle existe, ou la case 0 sinon*

### 4.15.1 Description détaillée

module des regles et d'enumeration des coups.

**Auteur**

Paraita Wohler

Définition dans le fichier regles.h.

### 4.15.2 Documentation des fonctions

#### 4.15.2.1 int compare_coups ( coup ∗ *liste1*, coup ∗ *liste2* )

**Paramètres**

| | |
|---|---|
| *liste1* | la premiere liste des coups |
| *liste2* | la seconde liste des coups |

**Renvoie**

0 si liste 1 == liste2, sinon 1

Définition à la ligne 913 du fichier regles.c.

#### 4.15.2.2 coup∗ completer_coup_dame ( const coup *c*, int *mvt*, plateau *p* )

**Paramètres**

| | |
|---|---|
| *mvt* | La diagonale du mouvement du coup précedent. |

**Renvoie**

La liste des coups possibles commençant par le coup c.

**Auteur**

Bastien Auda

Définition à la ligne 320 du fichier regles.c.

#### 4.15.2.3 coup∗ coupsPossibles ( const case_plateau *c*, const plateau *p* )

retourne le tableau des coups possible pour un coup donné.

**Paramètres**

| | |
|---|---|
| *c* | La case qui contient le pion de la recherche. |
| *p* | la plateau courant. |

Définition à la ligne 24 du fichier regles.c.

#### 4.15.2.4 case_plateau get_case_plateau_silent ( int *x*, int *y*, plateau *p* )

retourne la case si elle existe, ou la case 0 sinon

**Paramètres**

| | |
|---|---|
| *x* | la position horizontale sur le plateau |
| *y* | la position verticale sur le plateau |
| *p* | le plateau courant |

**Auteur**

Paraita Wohler

**Paramètres**

| | |
|---|---|
| *x* | la position en abscisse |
| *y* | la position en ordonnée |
| *p* | le plateau courant |

**Renvoie**

la case qui est en position x y dans p

Définition à la ligne 929 du fichier regles.c.

#### 4.15.2.5 coup* get_deplacements ( const case_plateau *c*, const plateau *p* )

retourne les deplacements possibles du pion de la case donné

**Paramètres**

| | |
|---|---|
| *c* | la case ou est situé le pion dont on va chercher les déplacements possibles. |

Définition à la ligne 859 du fichier regles.c.

#### 4.15.2.6 int get_possible_case_pos ( int *c*, int *diag*, plateau *p* )

**Paramètres**

| | |
|---|---|
| *c* | Notation officielle de la case de départ. |
| *diag* | Diagonale dans laquelle on recherche. |

**Renvoie**

0 si pas de cpions à prendre, la case sur laquelle se trouve le pion à prendre sinon.

Définition à la ligne 249 du fichier regles.c.

#### 4.15.2.7 coup* getCoups ( const joueur *j*, const plateau *p* )

retourne le tableau des coups possible pour le joueur donné.

**Paramètres**

| | |
|---|---|
| *j* | le joueur dont on chercher tout les coups |
| *p* | la plateau courant. |

Définition à la ligne 777 du fichier regles.c.

## 4.16 regles.h

```
00001
00006 /* ---------------------------------------------- */
00007
00008 #include "plateau.h"
00009
00017 coup* coupsPossibles(const case_plateau c, const plateau p);
00018
00019
00027 coup* getCoups(const joueur j, const plateau p);
00028
00029
00036 coup* getCoupsMax(const coup *cp);
00037
00038
00045 coup* get_deplacements(const case_plateau c, const plateau p);
00046
00047
00054 coup * completer_coup_dame(const coup c, int mvt, plateau p);
00055
00056
00063 int get_possible_case_pos(int c, int diag, plateau p) ;
00064
00065
00072 int compare_coups(coup *liste1, coup *liste2);
00073
00074
00082 case_plateau get_case_plateau_silent(int x, int y, plateau p);
```

# Index