

Reference Manual

Generated by Doxygen 1.7.4

Thu Jan 26 2012 17:34:20

Contents

1	Class Index	1
1.1	Class List	1
2	File Index	3
2.1	File List	3
3	Class Documentation	5
3.1	SDisplay Struct Reference	5
3.1.1	Detailed Description	6
3.2	SGame Struct Reference	6
3.2.1	Detailed Description	6
3.3	SGameState Struct Reference	7
3.3.1	Detailed Description	7
3.4	SIA_Functions Struct Reference	7
3.4.1	Detailed Description	8
3.5	SMove Struct Reference	8
3.5.1	Detailed Description	8
3.6	SZone Struct Reference	8
3.6.1	Detailed Description	8
4	File Documentation	11
4.1	backgammon.h File Reference	11
4.1.1	Detailed Description	12
4.1.2	Typedef Documentation	12
4.1.2.1	pfDoubleStack	12
4.1.2.2	pfEndGame	12

4.1.2.3	pfEndMatch	12
4.1.2.4	pfMakeDecision	13
4.1.2.5	pfStartGame	13
4.1.2.6	pfTakeDouble	13
4.2	definitions.h File Reference	13
4.2.1	Detailed Description	16
4.2.2	Enumeration Type Documentation	17
4.2.2.1	EGameMode	17
4.2.3	Function Documentation	17
4.2.3.1	Arbitrator_AuthorizedDeplacement	17
4.2.3.2	Arbitrator_EmptyPosition	17
4.2.3.3	Arbitrator_NumberOfDieCanPlay	17
4.2.3.4	Arbitrator_NumberOfDieForMove	18
4.2.3.5	Arbitrator_Pion_Depart_Autorise	18
4.2.3.6	Arbitrator_PlayerArrays	18
4.2.3.7	Arbitrator_Sens_rotation_correct	19
4.2.3.8	Arbitrator_Taille_Mouvement_Correcte	19
4.2.3.9	case_appartenant_joueur_adverse_avec_un_pion	19
4.2.3.10	Check_Args	20
4.2.3.11	CheckerWithScreenPosition	20
4.2.3.12	colorChecker	20
4.2.3.13	copyGameState	20
4.2.3.14	Display_Arrow_Possibilities	21
4.2.3.15	Display_CheckerDraw	21
4.2.3.16	Display_CheckerMove	21
4.2.3.17	Display_Checkers	22
4.2.3.18	Display_CheckersPossibilities	22
4.2.3.19	Display_Clear	22
4.2.3.20	Display_Die	23
4.2.3.21	Display_DrawBar	23
4.2.3.22	Display_DrawOut	23
4.2.3.23	Display_DrawSelectedArrow	23
4.2.3.24	Display_Exit	24
4.2.3.25	Display_GameActions	24

4.2.3.26	Display_Init	24
4.2.3.27	Display_Message	24
4.2.3.28	Display_RefreshGameBoard	25
4.2.3.29	Display_Score	25
4.2.3.30	Free_SAI	25
4.2.3.31	Game_FirstToPlay	25
4.2.3.32	Game_Init	26
4.2.3.33	Game_LaunchDie	26
4.2.3.34	Game_Play	26
4.2.3.35	get_distance	26
4.2.3.36	Init_SAI	27
4.2.3.37	inTab	27
4.2.3.38	Load_API	27
4.2.3.39	Menu_Click	27
4.2.3.40	Menu_Display	28
4.2.3.41	Menu_Fill	28
4.2.3.42	Menu_Text	28
4.2.3.43	Menu_TextInput	28

Chapter 1

Class Index

1.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

SDisplay (Structure d'affichage)	5
SGame (Structure contenant les informations du jeu)	6
SGameState (Structure de représentation du jeu avec son plateau et les dés)	7
SIA_Functions (Structure des fonctions de l'IA)	7
SMove (Structure de représentation d'un mouvement)	8
SZone (Structure de représentation d'une zone du jeu)	8

Chapter 2

File Index

2.1 File List

Here is a list of all documented files with brief descriptions:

- [backgammon.h](#) (Contient les signatures des fonctions et les différentes structures communes à la promotion) 11
- [definitions.h](#) (Contient les signatures des fonctions et les différentes structures) 13

Chapter 3

Class Documentation

3.1 SDisplay Struct Reference

Structure d'affichage.

```
#include <definitions.h>
```

Public Attributes

- SDL_Surface * **screen**
- char * **font_path**
- char * **img_path**
- int **window_width**
- int **window_height**
- SDL_Surface * **background_menu**
- SDL_Surface * **checked**
- SDL_Surface * **background**
- SDL_Rect **background_position**
- SDL_Surface * **gameBoard**
- SDL_Rect **gameBoard_position**
- TTF_Font * **font**
- SDL_Surface * **msg_box**
- SDL_Surface * **green_checker**
- SDL_Surface * **white_checker**
- SDL_Surface * **out_green_checker**
- SDL_Surface * **out_white_checker**
- SDL_Surface * **selected_checker**
- SDL_Surface * **videau**
- SDL_Rect **videau_position**
- SDL_Surface * **die** [6]
- SDL_Surface * **die_played** [6]
- SDL_Rect **die1_position**

- SDL_Rect **die2_position**
- SDL_Surface * **possibility1_12**
- SDL_Surface * **possibility13_24**
- SDL_Surface * **possibility_out**
- SDL_Surface * **gameActions** [2]
- SDL_Rect **positions** [28]

3.1.1 Detailed Description

Structure d'affichage.

Contient les éléments nécessaires à l'affichage

The documentation for this struct was generated from the following file:

- [definitions.h](#)

3.2 SGame Struct Reference

Structure contenant les informations du jeu.

```
#include <definitions.h>
```

Public Attributes

- char * **player1_name**
- char * **player2_name**
- int **player1_checker**
- int **player2_checker**
- int **withDouble**
- int **doubleValue**
- unsigned int **scoreLimit**
- int **die_To_Play** [4]

3.2.1 Detailed Description

Structure contenant les informations du jeu.

Contient les informations du jeu à savoir le nom des deux joueurs, la couleur de leurs pions, l'utilisation du double, sa valeur, la limite de score et le tableau de statut des dés.

The documentation for this struct was generated from the following file:

- [definitions.h](#)

3.3 SGameState Struct Reference

Structure de représentation du jeu avec son plateau et les dés.

```
#include <backgammon.h>
```

Public Attributes

- [SZone](#) **zones** [28]
- unsigned int **die1**
- unsigned int **die2**
- unsigned int **score**
- unsigned int **scoreP2**
- unsigned int **stake**

3.3.1 Detailed Description

Structure de représentation du jeu avec son plateau et les dés.

The documentation for this struct was generated from the following file:

- [backgammon.h](#)

3.4 SIA_Funcions Struct Reference

Structure des fonctions de l'IA.

```
#include <definitions.h>
```

Public Attributes

- void * **IA_Handle**
- pfInitLibrary **IA_InitLibrary**
- pfStartMatch **IA_StartMatch**
- [pfEndMatch](#) **IA_EndMatch**
- [pfStartGame](#) **IA_StartGame**
- [pfEndGame](#) **IA_EndGame**
- [pfDoubleStack](#) **IA_DoubleStack**
- [pfTakeDouble](#) **IA_TakeDouble**
- [pfMakeDecision](#) **IA_MakeDecision**

3.4.1 Detailed Description

Structure des fonctions de l'IA.

Contient les pointeurs de fonctions sur les fonctions de l'IA

The documentation for this struct was generated from the following file:

- [definitions.h](#)

3.5 SMove Struct Reference

Structure de représentation d'un mouvement.

```
#include <backgammon.h>
```

Public Attributes

- [EPosition](#) **src_point**
- [EPosition](#) **dest_point**

3.5.1 Detailed Description

Structure de représentation d'un mouvement.

The documentation for this struct was generated from the following file:

- [backgammon.h](#)

3.6 SZone Struct Reference

Structure de représentation d'une zone du jeu.

```
#include <backgammon.h>
```

Public Attributes

- [EPlayer](#) **player**
- unsigned int **nb_checkers**

3.6.1 Detailed Description

Structure de représentation d'une zone du jeu.

The documentation for this struct was generated from the following file:

- [backgammon.h](#)

Chapter 4

File Documentation

4.1 backgammon.h File Reference

Contient les signatures des fonctions et les différentes structures communes à la promotion.

Classes

- struct [SZone](#)
Structure de représentation d'une zone du jeu.
- struct [SGameState](#)
Structure de représentation du jeu avec son plateau et les dés.
- struct [SMove](#)
Structure de représentation d'un mouvement.

Typedefs

- typedef void(* [pfInitLibrary](#))(char[50])
- typedef void(* [pfStartMatch](#))(const unsigned int)
- typedef void(* [pfStartGame](#))()
- typedef void(* [pfEndGame](#))()
- typedef void(* [pfEndMatch](#))()
- typedef int(* [pfDoubleStack](#))(const [SGameState](#) *const)
- typedef int(* [pfTakeDouble](#))(const [SGameState](#) *const)
- typedef void(* [pfMakeDecision](#))(const [SGameState](#) *const, [SMove](#)[4], unsigned int)

Enumerations

- enum [EPlayer](#) { [EPlayer1](#), [EPlayer2](#) }

Représente un joueur.

- enum [EPosition](#) {
EPos_nopos = -1, **EPos_1** = 0, **EPos_2**, **EPos_3**,
EPos_4, **EPos_5**, **EPos_6**, **EPos_7**,
EPos_8, **EPos_9**, **EPos_10**, **EPos_11**,
EPos_12, **EPos_13**, **EPos_14**, **EPos_15**,
EPos_16, **EPos_17**, **EPos_18**, **EPos_19**,
EPos_20, **EPos_21**, **EPos_22**, **EPos_23**,
EPos_24, **EPos_OutP1**, **EPos_BarP1**, **EPos_OutP2**,
EPos_BarP2 }

Enumeration des zones pour le tableau points.

4.1.1 Detailed Description

Contient les signatures des fonctions et les différentes structures communes à la promotion.

Date

26 janvier 2012

4.1.2 Typedef Documentation

4.1.2.1 typedef int(* pfDoubleStack)(const SGameState *const)

Doubler la mise

Parameters

<i>const</i>	SGameState * const gameState l'état du jeu courant
--------------	--------------------------------------------------------------------

Returns

int vrai si on propose de doubler : faux sinon

4.1.2.2 typedef void(* pfEndGame)()

Fin d'une manche (d'un match)

4.1.2.3 typedef void(* pfEndMatch)()

Fin d'un match

4.1.2.4 `typedef void(* pfMakeDecision)(const SGameState *const, SMove[4], unsigned int)`

Prise de décision de la part de l'IA

Parameters

<i>const</i>	SGameState * const gameState l'état du jeu courant
SMove	moves[4] tableau des mouvements à effectuer par l'IA
<i>unsigned</i>	int lastTimeError vrai si la dernière action a causée une erreur

4.1.2.5 `typedef void(* pfStartGame)()`

Initialiser l'IA pour une manche (d'un match)

4.1.2.6 `typedef int(* pfTakeDouble)(const SGameState *const)`

Accepter ou refuser la nouvelle mise

Parameters

<i>const</i>	SGameState * const gameState l'état du jeu courant
--------------	--------------------------------------------------------------------

Returns

int vrai si on accepte la nouvelle mise ; faux sinon

4.2 definitions.h File Reference

Contient les signatures des fonctions et les différentes structures.

```
#include "backgammon.h"
#include <SDL/SDL.h>
#include <SDL/SDL_ttf.h>
```

Classes

- struct [SGame](#)
Structure contenant les informations du jeu.
- struct [SDisplay](#)
Structure d'affichage.
- struct [SIA_Functions](#)
Structure des fonctions de l'IA.

Defines

- `#define WHITE 1`
- `#define GREEN 0`

Enumerations

- enum `EGameMode` { `HUMAN_HUMAN`, `HUMAN_IA`, `IA_IA`, `ERROR = -1` }

Enumération des modes de jeu.

Functions

- `SGameState * Game_Init ()`
Fonction qui permet d'initialiser l'état du jeu (disposition des jetons au début de la partie).
- `int Game_Play (SDisplay *display, EGameMode gameMode, SGame *game, SIA_Functions *IA_struct)`
Lance la boucle de jeu.
- `void Game_LaunchDie (SGameState *gameState, SGame *game)`
Lance les dés et réinitialise le tableau d'état des dés.
- `int Game_FirstToPlay (SDisplay *display, EGameMode gameMode, SGame *game, SGameState *gameState)`
Détermine quel joueur va commencer la partie par lancement des dés.
- `SGameState * copyGameState (SGameState *gameState, EPlayer player)`
Effectue une copie de l'état du jeu suivant le joueur à qui cette copie est destinée.
- `void Display_Init (SDisplay *display, SGame *game)`
Initialise la SDL, la structure d'affichage et la structure d'informations sur le jeu.
- `void Display_Exit (SDisplay *display)`
Libère les surfaces utilisées dans la structure d'affichage et quitte la SDL.
- `void Display_Checkers (SDisplay *display, SGameState *gameState, SGame *game)`
Affiche les pions à l'écran.
- `void Display_Die (SDisplay *display, SGameState *gameState, SGame *game)`
Affiche les dés à l'écran.
- `void Display_Score (SDisplay *display, SGameState *gameState, SGame *game)`
Affiche les scores à l'écran.
- `void Display_Clear (SDisplay *display)`
Réinitialise l'affichage.
- `void Display_CheckerDraw (SDisplay *display, SDL_Rect position, int player, SGame *game)`
Dessine un pion d'un joueur à une position donnée.

- void [Display_CheckerMove](#) ([SDisplay](#) *display, [SGameState](#) *gameState, [SMove](#) *move, [SGame](#) *game)
Effectue le déplacement d'un pion suivant un mouvement donné et met à jour l'état du jeu.
- int [Display_Message](#) ([SDisplay](#) *display, char *message, [SDL_Rect](#) position, [SDL_Color](#) color, int box, int clic)
Affiche un message à l'écran.
- int * [Display_Arrow_Possibilities](#) ([SDisplay](#) *display, [SGameState](#) *gameState, [EPlayer](#) player, [EPosition](#) depart, [SGame](#) *game)
Determine en fonction du pion d'un joueur les flèches qui sont possibles à atteindre.
- void [Display_DrawSelectedArrow](#) ([SDisplay](#) *display, [EPosition](#) pos)
Colorie la flèche à la position pos en bleu.
- int [Display_CheckersPossibilities](#) ([SDisplay](#) *display, [SGameState](#) *gameState, [EPlayer](#) player, [SGame](#) *game)
Affiche à l'écran les pions qui peuvent être déplacé suivant l'état du jeu et le joueur courant.
- int [CheckerWithScreenPosition](#) (int x, int y, [EPosition](#) *pos)
Donne la position en fonction d'un clic sur le jeu dans le poiteur de [EPosition](#) pos.
- void [colorChecker](#) ([SDisplay](#) *display, [SGameState](#) *gameState, [EPosition](#) pos)

Colorie le dernier pion d'une flèche donnée.
- void [Display_RefreshGameBoard](#) ([SDisplay](#) *display, [SGameState](#) *gameState, [SGame](#) *game)
Raffraichit l'affichage du plateau de jeu(pions, dés et videau).
- int [inTab](#) ([EPosition](#) p, int *tab)
Détermine si la position p est dans le tableau tab.
- void [Display_DrawOut](#) ([SDisplay](#) *display, [SGameState](#) *gameState, [SGame](#) *game)

Affiche les pions présents dans les zones de sorties.
- void [Display_DrawBar](#) ([SDisplay](#) *display, [SGameState](#) *gameState, [SGame](#) *game)

Affiche les pions présents dans les barres (prisonniers).
- int [Display_GameActions](#) ([SDisplay](#) *display, [SGameState](#) *gameState, [SGame](#) *game)
Affiche à l'écran les actions de jeu possibles (doubler la mise, lancer les dés).
- void [Menu_Fill](#) ([SDisplay](#) *display, [EGameMode](#) gameMode, [SGame](#) *game)
Remplis le menu avec les options correspondants au mode de jeu.
- void [Menu_TextInput](#) (char *name, [SDL_keysym](#) key)
Ajoute à une chaine de caractère un caractère obtenu par l'événement [SDL_KEYUP](#).
- void [Menu_Text](#) ([SDisplay](#) *display, char *message, [SDL_Rect](#) position, [SDL_Color](#) color)
Affiche un texte à une certaine position d'une certaine couleur.
- int [Menu_Click](#) (int x, int y)
Renvoie le code de l'élément cliqué sur le menu.

- int `Menu_Display` (`SDisplay` *display, `EGameMode` gameMode, `SGame` *game)

Affiche le menu.
- `SIA_Functions` * `Init_SAI` ()
Retourne une structure de fonctions de l'IA initialisée.
- void `Free_SAI` (`SIA_Functions` *IA_struct)
Libère une structure SIA.
- int `Load_API` (char *path, `SIA_Functions` *IA_struct, int ind)
Charge la librairie passé en paramètre et initialise la structure IA.
- `EGameMode` `Check_Args` (int argc, char **argv, `SIA_Functions` *IA_struct)
Parse les arguments et détermine ainsi le mode de jeu et les SIA à utiliser.
- int `Arbitrator_AuthorizedDeplacement` (`SGameState` *gameState, `SMove` *move, `EPlayer` player, `SGame` *game)
Analyse la validité d'un mouvement.
- int `Arbitrator_PlayerArrays` (`SZone` zone, `EPlayer` player)
Détermine si une flèche est possédée par un joueur donné.
- int `case_appartenant_joueur_adverse_avec_un_pion` (`SZone` zone, `EPlayer` player)
- int `Arbitrator_EmptyPosition` (`SZone` zone)
Détermine si une flèche est vide (sans pions).
- unsigned int `get_distance` (`EPosition` depart, `EPosition` arrivee)
Détermine la distance entre une position de départ et une d'arrivée.
- int `Arbitrator_Sens_rotation_correct` (`EPlayer` player, `EPosition` depart, `EPosition` arrivee)
Détermine si le sens d'un mouvement est correct.
- int `Arbitrator_Pion_Depart_Autorise` (int x, int y, `EPlayer` player, `SGameState` *game, `EPosition` posDepart)
Détermine si le joueur donné a cliqué sur une zone contenant un pion déplaçable par ce joueur.
- int `Arbitrator_NumberOfDieForMove` (`SGameState` *game, `EPlayer` player, `SMove` move)
Détermine le nombre de dés utilisés pour faire un mouvement.
- int `Arbitrator_NumberOfDieCanPlay` (`SGameState` *gameState, `EPlayer` player)

Détermine le nombre de dés qui peuvent être joués dans la partie (utile si le joueur est bloqué par exemple).
- int `Arbitrator_Taille_Mouvement_Correcte` (unsigned int taille_mouvement, `SGame` *game, `SGameState` *gameState)
Détermine si le mouvement correspond à un des dés encore jouable.

4.2.1 Detailed Description

Contient les signatures des fonctions et les différentes structures.

Author

Alexandre BISIAUX et Antonin BIRET

Date

26 janvier 2012

4.2.2 Enumeration Type Documentation**4.2.2.1 enum EGameMode**

Enumération des modes de jeu.

4 Modes de jeu : 1) Humain contre Humain 2) Humain contre IA 3) IA contre IA 4) Invalide

4.2.3 Function Documentation**4.2.3.1 int Arbitrator_AuthorizedDeplacement (SGameState * *gameState*, SMove * *move*, EPlayer *player*, SGame * *game*)**

Analyse la validité d'un mouvement.

Parameters

<i>gameState</i>	Etat du jeu
<i>move</i>	Mouvement à analyser
<i>player</i>	Joueur voulant effectuer ce mouvement
<i>game</i>	Informations sur le jeu (noms des joueurs, couleurs des pions des joueurs,...)

Returns

Retourne vrai si le mouvement est autorisé, faux sinon.

4.2.3.2 int Arbitrator_EmptyPosition (SZone *zone*)

Détermine si une flèche est vide (sans pions).

Parameters

<i>zone</i>	Flèche à analyser
-------------	-------------------

Returns

Retourne vrai si la flèche est vide, faux sinon.

4.2.3.3 int Arbitrator_NumberOfDieCanPlay (SGameState * *gameState*, EPlayer *player*)

Détermine le nombre de dés qui peuvent être joués dans la partie (utile si le joueur est bloqué par exemple).

Parameters

<i>gameState</i>	Etat du jeu
<i>player</i>	Joueur à analyser

Returns

Retourne le nombre de dés pouvant être joués

4.2.3.4 int Arbitrator.NumberOfDieForMove (SGameState * game, EPlayer player, SMove move)

Détermine le nombre de dés utilisés pour faire un mouvement.

Parameters

<i>gameState</i>	Etat du jeu
<i>player</i>	Joueur voulant effectuer ce mouvement
<i>move</i>	Mouvement à analyser

Returns

Retourne le nombre de dés utilisés

4.2.3.5 int Arbitrator.Pion_Depart_Autorise (int x, int y, EPlayer player, SGameState * game, EPosition posDepart)

Détermine si le joueur donné a cliqué sur une zone contenant un pion déplaçable par ce joueur.

x Abscisse de la position du clic y Ordonnée de la position du clic

Parameters

<i>player</i>	Joueur voulant effectuer ce mouvement
<i>game</i>	Informations sur le jeu (noms des joueurs, couleurs des pions des joueurs,...)
<i>depart</i>	Position de départ

Returns

Retourne vrai si le joueur a cliqué sur une zone contenant un pion déplaçable, faux sinon.

4.2.3.6 int Arbitrator.PlayerArrays (SZone zone, EPlayer player)

Détermine si une flèche est possédée par un joueur donné.

Parameters

<i>zone</i>	Flèche à analyser
<i>player</i>	Joueur supposé détenir la flèche

Returns

Retourne vrai si la flèche est possédée par le joueur, faux sinon.

4.2.3.7 `int Arbitrator_Sens_rotation_correct (EPlayer player, EPosition depart, EPosition arrivee)`

Détermine si le sens d'un mouvement est correct.

Parameters

<i>player</i>	Joueur faisant le mouvement
<i>depart</i>	Position de départ
<i>arrivee</i>	Position d'arrivée

Returns

Retourne vrai si le sens est correct, faux sinon.

4.2.3.8 `int Arbitrator_Taille_Mouvement_Correcte (unsigned int taille_mouvement, SGame * game, SGameState * gameState)`

Détermine si le mouvement correspond à un des dés encore jouable.

Parameters

<i>taille_mouvement</i>	Taille du mouvement (Distance)
<i>game</i>	Informations sur le jeu (noms des joueurs, couleurs des pions des joueurs,...)
<i>gameState</i>	Etat du jeu

Returns

Retourne vrai si correspondance, faux sinon.

4.2.3.9 `int case_appartenant_joueur_adverse_avec_un_pion (SZone zone, EPlayer player)`

Parameters

<i>zone</i>	
<i>player</i>	

Returns**4.2.3.10 EGameMode Check_Args (int argc, char ** argv, SIA_Functions * IA_struct)**

Parse les arguments et détermine ainsi le mode de jeu et les SIA à utiliser.

Parameters

<i>argc</i>	Nombre d'arguments
<i>argv</i>	Liste des arguments
<i>IA_struct</i>	Tableau de deux structures SIA

Returns

Retourne le mode de jeu

4.2.3.11 int CheckerWithScreenPosition (int x, int y, EPosition * pos)

Donne la position en fonction d'un clic sur le jeu dans le poiteur de EPosition pos.

Parameters

<i>x</i>	Position en abscisse
<i>y</i>	Position en ordonnée
<i>pos</i>	Position sur laquelle on veut savoir si on a cliqué dessus

Returns

int Retourne Vrai si position trouve, faux sinon

4.2.3.12 void colorChecker (SDisplay * display, SGameState * gameState, EPosition pos)

Colorie le dernier pion d'une flèche donnée.

Parameters

<i>display</i>	Structure d'affichage
<i>gameState</i>	Etat du jeu
<i>pos</i>	Position de la flèche sur lequel on veut colorier le dernier pion

4.2.3.13 SGameState * copyGameState (SGameState * gameState, EPlayer player)

Effectue une copie de l'état du jeu suivant le joueur à qui cette copie est destinée.

Parameters

<i>gameState</i>	Etat du jeu
<i>player</i>	Joueur à qui est destinée cette copie

Returns

Retourne une copie de l'état du jeu

4.2.3.14 `int * Display_Arrow_Possibilities (SDisplay * display, SGameState * gameState, EPlayer player, EPosition depart, SGame * game)`

Determine en fonction du pion d'un joueur les flèches qui sont possibles à atteindre.

Parameters

<i>display</i>	Structure d'affichage
<i>gameState</i>	Etat du jeu
<i>player</i>	Le joueur qui veut faire le déplacement
<i>depart</i>	Le numéro de la flèche où se trouve le pion à déplacer
<i>game</i>	Informations sur le jeu (noms des joueurs, couleurs des pions des joueurs,...)

Returns

Retourne un tableau contenant le numéro des flèches possibles à atteindre

4.2.3.15 `void Display_CheckerDraw (SDisplay * display, SDL_Rect position, int player, SGame * game)`

Dessine un pion d'un joueur à une position donnée.

Parameters

<i>display</i>	Structure d'affichage
<i>position</i>	Position où l'on veut dessiner le pion
<i>player</i>	Joueur possédant ce pion
<i>game</i>	Informations sur le jeu (noms des joueurs, couleurs des pions des joueurs,...)

4.2.3.16 `void Display_CheckerMove (SDisplay * display, SGameState * gameState, SMove * move, SGame * game)`

Effectue le déplacement d'un pion suivant un mouvement donné et met à jour l'état du jeu.

Parameters

<i>display</i>	Structure d'affichage
<i>gameState</i>	Etat du jeu
<i>move</i>	Mouvement à effectuer
<i>game</i>	Informations sur le jeu (noms des joueurs, couleurs des pions des joueurs,...)

4.2.3.17 void Display_Checkers (SDisplay * *display*, SGameState * *gameState*, SGame * *game*)

Affiche les pions à l'écran.

Parameters

<i>display</i>	Structure d'affichage
<i>gameState</i>	Etat du jeu
<i>game</i>	Informations sur le jeu (noms des joueurs, couleurs des pions des joueurs,...)

4.2.3.18 int Display_CheckersPossibilities (SDisplay * *display*, SGameState * *gameState*, EPlayer *player*, SGame * *game*)

Affiche à l'écran les pions qui peuvent être déplacé suivant l'état du jeu et le joueur courant.

Parameters

<i>display</i>	Structure d'affichage
<i>gameState</i>	Etat du jeu
<i>player</i>	Le joueur qui veut faire le déplacement
<i>game</i>	Informations sur le jeu (noms des joueurs, couleurs des pions des joueurs,...)

Returns

Retourne 1 si quitter / 0 sinon

4.2.3.19 void Display_Clear (SDisplay * *display*)

Réinitialise l'affichage.

Parameters

<i>display</i>	Structure d'affichage
----------------	-----------------------

4.2.3.20 void Display_Die (SDisplay * *display*, SGameState * *gameState*, SGame * *game*)

Affiche les dés à l'écran.

Parameters

<i>display</i>	Structure d'affichage
<i>gameState</i>	Etat du jeu
<i>game</i>	Informations sur le jeu (noms des joueurs, couleurs des pions des joueurs,...)

4.2.3.21 void Display_DrawBar (SDisplay * *display*, SGameState * *gameState*, SGame * *game*)

Affiche les pions présents dans les barres (prisonniers).

Parameters

<i>display</i>	Structure d'affichage
<i>gameState</i>	Etat du jeu
<i>game</i>	Informations sur le jeu (noms des joueurs, couleurs des pions des joueurs,...)

4.2.3.22 void Display_DrawOut (SDisplay * *display*, SGameState * *gameState*, SGame * *game*)

Affiche les pions présents dans les zones de sorties.

Parameters

<i>display</i>	Structure d'affichage
<i>gameState</i>	Etat du jeu
<i>game</i>	Informations sur le jeu (noms des joueurs, couleurs des pions des joueurs,...)

4.2.3.23 void Display_DrawSelectedArrow (SDisplay * *display*, EPosition *pos*)

Colorie la flèche à la position pos en bleu.

Parameters

<i>display</i>	Structure d'affichage
<i>pos</i>	Le numéro de la flèche à colorer

4.2.3.24 void Display_Exit (SDisplay * *display*)

Libère les surfaces utilisées dans la structure d'affichage et quitte la SDL.

Parameters

<i>display</i>	Structure d'affichage
----------------	-----------------------

4.2.3.25 int Display_GameActions (SDisplay * *display*, SGameState * *gameState*, SGame * *game*)

Affiche à l'écran les actions de jeu possibles (doubler la mise, lancer les dés).

Parameters

<i>display</i>	Structure d'affichage
<i>gameState</i>	Etat du jeu
<i>game</i>	Informations sur le jeu (noms des joueurs, couleurs des pions des joueurs,...)

4.2.3.26 void Display_Init (SDisplay * *display*, SGame * *game*)

Initialise la SDL, la structure d'affichage et la structure d'informations sur le jeu.

Parameters

<i>display</i>	Structure d'affichage
<i>game</i>	Informations sur le jeu (noms des joueurs, couleurs des pions des joueurs,...)

4.2.3.27 int Display_Message (SDisplay * *display*, char * *message*, SDL_Rect *position*, SDL_Color *color*, int *box*, int *clic*)

Affiche un message à l'écran.

Parameters

<i>display</i>	Structure d'affichage
<i>message</i>	Message à afficher
<i>position</i>	Position du message
<i>color</i>	Couleur du message
<i>box</i>	Présence ou non d'un cadre autour du message

Returns

Retourne 1 si quitter / 0 sinon

4.2.3.28 void Display_RefreshGameBoard (SDisplay * *display*, SGameState * *gameState*, SGame * *game*)

Raffraichit l'affichage du plateau de jeu(pions, dés et videau).

Parameters

<i>display</i>	Structure d'affichage
<i>gameState</i>	Etat du jeu
<i>game</i>	Informations sur le jeu (noms des joueurs, couleurs des pions des joueurs,...)

4.2.3.29 void Display_Score (SDisplay * *display*, SGameState * *gameState*, SGame * *game*)

Affiche les scores à l'écran.

Parameters

<i>display</i>	Structure d'affichage
<i>gameState</i>	Etat du jeu
<i>game</i>	Informations sur le jeu (noms des joueurs, couleurs des pions des joueurs,...)

4.2.3.30 void Free_SAI (SIA_Functions * *IA_struct*)

Libère une structure SIA.

Parameters

<i>IA_struct</i>	Structure à libérer
------------------	---------------------

4.2.3.31 int Game_FirstToPlay (SDisplay * *display*, EGameMode *gameMode*, SGame * *game*, SGameState * *gameState*)

Détermine quel joueur va commencer la partie par lancement des dés.

Parameters

<i>display</i>	Structure d'affichage
<i>gameMode</i>	Mode de jeu
<i>game</i>	Informations sur le jeu (noms des joueurs, couleurs des pions des joueurs,...)
<i>gameState</i>	Etat du jeu

Returns

Retourne 1 si joueur1 commence / 2 si joueur2 / autre si quitter

4.2.3.32 SGameState * Game_Init ()

Fonction qui permet d'initialiser l'état du jeu (disposition des jetons au début de la partie).

Returns

Structure d'état du jeu

4.2.3.33 void Game_LaunchDie (SGameState * *gameState*, SGame * *game*)

Lance les dés et réinitialise le tableau d'état des dés.

Parameters

<i>gameState</i>	Etat du jeu
<i>game</i>	Informations sur le jeu (noms des joueurs, couleurs des pions des joueurs,...)

4.2.3.34 int Game_Play (SDisplay * *display*, EGameMode *gameMode*, SGame * *game*, SIA_Functions * *IA_struct*)

Lance la boucle de jeu.

Parameters

<i>display</i>	Structure d'affichage
<i>gameMode</i>	Mode de jeu
<i>game</i>	Informations sur le jeu (noms des joueurs, couleurs des pions des joueurs,...)
<i>IA_struct</i>	Tableau de structures contenant les fonctions de l'AI

Returns

Retourne 1 si quitter / 0 si bon déroulement

4.2.3.35 unsigned int get_distance (EPosition *depart*, EPosition *arrivee*)

Détermine la distance entre une position de départ et une d'arrivée.

Parameters

<i>depart</i>	Position de départ
<i>arrivee</i>	Position d'arrivee

Returns

Retourne la distance entre ces deux positions

4.2.3.36 SIA_Functions * Init_SAI ()

Retourne une structure de fonctions de l'IA initialisée.

Returns

Retourne une structure de fonctions d'une IA

4.2.3.37 int inTab (EPosition p, int * tab)

Détermine si la position p est dans le tableau tab.

Parameters

<i>p</i>	Position recherchée
<i>tab</i>	Tableau dans lequel on cherche

Returns

Retourne vrai si position trouvée, faux sinon

4.2.3.38 int Load_API (char * path, SIA_Functions * IA_struct, int ind)

Charge la librairie passé en paramètre et initialise la structure IA.

Parameters

<i>path</i>	Chemin de la librairie à charger
<i>IA_struct</i>	Structure à initialiser avec la librairie
<i>ind</i>	Indice de la structure SIA

Returns

Retourne 1 si chargement effectué, 0 si erreur de chargement

4.2.3.39 int Menu_Click (int x, int y)

Renvoie le code de l'élément cliqué sur le menu.

Parameters

<i>x</i>	Abscisse de la position du clic de souris
<i>y</i>	Ordonnée de la position du clic de souris

Returns

Retourne code de l'élément cliqué, -1 si ne correspond à aucun élément du menu

4.2.3.40 `int Menu_Display (SDisplay * display, EGameMode gameMode, SGame * game)`

Affiche le menu.

Parameters

<i>display</i>	Structure d'affichage
<i>gameMode</i>	Mode de jeu
<i>game</i>	Informations sur le jeu (noms des joueurs, couleurs des pions des joueurs,...)

Returns

Retourne 0 si jouer, 1 si quitter

4.2.3.41 `void Menu_Fill (SDisplay * display, EGameMode gameMode, SGame * game)`

Remplis le menu avec les options correspondants au mode de jeu.

Parameters

<i>display</i>	Structure d'affichage
<i>gameMode</i>	Mode de jeu
<i>game</i>	Informations sur le jeu (noms des joueurs, couleurs des pions des joueurs,...)

4.2.3.42 `void Menu_Text (SDisplay * display, char * message, SDL_Rect position, SDL_Color color)`

Affiche un texte à une certaine position d'une certaine couleur.

Parameters

<i>display</i>	Structure d'affichage
<i>message</i>	Message à afficher
<i>position</i>	Position à laquelle on veut afficher le message
<i>color</i>	Couleur dans laquelle on veut afficher le message

4.2.3.43 `void Menu_TextInput (char * name, SDL_keysym key)`

Ajoute à une chaîne de caractère un caractère obtenu par l'événement SDL_KEYUP.

Parameters

<i>name</i>	Chaine de caractère à modifier
<i>key</i>	Symbole qui a été tapé au clavier

Index

Arbitrator_AuthorizedDeplacement
definitions.h, [17](#)
Arbitrator_EmptyPosition
definitions.h, [17](#)
Arbitrator_NumberOfDieCanPlay
definitions.h, [17](#)
Arbitrator_NumberOfDieForMove
definitions.h, [18](#)
Arbitrator_Pion_Depart_Autorise
definitions.h, [18](#)
Arbitrator_PlayerArrays
definitions.h, [18](#)
Arbitrator_Sens_rotation_correct
definitions.h, [19](#)
Arbitrator_Taille_Mouvement_Correcte
definitions.h, [19](#)

backgammon.h, [11](#)
pfDoubleStack, [12](#)
pfEndGame, [12](#)
pfEndMatch, [12](#)
pfMakeDecision, [12](#)
pfStartGame, [13](#)
pfTakeDouble, [13](#)

case_appartenant_joueur_adverse_avec_-
un_pion
definitions.h, [19](#)
Check_Args
definitions.h, [20](#)
CheckerWithScreenPosition
definitions.h, [20](#)
colorChecker
definitions.h, [20](#)
copyGameState
definitions.h, [20](#)

definitions.h, [13](#)
Arbitrator_AuthorizedDeplacement, [17](#)
Arbitrator_EmptyPosition, [17](#)
Arbitrator_NumberOfDieCanPlay, [17](#)
Arbitrator_NumberOfDieForMove, [18](#)
Arbitrator_Pion_Depart_Autorise, [18](#)
Arbitrator_PlayerArrays, [18](#)
Arbitrator_Sens_rotation_correct, [19](#)
Arbitrator_Taille_Mouvement_Correcte, [19](#)
case_appartenant_joueur_adverse_avec_-
un_pion, [19](#)
Check_Args, [20](#)
CheckerWithScreenPosition, [20](#)
colorChecker, [20](#)
copyGameState, [20](#)
Display_Arrow_Possibilities, [21](#)
Display_CheckerDraw, [21](#)
Display_CheckerMove, [21](#)
Display_Checkers, [22](#)
Display_CheckersPossibilities, [22](#)
Display_Clear, [22](#)
Display_Die, [22](#)
Display_DrawBar, [23](#)
Display_DrawOut, [23](#)
Display_DrawSelectedArrow, [23](#)
Display_Exit, [23](#)
Display_GameActions, [24](#)
Display_Init, [24](#)
Display_Message, [24](#)
Display_RefreshGameBoard, [24](#)
Display_Score, [25](#)
EGameMode, [17](#)
Free_SAI, [25](#)
Game_FirstToPlay, [25](#)
Game_Init, [26](#)
Game_LaunchDie, [26](#)
Game_Play, [26](#)
get_distance, [26](#)
Init_SAI, [27](#)
inTab, [27](#)
Load_API, [27](#)
Menu_Click, [27](#)
Menu_Display, [28](#)
Menu_Fill, [28](#)

- Menu_Text, [28](#)
- Menu_TextInput, [28](#)
- Display_Arrow_Possibilities
 - definitions.h, [21](#)
- Display_CheckerDraw
 - definitions.h, [21](#)
- Display_CheckerMove
 - definitions.h, [21](#)
- Display_Checkers
 - definitions.h, [22](#)
- Display_CheckersPossibilities
 - definitions.h, [22](#)
- Display_Clear
 - definitions.h, [22](#)
- Display_Die
 - definitions.h, [22](#)
- Display_DrawBar
 - definitions.h, [23](#)
- Display_DrawOut
 - definitions.h, [23](#)
- Display_DrawSelectedArrow
 - definitions.h, [23](#)
- Display_Exit
 - definitions.h, [23](#)
- Display_GameActions
 - definitions.h, [24](#)
- Display_Init
 - definitions.h, [24](#)
- Display_Message
 - definitions.h, [24](#)
- Display_RefreshGameBoard
 - definitions.h, [24](#)
- Display_Score
 - definitions.h, [25](#)
- EGameMode
 - definitions.h, [17](#)
- Free_SAI
 - definitions.h, [25](#)
- Game_FirstToPlay
 - definitions.h, [25](#)
- Game_Init
 - definitions.h, [26](#)
- Game_LaunchDie
 - definitions.h, [26](#)
- Game_Play
 - definitions.h, [26](#)
- get_distance
 - definitions.h, [26](#)
- Init_SAI
 - definitions.h, [27](#)
- inTab
 - definitions.h, [27](#)
- Load_API
 - definitions.h, [27](#)
- Menu_Click
 - definitions.h, [27](#)
- Menu_Display
 - definitions.h, [28](#)
- Menu_Fill
 - definitions.h, [28](#)
- Menu_Text
 - definitions.h, [28](#)
- Menu_TextInput
 - definitions.h, [28](#)
- pfDoubleStack
 - backgammon.h, [12](#)
- pfEndGame
 - backgammon.h, [12](#)
- pfEndMatch
 - backgammon.h, [12](#)
- pfMakeDecision
 - backgammon.h, [12](#)
- pfStartGame
 - backgammon.h, [13](#)
- pfTakeDouble
 - backgammon.h, [13](#)
- SDisplay, [5](#)
- SGame, [6](#)
- SGameState, [7](#)
- SIA_Functions, [7](#)
- SMove, [8](#)
- SZone, [8](#)