Havannah

Generated by Doxygen 1.8.13

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 arbre< T, U > Class Template Reference 5

3.1.1 Constructor & Destructor Documentation 5

3.1.1.1 arbre() [1/2] 6

3.1.1.2 arbre() [2/2] 6

3.1.1.3 ~arbre() 6

3.1.2 Member Function Documentation 6

3.1.2.1 ajouter_noeud() 6

3.1.2.2 find() [1/2] 6

3.1.2.3 find() [2/2] 7

3.1.2.4 voirPere() 7

3.1.3 Member Data Documentation 7

3.1.3.1 racine 7

3.2 jeu Class Reference 7

3.2.1 Detailed Description 8

3.2.2 Constructor & Destructor Documentation 8

3.2.2.1 ~jeu() 9

3.2.2.2 jeu() 9

3.2.3 Member Function Documentation . . . 9
3.2.3.1 a() . . . 9
3.2.3.2 check_win() . . . 9
3.2.3.3 menu() . . . 9
3.2.3.4 tour() . . . 10
3.2.4 Member Data Documentation . . . 10
3.2.4.1 hav . . . 10
3.2.4.2 nb_ia . . . 10
3.2.4.3 nb_players . . . 10
3.2.4.4 nbturn . . . 10
3.2.4.5 part_plateau . . . 10
3.2.4.6 playerids . . . 11
3.2.4.7 players . . . 11
3.3 noeud< T, U > Class Template Reference . . . 11
3.3.1 Constructor & Destructor Documentation . . . 12
3.3.1.1 noeud() [1/3] . . . 12
3.3.1.2 ∼noeud() . . . 12
3.3.1.3 noeud() [2/3] . . . 12
3.3.1.4 noeud() [3/3] . . . 12
3.3.2 Member Function Documentation . . . 12
3.3.2.1 aff() . . . 13
3.3.2.2 operator==() . . . 13
3.3.3 Member Data Documentation . . . 13
3.3.3.1 filsgauche . . . 13
3.3.3.2 freredroit . . . 13
3.3.3.3 info . . . 13
3.3.3.4 pere . . . 14
3.4 plateau Class Reference . . . 14
3.4.1 Detailed Description . . . 15
3.4.2 Member Enumeration Documentation . . . 15

3.4.2.1 endGame . . . 15
3.4.3 Constructor & Destructor Documentation . . . 16
3.4.3.1 plateau() [1/2] . . . 16
3.4.3.2 ∼plateau() . . . 16
3.4.3.3 plateau() [2/2] . . . 16
3.4.4 Member Function Documentation . . . 16
3.4.4.1 aff_bas_hexa() . . . 17
3.4.4.2 aff_bas_hexa2() . . . 17
3.4.4.3 aff_contenu() . . . 18
3.4.4.4 aff_haut_hexa() . . . 18
3.4.4.5 aff_haut_hexa2() . . . 19
3.4.4.6 afficher() . . . 19
3.4.4.7 afficher2() . . . 20
3.4.4.8 check_win() . . . 20
3.4.4.9 find_in_node_vector() . . . 20
3.4.4.10 getTaille() . . . 21
3.4.4.11 getTailleReelle() . . . 21
3.4.4.12 placerpion() . . . 21
3.4.4.13 r_der_ligne() . . . 21
3.4.4.14 retParticuliers() . . . 21
3.4.4.15 Rtaillecote() . . . 21
3.4.4.16 suivreChemin() [1/2] . . . 22
3.4.4.17 suivreChemin() [2/2] . . . 22
3.4.5 Member Data Documentation . . . 22
3.4.5.1 der_ligne . . . 22
3.4.5.2 hexa . . . 23
3.4.5.3 taillecote . . . 23
3.5 player Class Reference . . . 23
3.5.1 Detailed Description . . . 24
3.5.2 Constructor & Destructor Documentation . . . 24

3.5.2.1 player() . . . 24
3.5.2.2 ∼player() . . . 25
3.5.3 Member Function Documentation . . . 25
3.5.3.1 affcoups() . . . 25
3.5.3.2 check_win() . . . 25
3.5.3.3 est_voisin() . . . 25
3.5.3.4 newid() . . . 26
3.5.3.5 placer_pion() . . . 26
3.5.3.6 returnIa() . . . 26
3.5.3.7 returnId() . . . 26
3.5.3.8 returnNom() . . . 26
3.5.4 Member Data Documentation . . . 27
3.5.4.1 ia . . . 27
3.5.4.2 id . . . 27
3.5.4.3 nom . . . 27

4 File Documentation **29**

4.1 /Users/thibault/Google Drive/PROG/S4/Projet/h/arbre.h File Reference . . . 29
4.2 /Users/thibault/Google Drive/PROG/S4/Projet/h/jeu.h File Reference . . . 29
4.3 /Users/thibault/Google Drive/PROG/S4/Projet/h/noeud.h File Reference . . . 29
4.4 /Users/thibault/Google Drive/PROG/S4/Projet/h/plateau.h File Reference . . . 30
4.5 /Users/thibault/Google Drive/PROG/S4/Projet/h/player.h File Reference . . . 30

Index **31**

Chapter 1

Class Index

1.1 Class List

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

arbre< T, U > 5
jeu
    Classe jeu 7
noeud< T, U > 11
plateau
    Classe plateau 14
player
    Class joueur 23

Chapter 2

File Index

2.1 File List

Here is a list of all files with brief descriptions:

/Users/thibault/Google Drive/PROG/S4/Projet/h/arbre.h 29
/Users/thibault/Google Drive/PROG/S4/Projet/h/jeu.h 29
/Users/thibault/Google Drive/PROG/S4/Projet/h/noeud.h 29
/Users/thibault/Google Drive/PROG/S4/Projet/h/plateau.h 30
/Users/thibault/Google Drive/PROG/S4/Projet/h/player.h 30

Chapter 3

Class Documentation

3.1 arbre< T, U > Class Template Reference

`#include <arbre.h>`

Public Member Functions

- arbre ()

  Constructeur par défaut.
- arbre (T info, U info2)

  Constructeur à deux arguments.
- ~arbre ()
- noeud< T, U > * find (T res, U res2)

  Cherche un noeud.
- noeud< T, U > * find (T res, U res2, noeud< T, U > *c)

  Cherche un noeud.
- noeud< T, U > * ajouter_noeud (std::pair< T, U > info, noeud< T, U > *source)

  Ajoute un noeud fils à un noeud spécifié
- noeud< T, U > * voirPere (noeud< T, U > *source, bool verifCoin, bool verifFourche, bool verifPont)

  Renvoie le père pénultime de la branche contenant le noeud source.

Public Attributes

- noeud< T, U > * racine

  Les données contenues dans les noeuds. Elles seront donc du type de l'arbre demandé.

3.1.1 Constructor & Destructor Documentation

3.1.1.1 arbre() [1/2]

```
template<class T, class U>
arbre< T, U >::arbre ( )  [inline]
```

Constructeur par défaut.

Il initialise la racine à NULL.

3.1.1.2 arbre() [2/2]

```
template<class T, class U>
arbre< T, U >::arbre (
            T info,
            U info2 )  [inline]
```

Constructeur à deux arguments.

3.1.1.3 ~arbre()

```
template<class T, class U>
arbre< T, U >::~arbre ( )  [inline]
```

3.1.2 Member Function Documentation

3.1.2.1 ajouter_noeud()

```
template<class T, class U>
noeud<T,U>* arbre< T, U >::ajouter_noeud (
            std::pair< T, U > info,
            noeud< T, U > * source )  [inline]
```

Ajoute un noeud fils à un noeud spécifié

3.1.2.2 find() [1/2]

```
template<class T, class U>
noeud<T,U>* arbre< T, U >::find (
            T res,
            U res2 )  [inline]
```

Cherche un noeud.

3.1.2.3 find() [2/2]

```
template<class T, class U>
noeud<T,U>* arbre< T, U >::find (
            T res,
            U res2,
            noeud< T, U > * c )  [inline]
```

Cherche un noeud.

3.1.2.4 voirPere()

```
template<class T, class U>
noeud<T,U>* arbre< T, U >::voirPere (
            noeud< T, U > * source,
            bool verifCoin,
            bool verifFourche,
            bool verifPont )  [inline]
```

Renvoie le père pénultime de la branche contenant le noeud source.

3.1.3 Member Data Documentation

3.1.3.1 racine

```
template<class T, class U>
noeud<T,U>* arbre< T, U >::racine
```

Les données contenues dans les noeuds. Elles seront donc du type de l'arbre demandé.

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

- /Users/thibault/Google Drive/PROG/S4/Projet/h/arbre.h

3.2 jeu Class Reference

classe jeu

```
#include <jeu.h>
```

Public Member Functions

- ~jeu ()

  Destructeur du jeu.
- jeu (int tp, int nbp)

  Constructeur du jeu.
- void menu ()

  Interface pour le menu.
- void a ()

  Affichage des informations du tour.
- void tour ()

  Fonction permettant aux joueurs de faire un nouveau tour.
- bool check_win ()

  Fonction permettant de vérifier si un joueur a gagné.

Private Attributes

- plateau * hav

  Le plateau en question.
- char nb_players

  Nombre de joueurs pour le jeu.
- char nb_ia

  Nombre d'IA à gérer.
- int nbturn

  Nombre de tours qui sont passés depuis le début de la partie.
- player ** players

  Tableau de pointeurs vers tous les joueurs (classe player)
- char * playerids

  Tableau de tous les identifiants joueur.
- std::multimap< char, std::pair< int, int > > part_plateau

  Map de toutes les cases particulières du plateau.

3.2.1 Detailed Description

classe jeu

Regroupe tous les éléments du jeu : le plateau, les joueurs, et les fonctions pour les coups.

Librairies incluses :

- iostream
- player.h
- plateau.h

3.2.2 Constructor & Destructor Documentation

3.2.2.1 ∼jeu()

```
jeu::∼jeu ( )
```

Destructeur du jeu.

3.2.2.2 jeu()

```
jeu::jeu (
            int tp,
            int nbp )
```

Constructeur du jeu.

Constructeur de jeu qui est appelée depuis la fonction main.

Parameters

tp	La taille désirée du plateau.
nbp	Nombre de joueurs.

3.2.3 Member Function Documentation

3.2.3.1 a()

```
void jeu::a ( )
```

Affichage des informations du tour.

3.2.3.2 check_win()

```
bool jeu::check_win ( )
```

Fonction permettant de vérifier si un joueur a gagné.

3.2.3.3 menu()

```
void jeu::menu ( )
```

Interface pour le menu.

3.2.3.4 tour()

```
void jeu::tour ( )
```

Fonction permettant aux joueurs de faire un nouveau tour.

3.2.4 Member Data Documentation

3.2.4.1 hav

```
plateau* jeu::hav  [private]
```

Le plateau en question.

3.2.4.2 nb_ia

```
char jeu::nb_ia  [private]
```

Nombre d'IA à gérer.

3.2.4.3 nb_players

```
char jeu::nb_players  [private]
```

Nombre de joueurs pour le jeu.

3.2.4.4 nbturn

```
int jeu::nbturn  [private]
```

Nombre de tours qui sont passés depuis le début de la partie.

3.2.4.5 part_plateau

```
std::multimap<char, std::pair<int,int> > jeu::part_plateau  [private]
```

Map de toutes les cases particulières du plateau.

3.2.4.6 playerids

```
char* jeu::playerids  [private]
```

Tableau de tous les identifiants joueur.

3.2.4.7 players

```
player** jeu::players  [private]
```

Tableau de pointeurs vers tous les joueurs (classe player)

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

- /Users/thibault/Google Drive/PROG/S4/Projet/h/jeu.h

3.3 noeud< T, U > Class Template Reference

```
#include <noeud.h>
```

Public Member Functions

- noeud ()
  Constructeur sans argument.
- ~noeud ()
  Destructeur de noeud.
- noeud (T info, U info2)
  Constructeur de noeud avec données à remplir.
- noeud (T i, U j, noeud *p, noeud *fg, noeud *fd)
  Constructeur de noeud relationnel (avec les pointeurs déjà attribués)
- void aff ()
  Fonction d'affichage d'un noeud.
- bool operator== (std::pair< T, U > paireComparee)
  Surcharge d'opérateur d'égalité pour voir si une paire est égale à celle d'un noeud.

Public Attributes

- noeud * pere
  Pointeur vers le père du noeud.
- noeud * freredroit
  Pointeur vers le frère du noeud.
- noeud * filsgauche
  Pointeur vers le fils du noeud.
- std::pair< T, U > info
  Information stockée.

3.3.1 Constructor & Destructor Documentation

3.3.1.1 noeud() [1/3]

```
template<class T, class U>
noeud< T, U >::noeud ( )  [inline]
```

Constructeur sans argument.

3.3.1.2 ~noeud()

```
template<class T, class U>
noeud< T, U >::~noeud ( )  [inline]
```

Destructeur de noeud.

3.3.1.3 noeud() [2/3]

```
template<class T, class U>
noeud< T, U >::noeud (
            T info,
            U info2 )  [inline]
```

Constructeur de noeud avec données à remplir.

3.3.1.4 noeud() [3/3]

```
template<class T, class U>
noeud< T, U >::noeud (
            T i,
            U j,
            noeud< T, U > * p,
            noeud< T, U > * fg,
            noeud< T, U > * fd )  [inline]
```

Constructeur de noeud relationnel (avec les pointeurs déjà attribués)

3.3.2 Member Function Documentation

3.3.2.1 aff()

```
template<class T, class U>
void noeud< T, U >::aff ( )  [inline]
```

Fonction d'affichage d'un noeud.

3.3.2.2 operator==()

```
template<class T, class U>
bool noeud< T, U >::operator== (
            std::pair< T, U > paireComparee )  [inline]
```

Surcharge d'opérateur d'égalité pour voir si une paire est égale à celle d'un noeud.

3.3.3 Member Data Documentation

3.3.3.1 filsgauche

```
template<class T, class U>
noeud* noeud< T, U >::filsgauche
```

Pointeur vers le fils du noeud.

3.3.3.2 freredroit

```
template<class T, class U>
noeud* noeud< T, U >::freredroit
```

Pointeur vers le frère du noeud.

3.3.3.3 info

```
template<class T, class U>
std::pair<T, U> noeud< T, U >::info
```

Information stockée.

3.3.3.4 pere

```
template<class T, class U>
noeud* noeud< T, U >::pere
```

Pointeur vers le père du noeud.

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

- /Users/thibault/Google Drive/PROG/S4/Projet/h/noeud.h

3.4 plateau Class Reference

classe plateau

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

Public Member Functions

- plateau ()
  Constructeur par défaut.
- ∼plateau ()
  Destructeur du plateau.
- plateau (int n)
  Constructeur avec arguments.
- void afficher (char ∗p)
  Affichage plateau avec des hexagones.
- void afficher2 (char ∗p)
  affichage du plateau avec des losanges
- void aff_contenu (int cont, int nb, char ∗p)
  Affiche contenu d'une case "n" fois.
- int r_der_ligne ()
  Donne la valeur de la dernière ligne.
- int Rtaillecote ()
  Donne la valeur de la taille demandée par le joueur.
- void aff_bas_hexa (int a, bool e_l, int sep, bool p_l, char ∗p)
  Affichage bas hexa normaux.
- void aff_haut_hexa (int a, int sep, bool e_l, bool p_l, bool l_l, char ∗p)
  Affichage bas hexa normaux.
- void aff_bas_hexa2 (int a, bool e_l, int sep, bool p_l, char ∗p)
  Affichage bas hexa en losange.
- void aff_haut_hexa2 (int a, int sep, bool e_l, bool p_l, bool l_l, char ∗p)
  Affichage haut hexa en losange.
- std::multimap< char, std::pair< int, int > > retParticuliers ()
  Retourne les coordonnées des pions aux coins et aux côtés.
- std::pair< int, int > placerpion (std::pair< std::pair< int, int >, char >)
  Permet de placer un pion (= affecter une valeur à une case)
- int getTaille ()
- int getTailleReelle ()

- bool check_win (player p)

  Fonction permettant de savoir si un joueur 'p' a gagné
- endGame suivreChemin (int i, int j, arbre< int, int > &cheminJoueurs, vector< noeud< int, int > *> &index↩ CheminJoueur)

  Fonction permettant de suivre le chemin qu'un joueur a créé sur le plateau.
- endGame suivreChemin (int i, int j, arbre< int, int > &cheminJoueurs, vector< noeud< int, int > *> &index↩ CheminJoueur, noeud< int, int > *nouedActuel)

  Suit le chemin créé par les pions voisins utilisateur.
- bool find_in_node_vector (vector< noeud< int, int > *> sourceVector, std::pair< int, int > information)

Private Types

- enum endGame { NONE, FORK, RING, BRIDGE }

  Différentes possibilités de finir le jeu.

Private Attributes

- int taillecote

  Taille d'un côté du plateau.
- int der_ligne

  Indice de la dernière ligne.
- vector< int > * hexa

  La structure de données pour le jeu.

3.4.1 Detailed Description

classe plateau

La classe contenant tout le plateau. Interagit fortement avec la classe jeu, et à moindre envergure avec la classe player.

Librairies incluses :

- iostream
- vector
- map

3.4.2 Member Enumeration Documentation

3.4.2.1 endGame

```
enum plateau::endGame  [private]
```

Différentes possibilités de finir le jeu.

Enumerator

NONE	Le joueur n'a pas gagné la partie.
FORK	Le joueur a fait une fourche entre 3 côtés.
RING	Le joueur a fait un cercle avec au moins un trou au mileu.
BRIDGE	Le joueur a fait un pont entre 2 coins.

3.4.3 Constructor & Destructor Documentation

3.4.3.1 plateau() [1/2]

```
plateau::plateau ( )
```

Constructeur par défaut.

Demande une taille de plateau au joueur et ensuite crée un plateau de la taille demandée, avec des cases contenant la valeur minimale des entiers dans C++ (-2147483648) pour signifier que la case est vide.

3.4.3.2 ∼plateau()

```
plateau::∼plateau ( )
```

Destructeur du plateau.

3.4.3.3 plateau() [2/2]

```
plateau::plateau (
            int n )
```

Constructeur avec arguments.

Permet à la classe jeu de créer un plateau sans pour autant demander au joueur quoi que ce soit. La taille passée en argument sera donc déterminée par la classe jeu.

Parameters

n	taille demandée pour le plateau.

3.4.4 Member Function Documentation

3.4.4.1 aff_bas_hexa()

```
void plateau::aff_bas_hexa (
            int a,
            bool e_l,
            int sep,
            bool p_l,
            char * p )
```

Affichage bas hexa normaux.

Affiche le bas des hexagones.

Parameters

a	indice de la ligne de l'hexa a afficher
e_l	est-ce que l'on est entre 2 lignes ? si oui, afficher des underscore
sep	le nombre d'espaces à mettre avant chaque ligne (x4)
p_l	sert à déterminer si on est sur la première ligne de la grille. ce booléen permet d'enlever un des appels de la fonction et de rajouter une ligne pour rien
p	les identifiants joueurs

3.4.4.2 aff_bas_hexa2()

```
void plateau::aff_bas_hexa2 (
            int a,
            bool e_l,
            int sep,
            bool p_l,
            char * p )
```

Affichage bas hexa en losange.

Affiche le bas des hexagones.

Parameters

a	indice de la ligne de l'hexa a afficher
e_l	est-ce que l'on est entre 2 lignes ? si oui, afficher des underscore
sep	le nombre d'espaces à mettre avant chaque ligne (x4)
p_l	sert à déterminer si on est sur la première ligne de la grille. ce booléen permet d'enlever un des appels de la fonction et de rajouter une ligne pour rien
p	les identifiants joueurs

3.4.4.3 aff_contenu()

```
void plateau::aff_contenu (
            int cont,
            int nb,
            char * p )
```

Affiche contenu d'une case "n" fois.

Permet d'afficher le contenu (l'identifiant qui est contenu dans la case) un nombre de fois donné par la fonction aff_bas_hexa(), aff_bas_hexa2(), aff_haut_hexa(), et aff_haut_hexa2().

Parameters

cont	Le contenu de la case en question
nb	Le nombre de fois qu'on souhaite afficher les caractères
p	Les identifiants joueur pour savoir quel caractère afficher

See also

aff_bas_hexa(), aff_bas_hexa2(), aff_haut_hexa() and aff_haut_hexa2()

3.4.4.4 aff_haut_hexa()

```
void plateau::aff_haut_hexa (
            int a,
            int sep,
            bool e_l,
            bool p_l,
            bool l_l,
            char * p )
```

Affichage bas hexa normaux.

Haut de l'hexagone :

```
  ____       = Ligne 1 (necessaire seulement pour le haut de la grille)
 /    \      = Ligne 2
/      \     = Ligne 3
```

Parameters

a	indice de la ligne de l'hexa a afficher
sep	le nombre d'espaces à mettre avant chaque ligne (x6)
e_l	est-ce que l'on est entre 2 lignes ? si oui, afficher des underscore
p_l	sert à déterminer si on est sur la première ligne de la grille. ce booléen permet d'enlever un des appels de la fonction et de rajouter une ligne pour rien
l_l	determine si on est en derniere ligne
p	donne les player id

See also

afficher()

3.4.4.5 aff_haut_hexa2()

```
void plateau::aff_haut_hexa2 (
            int a,
            int sep,
            bool e_l,
            bool p_l,
            bool l_l,
            char * p )
```

Affichage haut hexa en losange.

Haut de l'hexagone :

```
        = Ligne 1 (necessaire seulement pour le haut de la grille)
 /\     = Ligne 2
/  \    = Ligne 3
```

Parameters

a	indice de la ligne de l'hexa a afficher
e_l	est-ce que l'on est entre 2 lignes ? si oui, afficher des underscore
sep	le nombre d'espaces à mettre avant chaque ligne (x6)
p_l	sert à déterminer si on est sur la première ligne de la grille. ce booléen permet d'enlever un des appels de la fonction et de rajouter une ligne pour rien
l_l	determine si on est en derniere ligne
p	donne les player id

3.4.4.6 afficher()

```
void plateau::afficher (
            char * p )
```

Affichage plateau avec des hexagones.

Cette visualisation du plateau est une visualisation en hexagone, donnant une vision facile de l'état du plateau. En revanche, celle-ci est assez compressée en hauteur. Cette méthode contient 2 étapes :

1. On va faire la première partie de l'affichage des lignes, avant que l'on atteigne la ligne du milieu de la grille (qui est, par ailleurs, la plus longue). Le haut des haxagones sera donc affiché avec le fonction aff_haut_hexa()
2. Pareil que la partie d'avant, mais pour le bas du plateau, avec la fonction aff_bas_hexa()

Parameters

p	Les identifiants joueur pour que l'on puisse afficher leur cases avec les identifiants choisis auparavant.

See also

aff_haut_hexa() and aff_bas_hexa()

3.4.4.7 afficher2()

```
void plateau::afficher2 (
            char * p )
```

affichage du plateau avec des losanges

Les losanges représentent les hexagones. Cette visualisation permet de voir le plateau de manière plus compacte. Cette méthode contient 2 étapes :

1. On va faire la première partie de l'affichage des lignes, avant que l'on atteigne la ligne du milieu de la grille (qui est, par ailleurs, la plus longue). Le haut des haxagones sera donc affiché avec le fonction aff_haut_hexa2()
2. Pareil que la partie d'avant, mais pour le bas du plateau, avec la fonction aff_bas_hexa2()

Parameters

p	Les identifiants joueur pour que l'on puisse afficher leur cases avec les identifiants choisis auparavant.

See also

aff_haut_hexa2() and aff_bas_hexa2()

3.4.4.8 check_win()

```
bool plateau::check_win (
            player p )
```

Fonction permettant de savoir si un joueur 'p' a gagné

3.4.4.9 find_in_node_vector()

```
bool plateau::find_in_node_vector (
            vector< noeud< int, int > *> sourceVector,
            std::pair< int, int > information )
```

3.4.4.10 getTaille()

```
int plateau::getTaille ( )  [inline]
```

3.4.4.11 getTailleReelle()

```
int plateau::getTailleReelle ( )  [inline]
```

3.4.4.12 placerpion()

```
std::pair<int,int> plateau::placerpion (
            std::pair< std::pair< int, int >, char >  )
```

Permet de placer un pion (= affecter une valeur à une case)

3.4.4.13 r_der_ligne()

```
int plateau::r_der_ligne ( )  [inline]
```

Donne la valeur de la dernière ligne.

3.4.4.14 retParticuliers()

```
std::multimap<char, std::pair<int,int> > plateau::retParticuliers ( )
```

Retourne les coordonnées des pions aux coins et aux côtés.

Permet de savoir quels cases sont aux coins pour pouvoir vérifier si le joueur

3.4.4.15 Rtaillecote()

```
int plateau::Rtaillecote ( )  [inline]
```

Donne la valeur de la taille demandée par le joueur.

3.4.4.16 suivreChemin() [1/2]

```
endGame plateau::suivreChemin (
            int i,
            int j,
            arbre< int, int > & cheminJoueurs,
            vector< noeud< int, int > *> & indexCheminJoueur )
```

Fonction permettant de suivre le chemin qu'un joueur a créé sur le plateau.

Appelle suivreChemin(int i, int j, arbre<int,int>& cheminJoueurs, vector<noeud<int,int>*>& indexCheminJoueur, noeud<int,int>* nouedActuel) avec un noeud créé en racine de l'arbre

3.4.4.17 suivreChemin() [2/2]

```
endGame plateau::suivreChemin (
            int i,
            int j,
            arbre< int, int > & cheminJoueurs,
            vector< noeud< int, int > *> & indexCheminJoueur,
            noeud< int, int > * nouedActuel )
```

Suit le chemin créé par les pions voisins utilisateur.

- Regarde tous les vosins possibles selon la position dans le plateau
- Si un voisin est trouvé, on le cherche dans le vecteur des noeuds de l'arbre :
  - Il n'est pas trouvé, donc on l'ajoute et on continue la recherche a partir de ce point là (on met le booléen node_found à 1)
  - Si il est trouvé, on continue l'analyse
- Si le booléen node_found est à 1, alors est pas en fin de chemin, donc on ne fait rien
- Sinon, on regarde si le joueur a fait un cercle, une fourche ou un pont.

3.4.5 Member Data Documentation

3.4.5.1 der_ligne

```
int plateau::der_ligne  [private]
```

Indice de la dernière ligne.

3.4.5.2 hexa

```
vector<int>* plateau::hexa  [private]
```

La structure de données pour le jeu.

3.4.5.3 taillecote

```
int plateau::taillecote  [private]
```

Taille d'un côté du plateau.

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

- /Users/thibault/Google Drive/PROG/S4/Projet/h/plateau.h

3.5 player Class Reference

class joueur

```
#include <player.h>
```

Public Member Functions

- player ()
  Constructeur vide.
- ~player ()
  Destructeur par défaut.
- char returnId ()
  Retour de l'identifiant joueur.
- bool returnIa ()
  Retour de l'etat du joueur (humain ou non).
- void newid (char t)
  Réaffectation de l'identifiant joueur.
- std::string returnNom ()
  Retour du nom du joueur.
- std::pair< std::pair< int, int >, char > placer_pion ()
  Demande du choix joueur pour son tour.
- bool est_voisin (std::pair< int, int > point_orig, std::pair< int, int > point_nouv, int t)
  Vérifie le voisinage d'une case.
- bool check_win ()
  Vérification de la victoire d'un joueur.
- void affcoups ()
  Affichage des coups joueur.

Private Attributes

- std::string nom

  Stocke le nom choisi par le joueur.
- bool ia

  Sert à savoir si un joueur est une IA ou pas.
- char id

  Identificateur qui sert à différencier les joueur entre eux.

3.5.1 Detailed Description

class joueur

La classe player sert à gérer toutes les infos du joueur lors du jeu. Permet de créer, détruire, stocker les coups, jouer, afficher l'état du joueur. Librairies incluses :

- iostream
- vector
- list

3.5.2 Constructor & Destructor Documentation

3.5.2.1 player()

```
player::player ( )
```

Constructeur vide.

Il sert à créer un joueur en demandant un nom, un identifiant et en demandant si le joueur sera une IA ou pas.

Fonctionnement :

1. On informe le joueur qu'on va créer son personnage dans le jeu
2. On lui demande un nom (via un getline()):
   - Tant qu'il ne fournit pas de nom, on redemande la saisie.
   - On affecte ensuite le nom choisi à la variable nom.
3. On demande si il souhaite que ce personnage soit une IA (via un getline() suivi d'un tolower() pour normaliser l'entrée utilisateur):
   - Si oui, on affecte le booléen ia à 1
4. On lui demande un identifiant entre 1 et 10 :
   - Si il rentre un identifiant supèrieur, on ne gère pas
   - Si l'entrée utilisateur est non comprise, on affecte la valeur 10 par défaut

3.5.2.2 ∼player()

```
player::∼player ( )
```

Destructeur par défaut.

3.5.3 Member Function Documentation

3.5.3.1 affcoups()

```
void player::affcoups ( )
```

Affichage des coups joueur.

3.5.3.2 check_win()

```
bool player::check_win ( )
```

Vérification de la victoire d'un joueur.

3.5.3.3 est_voisin()

```
bool player::est_voisin (
            std::pair< int, int > point_orig,
            std::pair< int, int > point_nouv,
            int t )
```

Vérifie le voisinage d'une case.

Prends en argument 2 coordonnées x et y ainsi que la taille du plateau. Ensuite, vérifie si ceux ci sont voisins ou non selon leur place dans le plateau.

Parameters

point_orig	Le point de référence à partir duquel on vérifie le voisinage.
point_nouv	Le point dont on veut vérifier la position.
t	Taille du plateau (pour déterminer la forme du voisinage dans le tableau).

Returns

Un booléen permettant de savoir si les points sont voisins.

3.5.3.4 newid()

```
void player::newid (
            char t )  [inline]
```

Réaffectation de l'identifiant joueur.

3.5.3.5 placer_pion()

```
std::pair<std::pair<int,int>,char> player::placer_pion ( )
```

Demande du choix joueur pour son tour.

Demande au joueur sur quelle case il souhaite placer le pion, sans se soucier de la taille du plateau.

Returns

Une paire composée d'un identifiant, et de coordonnées x et y pour le placement sur le plateau.

See also

ajouter_coup()

3.5.3.6 returnIa()

```
bool player::returnIa ( )  [inline]
```

Retour de l'etat du joueur (humain ou non).

3.5.3.7 returnId()

```
char player::returnId ( )  [inline]
```

Retour de l'identifiant joueur.

3.5.3.8 returnNom()

```
std::string player::returnNom ( )  [inline]
```

Retour du nom du joueur.

3.5.4 Member Data Documentation

3.5.4.1 ia

```
bool player::ia  [private]
```

Sert à savoir si un joueur est une IA ou pas.

3.5.4.2 id

```
char player::id  [private]
```

Identificateur qui sert à différencier les joueur entre eux.

3.5.4.3 nom

```
std::string player::nom  [private]
```

Stocke le nom choisi par le joueur.

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

- /Users/thibault/Google Drive/PROG/S4/Projet/h/player.h

Chapter 4

File Documentation

4.1 /Users/thibault/Google Drive/PROG/S4/Projet/h/arbre.h File Reference

```
#include "../h/noeud.h"
#include <vector>
```

Classes

- class arbre< T, U >

4.2 /Users/thibault/Google Drive/PROG/S4/Projet/h/jeu.h File Reference

```
#include <iostream>
#include "../h/plateau.h"
#include "../h/player.h"
```

Classes

- class jeu

  classe jeu

4.3 /Users/thibault/Google Drive/PROG/S4/Projet/h/noeud.h File Reference

Classes

- class noeud< T, U >

4.4 /Users/thibault/Google Drive/PROG/S4/Projet/h/plateau.h File Reference

```
#include <iostream>
#include <vector>
#include <map>
#include "./player.h"
#include "../h/arbre.h"
#include "../h/noeud.h"
```

Classes

- class plateau

  classe plateau

4.5 /Users/thibault/Google Drive/PROG/S4/Projet/h/player.h File Reference

```
#include <iostream>
#include <vector>
#include <list>
#include "./arbre.h"
#include "./noeud.h"
```

Classes

- class player

  class joueur

Index

/Users/thibault/Google Drive/PROG/S4/Projet/h/arbre.h, 29
/Users/thibault/Google Drive/PROG/S4/Projet/h/jeu.h, 29
/Users/thibault/Google Drive/PROG/S4/Projet/h/noeud.h, 29
/Users/thibault/Google Drive/PROG/S4/Projet/h/plateau.h, 30
/Users/thibault/Google Drive/PROG/S4/Projet/h/player.h, 30
~arbre
    arbre, 6
~jeu
    jeu, 8
~noeud
    noeud, 12
~plateau
    plateau, 16
~player
    player, 24

a
    jeu, 9
aff
    noeud, 12
aff_bas_hexa
    plateau, 16
aff_bas_hexa2
    plateau, 17
aff_contenu
    plateau, 17
aff_haut_hexa
    plateau, 18
aff_haut_hexa2
    plateau, 19
affcoups
    player, 25
afficher
    plateau, 19
afficher2
    plateau, 20
ajouter_noeud
    arbre, 6
arbre
    ~arbre, 6
    ajouter_noeud, 6
    arbre, 5, 6
    find, 6
    racine, 7
    voirPere, 7
arbre< T, U >, 5

check_win
    jeu, 9
    plateau, 20
    player, 25

der_ligne
    plateau, 22

endGame
    plateau, 15
est_voisin
    player, 25

filsgauche
    noeud, 13
find
    arbre, 6
find_in_node_vector
    plateau, 20
freredroit
    noeud, 13

getTaille
    plateau, 20
getTailleReelle
    plateau, 21

hav
    jeu, 10
hexa
    plateau, 22

ia
    player, 27
id
    player, 27
info
    noeud, 13

jeu, 7
    ~jeu, 8
    a, 9
    check_win, 9
    hav, 10
    jeu, 9
    menu, 9
    nb_ia, 10
    nb_players, 10
    nbturn, 10

part_plateau, 10
playerids, 10
players, 11
tour, 9

menu
jeu, 9

nb_ia
jeu, 10
nb_players
jeu, 10
nbturn
jeu, 10
newid
player, 25
noeud
∼noeud, 12
aff, 12
filsgauche, 13
freredroit, 13
info, 13
noeud, 12
operator==, 13
pere, 13
noeud< T, U >, 11
nom
player, 27

operator==
noeud, 13

part_plateau
jeu, 10
pere
noeud, 13
placer_pion
player, 26
placerpion
plateau, 21
plateau, 14
∼plateau, 16
aff_bas_hexa, 16
aff_bas_hexa2, 17
aff_contenu, 17
aff_haut_hexa, 18
aff_haut_hexa2, 19
afficher, 19
afficher2, 20
check_win, 20
der_ligne, 22
endGame, 15
find_in_node_vector, 20
getTaille, 20
getTailleReelle, 21
hexa, 22
placerpion, 21
plateau, 16
r_der_ligne, 21
retParticuliers, 21
Rtaillecote, 21
suivreChemin, 21, 22
taillecote, 23
player, 23
∼player, 24
affcoups, 25
check_win, 25
est_voisin, 25
ia, 27
id, 27
newid, 25
nom, 27
placer_pion, 26
player, 24
returnIa, 26
returnId, 26
returnNom, 26
playerids
jeu, 10
players
jeu, 11

r_der_ligne
plateau, 21
racine
arbre, 7
retParticuliers
plateau, 21
returnIa
player, 26
returnId
player, 26
returnNom
player, 26
Rtaillecote
plateau, 21

suivreChemin
plateau, 21, 22

taillecote
plateau, 23
tour
jeu, 9

voirPere
arbre, 7