Projet POA

Généré par Doxygen 1.8.13

Table des matières

Chapitre 1

Index hiérarchique

1.1 Hiérarchie des classes

Cette liste d'héritage est classée approximativement par ordre alphabétique :

Chapitre 2

Index des classes

2.1 Liste des classes

Liste des classes, structures, unions et interfaces avec une brève description :

Chapitre 3

Index des fichiers

3.1 Liste des fichiers

Liste de tous les fichiers avec une brève description :

Chapitre 4

Documentation des classes

4.1 Référence de la structure Box

`#include <Environnement.h>`

Attributs publics

- int _x
- int _y
- int _ntex

4.1.1 Documentation des données membres

4.1.1.1 _ntex

`int Box::_ntex`

4.1.1.2 _x

`int Box::_x`

4.1.1.3 _y

`int Box::_y`

La documentation de cette structure a été générée à partir du fichier suivant :

- Environnement.h

4.2 Référence de la classe Chasseur

`#include <Chasseur.h>`

Graphe d'héritage de Chasseur :

Graphe de collaboration de Chasseur :

Fonctions membres publiques

- Chasseur (Labyrinthe *l)
  constructeur de chasseur, en prenant en param un pointeur de Labyrinthe
- void hurt ()
  fonction indiquant au chasseur qu'il a été touché
- bool move (double dx, double dy)
  fonction de mouvement du chasseur
- void update (void)
- bool process_fireball (float dx, float dy)
  fonction permettant le déplacement des boules de feu
- void fire (int angle_vertical)
  fonction de tir. Crée une boule de feu et initie son mouvement
- void right_click (bool shift, bool control)

Attributs publics statiques

- static Sound * _hunter_fire
  bruit de l'arme du chasseur.
- static Sound * _hunter_hit
  cri du chasseur touché.
- static Sound * _wall_hit
  on a tapé un mur.

Fonctions membres privées

- bool move_aux (double dx, double dy)
  accepte ou non un déplacement
- void die ()
  fonction indiquant au chasseur qu'il est mort

Attributs privés

- int _pv = 10
  nombre de points de vie du chasseur
- Labyrinthe * l
  Labyrinthe correspondant l'environnement.
- int perte_precision = 0
  perte de précision du chasseur

Membres hérités additionnels

4.2.1 Documentation des constructeurs et destructeur

4.2.1.1 Chasseur()

```
Chasseur::Chasseur (
            Labyrinthe * l )
```

constructeur de chasseur, en prenant en param un pointeur de Labyrinthe

4.2.2 Documentation des fonctions membres

4.2.2.1 die()

```
void Chasseur::die ( )  [private]
```

fonction indiquant au chasseur qu'il est mort

Va simplement entraner la fin de la partie

4.2.2.2 fire()

```
void Chasseur::fire (
            int angle_vertical )  [virtual]
```

fonction de tir. Crée une boule de feu et initie son mouvement

Paramètres

angle_vertical	l'angle vertical

Implémente Mover.

4.2.2.3 hurt()

```
void Chasseur::hurt ( )
```

fonction indiquant au chasseur qu'il a été touché

Va baisser son nombre de points de vie et sa précision

4.2.2.4 move()

```
bool Chasseur::move (
            double dx,
            double dy )  [inline], [virtual]
```

fonction de mouvement du chasseur

Implémente Mover.

4.2.2.5 move_aux()

```
bool Chasseur::move_aux (
            double dx,
            double dy )  [private]
```

accepte ou non un déplacement

4.2.2.6 process_fireball()

```
bool Chasseur::process_fireball (
            float dx,
            float dy )  [virtual]
```

fonction permettant le déplacement des boules de feu

Paramètres

dx	le déplacement de la boule de feu sur l'axe x
dy	le déplacement de la boule de feu sur l'axe y

Renvoie

true si la boule de feu n'a pas explosé, false si la boule de feu a explosé (par exemple la rencontre d'un mur)

Implémente Mover.

4.2.2.7 right_click()

```
void Chasseur::right_click (
            bool shift,
            bool control )  [virtual]
```

Réimplémentée à partir de Mover.

4.2.2.8 update()

```
void Chasseur::update (
            void  )  [inline], [virtual]
```

Implémente Mover.

4.2.3 Documentation des données membres

4.2.3.1 _hunter_fire

```
Sound * Chasseur::_hunter_fire  [static]
```

bruit de l'arme du chasseur.

4.2.3.2 _hunter_hit

```
Sound * Chasseur::_hunter_hit  [static]
```

cri du chasseur touché.

4.2.3.3 _pv

```
int Chasseur::_pv = 10  [private]
```

nombre de points de vie du chasseur

4.2.3.4 _wall_hit

```
Sound * Chasseur::_wall_hit  [static]
```

on a tapé un mur.

4.2.3.5 l

```
Labyrinthe* Chasseur::l  [private]
```

Labyrinthe correspondant l'environnement.

4.2.3.6 perte_precision

```
int Chasseur::perte_precision = 0  [private]
```

perte de précision du chasseur

La documentation de cette classe a été générée à partir des fichiers suivants :
- Chasseur.h
- Chasseur.cc
- Labyrinthe.cc

4.3 Référence de la structure coord

Structure représentant une coordonnée du labyrinthe.

```
#include <Labyrinthe.h>
```

Attributs publics

- int x
- int y

4.3.1 Description détaillée

Structure représentant une coordonnée du labyrinthe.

4.3.2 Documentation des données membres

4.3.2.1 x

```
int coord::x
```

4.3.2.2 y

```
int coord::y
```

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

4.4 Référence de la classe Environnement

`#include <Environnement.h>`

Graphe d'héritage de Environnement :

Graphe de collaboration de Environnement :

Fonctions membres publiques

- virtual int width ()=0
- virtual int height ()=0
- virtual char data (int i, int j)=0
- virtual ~Environnement ()
- void reconfigure ()
- int wall_texture (char *)
- void make_fireballs (void)

Fonctions membres publiques statiques

- static Environnement * init (char *filename)

Attributs publics

- Wall * _walls
- int _nwall
- Wall * _picts
- int _npicts
- Box * _boxes
- int _nboxes
- Box _treasor
- Mover ** _guards
- int _nguards

Attributs publics statiques

- static const int scale
- static const char * texture_dir
- static const char * modele_dir

4.4.1 Documentation des constructeurs et destructeur

4.4.1.1 ~Environnement()

`virtual Environnement::~Environnement ( )  [inline], [virtual]`

4.4.2 Documentation des fonctions membres

4.4.2.1 data()

```
virtual char Environnement::data (
            int i,
            int j )  [pure virtual]
```

Implémenté dans Labyrinthe.

4.4.2.2 height()

```
virtual int Environnement::height ( )  [pure virtual]
```

Implémenté dans Labyrinthe.

4.4.2.3 init()

```
Environnement * Environnement::init (
            char * filename )  [static]
```

4.4.2.4 make_fireballs()

```
void Environnement::make_fireballs (
            void  )
```

4.4.2.5 reconfigure()

```
void Environnement::reconfigure ( )
```

4.4.2.6 wall_texture()

```
int Environnement::wall_texture (
            char *  )
```

4.4.2.7 width()

```
virtual int Environnement::width ( )  [pure virtual]
```

Implémenté dans Labyrinthe.

4.4.3 Documentation des données membres

4.4.3.1 _boxes

```
Box* Environnement::_boxes
```

4.4.3.2 _guards

```
Mover** Environnement::_guards
```

4.4.3.3 _nboxes

```
int Environnement::_nboxes
```

4.4.3.4 _nguards

```
int Environnement::_nguards
```

4.4.3.5 _npicts

```
int Environnement::_npicts
```

4.4.3.6 _nwall

```
int Environnement::_nwall
```

4.4.3.7 _picts

```
Wall* Environnement::_picts
```

4.4.3.8 _treasor

```
Box Environnement::_treasor
```

4.4.3.9 _walls

```
Wall* Environnement::_walls
```

4.4.3.10 modele_dir

```
const char* Environnement::modele_dir  [static]
```

4.4.3.11 scale

```
const int Environnement::scale  [static]
```

4.4.3.12 texture_dir

```
const char* Environnement::texture_dir  [static]
```

La documentation de cette classe a été générée à partir des fichiers suivants :
- Environnement.h
- Labyrinthe.cc

4.5 Référence de la classe FireBall

```
#include <FireBall.h>
```

Graphe de collaboration de FireBall :

Fonctions membres publiques

- FireBall (float size, unsigned int tex, Mover ∗)
- void display ()
- void move_step ()
- void init (float x, float y, float z, int angle_vertical, int angle_horizontal)
- void set_orig_size (float size)
- float get_x ()
- float get_y ()

Fonctions membres privées

- void explode ()

Attributs privés

- float _x
- float _y
- float _z
- float _size
- float _orig_size
- int _angle
- int _angle2
- unsigned int _fire_texture
- int _move_angle
- int _azimuth
- Mover ∗ _owner
- FBstat _state

Attributs privés statiques

- static unsigned int _fire_list

4.5.1 Documentation des constructeurs et destructeur

4.5.1.1 FireBall()

```
FireBall::FireBall (
            float size,
            unsigned int tex,
            Mover *  )
```

4.5.2 Documentation des fonctions membres

4.5.2.1 display()

```
void FireBall::display ( )
```

4.5.2.2 explode()

```
void FireBall::explode ( )  [private]
```

4.5.2.3 get_x()

```
float FireBall::get_x ( )  [inline]
```

4.5.2.4 get_y()

```
float FireBall::get_y ( )  [inline]
```

4.5.2.5 init()

```
void FireBall::init (
            float x,
            float y,
            float z,
            int angle_vertical,
            int angle_horizontal )
```

4.5.2.6 move_step()

```
void FireBall::move_step ( )
```

4.5.2.7 set_orig_size()

```
void FireBall::set_orig_size (
            float size )  [inline]
```

4.5.3 Documentation des données membres

4.5.3.1 _angle

```
int FireBall::_angle  [private]
```

4.5.3.2 _angle2

```
int FireBall::_angle2  [private]
```

4.5.3.3 _azimuth

```
int FireBall::_azimuth  [private]
```

4.5.3.4 _fire_list

```
unsigned int FireBall::_fire_list  [static], [private]
```

4.5.3.5 _fire_texture

```
unsigned int FireBall::_fire_texture  [private]
```

4.5.3.6 _move_angle

```
int FireBall::_move_angle  [private]
```

4.5.3.7 _orig_size

```
float FireBall::_orig_size  [private]
```

4.5.3.8 _owner

```
Mover* FireBall::_owner  [private]
```

4.5.3.9 _size

```
float FireBall::_size  [private]
```

4.5.3.10 _state

```
FBstat FireBall::_state  [private]
```

4.5.3.11 _x

```
float FireBall::_x  [private]
```

4.5.3.12 _y

```
float FireBall::_y  [private]
```

4.5.3.13 _z

```
float FireBall::_z  [private]
```

La documentation de cette classe a été générée à partir du fichier suivant :

— FireBall.h

4.6 Référence de la classe Gardien

Classe implémentant les gardes.

```
#include <Gardien.h>
```

Graphe d'héritage de Gardien :

Graphe de collaboration de Gardien :

Fonctions membres publiques

- Gardien (Labyrinthe *laby, const char *modele)
- ~Gardien ()
- void update (void)
- void hurt ()
  fonction indiquant au gardien qu'il a été touché par une boule de feu provenant du joueur. Baissera sa précision et son nombre de points de vie
- bool move (double dx, double dy)
  Tente d'accomplir le mouvement en paramtre. Meme principe que try_move, mais va faire en sorte de glisser sur les murs.
- void fire (int angle_vertical)
  initialisera une boule de feu, qui sera envoyée
- bool process_fireball (float dx, float dy)
  Gre le déplacement de la boule de feu venant du garde.

Fonctions membres privées

- bool can_see_player ()
  indique si le gardien peut voir le joueur (ie s'il n'y a ni mur ni obstacle entre les deux).
- bool is_legit_move (double dx, double dy)
  teste si le mouvement de coordonnées (dx, dy) est acceptable, id est que a implique pas de marcher au travers d'un mur
- bool try_move (double dx, double dy)
  tente d'accomplir le mouvement en paramtre
- bool avancer ()
  fait marcher le gardien vers l'avant
- bool move_to_treasure ()
  fait en sorte que le gardien aille vers le trésor
- void die ()
  fonction appellée lors de la mort du gardien

Attributs privés

- bool fired = false
  Indique si une boule de feu a déj été tirée. Si c'est le cas, il est impossible pour le gardien d'en tirer une autre.
- bool reloading = false
  indique si le gardien est en train de recharger (si c'est le cas, il ne peut pas tirer)
- int reload = 0
  indique le statut du rechargement
- int perte_precision = 5
  indique la perte de précision, en degrés, du gardien
- bool dead = false
  indique si le gardien est mort
- Labyrinthe * l
  pointeur vers le labyrinthe
- int _pv = BASE_PV
  Nombre de PV restants au gardien.

Attributs privés statiques

- static const int RELOAD_TIME = 100
  Temps de rechargement pour les gardiens (en nombre d'appels de update())
- static const int BASE_PV = 10
  Nombre de PV de base du gardien.

Membres hérités additionnels

4.6.1 Description détaillée

Classe implémentant les gardes.

4.6.2 Documentation des constructeurs et destructeur

4.6.2.1 Gardien()

```
Gardien::Gardien (
            Labyrinthe * laby,
            const char * modele )  [inline]
```

4.6.2.2 ~Gardien()

```
Gardien::~Gardien ( )  [inline]
```

4.6.3 Documentation des fonctions membres

4.6.3.1 avancer()

```
bool Gardien::avancer ( )  [private]
```

fait marcher le gardien vers l'avant

4.6.3.2 can_see_player()

```
bool Gardien::can_see_player ( )  [private]
```

indique si le gardien peut voir le joueur (ie s'il n'y a ni mur ni obstacle entre les deux).

Renvoie

true si pas d'obstacle entre le garde et le joueur, false inon

4.6.3.3 die()

```
void Gardien::die ( )  [private]
```

fonction appellée lors de la mort du gardien

4.6.3.4 fire()

```
void Gardien::fire (
            int angle_vertical )  [virtual]
```

initialisera une boule de feu, qui sera envoyée

Implémente Mover.

4.6.3.5 hurt()

```
void Gardien::hurt ( )
```

fonction indiquant au gardien qu'il a été touché par une boule de feu provenant du joueur. Baissera sa précision et son nombre de points de vie

4.6.3.6 is_legit_move()

```
bool Gardien::is_legit_move (
            double dx,
            double dy )  [private]
```

teste si le mouvement de coordonnées (dx, dy) est acceptable, id est que a implique pas de marcher au travers d'un mur

4.6.3.7 move()

```
bool Gardien::move (
            double dx,
            double dy )  [virtual]
```

Tente d'accomplir le mouvement en paramtre. Meme principe que try_move, mais va faire en sorte de glisser sur les murs.

Paramètres

dx	mouvement en x tenté par le gardien
dy	mouvement en y tenté par le gardien

Renvoie

true si le mouvement a été accompli, false sinon

Implémente Mover.

4.6.3.8 move_to_treasure()

```
bool Gardien::move_to_treasure ( )  [private]
```

fait en sorte que le gardien aille vers le trésor

4.6.3.9 process_fireball()

```
bool Gardien::process_fireball (
            float dx,
            float dy )  [virtual]
```

Gre le déplacement de la boule de feu venant du garde.

Paramètres

dx	le déplacement sur la coordonnée x
dy	le déplacement sur la coordonnée y

Implémente Mover.

4.6.3.10 try_move()

```
bool Gardien::try_move (
            double dx,
            double dy )  [private]
```

tente d'accomplir le mouvement en paramtre

4.6.3.11 update()

```
void Gardien::update (
            void  )  [virtual]
```

Implémente Mover.

4.6.4 Documentation des données membres

4.6.4.1 _pv

```
int Gardien::_pv = BASE_PV  [private]
```

Nombre de PV restants au gardien.

4.6.4.2 BASE_PV

```
const int Gardien::BASE_PV = 10  [static], [private]
```

Nombre de PV de base du gardien.

4.6.4.3 dead

```
bool Gardien::dead = false  [private]
```

indique si le gardien est mort

4.6.4.4 fired

```
bool Gardien::fired = false  [private]
```

Indique si une boule de feu a déj été tirée. Si c'est le cas, il est impossible pour le gardien d'en tirer une autre.

4.6.4.5 l

```
Labyrinthe* Gardien::l  [private]
```

pointeur vers le labyrinthe

4.6.4.6 perte_precision

```
int Gardien::perte_precision = 5  [private]
```

indique la perte de précision, en degrés, du gardien

4.6.4.7 reload

```
int Gardien::reload = 0  [private]
```

indique le statut du rechargement

4.6.4.8 RELOAD_TIME

```
const int Gardien::RELOAD_TIME = 100  [static], [private]
```

Temps de rechargement pour les gardiens (en nombre d'appels de update())

4.6.4.9 reloading

```
bool Gardien::reloading = false  [private]
```

indique si le gardien est en train de recharger (si c'est le cas, il ne peut pas tirer)

La documentation de cette classe a été générée à partir des fichiers suivants :
- Gardien.h
- Gardien.cc
- old_gardien.cc

4.7 Référence de la classe Labyrinthe

```
#include <Labyrinthe.h>
```

Graphe d'héritage de Labyrinthe :

Graphe de collaboration de Labyrinthe :

Fonctions membres publiques

- Labyrinthe ()
- Labyrinthe (char *)
- int width ()
  retourne la largeur du labyrinthe.
- int height ()
  retourne la longueur du labyrinthe.
- char data (int i, int j)
  Informe sur l'état d'occupation de la case (i, j).
- void iamdying ()
  indique au labyrinthe qu'un des gardes est mort
- int nb_alive ()
  Indique le nombre de gardiens encore en vie dans le labyrinthe.
- void hurt_gardien_at (int x, int y)
  indique que une instance de Fireball du joueur a explosé l'endroit o était placé un (ou plusieurs) des gardiens. Va donc "Blesser" le gardien via la méthode Gardien::hurt()
- void hurt_joueur ()
  indique au labyrinthe que le joueur a été touché par une boule de feu provenant d'un des gardiens
- void set_data (int x, int y, char value)
  change la valeur d'une case de la matrice _data, par exemple lors du déplacement d'un gardien
- int dist_of_treasure (int x, int y)
  rend la distance de la case courante

Fonctions membres privées

- void init_data ()
  initialise _data[][], en fonction de lab_with et lab_height
- void fill_dist (int x, int y)
  remplit la matrice de distance au trésor, récursivement, en partant de la case de coordonnées (x, y) qu'on sait déj remplie
- void build_guards (list< coord > guards, const coord &player_pos)
  Initialise _guards et _nguard dans la classe.
- void build_walls (list< pair< coord, coord >> wall_list)
  Initialise les murs (_nwall et _walls[])
- void build_boxes (list< coord > box_list)
  initialise les boites
- void build_treasure (coord c)
  initialisation du trésor
- void build_text (map< char, string > text_map, list< tuple< coord, char, bool >> text_list)
  initialisation des textures
- void init_vector_dist ()
  initialise le vecteur des distances. Doit tre appellé la fin du constructeur

Attributs privés

- char ** _data
  indique si la case est libre ou occupée.
- int lab_width
  Dimension de l'axe 'x' du labyrinthe.
- int lab_height
  Dimension de l'axe 'y' du labyrinthe.
- int _nb_alive
  Indique le nombre.
- vector< vector< int > > _dist_vect
  Vecteur contenant les distances entre chaque case et le trésor.

Membres hérités additionnels

4.7.1 Documentation des constructeurs et destructeur

4.7.1.1 Labyrinthe() [1/2]

```
Labyrinthe::Labyrinthe ( )
```

4.7.1.2 Labyrinthe() [2/2]

```
Labyrinthe::Labyrinthe (
            char * path )
```

4.7.2 Documentation des fonctions membres

4.7.2.1 build_boxes()

```
void Labyrinthe::build_boxes (
            list< coord > box_list )  [private]
```

initialise les boites

Paramètres

box_list	(in) : liste des coordonnées des boites

4.7.2.2 build_guards()

```
void Labyrinthe::build_guards (
            list< coord > guards,
            const coord & player_pos )  [private]
```

Initialise _guards et _nguard dans la classe.

Paramètres

guards	(in) : une liste des coordonnées des gardes
player_pos	(in) : La coordonnée du joueur TODO : génération de gardes différents

4.7.2.3 build_text()

```
void Labyrinthe::build_text (
            map< char, string > text_map,
            list< tuple< coord, char, bool >> text_list )  [private]
```

initialisation des textures

Paramètres

text_map	(in) : la structure associant un caractre représentant une texture dans le .txt du labyrinthe au nom de la texture
text_list	(in) : Une liste des textures, o chaque élément contient la coordonnée de la texture, le caractre représentant la texture, ainsi que l'orientation de la texture (égale HORIZONTAL ou VERTICAL)

4.7.2.4 build_treasure()

```
void Labyrinthe::build_treasure (
            coord c )  [private]
```

initialisation du trésor

Paramètres

c	la coordonnée du trésor

4.7.2.5 build_walls()

```
void Labyrinthe::build_walls (
            list< pair< coord, coord >> wall_list )  [private]
```

Initialise les murs (_nwall et _walls[])

Paramètres

wall_list	(in) : la liste des murs, un mur étant représenté par une paire de coordonnées. Pour chaque paire de coordonnées (x1 y1) (x2 y2) de la liste, on fait l'hypothse que La premire coordonnées est "plus petite" que la seconde, id est (x1 $<$ x2 $||$ y1 $<$ y2).

4.7.2.6 data()

```
char Labyrinthe::data (
            int i,
            int j )  [inline], [virtual]
```

Informe sur l'état d'occupation de la case (i, j).

Renvoie

WALL si la case est un mur, BOX si la case est occupée par une bote, GARDE si la case est occupée par un garde, JOUEUR si la case est occupée par le joueur, TREASURE si la case est occupée par le trésor

Implémente Environnement.

4.7.2.7 dist_of_treasure()

```
int Labyrinthe::dist_of_treasure (
            int x,
            int y )  [inline]
```

rend la distance de la case courante

Paramètres

x	la coordonnée dans l'axe des abcisses
y	la coordonnée dans l'axe des ordonnées

Renvoie

-1 si la case est un mur, 1 000 000 s'il n'existe pas de chemin entre le point de coordonnées (x,y), La distance en nombre de cases entre le point (x,y) et le trésor dans les autres cas

4.7.2.8 fill_dist()

```
void Labyrinthe::fill_dist (
            int x,
            int y )  [private]
```

remplit la matrice de distance au trésor, récursivement, en partant de la case de coordonnées (x, y) qu'on sait déj remplie

4.7.2.9 height()

```
int Labyrinthe::height ( )  [inline], [virtual]
```

retourne la longueur du labyrinthe.

Implémente Environnement.

4.7.2.10 hurt_gardien_at()

```
void Labyrinthe::hurt_gardien_at (
            int x,
            int y )
```

indique que une instance de Fireball du joueur a explosé l'endroit o était placé un (ou plusieurs) des gardiens. Va donc "Blesser" le gardien via la méthode Gardien::hurt()

ů

Paramètres

x	la composante dans l'axe des abcisses de la case dans laquelle la boule de feu explose y la composante dans l'axe des ordonnées de la case dans laquelle la boule de feu explose

4.7.2.11 hurt_joueur()

```
void Labyrinthe::hurt_joueur ( )
```

indique au labyrinthe que le joueur a été touché par une boule de feu provenant d'un des gardiens

Va donc appeller la méthode Chasseur::hurt()

4.7.2.12 iamdying()

```
void Labyrinthe::iamdying ( )  [inline]
```

indique au labyrinthe qu'un des gardes est mort

4.7.2.13 init_data()

```
void Labyrinthe::init_data ( )  [private]
```

initialise _data[][], en fonction de lab_with et lab_height

4.7.2.14 init_vector_dist()

```
void Labyrinthe::init_vector_dist ( )  [private]
```

initialise le vecteur des distances. Doit tre appellé la fin du constructeur

4.7.2.15 nb_alive()

```
int Labyrinthe::nb_alive ( )  [inline]
```

Indique le nombre de gardiens encore en vie dans le labyrinthe.

Renvoie

le nombre de gardes encore en vie.

4.7.2.16 set_data()

```
void Labyrinthe::set_data (
            int x,
            int y,
            char value )
```

change la valeur d'une case de la matrice _data, par exemple lors du déplacement d'un gardien

Paramètres

x	
y	
value	La nouvelle valeur de la case _data[x][y]. En théorie, compris entre EMPTY et TREASURE, mais dans les faits ce sera EMPTY, JOUEUR, ou GARDE, étant donné que seules ces entités se déplacent)

4.7.2.17 width()

```
int Labyrinthe::width ( )  [inline], [virtual]
```

retourne la largeur du labyrinthe.

Implémente Environnement.

4.7.3 Documentation des données membres

4.7.3.1 _data

```
char** Labyrinthe::_data  [private]
```

indique si la case est libre ou occupée.

4.7.3.2 _dist_vect

```
vector<vector<int> > Labyrinthe::_dist_vect  [private]
```

Vecteur contenant les distances entre chaque case et le trésor.

4.7.3.3 _nb_alive

```
int Labyrinthe::_nb_alive  [private]
```

Indique le nombre.

4.7.3.4 lab_height

```
int Labyrinthe::lab_height  [private]
```

Dimension de l'axe 'y' du labyrinthe.

4.7.3.5 lab_width

```
int Labyrinthe::lab_width  [private]
```

Dimension de l'axe 'x' du labyrinthe.

La documentation de cette classe a été générée à partir des fichiers suivants :
- Labyrinthe.h
- Labyrinthe.cc

4.8 Référence de la classe Mover

```
#include <Mover.h>
```

Graphe d'héritage de Mover :

Graphe de collaboration de Mover :

Fonctions membres publiques

- Mover (int x, int y, Labyrinthe *l, const char *modele)
- virtual ~Mover ()
- void tomber ()
- void rester_au_sol ()
- virtual void update (void)=0
- virtual bool process_fireball (float dx, float dy)=0
- virtual bool move (double dx, double dy)=0
- virtual void fire (int angle_vertical)=0
- virtual void right_click (bool shift, bool control)

Attributs publics

- Environnement * _l
- FireBall * _fb
- float _x
- float _y
- int _angle
- void * _model

Fonctions membres privées statiques

- static void * init (const char *)

4.8.1 Documentation des constructeurs et destructeur

4.8.1.1 Mover()

```
Mover::Mover (
            int x,
            int y,
            Labyrinthe * l,
            const char * modele )  [inline]
```

4.8.1.2 ~Mover()

```
virtual Mover::~Mover ( )  [inline], [virtual]
```

4.8.2 Documentation des fonctions membres

4.8.2.1 fire()

```
virtual void Mover::fire (
            int angle_vertical )  [pure virtual]
```

Implémenté dans Gardien, et Chasseur.

4.8.2.2 init()

```
static void* Mover::init (
            const char *  )  [static], [private]
```

4.8.2.3 move()

```
virtual bool Mover::move (
            double dx,
            double dy )  [pure virtual]
```

Implémenté dans Gardien, et Chasseur.

4.8.2.4 process_fireball()

```
virtual bool Mover::process_fireball (
            float dx,
            float dy )  [pure virtual]
```

Implémenté dans Gardien, et Chasseur.

4.8.2.5 rester_au_sol()

```
void Mover::rester_au_sol ( )
```

4.8.2.6 right_click()

```
virtual void Mover::right_click (
            bool shift,
            bool control )  [inline], [virtual]
```

Réimplémentée dans Chasseur.

4.8.2.7 tomber()

```
void Mover::tomber ( )
```

4.8.2.8 update()

```
virtual void Mover::update (
            void  )  [pure virtual]
```

Implémenté dans Gardien, et Chasseur.

4.8.3 Documentation des données membres

4.8.3.1 _angle

```
int Mover::_angle
```

4.8.3.2 _fb

```
FireBall* Mover::_fb
```

4.8.3.3 _l

```
Environnement* Mover::_l
```

4.8.3.4 _model

```
void* Mover::_model
```

4.8.3.5 _x

```
float Mover::_x
```

4.8.3.6 _y

```
float Mover::_y
```

La documentation de cette classe a été générée à partir du fichier suivant :
— Mover.h

4.9 Référence de la structure node

```
#include <Gardien.h>
```

Attributs publics

— int x
— int y

4.9.1 Documentation des données membres

4.9.1.1 x

```
int node::x
```

4.9.1.2 y

```
int node::y
```

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

4.10 Référence de la structure node_comparator

Fonctions membres publiques

— bool operator() (const node &a, const node &b) const

4.10.1 Documentation des fonctions membres

4.10.1.1 operator()()

```
bool node_comparator::operator() (
            const node & a,
            const node & b ) const  [inline]
```

La documentation de cette structure a été générée à partir du fichier suivant :
— old_gardien.cc

4.11 Référence de la classe Sound

```
#include <Sound.h>
```

Fonctions membres publiques

— Sound (const char *)
— ~Sound ()
— void play (float volume=1., float pan=0.)

Fonctions membres privées

— void init (void)

Attributs privés

— FMOD: :Sound * _sound

Attributs privés statiques

— static FMOD: :System * _system
— static FMOD: :Channel * _channel
— static int _nsounds

4.11.1 Documentation des constructeurs et destructeur

4.11.1.1 Sound()

```
Sound::Sound (
            const char *  )
```

4.11.1.2 ~Sound()

```
Sound::~Sound ( )
```

4.11.2 Documentation des fonctions membres

4.11.2.1 init()

```
void Sound::init (
            void  )  [private]
```

4.11.2.2 play()

```
void Sound::play (
            float volume = 1.,
            float pan = 0. )
```

4.11.3 Documentation des données membres

4.11.3.1 _channel

```
FMOD::Channel* Sound::_channel  [static], [private]
```

4.11.3.2 _nsounds

```
int Sound::_nsounds  [static], [private]
```

4.11.3.3 _sound

```
FMOD::Sound* Sound::_sound  [private]
```

4.11.3.4 _system

```
FMOD::System* Sound::_system  [static], [private]
```

La documentation de cette classe a été générée à partir du fichier suivant :
— Sound.h

4.12 Référence de la structure Wall

```
#include <Environnement.h>
```

Attributs publics

— int _x1
— int _y1
— int _x2
— int _y2
— int _ntex

4.12.1 Documentation des données membres

4.12.1.1 _ntex

```
int Wall::_ntex
```

4.12.1.2 _x1

```
int Wall::_x1
```

4.12.1.3 _x2

```
int Wall::_x2
```

4.12.1.4 _y1

```
int Wall::_y1
```

4.12.1.5 _y2

```
int Wall::_y2
```

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

Chapitre 5

Documentation des fichiers

5.1 Référence du fichier Chasseur.cc

```
#include "Chasseur.h"
```
Graphe des dépendances par inclusion de Chasseur.cc :

5.2 Référence du fichier Chasseur.h

```
#include <stdio.h>
#include "Mover.h"
#include "Sound.h"
#include "Labyrinthe.h"
```
Graphe des dépendances par inclusion de Chasseur.h : Ce graphe montre quels fichiers incluent directement ou indirectement ce fichier :

Classes

— class Chasseur

5.3 Référence du fichier Environnement.h

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

Classes

— struct Wall
— struct Box
— class Environnement

Macros

— #define EMPTY 0

Fonctions

— void partie_terminee (bool win)
— void message (const char ∗format,...)

5.3.1 Documentation des macros

5.3.1.1 EMPTY

```
#define EMPTY 0
```

5.3.2 Documentation des fonctions

5.3.2.1 message()

```
void message (
            const char * format,
             ... )
```

5.3.2.2 partie_terminee()

```
void partie_terminee (
            bool win )
```

5.4 Référence du fichier FireBall.h

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

Classes

— class FireBall

Macros

— #define M_PI 3.1415926

Énumérations

— enum FBstat { FB_NONE, FB_MOVE, FB_EXPLODE }

5.4.1 Documentation des macros

5.4.1.1 M_PI

```
#define M_PI 3.1415926
```

5.4.2 Documentation du type de l'énumération

5.4.2.1 FBstat

```
enum FBstat
```

Valeurs énumérées

FB_NONE	
FB_MOVE	
FB_EXPLODE	

5.5 Référence du fichier Gardien.cc

```
#include "Gardien.h"
```

Graphe des dépendances par inclusion de Gardien.cc :

5.6 Référence du fichier Gardien.h

```
#include "Mover.h"
#include <stdlib.h>
#include <iostream>
#include <cstdlib>
#include <cmath>
#include <map>
#include <list>
#include <limits.h>
#include <set>
#include <iterator>
```

```
#include <stdexcept>
#include "Labyrinthe.h"
```

Graphe des dépendances par inclusion de Gardien.h : Ce graphe montre quels fichiers incluent directement ou indirectement ce fichier :

Classes

- struct node
- class Gardien
  Classe implémentant les gardes.

5.7 Référence du fichier laby2.c

5.8 Référence du fichier Labyrinthe.cc

```
#include "Labyrinthe.h"
#include "Chasseur.h"
#include "Gardien.h"
#include <sstream>
```

Graphe des dépendances par inclusion de Labyrinthe.cc :

Fonctions

- bool is_blank (const string &str)
  Décide si une ligne est considérée comme vide (commentaires inclus), id est ne contenant que des caractres espace (ou un commentaire)
- char first_char (const string &str)
  rend le premier caractre non-nul de la chaine, ou 0 si la chaine est vide
- void build_map (char *path, vector< string > &terrain, map< char, string > &tex_list)
- void parse_wall_extr (const vector< string > &terrain, const coord &base_coord, list< coord > &extr_list, list< tuple< coord, char, bool >> &text_list)
  fait la liste des extrémités des murs partant de la coordonnée base_coord sur le terrain.
- void add_to_wall_list (const coord &base_coord, list< coord > other_extr, list< pair< coord, coord >> &wall_lst)
  fusionne les extrémités et la coordonnée de base, pour créer un ou deux murs
- void parse_map (const vector< string > &terrain, list< coord > &guard_list, list< pair< coord, coord >> &wall_list, list< tuple< coord, char, bool >> &text_list, list< coord > &box_list, coord &player_pos, coord &treasure_pos, int &height, int &width)
  construit les listes

5.8.1 Documentation des fonctions

5.8.1.1 add_to_wall_list()

```
void add_to_wall_list (
            const coord & base_coord,
            list< coord > other_extr,
            list< pair< coord, coord >> & wall_lst )
```

fusionne les extrémités et la coordonnée de base, pour créer un ou deux murs

Paramètres

other_extr	(in) : La liste des extrémités
wall_lst	(inout) : La liste des murs

5.8.1.2 build_map()

```
void build_map (
            char * path,
            vector< string > & terrain,
            map< char, string > & tex_list )
```

5.8.1.3 first_char()

```
char first_char (
            const string & str )
```

rend le premier caractre non-nul de la chaine, ou 0 si la chaine est vide

Paramètres

la	str tester

5.8.1.4 is_blank()

```
bool is_blank (
            const string & str )
```

Décide si une ligne est considérée comme vide (commentaires inclus), id est ne contenant que des caractres espace (ou un commentaire)

Paramètres

str	la chane de caractre testée

Renvoie

true si la chane est vide (uniquement des espaces ou des commentaires), false sinon

5.8.1.5 parse_map()

```
void parse_map (
            const vector< string > & terrain,
            list< coord > & guard_list,
            list< pair< coord, coord >> & wall_list,
            list< tuple< coord, char, bool >> & text_list,
            list< coord > & box_list,
            coord & player_pos,
            coord & treasure_pos,
            int & height,
            int & width )
```

construit les listes

Paramètres

terrain	(in) : Le terrain/labyrinthe
guard_list	(out) : La liste des coordonnées des gardes
wall_list	(out) : La liste des coordonnées des murs
text_list	(out) : La liste des textures (une texture = coordonnée, char-id de texture et un indicateur sur le sens de la texture)
box_list	(out) : La liste des boites
player_pos	(out) : La position du joueur
treasure_pos(out)	La position du trésor

5.8.1.6 parse_wall_extr()

```
void parse_wall_extr (
            const vector< string > & terrain,
            const coord & base_coord,
            list< coord > & extr_list,
            list< tuple< coord, char, bool >> & text_list )
```

fait la liste des extrémités des murs partant de la coordonnée base_coord sur le terrain.

Paramètres

terrain	(in) : Le labyrinthe
base_coord	(in) : La coordonnée de départ du mur
extr_list	(out) : La liste des extrémités des murs
text_list	(inout) : La liste des coordonnées+caractres+"verticalité" des textures

5.9 Référence du fichier Labyrinthe.h

```
#include <stdio.h>
#include <stdlib.h>
```

```
#include <utility>
#include <list>
#include <iostream>
#include <string>
#include <vector>
#include <ios>
#include <map>
#include <fstream>
#include <unistd.h>
#include "Environnement.h"
```

Graphe des dépendances par inclusion de Labyrinthe.h : Ce graphe montre quels fichiers incluent directement ou indirectement ce fichier :

Classes

- struct coord
  Structure représentant une coordonnée du labyrinthe.
- class Labyrinthe

Fonctions

- void build_map (const char *path, vector< string > &terrain, map< char, string > &tex_list)
  fonction construisant la fois le terrain et la liste des textures
- void parse_map (const vector< string > &terrain, list< coord > &guard_list, list< pair< coord, coord >> &wall_list, list< tuple< coord, char, bool >> &text_list, list< coord > &box_list, coord &player_pos, coord &treasure_pos, int &height, int &width)
  construit les listes
- void parse_wall_extr (const vector< string > &terrain, const coord &base_coord, list< coord > &extr_list, list< tuple< coord, char, bool >> &text_list)
  fait la liste des extrémités des murs partant de la coordonnée base_coord sur le terrain.
- void add_to_wall_list (const coord &base_coord, list< coord > other_extr, list< pair< coord, coord >> &wall_lst)
  fusionne les extrémités et la coordonnée de base, pour créer un ou deux murs
- bool is_blank (const string &str)
  Décide si une ligne est considérée comme vide (commentaires inclus), id est ne contenant que des caractres espace (ou un commentaire)
- char first_char (const string &str)
  rend le premier caractre non-nul de la chaine, ou 0 si la chaine est vide

Variables

- const bool HORIZONTAL = false
  booléen permettant d'indiquer le sens des affiches
- const bool VERTICAL = true
  booléen permettant d'indiquer le sens des affiches
- const char WALL = 1
- const char BOX = 2
- const char GARDE = 3
- const char JOUEUR = 4
- const char TREASURE = 5

5.9.1 Documentation des fonctions

5.9.1.1 add_to_wall_list()

```
void add_to_wall_list (
            const coord & base_coord,
            list< coord > other_extr,
            list< pair< coord, coord >> & wall_lst )
```

fusionne les extrémités et la coordonnée de base, pour créer un ou deux murs

Paramètres

other_extr	(in) : La liste des extrémités
wall_lst	(inout) : La liste des murs

5.9.1.2 build_map()

```
void build_map (
            const char * path,
            vector< string > & terrain,
            map< char, string > & tex_list )
```

fonction construisant la fois le terrain et la liste des textures

Paramètres

path	(in) : Le chemin d'accs au fichier du labyrinthe
terrain	(out) : Le tableau de string représentant le labyrinthe
tex_list	(out) : La hashmap associant un chemin d'accs un caractre

5.9.1.3 first_char()

```
char first_char (
            const string & str )
```

rend le premier caractre non-nul de la chaine, ou 0 si la chaine est vide

Paramètres

la	str tester

5.9.1.4 is_blank()

```
bool is_blank (
```

```
            const string & str )
```

Décide si une ligne est considérée comme vide (commentaires inclus), id est ne contenant que des caractres espace (ou un commentaire)

Paramètres

str	la chane de caractre testée

Renvoie

true si la chane est vide (uniquement des espaces ou des commentaires), false sinon

5.9.1.5 parse_map()

```
void parse_map (
            const vector< string > & terrain,
            list< coord > & guard_list,
            list< pair< coord, coord >> & wall_list,
            list< tuple< coord, char, bool >> & text_list,
            list< coord > & box_list,
            coord & player_pos,
            coord & treasure_pos,
            int & height,
            int & width )
```

construit les listes

Paramètres

terrain	(in) : Le terrain/labyrinthe
guard_list	(out) : La liste des coordonnées des gardes
wall_list	(out) : La liste des coordonnées des murs
text_list	(out) : La liste des textures (une texture = coordonnée, char-id de texture et un indicateur sur le sens de la texture)
box_list	(out) : La liste des boites
player_pos	(out) : La position du joueur
treasure_pos(out)	La position du trésor

5.9.1.6 parse_wall_extr()

```
void parse_wall_extr (
            const vector< string > & terrain,
            const coord & base_coord,
            list< coord > & extr_list,
            list< tuple< coord, char, bool >> & text_list )
```

fait la liste des extrémités des murs partant de la coordonnée base_coord sur le terrain.

Paramètres

terrain	(in) : Le labyrinthe
base_coord	(in) : La coordonnée de départ du mur
extr_list	(out) : La liste des extrémités des murs
text_list	(inout) : La liste des coordonnées+caractres+"verticalité" des textures

5.9.2 Documentation des variables

5.9.2.1 BOX

```
const char BOX = 2
```

5.9.2.2 GARDE

```
const char GARDE = 3
```

5.9.2.3 HORIZONTAL

```
const bool HORIZONTAL = false
```

booléen permettant d'indiquer le sens des affiches

5.9.2.4 JOUEUR

```
const char JOUEUR = 4
```

5.9.2.5 TREASURE

```
const char TREASURE = 5
```

5.9.2.6 VERTICAL

```
const bool VERTICAL = true
```

booléen permettant d'indiquer le sens des affiches

5.9.2.7 WALL

```
const char WALL = 1
```

5.10 Référence du fichier Mover.h

```
#include "FireBall.h"
#include "Environnement.h"
```

Graphe des dépendances par inclusion de Mover.h : Ce graphe montre quels fichiers incluent directement ou indirectement ce fichier :

Classes

— class Mover

5.11 Référence du fichier old_gardien.cc

```
#include "Gardien.h"
```

Graphe des dépendances par inclusion de old_gardien.cc :

Classes

— struct node_comparator

Fonctions

— bool operator== (const node &a, const node &b)
— int get (map< node, int, node_comparator > &m, node &key, int def)
— list< node > build_solution (const node &goal, map< node, node, node_comparator > &predecesseurs)
— int estimation_dist (const node &a, const node &b)
— list< node > get_shortest_path (node start, node goal, Environnement *evt)

Variables

— int MIN = -1000000
— int MAX = 1000000

5.11.1 Documentation des fonctions

5.11.1.1 build_solution()

```
list<node> build_solution (
            const node & goal,
            map< node, node, node_comparator > & predecesseurs )
```

5.11.1.2 estimation_dist()

```
int estimation_dist (
            const node & a,
            const node & b )
```

5.11.1.3 get()

```
int get (
            map< node, int, node_comparator > & m,
            node & key,
            int def )
```

5.11.1.4 get_shortest_path()

```
list<node> get_shortest_path (
            node start,
            node goal,
            Environnement * evt )
```

5.11.1.5 operator==()

```
bool operator== (
            const node & a,
            const node & b )
```

5.11.2 Documentation des variables

5.11.2.1 MAX

```
int MAX = 1000000
```

5.11.2.2 MIN

```
int MIN = -1000000
```

5.12 Référence du fichier Sound.h

```
#include "fmod.hpp"
#include "fmod_errors.h"
```

Graphe des dépendances par inclusion de Sound.h : Ce graphe montre quels fichiers incluent directement ou indirectement ce fichier :

Classes

— class Sound