Visualiseur d'algorithme

Généré par Doxygen 1.8.8

Jeudi 19 Mars 2015 17 :00 :17

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

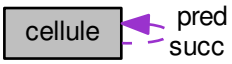
Liste de tous les fichiers documentés avec une brève description :

Chapitre 4

Documentation des classes

4.1 Référence de la structure cellule

Graphe de collaboration de cellule :



Attributs publics

- int **val**
- struct cellule * **pred**
- struct cellule * **succ**

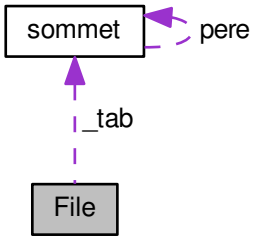
4.1.1 Description détaillée

Définition à la ligne 7 du fichier cellule.hpp.

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

4.2 Référence de la classe File

Graphe de collaboration de File :



Fonctions membres publiques

- **File** (File ∗)
- int **file_vide** ()
- void **enfiler** (sommet_t ∗)
- sommet_t ∗ **defiler** ()
- void **afficher_file** ()

Attributs publics

- sommet_t ∗∗ **_tab**
- int **_tete**
- int **_queue**

4.2.1 Description détaillée

Définition à la ligne 14 du fichier file.hpp.

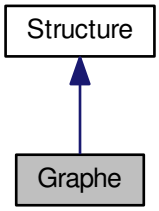
La documentation de cette classe a été générée à partir des fichiers suivants :
- file.hpp
- file.cpp

4.3 Référence de la classe Graphe

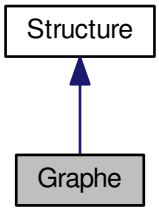
Classe gerant la structure de donnée de graphe.

`#include <graphe.hpp>`

Graphe d'héritage de Graphe :



Graphe de collaboration de Graphe :



Fonctions membres publiques

- Graphe (std : :string path)
  Constructeur de la structure de graphe.
- ∼Graphe ()
  Destructeur.
- void charger ()
  Chargement du graphe.
- void creer_listes_adjacences ()
  Creer liste d'adjacence.
- void creer_matrice_adjacences ()
  Creer matrice d'adjacence.
- void creer_matrice_laplace ()
  Creer matrice laplacienne.
- parcours_t ∗ parcours_largeur (sommet_t ∗s)
  Parcours en largeur.
- parcours_t ∗ parcours_profondeur ()
  Parcours en profondeur.
- void afficher_parcours_profondeur ()
  Affichage du parcours en profondeur.
- void DFS_visiter_noeud (sommet_t ∗u, int ∗time, parcours_t ∗p)
  Fonction interne à l'algorithme DFS.
- void afficher_listes_adjacences (liste_adjacence_t ∗l)
  Affichage liste d'adjacences.
- void afficher_matrice_adjacences (matrice_adjacence_t ∗l)
  Affichage matrice d'adjacences.

— void afficher_chemin (sommet_t *i, sommet_t *j)
Affichage chemin.

Membres hérités additionnels

4.3.1 Description détaillée

Classe gerant la structure de donnée de graphe.

Définition à la ligne 39 du fichier graphe.hpp.

4.3.2 Documentation des constructeurs et destructeur

4.3.2.1 Graphe : :Graphe (std : :string *path*)

Constructeur de la structure de graphe.

Structure de données permettant de manipuler des graphes et des algos de graphe. Le graphe ne sera pas charger, il est juste créer, pour le charger il faut appelé explicitement la fonction charger()

Paramètres

in	*path*	chemin vers le fichier contenant le graphe

Définition à la ligne 14 du fichier graphe.cpp.

4.3.3 Documentation des fonctions membres

4.3.3.1 void Graphe : :afficher_chemin (sommet_t * *i*, sommet_t * *j*)

Affichage chemin.

affiche le chemin après un BFS, permet de voir le plus court chemin entre deux noeud passé en parametre

Paramètres

in	*i*	sommet de depart
in	*j*	sommet d'arrivé

Définition à la ligne 588 du fichier graphe.cpp.

4.3.3.2 void Graphe : :afficher_listes_adjacences (liste_adjacence_t * *l*)

Affichage liste d'adjacences.

TODO : utiliser operator<<

Définition à la ligne 426 du fichier graphe.cpp.

4.3.3.3 void Graphe : :afficher_matrice_adjacences (matrice_adjacence_t * *l*)

Affichage matrice d'adjacences.

TODO : utiliser operator<<

Définition à la ligne 447 du fichier graphe.cpp.

4.3.3.4 void Graphe : :afficher_parcours_profondeur ( )

Affichage du parcours en profondeur.

Affiche le parcours en profondeur resultant, effectuer l'appel à cette fonction après avoir fait un parcours en profndeur

Définition à la ligne 603 du fichier graphe.cpp.

4.3.3.5 void Graphe : :charger ( ) [virtual]

Chargement du graphe.

Chargement du graphe de maniere explicite à partir de l'appel de cette fonction

Implémente Structure.

Définition à la ligne 38 du fichier graphe.cpp.

4.3.3.6 void Graphe : :DFS_visiter_noeud (sommet_t ∗ *u*, int ∗ *time*, parcours_t ∗ *p*)

Fonction interne à l'algorithme DFS.

Paramètres

in	*u*	sommet à visiter
in	*time*	date à laquelle on visite le noeud
in	*p*	parcours dans lequel est integré la visite du noeud

Définition à la ligne 639 du fichier graphe.cpp.

4.3.3.7 parcours_t ∗ Graphe : :parcours_largeur (sommet_t ∗ *s*)

Parcours en largeur.

Algorithme BFS, parcours en largeur sur le graphe courant

Paramètres

in	*s*	sommet de depart pour le parcours en largeur
out	*parcours*	en largeur resultant

Définition à la ligne 523 du fichier graphe.cpp.

4.3.3.8 parcours_t ∗ Graphe : :parcours_profondeur ( )

Parcours en profondeur.

Algorithme DFS, parcours en profondeur sur le graphe courant

Paramètres

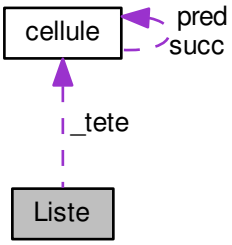
out	*parcours*	en largeur resultant

Définition à la ligne 612 du fichier graphe.cpp.

La documentation de cette classe a été générée à partir des fichiers suivants :
- graphe.hpp
- graphe.cpp

4.4 Référence de la classe Liste

Graphe de collaboration de Liste :



Fonctions membres publiques

- void **inserer** (cellule_t *)
- cellule_t * **rechercher** (int)
- void **supprimer** (cellule_t *)
- int **compter_liste** ()
- void **afficher_liste** ()

Attributs publics

- cellule_t * **_tete**

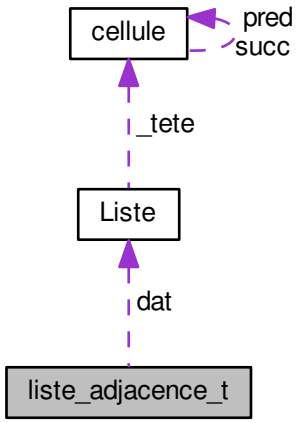
4.4.1 Description détaillée

Définition à la ligne 7 du fichier liste.hpp.

La documentation de cette classe a été générée à partir des fichiers suivants :
- liste.hpp
- liste.cpp

4.5 Référence de la structure liste_adjacence_t

Graphe de collaboration de liste_adjacence_t :



Attributs publics

- Liste ** **dat**
- int **nbrA**
- int **nbrS**

4.5.1 Description détaillée

Définition à la ligne 13 du fichier graphe.hpp.

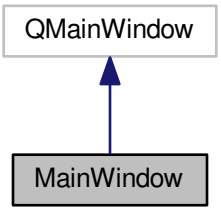
La documentation de cette structure a été générée à partir du fichier suivant :
- graphe.hpp

4.6 Référence de la classe MainWindow

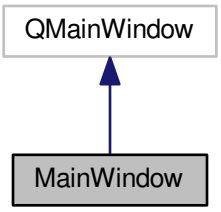
classe gerant le placement et interactions des principaux widgets dans la fenetre principale

`#include <mainwindow.hpp>`

Graphe d'héritage de MainWindow :



Graphe de collaboration de MainWindow :



Fonctions membres publiques

- MainWindow (QWidget *parent=0)
  Constructeur de la fenetre principale.
- void selection_graphe ()
  Selection d'un graphe.
- void charger_menu ()
  Chargeur du menu.
- void charger_opengl ()
  Chargeur OpenGL.

4.6.1 Description détaillée

classe gerant le placement et interactions des principaux widgets dans la fenetre principale

Définition à la ligne 38 du fichier mainwindow.hpp.

4.6.2 Documentation des constructeurs et destructeur

4.6.2.1 MainWindow : :MainWindow (QWidget * *parent* = 0)

Constructeur de la fenetre principale.

il s'agit de la fenetre principale, elle n'a pas de parent, mais on laisse le parametre par commodité

Paramètres

in	*parent*	le widget parent

Définition à la ligne 7 du fichier mainwindow.cpp.

4.6.3 Documentation des fonctions membres

4.6.3.1 void MainWindow : :charger_menu ( ) [inline]

Chargeur du menu.

Appelé lors de la construction de la fenetre principale, construit la barre de menu, et associe les signaux aux actions

Définition à la ligne 66 du fichier mainwindow.cpp.

4.6.3.2 void MainWindow : :charger_opengl ( ) [inline]

Chargeur OpenGL.

Appelé lors de la construction de la fenetre principale, cette fonction appelle le constructeur de SceneGL créant le widget OpenGL qui sera integré dans la fenetre principale

Définition à la ligne 116 du fichier mainwindow.cpp.

4.6.3.3 void MainWindow : :selection_graphe ( )

Selection d'un graphe.

Appelé lors de la reception d'un signal emis par l'utilisateur lorsqu'il demande d'ouvrir un nouveau graphe dans le menu. Cette fonction ouvre une boite de dialogue permettant de selectionner le graphe, puis va créer une structure de graphe à partir du fichier sélectionné.

Définition à la ligne 34 du fichier mainwindow.cpp.

La documentation de cette classe a été générée à partir des fichiers suivants :
- mainwindow.hpp
- mainwindow.cpp

4.7 Référence de la structure matrice_adjacence_t

Attributs publics

- int ** **dat**
- int **nbrA**
- int **nbrS**

4.7.1 Description détaillée

Définition à la ligne 25 du fichier graphe.hpp.

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

4.8 Référence de la structure matrice_laplace_t

Attributs publics

— int ** **dat**
— int **nbrA**
— int **nbrS**

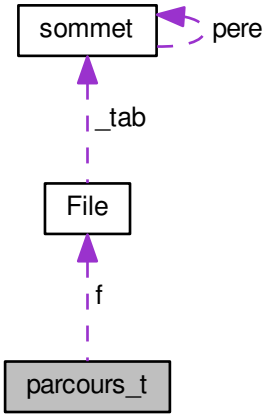
4.8.1 Description détaillée

Définition à la ligne 19 du fichier graphe.hpp.

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

4.9 Référence de la structure parcours_t

Graphe de collaboration de parcours_t :



Attributs publics

— File * **f**

4.9.1 Description détaillée

Définition à la ligne 31 du fichier graphe.hpp.

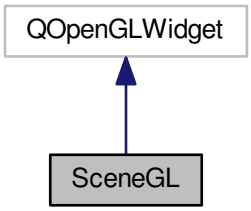
La documentation de cette structure a été générée à partir du fichier suivant :
— graphe.hpp

4.10 Référence de la classe SceneGL

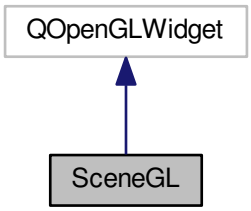
classe gerant le contexte OpenGL de l'application

`#include <scene.hpp>`

Graphe d'héritage de SceneGL :



Graphe de collaboration de SceneGL :



Fonctions membres publiques

- SceneGL (std : :string vertex_shad, std : :string fragment_shad, QWidget ∗parent=0, Structure ∗str=nullptr, Qt : :WindowFlags f=0)
  Constructeur de la scene GL.
- ∼SceneGL ()
  Destructeur.
- void initializeGL ()
  Initialise le contexte OpenGL.
- void cleanupGL ()
  Fonction de formattage.
- void resizeGL (int w, int h)
  Fonction de redimensionnement.
- void paintGL ()
  Fonction qui est appelé quand le widget a besoin d'etre repaint.
- void Zoompos ()
  Fonction de zoom avant de la camera.
- void Zoomneg ()
  Fonction de zoom arriere de la camera.
- Structure ∗ get_structure ()
  Getters de la structure.
- void charger_contenu_graphique ()
  Charger contenu graphique.

4.10.1 Description détaillée

classe gerant le contexte OpenGL de l'application

Définition à la ligne 27 du fichier scene.hpp.

4.10.2 Documentation des constructeurs et destructeur

4.10.2.1 SceneGL : :SceneGL (std : :string *vertex_shad,* std : :string *fragment_shad,* QWidget ∗ *parent* = `0`, Structure ∗ *str* = `nullptr`, Qt : :WindowFlags *f* = `0`)

Constructeur de la scene GL.

il s'agit de la fenetre principale, elle n'a pas de parent, mais on laisse le parametre par commodité

Paramètres

in	*vertex_shad*	le vertex shaders
in	*fragment_shad*	le fragment shaders
in	*parent*	le widget parent où sera contenu le contexte OpenGL
in	*str*	la structure
in	*flags*	flags de widget

Définition à la ligne 20 du fichier scene.cpp.

4.10.2.2 SceneGL : :∼SceneGL ( )

Destructeur.

Destructeur

Définition à la ligne 284 du fichier scene.cpp.

4.10.3 Documentation des fonctions membres

4.10.3.1 void SceneGL : :charger_contenu_graphique ( )

Charger contenu graphique.

Charge le contenu graphique à afficher dans le contexte OpenGL

Définition à la ligne 128 du fichier scene.cpp.

4.10.3.2 void SceneGL : :cleanupGL ( )

Fonction de formattage.

Cette fonction supprime les identifiants de shaders et vide les buffers

Définition à la ligne 194 du fichier scene.cpp.

4.10.3.3 Structure ∗ SceneGL : :get_structure ( )

Getters de la structure.

Getters de la structure chargé dans la scene OpenGL courante

Définition à la ligne 267 du fichier scene.cpp.

4.10.3.4 void SceneGL : :initializeGL ( )

Initialise le contexte OpenGL.

Cette fonction est appelé une fois avant le premier appel à paintGL ou à resizeGL

Définition à la ligne 52 du fichier scene.cpp.

4.10.3.5 void SceneGL : :paintGL ( )

Fonction qui est appelé quand le widget a besoin d'etre repaint.

Cette fonction est appelé une fois avant le premier appel à paintGL ou à resizeGL

Définition à la ligne 224 du fichier scene.cpp.

4.10.3.6 void SceneGL : :resizeGL (int *w*, int *h*)

Fonction de redimensionnement.

Cette fonction est appelé lorsque le widget conteneur du contexte OpenGL est redimensionné

Paramètres

in	*w*	nouvelle dimension de largeur
in	*h*	nouvelle dimension de hauteur

Définition à la ligne 219 du fichier scene.cpp.

4.10.3.7 void SceneGL : :Zoomneg ( )

Fonction de zoom arriere de la camera.

Fonction appelé lors d'un zoom arriere commandé par l'IHM

Définition à la ligne 278 du fichier scene.cpp.

4.10.3.8 void SceneGL : :Zoompos ( )

Fonction de zoom avant de la camera.

Fonction appelé lors d'un zoom arriere commandé par l'IHM

Définition à la ligne 272 du fichier scene.cpp.

La documentation de cette classe a été générée à partir des fichiers suivants :
- scene.hpp
- scene.cpp

4.11 Référence de la structure SceneVertex

structure d'un vertex incluant sa position et couleur

`#include <structure.hpp>`

Attributs publics

- float **Position** [PositionSize]
- float **Color** [ColorSize]

Attributs publics statiques

— static const int **PositionSize** = 3
— static const int **ColorSize** = 3

4.11.1 Description détaillée

structure d'un vertex incluant sa position et couleur

Définition à la ligne 21 du fichier structure.hpp.

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

4.12 Référence de la classe Shader

Classe gerant la compilation et verrouillage du vertex et fragment shaders.

`#include <shader.hpp>`

Fonctions membres publiques

— Shader (std : :string ver_shad, std : :string fra_shad)
Constructeur shader.
— ∼Shader ()
Destructeur shader.
— void del ()
Fonction de formattage.
— void charger (QOpenGLFunctions ∗function_contexte)
Fonction de chargement.
— bool verif_compil_shader (GLint id_shader, std : :string nom_shader)
Fonction de verification de la compilation.
— bool verif_link_shader (GLint id_program, std : :string nom)
Fonction de verification de l'edition des liens.
— GLint **get_id_position** ()
— GLint **get_id_texture** ()
— GLint **get_id_color** ()
— GLint **get_id_shader_program** ()
— GLint **get_id_vertex_shader** ()
— GLint **get_id_fragme_shader** ()
— GLint **get_vertex_source_length** ()
— GLint **get_fragme_source_length** ()
— const GLchar ∗ **get_vertex_source** ()
— const GLchar ∗ **get_fragme_source** ()
— void **set_id_position** (GLint)
— void **set_id_texture** (GLint)
— void **set_id_color** (GLint)
— void **set_id_shader_program** (GLint)
— void **set_id_vertex_shader** (GLint)
— void **set_id_fragme_shader** (GLint)

4.12.1 Description détaillée

Classe gerant la compilation et verrouillage du vertex et fragment shaders.

Définition à la ligne 22 du fichier shader.hpp.

4.12.2 Documentation des constructeurs et destructeur

4.12.2.1 Shader : :Shader (std : :string *ver_shad,* std : :string *fra_shad*)

Constructeur shader.

Constructeur shaders

Paramètres

in	*ver_shad*	le vertex shaders
in	*fra_shad*	le fragment shaders

Définition à la ligne 10 du fichier shader.cpp.

4.12.2.2 Shader : :∼Shader ( )

Destructeur shader.

Destructeur shader

Définition à la ligne 42 du fichier shader.cpp.

4.12.3 Documentation des fonctions membres

4.12.3.1 void Shader : :charger (QOpenGLFunctions ∗ *function_contexte*)

Fonction de chargement.

Compile le vertex et fragment shaders, puis fait l'edition des liens des deux shaders

Paramètres

in	*function_↩ contexte*	resolver de fonctions

Définition à la ligne 51 du fichier shader.cpp.

4.12.3.2 void Shader : :del ( )

Fonction de formattage.

Fonction de formattage

Définition à la ligne 173 du fichier shader.cpp.

4.12.3.3 bool Shader : :verif_compil_shader (GLint *id_shader,* std : :string *nom_shader*)

Fonction de verification de la compilation.

Verifie les flag de status de compilation de chacun des shaders

Paramètres

in	*id_shader*	identifiant du shader dont on verifie sa compilation
in	*nom_shader*	nom du shader dont on verifie sa compilation

Définition à la ligne 137 du fichier shader.cpp.

4.12.3.4 bool Shader : :verif_link_shader (GLint *id_program,* std : :string *nom*)

Fonction de verification de l'edition des liens.

Verifie les flag de status d'edition des liens de chacun des shaders

Paramètres

in	*id_program*	identifiant du programme resultant de l'edition des liens du vertex et fragment shaders
in	*nom*	nom du programme resultant de l'edition des liens du vertex et fragment shaders

Définition à la ligne 101 du fichier shader.cpp.

La documentation de cette classe a été générée à partir des fichiers suivants :
- shader.hpp
- shader.cpp

4.13 Référence de la structure sommet

Graphe de collaboration de sommet :

Attributs publics

- couleur_t **couleur**
- int **f**
- int **d**
- int **val**
- struct sommet * **pere**

4.13.1 Description détaillée

Définition à la ligne 6 du fichier file.hpp.

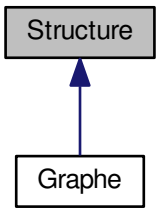
La documentation de cette structure a été générée à partir du fichier suivant :
- file.hpp

4.14 Référence de la classe Structure

Classe gerant les structures de données chargées dans l'IHM.

`#include <structure.hpp>`

Graphe d'héritage de Structure :



Fonctions membres publiques

- Structure (std : :string nom_fichier)
  Constructeur de la structure de donnée.
- virtual ~Structure ()
  Destructeur virtuel.
- void compute_coordonnes ()
  Calcul du placement.
- virtual void charger ()=0
  Chargement de la structure de donnée.
- bool est_init ()
  Est initialisé
- std : :vector< SceneVertex > * **get_vertices** ()
- std : :vector< GLuint > * **get_indices** ()

Attributs protégés

- const char * **_path_fichier**
- std : :vector< SceneVertex > **_vertices**
- bool **_est_init**
- std : :vector< GLuint > **_indices**

4.14.1 Description détaillée

Classe gerant les structures de données chargées dans l'IHM.

Définition à la ligne 39 du fichier structure.hpp.

4.14.2 Documentation des constructeurs et destructeur

4.14.2.1 Structure : :Structure (std : :string *nom_fichier*)

Constructeur de la structure de donnée.

Structure de données qui sera chargé de maniere à etre interfaçable dans l'IHM

Paramètres

in	*nom_fichier*	nom de fichier à charger pour la structure

Définition à la ligne 10 du fichier structure.cpp.

4.14.3 Documentation des fonctions membres

4.14.3.1 virtual void Structure : :charger ( ) [pure virtual]

Chargement de la structure de donnée.

Chargement de la structure de donnée à partir du fichier, appelé lors de la creation de l'instance courante

Paramètres

in	*nom_fichier*	nom de fichier qui va etre chargé pour la structure

Implémenté dans Graphe.

4.14.3.2 void Structure : :compute_coordonnes ( )

Calcul du placement.

Calculer les coordonnées de placement pour l'affichage de la structure de donnée courante

Définition à la ligne 16 du fichier structure.cpp.

4.14.3.3 bool Structure : :est_init ( )

Est initialisé

permet de savoir si la structure courante est bien initialisé et prete à etre chargé dans la scene opengl

Paramètres

out	*retourne*	un booleen specifiant si la structure courante est bien initialisé

Définition à la ligne 31 du fichier structure.cpp.

La documentation de cette classe a été générée à partir des fichiers suivants :
- structure.hpp
- structure.cpp

Chapitre 5

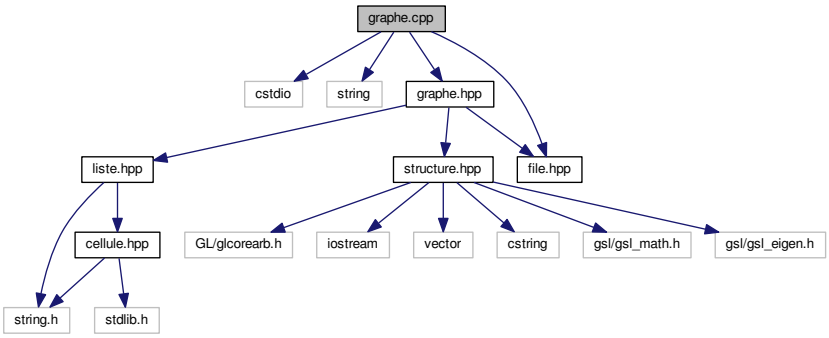
Documentation des fichiers

5.1 Référence du fichier graphe.cpp

Implementation de graphe.hpp.

```
#include <cstdio>
#include <string>
#include "graphe.hpp"
#include "file.hpp"
```

Graphe des dépendances par inclusion de graphe.cpp :



5.1.1 Description détaillée

Implementation de graphe.hpp.

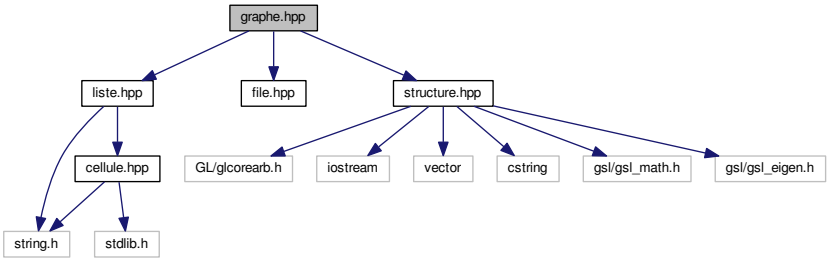
Définition dans le fichier graphe.cpp.

5.2 Référence du fichier graphe.hpp

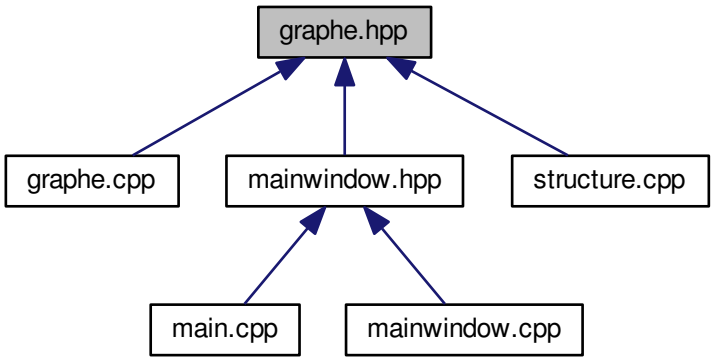
Gère la structure de donnée de graphe.

```
#include "liste.hpp"
#include "file.hpp"
#include "structure.hpp"
```

Graphe des dépendances par inclusion de graphe.hpp :



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



Classes

- struct liste_adjacence_t
- struct matrice_laplace_t
- struct matrice_adjacence_t
- struct parcours_t
- class Graphe

  Classe gerant la structure de donnée de graphe.

5.2.1 Description détaillée

Gère la structure de donnée de graphe.

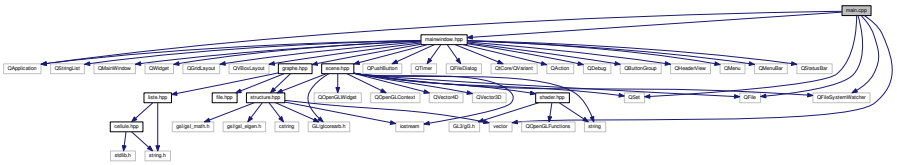
Définition dans le fichier graphe.hpp.

5.3 Référence du fichier main.cpp

Programme principale.

```
#include <QApplication>
#include <QSet>
#include <QFile>
#include <QFileSystemWatcher>
#include <vector>
#include "mainwindow.hpp"
```

Graphe des dépendances par inclusion de main.cpp :



Fonctions

— int **main** (int argc, char ∗argv[])

5.3.1 Description détaillée

Programme principale.

Définition dans le fichier main.cpp.

5.4 Référence du fichier mainwindow.cpp

Implementation de mainwindow.hpp.

```
#include "mainwindow.hpp"
```

Graphe des dépendances par inclusion de mainwindow.cpp :

5.4.1 Description détaillée

Implementation de mainwindow.hpp.

Définition dans le fichier mainwindow.cpp.

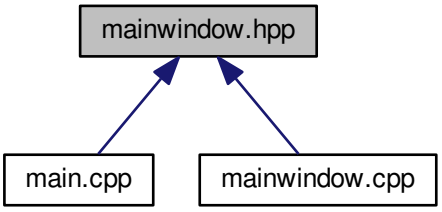
5.5 Référence du fichier mainwindow.hpp

Gère la fenetre principale.

```
#include <QMainWindow>
#include <QWidget>
#include <QGridLayout>
#include <QVBoxLayout>
#include <QPushButton>
#include <QTimer>
#include <QFileDialog>
#include <QtCore/QVariant>
#include <QAction>
#include <QDebug>
#include <QApplication>
#include <QButtonGroup>
#include <QHeaderView>
#include <QMenu>
#include <QMenuBar>
#include <QStatusBar>
#include <QStringList>
#include "scene.hpp"
#include "graphe.hpp"
```

Graphe des dépendances par inclusion de mainwindow.hpp :

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



Classes

— class MainWindow
classe gerant le placement et interactions des principaux widgets dans la fenetre principale

5.5.1 Description détaillée

Gère la fenetre principale.

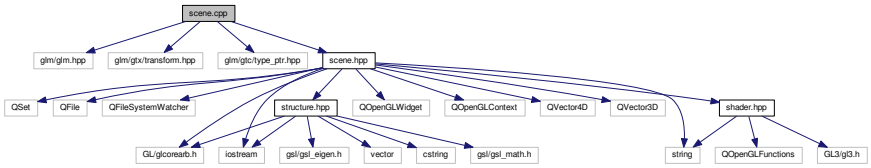
Définition dans le fichier mainwindow.hpp.

5.6 Référence du fichier scene.cpp

Implementation de scene.hpp.

```
#include <glm/glm.hpp>
#include <glm/gtx/transform.hpp>
#include <glm/gtc/type_ptr.hpp>
#include "scene.hpp"
```

Graphe des dépendances par inclusion de scene.cpp :



5.6.1 Description détaillée

Implementation de scene.hpp.

Définition dans le fichier scene.cpp.

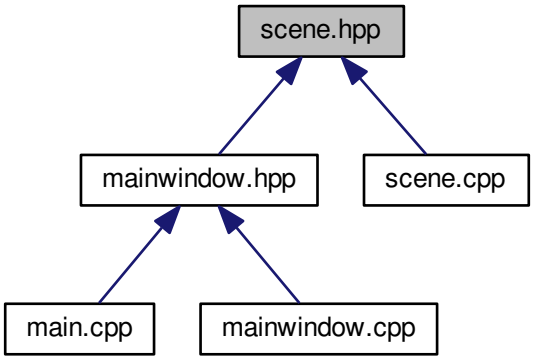
5.7 Référence du fichier scene.hpp

Gère le contexte OpenGL.

```
#include <QSet>
#include <QFile>
#include <QFileSystemWatcher>
#include <GL/glcorearb.h>
#include <QOpenGLWidget>
#include <QOpenGLContext>
#include <QVector4D>
#include <QVector3D>
#include <iostream>
#include <string>
#include "shader.hpp"
#include "structure.hpp"
```

Graphe des dépendances par inclusion de scene.hpp :

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



Classes

— class SceneGL
classe gerant le contexte OpenGL de l'application

5.7.1 Description détaillée

Gère le contexte OpenGL.

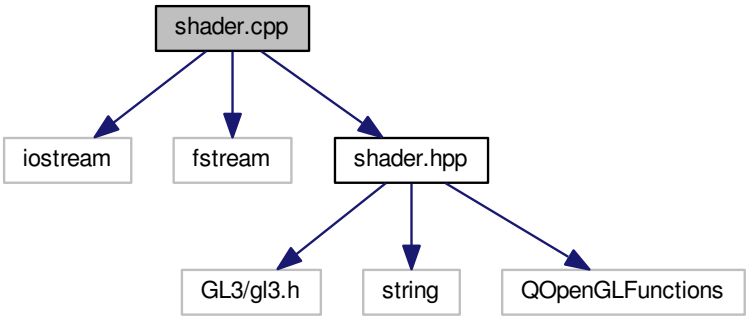
Définition dans le fichier scene.hpp.

5.8 Référence du fichier shader.cpp

Implementation de shader.hpp.

```
#include <iostream>
#include <fstream>
#include "shader.hpp"
```

Graphe des dépendances par inclusion de shader.cpp :



5.8.1 Description détaillée

Implementation de shader.hpp.

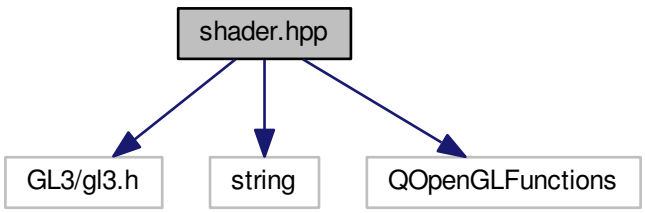
Définition dans le fichier shader.cpp.

5.9 Référence du fichier shader.hpp

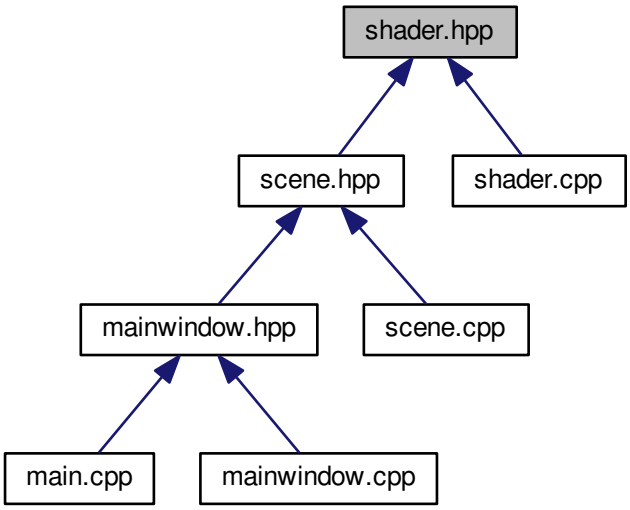
Gère les shaders.

```
#include <GL3/gl3.h>
#include <string>
#include <QOpenGLFunctions>
```

Graphe des dépendances par inclusion de shader.hpp :



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



Classes

— class Shader
Classe gerant la compilation et verrouillage du vertex et fragment shaders.

Macros

— #define **GL3_PROTOTYPES** 1

5.9.1 Description détaillée

Gère les shaders.

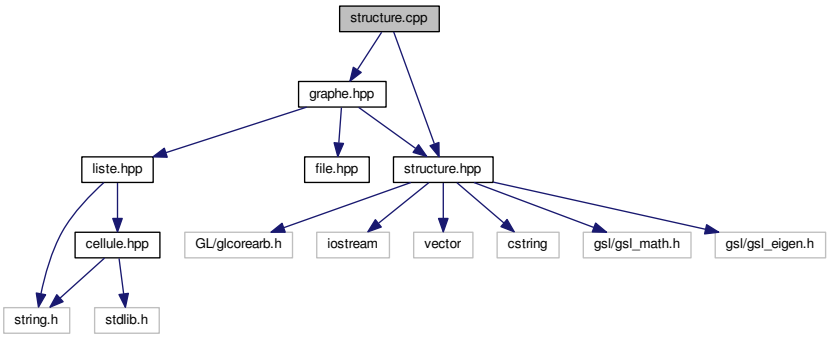
Définition dans le fichier shader.hpp.

5.10 Référence du fichier structure.cpp

Implementation de structure.hpp.

```
#include "graphe.hpp"
#include "structure.hpp"
```

Graphe des dépendances par inclusion de structure.cpp :



5.10.1 Description détaillée

Implementation de structure.hpp.

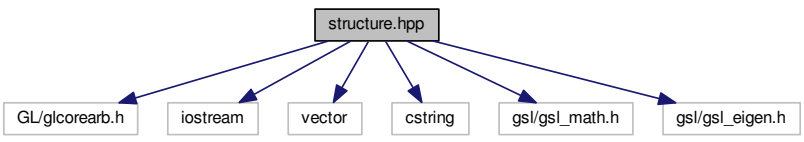
Définition dans le fichier structure.cpp.

5.11 Référence du fichier structure.hpp

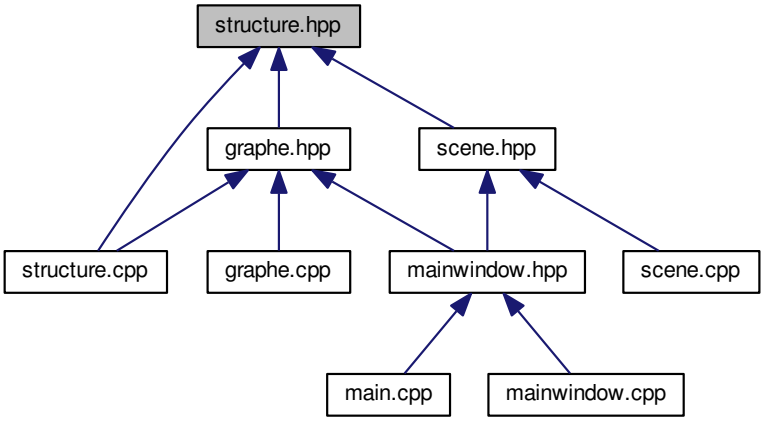
Gère les structures de données.

```
#include <GL/glcorearb.h>
#include <iostream>
#include <vector>
#include <cstring>
#include <gsl/gsl_math.h>
#include <gsl/gsl_eigen.h>
```

Graphe des dépendances par inclusion de structure.hpp :



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



Classes

— struct SceneVertex
structure d'un vertex incluant sa position et couleur
— class Structure
Classe gerant les structures de données chargées dans l'IHM.

5.11.1 Description détaillée

Gère les structures de données.

Définition dans le fichier structure.hpp.

Index