# Visualiseur d'algorithme

Généré par Doxygen 1.8.8

Dimanche 17 Mai 2015 19 :48 :56

# 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

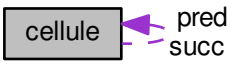
Liste de tous les fichiers documentés avec une brève description :

# 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 structure DataNode

### Attributs publics

- QVector3D **_lumiere_diffuse**
- std : :vector< std : :pair< GLuint, ColGeom > > **_list_vao_mesh**
- QFont : :StyleHint **_style_font**
- QColor **_color_font**

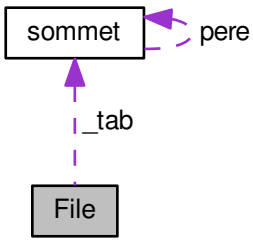
### 4.2.1 Description détaillée

Définition à la ligne 28 du fichier scene.hpp.

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

## 4.3 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**

### 4.3.1 Description détaillée

Définition à la ligne 14 du fichier file.hpp.

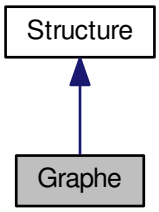
La documentation de cette classe a été générée à partir des fichiers suivants :
- file.hpp
- file.cpp

## 4.4 Référence de la classe Graphe

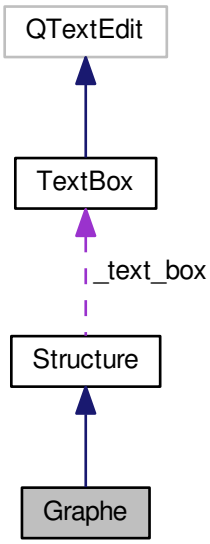
Classe gerant la structure de donnée de graphe.

`#include <graphe.hpp>`

Graphe d'héritage de Graphe :




Graphe de collaboration de Graphe :




## Fonctions membres publiques

- Graphe (std : :string path, TextBox ∗textbox)
  *Constructeur de la structure de graphe.*
- ∼Graphe ()
  *Destructeur.*
- void charger ()
  *Chargement du graphe.*
- void compute_scale ()
  *Calcul du scale.*
- 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_matrice_laplace ()
*Affichage matrice laplace.*
— void afficher_chemin (sommet_t *i, sommet_t *j)
*Affichage chemin.*

### Membres hérités additionnels

### 4.4.1 Description détaillée

Classe gerant la structure de donnée de graphe.

Définition à la ligne 41 du fichier graphe.hpp.

### 4.4.2 Documentation des constructeurs et destructeur

#### 4.4.2.1 Graphe : :Graphe ( std : :string *path,* TextBox * *textbox* )

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 |
| in | *textbox* | widget permettant l'affichage de texte |

Définition à la ligne 18 du fichier graphe.cpp.

### 4.4.3 Documentation des fonctions membres

#### 4.4.3.1 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.

#### 4.4.3.2 void Graphe : :afficher_listes_adjacences ( liste_adjacence_t * *l* )

Affichage liste d'adjacences.

TODO : utiliser operator<<

Définition à la ligne 409 du fichier graphe.cpp.

#### 4.4.3.3 void Graphe : :afficher_matrice_adjacences ( matrice_adjacence_t * *l* )

Affichage matrice d'adjacences.

TODO : utiliser operator<<

Définition à la ligne 430 du fichier graphe.cpp.

#### 4.4.3.4 void Graphe : :afficher_matrice_laplace ( )

Affichage matrice laplace.

TODO : utiliser operator<<

Définition à la ligne 471 du fichier graphe.cpp.

#### 4.4.3.5 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.

#### 4.4.3.6 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 46 du fichier graphe.cpp.

#### 4.4.3.7 void Graphe : :compute_scale ( ) [inline]

Calcul du scale.

Calcul du scale pour le dessin des mesh representant les noeuds

Définition à la ligne 122 du fichier graphe.cpp.

#### 4.4.3.8 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 641 du fichier graphe.cpp.

#### 4.4.3.9 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.

#### 4.4.3.10 parcours_t * Graphe : :parcours_profondeur ( )

Parcours en profondeur.

Algorithme DFS, parcours en profondeur sur le graphe courant

**Paramètres**

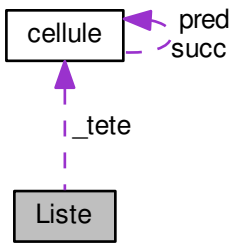
| | | |
|---|---|---|
| out | *parcours* | en largeur resultant |

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

## 4.5 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**

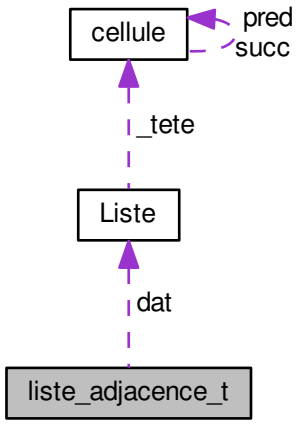
### 4.5.1 Description détaillée

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

## 4.6 Référence de la structure liste_adjacence_t

Graphe de collaboration de liste_adjacence_t :




### Attributs publics

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

### 4.6.1 Description détaillée

Définition à la ligne 15 du fichier graphe.hpp.

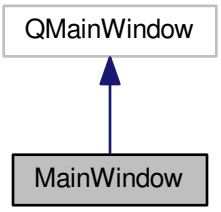
La documentation de cette structure a été générée à partir du fichier suivant :
- graphe.hpp

## 4.7 Référence de la classe MainWindow

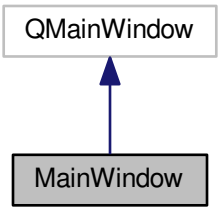
classe gerant le placement et interactions des principaux widgets dans la fenetre principale

```
#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.*

## 4.7.1 Description détaillée

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

Définition à la ligne 39 du fichier mainwindow.hpp.

## 4.7.2 Documentation des constructeurs et destructeur

### 4.7.2.1 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 8 du fichier mainwindow.cpp.

### 4.7.3 Documentation des fonctions membres

#### 4.7.3.1 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 68 du fichier mainwindow.cpp.

#### 4.7.3.2 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 118 du fichier mainwindow.cpp.

#### 4.7.3.3 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 31 du fichier mainwindow.cpp.

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

## 4.8 Référence de la structure matrice_adjacence_t

### Attributs publics

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

### 4.8.1 Description détaillée

Définition à la ligne 27 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 matrice_laplace_t

### Attributs publics

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

#### 4.9.1 Description détaillée

Définition à la ligne 21 du fichier graphe.hpp.

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

## 4.10 Référence de la structure param_lux

### Attributs publics

— glm : :vec4 **diffuse_intensity**
— glm : :vec4 **ambient_intensity**
— glm : :vec4 **light_direction**

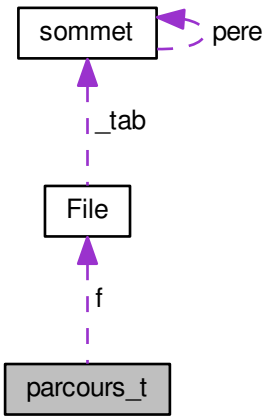
#### 4.10.1 Description détaillée

Définition à la ligne 25 du fichier scene.cpp.

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

## 4.11 Référence de la structure parcours_t

Graphe de collaboration de parcours_t :



### Attributs publics

— File ∗ **f**

### 4.11.1 Description détaillée

Définition à la ligne 33 du fichier graphe.hpp.

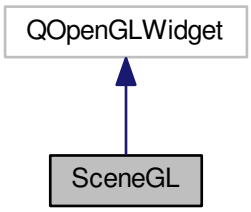
La documentation de cette structure a été générée à partir du fichier suivant :
— graphe.hpp

## 4.12 Référence de la classe SceneGL

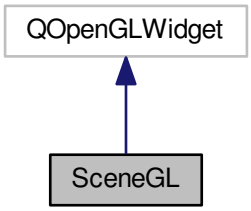
classe gerant le contexte OpenGL de l'application

```
#include <scene.hpp>
```

Graphe d'héritage de SceneGL :




Graphe de collaboration de SceneGL :




### Fonctions membres publiques

— SceneGL (QWidget ∗parent=0, Structure ∗str=nullptr, Qt : :WindowFlags f=0)
*Constructeur de la scene GL.*
— ∼SceneGL ()
*Destructeur.*

— void initializeGL ()
*Initialise le contexte OpenGL.*
— void paintGL ()
*Fonction qui est appelé quand le widget a besoin d'etre repaint.*
— void cleanupGL ()
*Fonction de formattage.*
— void resizeGL (int w, int h)
*Fonction de redimensionnement.*
— Structure * get_structure ()
*Getters de la structure.*
— void charger_contenu_graphique ()
*Charger contenu graphique.*
— void mousePressEvent (QMouseEvent *event) override
*handler evenement clique souris*
— void mouseMoveEvent (QMouseEvent *event) override
*handler evenement mouvement souris*
— void wheelEvent (QWheelEvent *event) override
*handler evenement scroll souris*

### 4.12.1 Description détaillée

classe gerant le contexte OpenGL de l'application

Définition à la ligne 45 du fichier scene.hpp.

### 4.12.2 Documentation des constructeurs et destructeur

#### 4.12.2.1 SceneGL : :SceneGL ( 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` | *parent* | le widget parent où sera contenu le contexte OpenGL |
| `in` | *str* | la structure |
| `in` | *flags* | flags de widget |

Définition à la ligne 49 du fichier scene.cpp.

#### 4.12.2.2 SceneGL : :~SceneGL ( )

Destructeur.

Destructeur

Définition à la ligne 590 du fichier scene.cpp.

### 4.12.3 Documentation des fonctions membres

#### 4.12.3.1 void SceneGL : :charger_contenu_graphique ( )

Charger contenu graphique.

Charge le contenu graphique à afficher dans le contexte OpenGL

Définition à la ligne 175 du fichier scene.cpp.

#### 4.12.3.2 void SceneGL : :cleanupGL ( )

Fonction de formattage.

Cette fonction supprime les identifiants de shaders et vide les buffers

Définition à la ligne 399 du fichier scene.cpp.

#### 4.12.3.3 Structure ∗ SceneGL : :get_structure ( )

Getters de la structure.

Getters de la structure chargé dans la scene OpenGL courante

Définition à la ligne 585 du fichier scene.cpp.

#### 4.12.3.4 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 89 du fichier scene.cpp.

#### 4.12.3.5 void SceneGL : :mouseMoveEvent ( QMouseEvent ∗ *event* ) [override]

handler evenement mouvement souris

handler appelé lors d'un mouvement de la souris sur le widget courant

Définition à la ligne 609 du fichier scene.cpp.

#### 4.12.3.6 void SceneGL : :mousePressEvent ( QMouseEvent ∗ *event* ) [override]

handler evenement clique souris

handler appelé lors d'un clique sur le widget courant

Définition à la ligne 603 du fichier scene.cpp.

#### 4.12.3.7 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 429 du fichier scene.cpp.

#### 4.12.3.8 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 424 du fichier scene.cpp.

#### 4.12.3.9 void SceneGL : :wheelEvent ( QWheelEvent * *event* ) [override]

handler evenement scroll souris

handler appelé lors d'un scroll de la souris sur le widget courant

Définition à la ligne 644 du fichier scene.cpp.

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

## 4.13 Référence de la structure SceneVertex

structure d'un vertex incluant sa position et couleur

`#include <structure.hpp>`

### Attributs publics

- float **Position** [PositionSize]
- float **Normal** [NormalSize]

### Attributs publics statiques

- static const int **PositionSize** = 3
- static const int **NormalSize** = 3

### 4.13.1 Description détaillée

structure d'un vertex incluant sa position et couleur

Définition à la ligne 32 du fichier structure.hpp.

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

## 4.14 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, bool text)
  *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_normal** ()
- 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_normal** (GLint)
- void **set_id_shader_program** (GLint)
- void **set_id_vertex_shader** (GLint)
- void **set_id_fragme_shader** (GLint)

### 4.14.1 Description détaillée

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

Définition à la ligne 22 du fichier shader.hpp.

### 4.14.2 Documentation des constructeurs et destructeur

#### 4.14.2.1 Shader : :Shader ( std : :string *ver_shad,* std : :string *fra_shad,* bool *text* )

Constructeur shader.

Constructeur shaders

**Paramètres**

| | | |
|---|---|---|
| in | *ver_shad* | le vertex shaders |
| in | *fra_shad* | le fragment shaders |
| in | *fra_shad* | gestion du texte ? Si oui alors on gere les textures sinon on gere les normals pour la lumiere/ombre |

Définition à la ligne 10 du fichier shader.cpp.

#### 4.14.2.2 Shader : :~Shader ( )

Destructeur shader.

Destructeur shader

Définition à la ligne 44 du fichier shader.cpp.

### 4.14.3 Documentation des fonctions membres

#### 4.14.3.1 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 53 du fichier shader.cpp.

#### 4.14.3.2 void Shader : :del ( )

Fonction de formattage.

Fonction de formattage

Définition à la ligne 178 du fichier shader.cpp.

#### 4.14.3.3 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 142 du fichier shader.cpp.

#### 4.14.3.4 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 106 du fichier shader.cpp.

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

## 4.15 Référence de la structure sommet

Graphe de collaboration de sommet :

### Attributs publics

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

### 4.15.1 Description détaillée

Définition à la ligne 6 du fichier file.hpp.

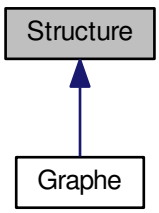
La documentation de cette structure a été générée à partir du fichier suivant :
- file.hpp

## 4.16 Référence de la classe Structure

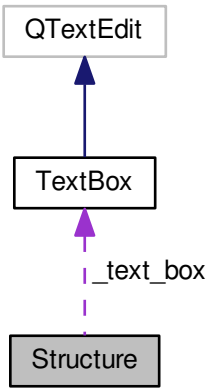
Classe gerant les structures de données chargées dans l'IHM.

`#include <structure.hpp>`

Graphe d'héritage de Structure :



Graphe de collaboration de Structure :




### Fonctions membres publiques

- Structure (std : :string nom_fichier, TextBox ∗textbox)
  *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< GLuint > ∗ **get_indices** ()
- std : :vector< SceneVertex > ∗ **get_vertices** ()
- std : :vector< Etat : :Noeud > ∗ **get_vertex_to_etat** ()
- double **get_scale** ()

### Attributs protégés

- const char ∗ **_path_fichier**
- std : :vector< SceneVertex > **_vertices**
- bool **_est_init**
- std : :vector< GLuint > **_indices**
- std : :vector< Etat : :Noeud > **_vertex_to_etat**
- double **_scale**
- TextBox ∗ **_text_box**

#### 4.16.1 Description détaillée

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

Définition à la ligne 50 du fichier structure.hpp.

#### 4.16.2 Documentation des constructeurs et destructeur

#### 4.16.2.1 Structure : :Structure ( std : :string *nom_fichier,* TextBox * *textbox* )

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 |
| in | *textbox* | widget permettant l'affichage de texte |

Définition à la ligne 10 du fichier structure.cpp.

### 4.16.3 Documentation des fonctions membres

#### 4.16.3.1 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.

#### 4.16.3.2 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 17 du fichier structure.cpp.

#### 4.16.3.3 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 42 du fichier structure.cpp.

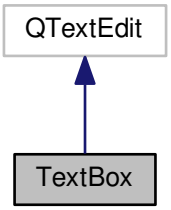
La documentation de cette classe a été générée à partir des fichiers suivants :
- structure.hpp
- structure.cpp

## 4.17 Référence de la classe TextBox

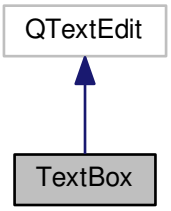
classe gerant l'affichage de texte durant l'execution

`#include <textbox.hpp>`

Graphe d'héritage de TextBox :




Graphe de collaboration de TextBox :




### Fonctions membres publiques

- TextBox (QWidget *parent=0)
  *Constructeur de la boite de texte.*
- void afficher ()
  *Met à jour le contenu texte.*
- void ajouter (std : :string str, int num_ligne)
  *Ajoute du texte.*
- void remplacer (std : :string str, int num_ligne, int npos)
  *Remplace le texte.*
- int newline ()
  *Retourne un numero d'une nouvelle ligne.*
- void clear ()
  *efface le buffer actuel*

#### 4.17.1 Description détaillée

classe gerant l'affichage de texte durant l'execution

Définition à la ligne 17 du fichier textbox.hpp.

#### 4.17.2 Documentation des constructeurs et destructeur

#### 4.17.2.1 TextBox : :TextBox ( QWidget * *parent* = 0 )

Constructeur de la boite de texte.

il s'agit de text box, on y affichera des informations durant l'execution

**Paramètres**

| | | |
|---|---|---|
| in | *parent* | le widget parent |

Définition à la ligne 7 du fichier textbox.cpp.

### 4.17.3 Documentation des fonctions membres

#### 4.17.3.1 void TextBox : :afficher ( )

Met à jour le contenu texte.

est appelé de manière reguliere par le main thread

Définition à la ligne 29 du fichier textbox.cpp.

#### 4.17.3.2 void TextBox : :ajouter ( std : :string *str*, int *num_ligne* )

Ajoute du texte.

Ajoute du texte à la suite de la ligne

**Paramètres**

| | | |
|---|---|---|
| in | *str* | texte à ajouter |
| in | *num_ligne* | numero de la ligne |

Définition à la ligne 51 du fichier textbox.cpp.

#### 4.17.3.3 int TextBox : :newline ( )

Retourne un numero d'une nouvelle ligne.

Permet d'avoir un checkpoint vers une nouvelle ligne pour pouvoir modifier cette ligne grace au numero renvoyé

**Paramètres**

| | | |
|---|---|---|
| out | *numero* | de ligne |

Définition à la ligne 63 du fichier textbox.cpp.

#### 4.17.3.4 void TextBox : :remplacer ( std : :string *str*, int *num_ligne*, int *npos* )

Remplace le texte.

Remplace le texte passé en parametre

**Paramètres**

| | | |
|---|---|---|
| in | *str* | texte à remplacer |
| in | *num_ligne* | numero de la ligne |
| in | *npos* | position du premier caractere à remplacer |

Définition à la ligne 56 du fichier textbox.cpp.

La documentation de cette classe a été générée à partir des fichiers suivants :

- textbox.hpp
- textbox.cpp

# Chapitre 5

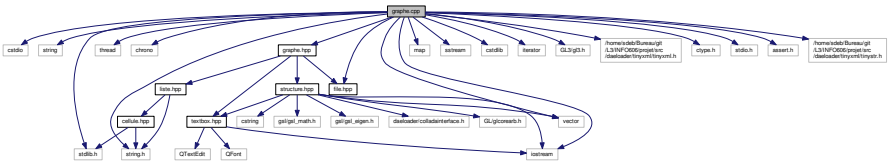
# Documentation des fichiers

## 5.1 Référence du fichier graphe.cpp

Implementation de graphe.hpp.

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

Graphe des dépendances par inclusion de graphe.cpp :




### 5.1.1 Description détaillée

Implementation de graphe.hpp.

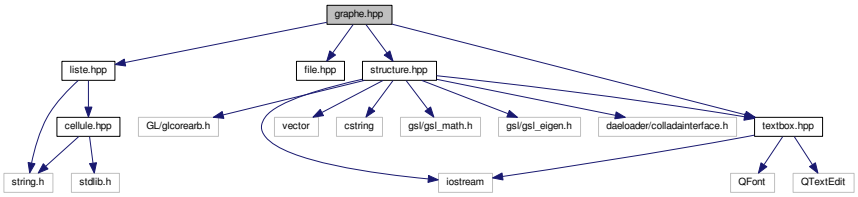
Définition dans le fichier graphe.cpp.

## 5.2 Référence du fichier graphe.hpp

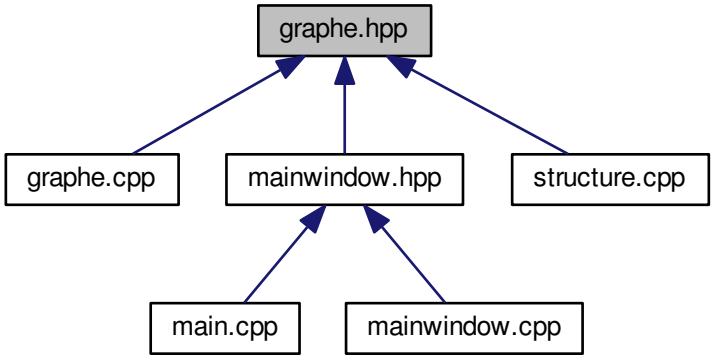
Gère la structure de donnée de graphe.

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

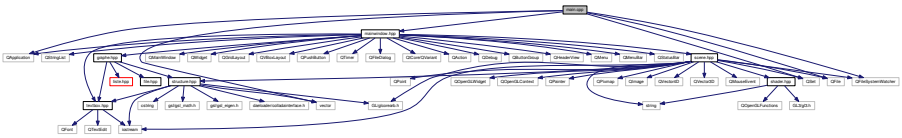
Définition dans le fichier graphe.hpp.

## 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.

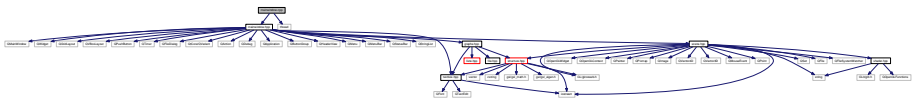
Définition dans le fichier main.cpp.

## 5.4 Référence du fichier mainwindow.cpp

Implementation de mainwindow.hpp.

```
#include "mainwindow.hpp"
#include <thread>
```

Graphe des dépendances par inclusion de mainwindow.cpp :



### 5.4.1 Description détaillée

Implementation de mainwindow.hpp.

Définition dans le fichier mainwindow.cpp.

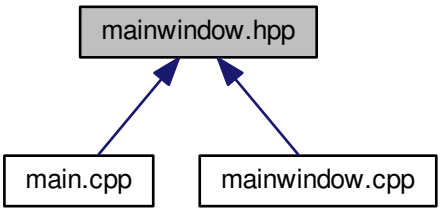
## 5.5 Référence du fichier mainwindow.hpp

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 "textbox.hpp"
#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.

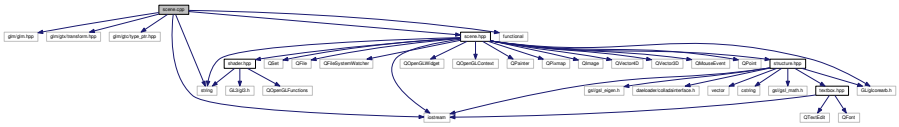
Définition dans le fichier mainwindow.hpp.

## 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 <iostream>
#include <string>
#include <functional>
#include "scene.hpp"
```

Graphe des dépendances par inclusion de scene.cpp :



### Classes

— struct param_lux

### Macros

— #define **GLM_FORCE_RADIANS**
— #define **AFFDEBUG** 1
— #define **WARN**(MSG)
— #define **ISOK**(MSG)

### Variables

— double **angleY** = 0.0
— double **angleZ** = 0.0
— int **old_posx** = 0
— int **old_posy** = 0
— bool **_hold** = false

### 5.6.1 Description détaillée

Implementation de scene.hpp.

Définition dans le fichier scene.cpp.

### 5.6.2 Documentation des macros

#### 5.6.2.1 #define ISOK( *MSG* )

**Valeur :**

```
printf("\n\033[42;01m"                                  \
                        "[_POSITION_]\t[%s:%d] -> [%s]\n"              \
                        "[_DEBUG_MSG_]\t[%s]"                          \
                        "\033[00m\n", __FILE__, __LINE__, __func__, MSG);
```

Définition à la ligne 36 du fichier scene.cpp.

#### 5.6.2.2 #define WARN( *MSG* )

**Valeur :**

```
printf("\n\033[41;01m"                                  \
                        "[_POSITION_]\t[%s:%d] -> [%s]\n"              \
                        "[_DEBUG_MSG_]\t[%s]"                          \
                        "\033[00m\n", __FILE__, __LINE__, __func__, MSG);
```

Définition à la ligne 31 du fichier scene.cpp.

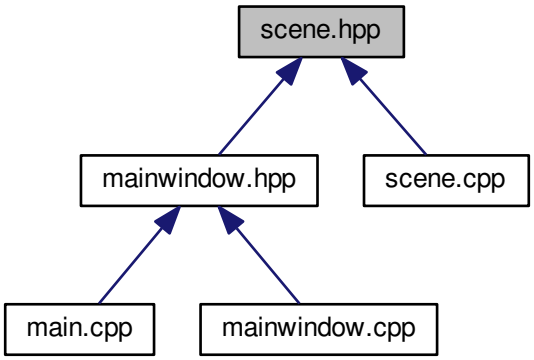
## 5.7 Référence du fichier scene.hpp

Gère le contexte OpenGL.

```
#include <QSet>
#include <QFile>
#include <QFileSystemWatcher>
#include <GL/glcorearb.h>
#include <QOpenGLWidget>
#include <QOpenGLContext>
#include <QPainter>
#include <QPixmap>
#include <QImage>
#include <QVector4D>
#include <QVector3D>
#include <QMouseEvent>
#include <QPoint>
#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

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

### 5.7.1 Description détaillée

Gère le contexte OpenGL.

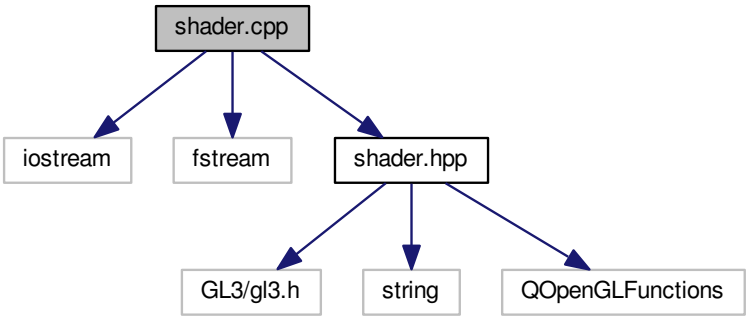
Définition dans le fichier scene.hpp.

## 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.

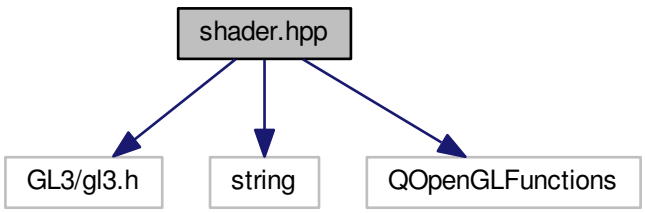
Définition dans le fichier shader.cpp.

## 5.9 Référence du fichier shader.hpp

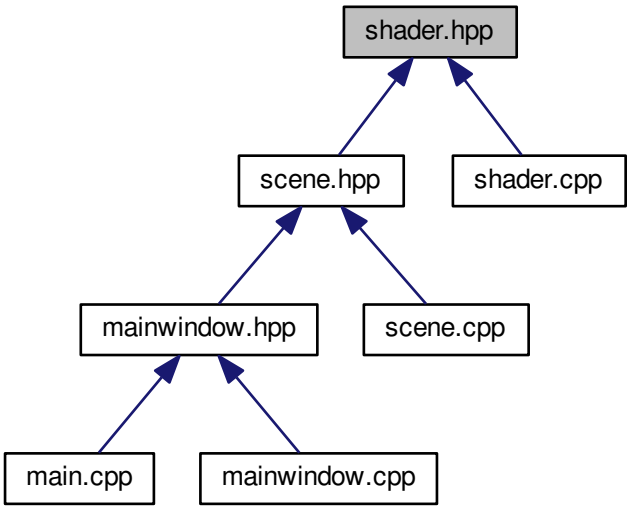
Gère les shaders.

```
#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.

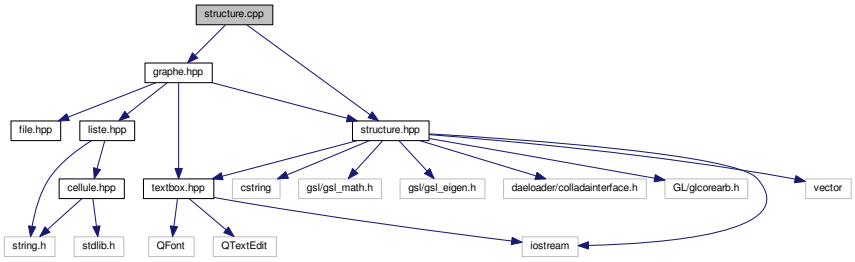
Définition dans le fichier shader.hpp.

## 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.

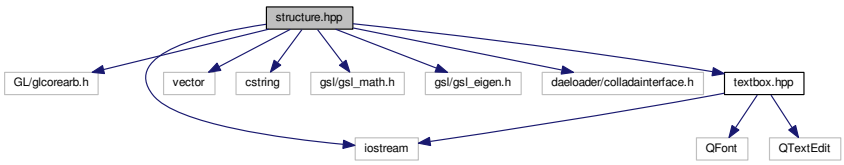
Définition dans le fichier structure.cpp.

## 5.11 Référence du fichier structure.hpp

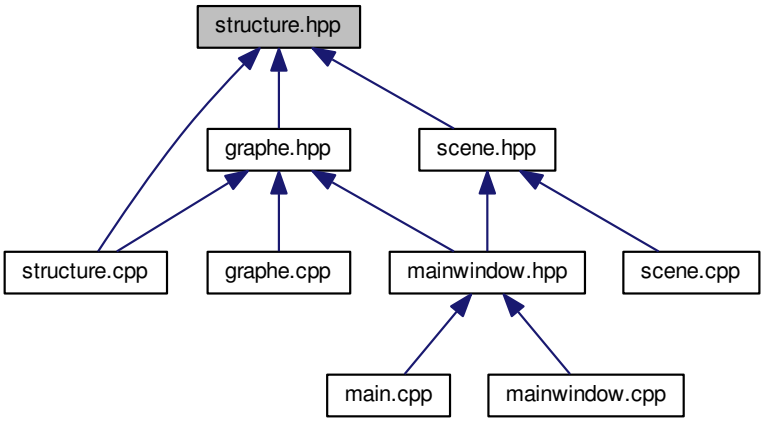
Gère les structures de données.

```
#include <GL/glcorearb.h>
#include <iostream>
#include <vector>
#include <cstring>
#include <gsl/gsl_math.h>
#include <gsl/gsl_eigen.h>
#include "daeloader/colladainterface.h"
#include "textbox.hpp"
```

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.*

### Énumérations

— enum **Noeud** {
**SIMPLE**, **COURANT**, **VISITED**, **SELECT**,
**CHOIX** }

#### 5.11.1 Description détaillée

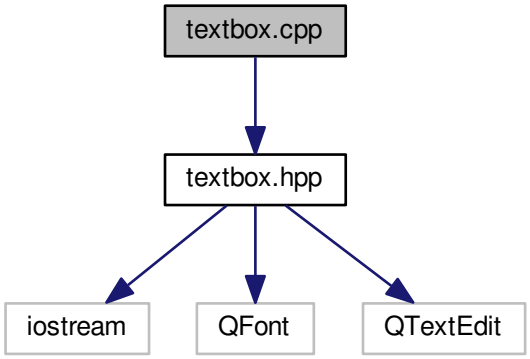
Gère les structures de données.

Définition dans le fichier structure.hpp.

## 5.12 Référence du fichier textbox.cpp

Implementation de textbox.hpp.

```
#include "textbox.hpp"
```
Graphe des dépendances par inclusion de textbox.cpp :




### 5.12.1 Description détaillée

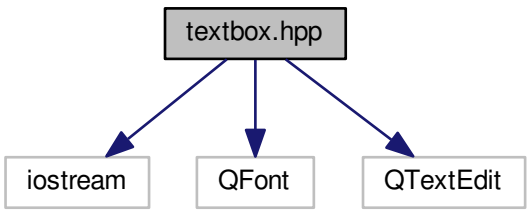
Implementation de textbox.hpp.

Définition dans le fichier textbox.cpp.

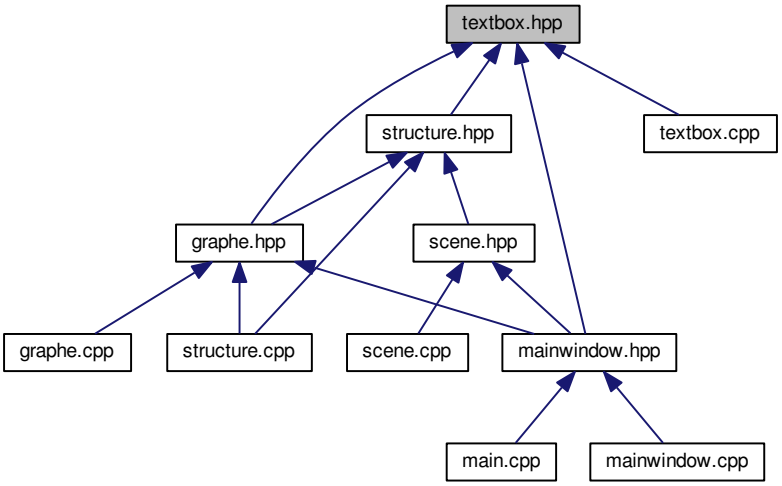
## 5.13 Référence du fichier textbox.hpp

Gère la text box.

```
#include <iostream>
#include <QFont>
#include <QTextEdit>
```
Graphe des dépendances par inclusion de textbox.hpp :



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




## Classes

— class TextBox
*classe gerant l'affichage de texte durant l'execution*

### 5.13.1 Description détaillée

Gère la text box.

Définition dans le fichier textbox.hpp.

# Index