lrt

0.1

Généré par Doxygen 1.7.5.1

Mardi Décembre 6 2011 11 :36 :22

Table des matières

Chapitre 1

Index des classes

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 :

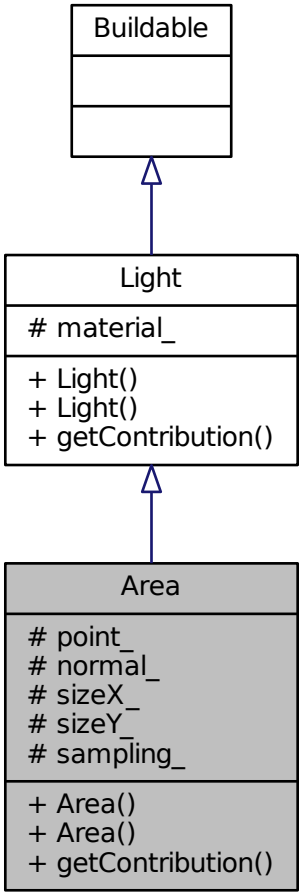
Chapitre 4

Documentation des classes

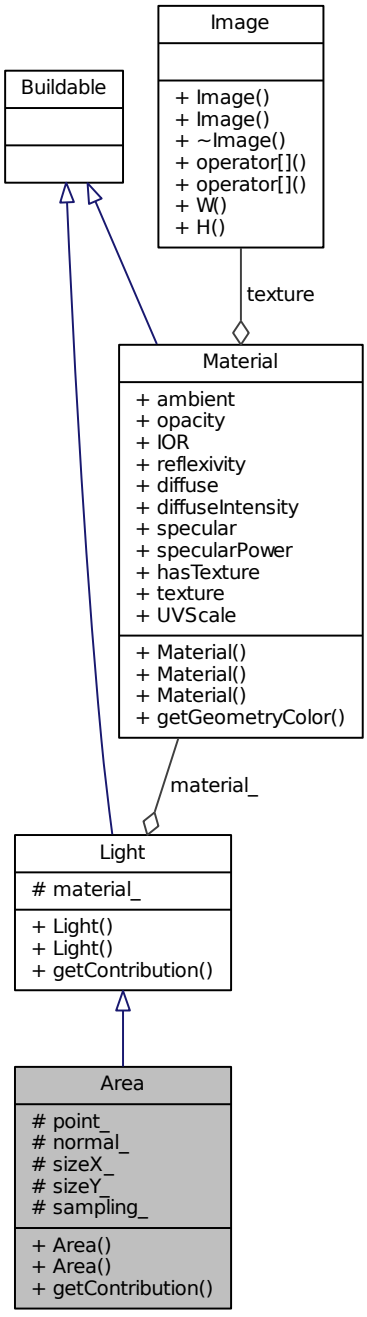
4.1 Référence de la classe Area

`#include <Area.hpp>`

Graphe d'héritage de Area :



Graphe de collaboration de Area :



Fonctions membres publiques

- Area (Vector3d const &point, Vector3d const &normal, double sizeX, double sizeY, unsigned int sampling)
- Area (Vector3d const &point, Vector3d const &normal, double sizeX, double sizeY, unsigned int sampling, Material ∗material)
- Color_d getContribution (Camera ∗camera, std : :vector< Geometry ∗ > const &geometries, HitRecord const &record) const

Attributs protégés

- Vector3d point_
- Vector3d normal_
- double sizeX_
- double sizeY_
- unsigned int sampling_

4.1.1 Description détaillée

Définition à la ligne 11 du fichier Area.hpp.

4.1.2 Documentation des constructeurs et destructeur

4.1.2.1 Area : :Area (Vector3d const & *point,* Vector3d const & *normal,* double *sizeX,* double *sizeY,* unsigned int *sampling*)

Définition à la ligne 11 du fichier Area.cc.

4.1.2.2 Area : :Area (Vector3d const & *point,* Vector3d const & *normal,* double *sizeX,* double *sizeY,* unsigned int *sampling,* Material ∗ *material*)

Définition à la ligne 23 du fichier Area.cc.

4.1.3 Documentation des fonctions membres

4.1.3.1 Color_d Area : :getContribution (Camera ∗ *camera,* std : :vector< Geometry ∗ > const & *geometries,* HitRecord const & *record*) const [virtual]

Renvoie

La contribution, c'est à dire l'ajout de lumière de la source à la lumière totale du point considéré.

Paramètres

camera	La caméra à considérer.
geometries	Listes des géométries de la scène.
record	Enregistrement de l'intersection.

Implémente Light.

Définition à la ligne 49 du fichier Area.cc.

Voici le graphe d'appel pour cette fonction :

4.1.4 Documentation des données membres

4.1.4.1 Vector3d Area : :normal_ [protected]

Définition à la ligne 24 du fichier Area.hpp.

4.1.4.2 Vector3d Area : :point_ [protected]

Définition à la ligne 23 du fichier Area.hpp.

4.1.4.3 unsigned int Area : :sampling_ [protected]

Définition à la ligne 27 du fichier Area.hpp.

4.1.4.4 double Area : :sizeX_ [protected]

Définition à la ligne 25 du fichier Area.hpp.

4.1.4.5 double Area : :sizeY_ [protected]

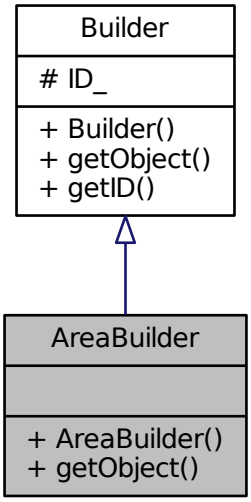
Définition à la ligne 26 du fichier Area.hpp.

La documentation de cette classe a été générée à partir des fichiers suivants :
– src/Lights/Area.hpp
– src/Lights/Area.cc

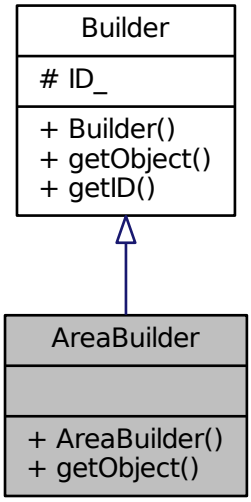
4.2 Référence de la classe AreaBuilder

```
#include <AreaBuilder.hpp>
```

Graphe d'héritage de AreaBuilder :



Graphe de collaboration de AreaBuilder :



Fonctions membres publiques

– AreaBuilder ()
– Buildable ∗ getObject (boost : :property_tree : :ptree pt, Material ∗material) const

4.2.1 Description détaillée

Définition à la ligne 17 du fichier AreaBuilder.hpp.

4.2.2 Documentation des constructeurs et destructeur

4.2.2.1 AreaBuilder : :AreaBuilder ( )

Définition à la ligne 6 du fichier AreaBuilder.cc.

4.2.3 Documentation des fonctions membres

4.2.3.1 Buildable ∗ AreaBuilder : :getObject (boost : :property_tree : :ptree *pt,* Material ∗ *material*) const [virtual]

Paramètres

pt	L'arbre DOM contenant les paramètres du fichier de configuration de la scène,
material	Matériau associé.

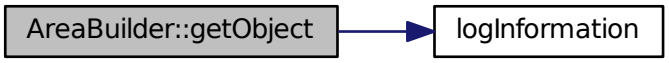
Renvoie

Un pointeur sur l'objet construit.

Implémente Builder.

Définition à la ligne 10 du fichier AreaBuilder.cc.

Voici le graphe d'appel pour cette fonction :

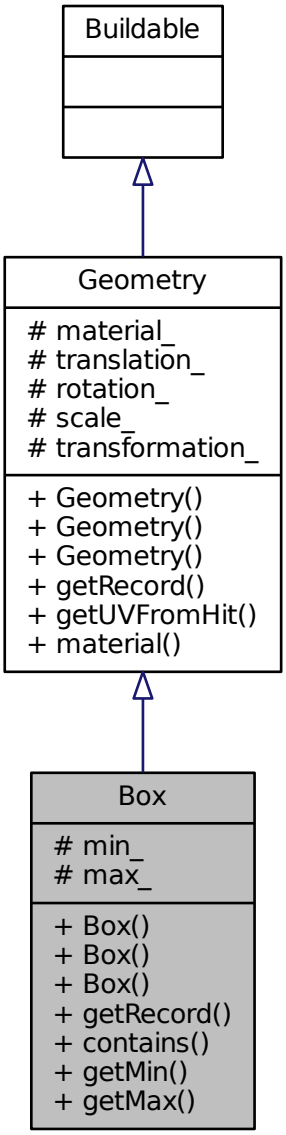


La documentation de cette classe a été générée à partir des fichiers suivants :
– src/Builders/AreaBuilder.hpp
– src/Builders/AreaBuilder.cc

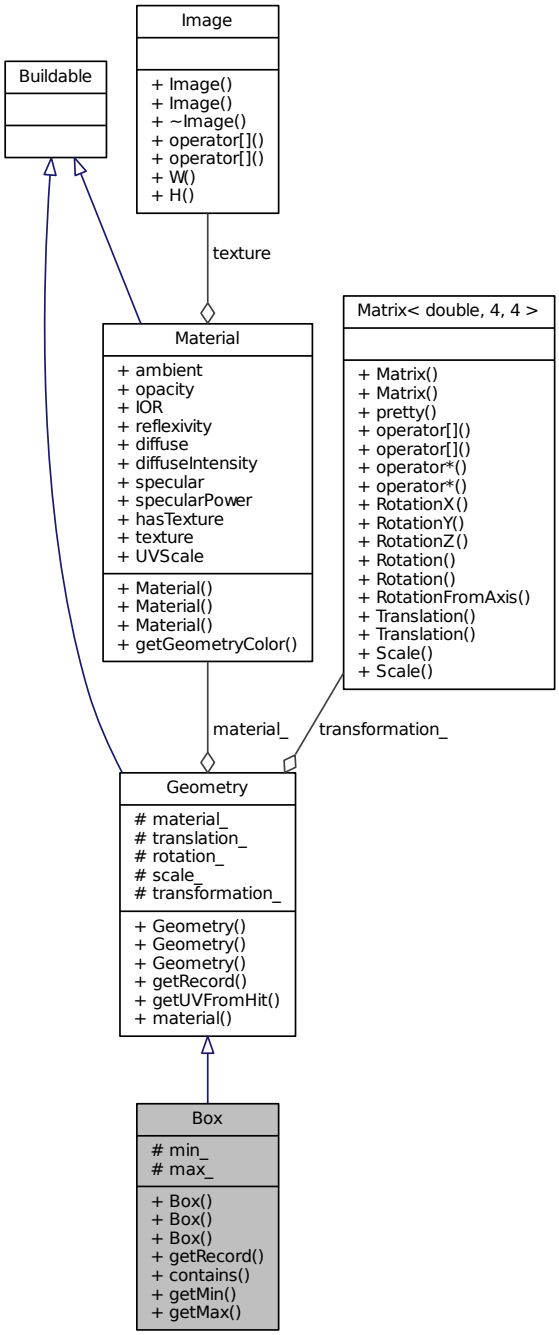
4.3 Référence de la classe Box

```
#include <Box.hpp>
```

Graphe d'héritage de Box :



Graphe de collaboration de Box :



Fonctions membres publiques

- Box ()
- Box (Vector3d const &min, Vector3d const &max)
- Box (Vector3d const &min, Vector3d const &max, Material ∗material)
- HitRecord getRecord (Ray const &ray) const
- bool contains (const Triangle ∗const t) const
- Vector3d getMin () const
- Vector3d getMax () const

Attributs protégés

- Vector3d min_
- Vector3d max_

4.3.1 Description détaillée

Définition à la ligne 8 du fichier Box.hpp.

4.3.2 Documentation des constructeurs et destructeur

4.3.2.1 Box : :Box ( )

Créer une boite dummy

Définition à la ligne 3 du fichier Box.cc.

4.3.2.2 Box : :Box (Vector3d const & *min*, Vector3d const & *max*)

Paramètres

min	Coin inférieur gauche de la boite.
min	Coin supérieur droit de la boite.

Définition à la ligne 9 du fichier Box.cc.

4.3.2.3 Box : :Box (Vector3d const & *min*, Vector3d const & *max*, Material ∗ *material*)

Paramètres

min	Coin inférieur gauche de la boite.
min	Coin supérieur droit de la boite.
material	Matériaux à associer à la géométrie.

Définition à la ligne 15 du fichier Box.cc.

4.3.3 Documentation des fonctions membres

4.3.3.1 bool Box : :contains (const Triangle ∗const *t*) const

Paramètres

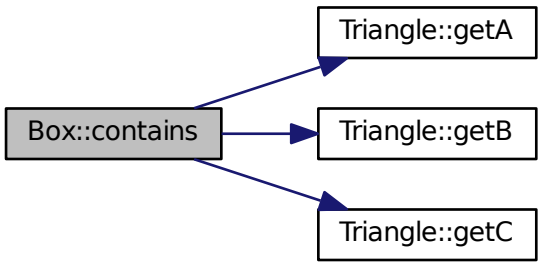
t	Triangle dont il faut vérifier l'appartenance à la boite.

Renvoie

true, si le triangle est géométriquement inclu dans la boite, false, sinon.

Définition à la ligne 48 du fichier Box.cc.

Voici le graphe d'appel pour cette fonction :



4.3.3.2 Vector3d Box : :getMax ( ) const [inline]

Renvoie

Le coin supérieur droitde la boite.

Définition à la ligne 47 du fichier Box.hpp.

4.3.3.3 Vector3d Box : :getMin ( ) const [inline]

Renvoie

Le coin inférieur gauche de la boite.

Définition à la ligne 42 du fichier Box.hpp.

4.3.3.4 HitRecord Box : :getRecord (Ray const & *ray*) const [virtual]

Renvoie

L'enregistrement lié à la collision du rayon *ray* et la géométrie.

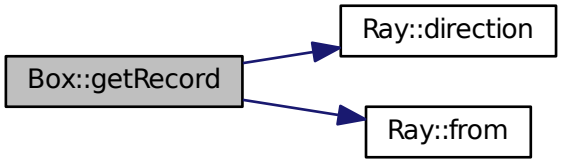
Paramètres

ray	Rayon à intersecter.

Implémente Geometry.

Définition à la ligne 21 du fichier Box.cc.

Voici le graphe d'appel pour cette fonction :



4.3.4 Documentation des données membres

4.3.4.1 Vector3d Box : :max_ [protected]

Définition à la ligne 51 du fichier Box.hpp.

4.3.4.2 Vector3d Box : :min_ [protected]

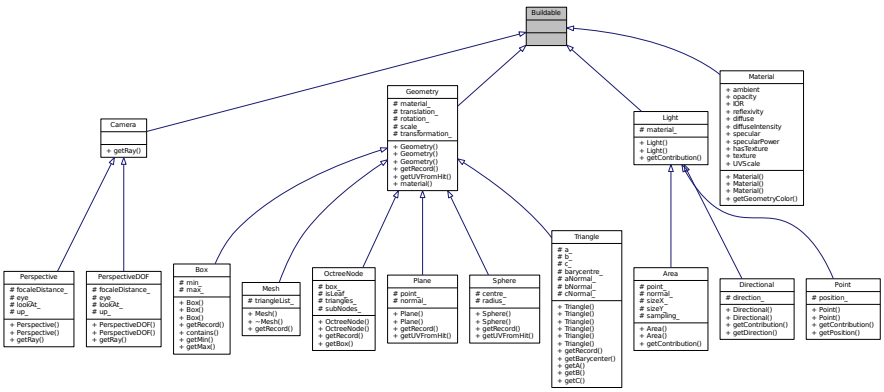
Définition à la ligne 50 du fichier Box.hpp.

La documentation de cette classe a été générée à partir des fichiers suivants :
– src/Geometry/Box.hpp
– src/Geometry/Box.cc

4.4 Référence de la classe Buildable

```
#include <Buildable.hpp>
```

Graphe d'héritage de Buildable :



4.4.1 Description détaillée

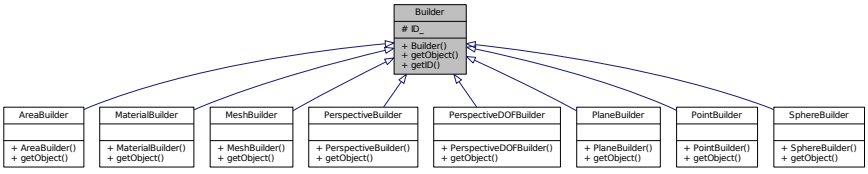
Définition à la ligne 13 du fichier Buildable.hpp.

La documentation de cette classe a été générée à partir du fichier suivant :
– src/Buildable.hpp

4.5 Référence de la classe Builder

`#include <Builder.hpp>`

Graphe d'héritage de Builder :



Fonctions membres publiques

– Builder (std : :string const &ID)
– virtual Buildable ∗ getObject (boost : :property_tree : :ptree pt, Material ∗material) const =0
– std : :string getID () const

Attributs protégés

– std : :string ID_

4.5.1 Description détaillée

Définition à la ligne 18 du fichier Builder.hpp.

4.5.2 Documentation des constructeurs et destructeur

4.5.2.1 Builder : :Builder (std : :string const & *ID*)

Paramètres

ID	Identifiant du type d'objet construit.

Définition à la ligne 3 du fichier Builder.cc.

4.5.3 Documentation des fonctions membres

4.5.3.1 std : :string Builder : :getID ( ) const

Définition à la ligne 7 du fichier Builder.cc.

4.5.3.2 virtual Buildable* Builder : :getObject (boost : :property_tree : :ptree *pt,* Material * *material*) const [pure virtual]

Paramètres

pt	L'arbre DOM contenant les paramètres du fichier de configuration de la scène,
material	Matériau associé.

Renvoie

Un pointeur sur l'objet construit.

Implémenté dans AreaBuilder, MaterialBuilder, MeshBuilder, PerspectiveBuilder, PerspectiveDOFBuilder, PlaneBuilder, PointBuilder, et SphereBuilder.

4.5.4 Documentation des données membres

4.5.4.1 std : :string Builder : :ID_ [protected]

Définition à la ligne 42 du fichier Builder.hpp.

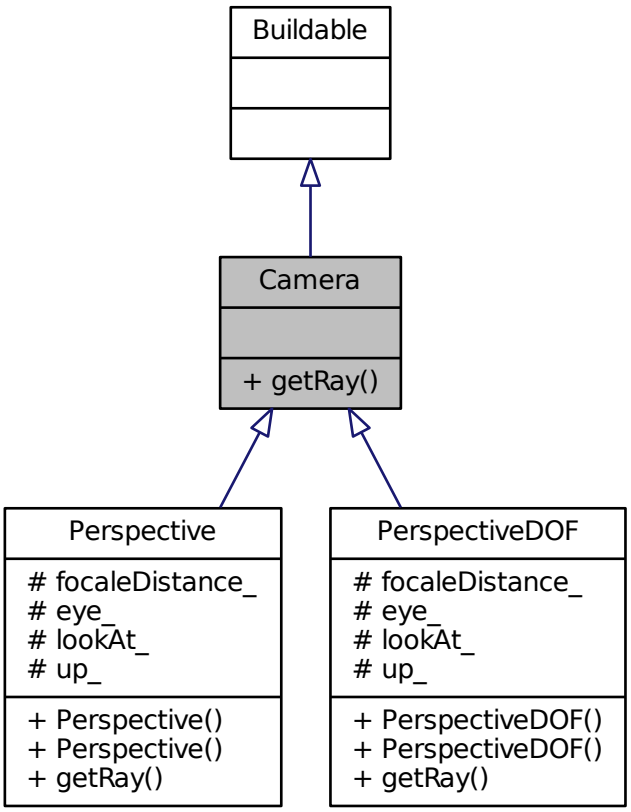
La documentation de cette classe a été générée à partir des fichiers suivants :
– src/Builders/Builder.hpp

– src/Builders/Builder.cc

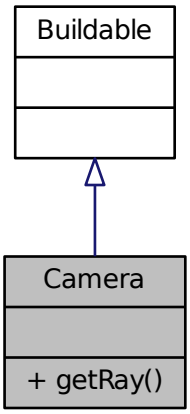
4.6 Référence de la classe Camera

`#include <Camera.hpp>`

Graphe d'héritage de Camera :



Graphe de collaboration de Camera :



Fonctions membres publiques

– virtual std : :vector< Ray > getRay (double u, double v) const =0

4.6.1 Description détaillée

Définition à la ligne 25 du fichier Camera.hpp.

4.6.2 Documentation des fonctions membres

4.6.2.1 virtual std : :vector<Ray> Camera : :getRay (double *u,* double *v*) const [pure virtual]

Renvoie

L'ensemble des rayons associés à la coordonnée (u,v)

Implémenté dans PerspectiveDOF, et Perspective.

La documentation de cette classe a été générée à partir du fichier suivant :
– src/Cameras/Camera.hpp

4.7 Référence du modèle de la classe Color< P >

`#include <Color.hpp>`

Fonctions membres publiques

- Color ()
- Color (P const &v)
- Color (P const &R, P const &G, P const &B)
- Color (Color< P > const &C)
- string pretty () const
- Color< P > Clamped () const
- Color< P > operator+ (Color< P > const &C2) const
- Color< P > operator+ (P const &C2) const
- Color< P > operator- (Color< P > const &C2) const
- Color< P > operator- (P const &C2) const
- Color< P > operator* (Color< P > const &C2) const
- Color< P > operator* (P const &C2) const
- Color< P > operator/ (P const &C2) const
- Color< P > & operator+= (Color< P > const &C2)
- Color< P > & operator*= (Color< P > const &C2)
- Color< P > & operator*= (P const &C2)
- Color< P > & operator[] (unsigned int i)
- Color< P > & operator[] (unsigned int i) const
- P R () const
- P G () const
- P B () const

Amis

- Color< P > operator* (P const &v, Color< P > const &C1)
- ostream & operator<< (ostream &oss, Color< P > const &c)

4.7.1 Description détaillée

template<typename P>class Color< P >

Définition à la ligne 24 du fichier Color.hpp.

4.7.2 Documentation des constructeurs et destructeur

4.7.2.1 template<typename P> Color< P > : :Color ( )

Toutes les composantes de la couleur sont à 0.

4.7.2.2 template<typename P> Color< P > : :Color (P const & *v*)

Paramètres

v	Valeur de toutes les composantes.

4.7.2.3 template<typename P> Color< P > : :Color (P const & *R*, P const & *G*, P const & *B*)

Paramètres

R	Valeur de la composante rouge.
G	Valeur de la composante Verte.
B	Valeur de la composante bleue.

4.7.2.4 template<typename P> Color< P > : :Color (Color< P > const & *C*)

4.7.3 Documentation des fonctions membres

4.7.3.1 template<typename P> P Color< P > : :B ( ) const

4.7.3.2 template<typename P> Color<P> Color< P > : :Clamped ( ) const

4.7.3.3 template<typename P> P Color< P > : :G ( ) const

4.7.3.4 template<typename P> Color<P> Color< P > : :operator∗ (Color< P > const & *C2*) const

4.7.3.5 template<typename P> Color<P> Color< P > : :operator∗ (P const & *C2*) const

4.7.3.6 template<typename P> Color<P>& Color< P > : :operator∗= (Color< P > const & *C2*)

4.7.3.7 template<typename P> Color<P>& Color< P > : :operator∗= (P const & *C2*)

4.7.3.8 template<typename P> Color<P> Color< P > : :operator+ (Color< P > const & *C2*) const

4.7.3.9 template<typename P> Color<P> Color< P > : :operator+ (P const & *C2*) const

4.7.3.10 template<typename P> Color<P>& Color< P > : :operator+= (Color< P > const & *C2*)

4.7.3.11 template<typename P> Color<P> Color< P > : :operator- (Color< P > const & *C2*) const

4.7.3.12 template<typename P> Color<P> Color< P > : :operator- (P const & *C2*) const

4.7.3.13 template<typename P> Color<P> Color< P > : :operator/ (P const & *C2*) const

4.7.3.14 template<typename P> Color<P>& Color< P > : :operator[] (unsigned int *i*)

4.7.3.15 template<typename P> Color<P>& Color< P > : :operator[] (unsigned int *i*) const

4.7.3.16 template<typename P> string Color< P > : :pretty ( ) const

4.7.3.17 template<typename P> P Color< P > : :R ( ) const

4.7.4 Documentation des fonctions amies et associées

4.7.4.1 template<typename P> Color<P> operator∗ (P const & *v,* Color< P > const & *C1*) [friend]

Définition à la ligne 62 du fichier Color.hpp.

4.7.4.2 template<typename P> ostream& operator<< (ostream & *oss,* Color< P > const & *c*) [friend]

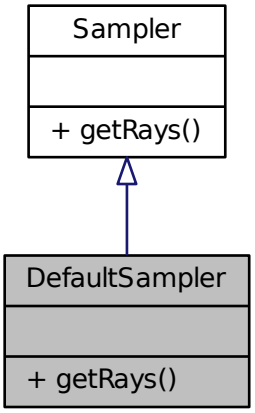
Définition à la ligne 73 du fichier Color.hpp.

La documentation de cette classe a été générée à partir du fichier suivant :
– src/Graphics/Color.hpp

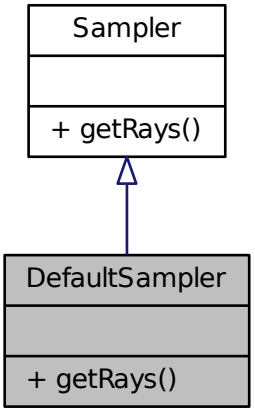
4.8 Référence de la classe DefaultSampler

`#include <DefaultSampler.hpp>`

Graphe d'héritage de DefaultSampler :



Graphe de collaboration de DefaultSampler :



Fonctions membres publiques

– std : :vector< Ray > getRays (Scene const &scene, unsigned int X, unsigned int Y)

4.8.1 Description détaillée

Définition à la ligne 10 du fichier DefaultSampler.hpp.

4.8.2 Documentation des fonctions membres

4.8.2.1 std : :vector< Ray > DefaultSampler : :getRays (Scene const & *scene,* unsigned int *X,* unsigned int *Y*) [virtual]

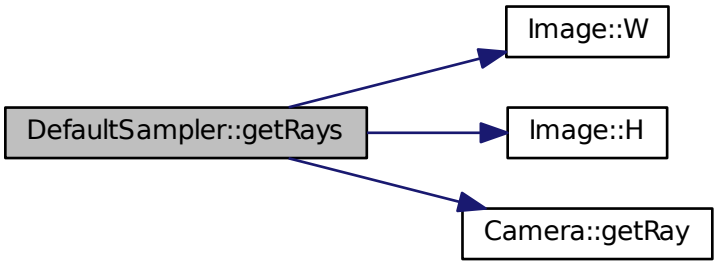
Renvoie

Ensemble des rayons à lancer pour la scène et les coordonnées passés en paramètres.

Implémente Sampler.

Définition à la ligne 5 du fichier DefaultSampler.cc.

Voici le graphe d'appel pour cette fonction :

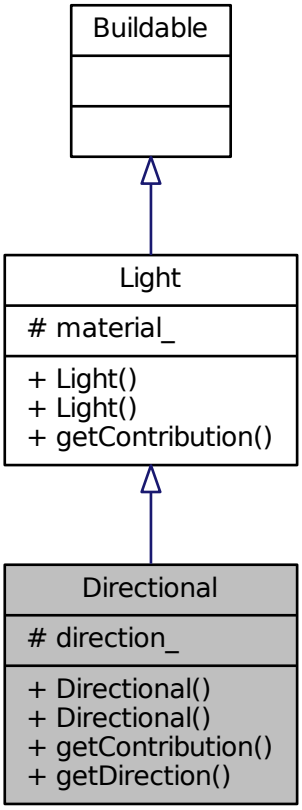


La documentation de cette classe a été générée à partir des fichiers suivants :
– src/Samplers/DefaultSampler.hpp
– src/Samplers/DefaultSampler.cc

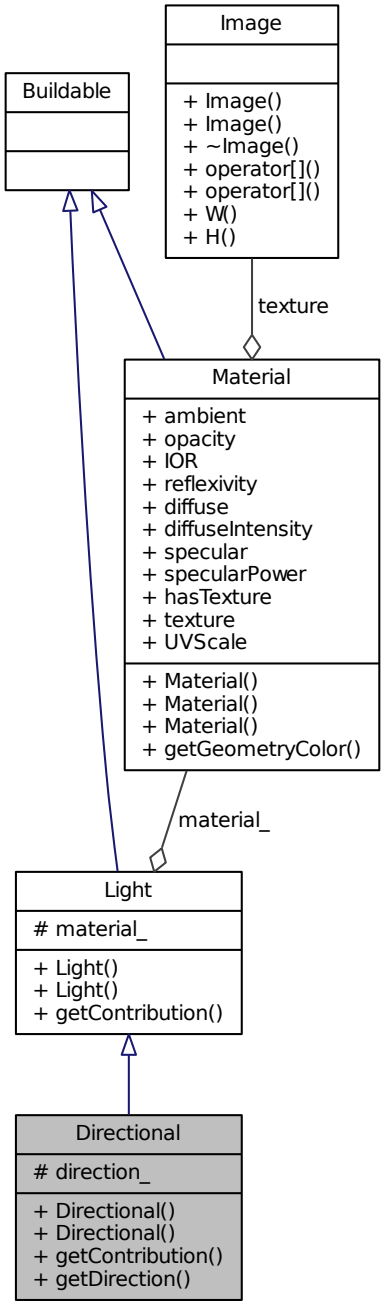
4.9 Référence de la classe Directional

`#include <Directional.hpp>`

Graphe d'héritage de Directional :



Graphe de collaboration de Directional :



Fonctions membres publiques

- Directional (Vector3d const &direction)
- Directional (Vector3d const &direction, Material ∗material)
- Color_d getContribution (Camera ∗camera, std : :vector< Geometry ∗ > const &geometries, HitRecord const &record) const
- Vector3d getDirection () const

Attributs protégés

- Vector3d direction_

4.9.1 Description détaillée

Définition à la ligne 8 du fichier Directional.hpp.

4.9.2 Documentation des constructeurs et destructeur

4.9.2.1 Directional : :Directional (Vector3d const & *direction*)

Définition à la ligne 9 du fichier Directional.cc.

4.9.2.2 Directional : :Directional (Vector3d const & *direction,* Material ∗ *material*)

Définition à la ligne 14 du fichier Directional.cc.

4.9.3 Documentation des fonctions membres

4.9.3.1 Color_d Directional : :getContribution (Camera ∗ *camera,* std : :vector< Geometry ∗ > const & *geometries,* HitRecord const & *record*) const [virtual]

Renvoie

La contribution, c'est à dire l'ajout de lumière de la source à la lumière totale du point considéré.

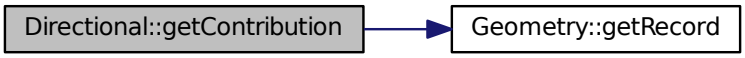
Paramètres

camera	La caméra à considérer.
geometries	Listes des géométries de la scène.
record	Enregistrement de l'intersection.

Implémente Light.

Définition à la ligne 20 du fichier Directional.cc.

Voici le graphe d'appel pour cette fonction :



4.9.3.2 Vector3d Directional : :getDirection ( ) const [inline]

Définition à la ligne 19 du fichier Directional.hpp.

4.9.4 Documentation des données membres

4.9.4.1 Vector3d Directional : :direction_ [protected]

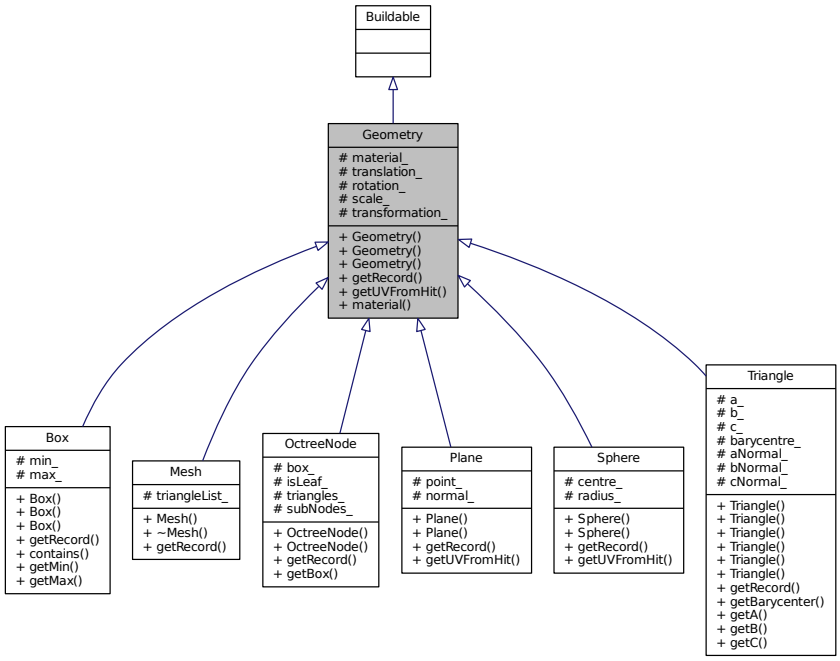
Définition à la ligne 22 du fichier Directional.hpp.

La documentation de cette classe a été générée à partir des fichiers suivants :
– src/Lights/Directional.hpp
– src/Lights/Directional.cc

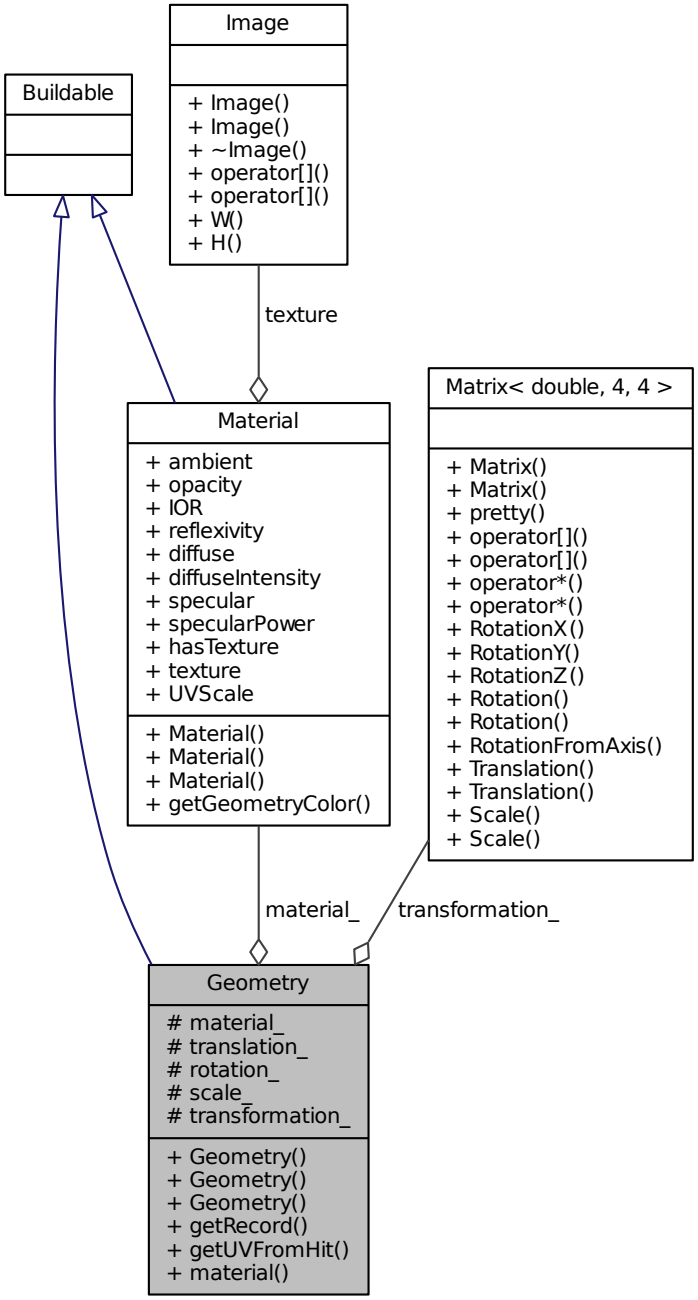
4.10 Référence de la classe Geometry

```
#include <Geometry.hpp>
```

Graphe d'héritage de Geometry :



Graphe de collaboration de Geometry :



Fonctions membres publiques

- Geometry ()
- Geometry (Material ∗material)
- Geometry (Material ∗material, Vector3d const &translation, Vector3d const &rotation, Vector3d const &scale)
- virtual HitRecord getRecord (Ray const &ray) const =0
- virtual Vector< double, 2 > getUVFromHit (HitRecord const &record) const
- Material ∗ material () const

Attributs protégés

- Material ∗ material_
- Vector3d translation_
- Vector3d rotation_
- Vector3d scale_
- Matrix< double, 4, 4 > transformation_

4.10.1 Description détaillée

Définition à la ligne 18 du fichier Geometry.hpp.

4.10.2 Documentation des constructeurs et destructeur

4.10.2.1 Geometry : :Geometry ( )

Définition à la ligne 3 du fichier Geometry.cc.

4.10.2.2 Geometry : :Geometry (Material ∗ *material*)

Paramètres

material	Matériaux lié à la géométrie

Définition à la ligne 11 du fichier Geometry.cc.

4.10.2.3 Geometry : :Geometry (Material ∗ *material,* Vector3d const & *translation,* Vector3d const & *rotation,* Vector3d const & *scale*)

Paramètres

material	Matériaux lié à la géométrie.
translation	Translation à appliquer à la géométrie.
rotation	Rotation à appliquer à la géométrie.
scale	Mise à l'échelle à appliquer à la géométrie.

Définition à la ligne 19 du fichier Geometry.cc.

4.10.3 Documentation des fonctions membres

4.10.3.1 virtual HitRecord Geometry : :getRecord (Ray const & *ray*) const [pure virtual]

Renvoie

L'enregistrement lié à la collision du rayon *ray* et la géométrie.

Paramètres

ray	Rayon à intersecter.

Implémenté dans Triangle, Box, Mesh, Sphere, OctreeNode, et Plane.

4.10.3.2 Vector< double, 2 > Geometry : :getUVFromHit (HitRecord const & *record*) const [virtual]

Renvoie

La coordonnées de texture correspondant à l'intersection enregistrée

Paramètres

record	Enregistrement de l'intersection.

Réimplémentée dans Sphere, et Plane.

Définition à la ligne 34 du fichier Geometry.cc.

4.10.3.3 Material* Geometry : :material ( ) const [inline]

Renvoie

Le matériau de la géométrie.

Définition à la ligne 59 du fichier Geometry.hpp.

4.10.4 Documentation des données membres

4.10.4.1 Material* Geometry : :material_ [protected]

Définition à la ligne 62 du fichier Geometry.hpp.

4.10.4.2 Vector3d Geometry : :rotation_ [protected]

Définition à la ligne 65 du fichier Geometry.hpp.

4.10.4.3 Vector3d Geometry : :scale_ [protected]

Définition à la ligne 66 du fichier Geometry.hpp.

4.10.4.4 Matrix<double, 4, 4> Geometry : :transformation_ [protected]

Définition à la ligne 67 du fichier Geometry.hpp.

4.10.4.5 Vector3d Geometry : :translation_ [protected]

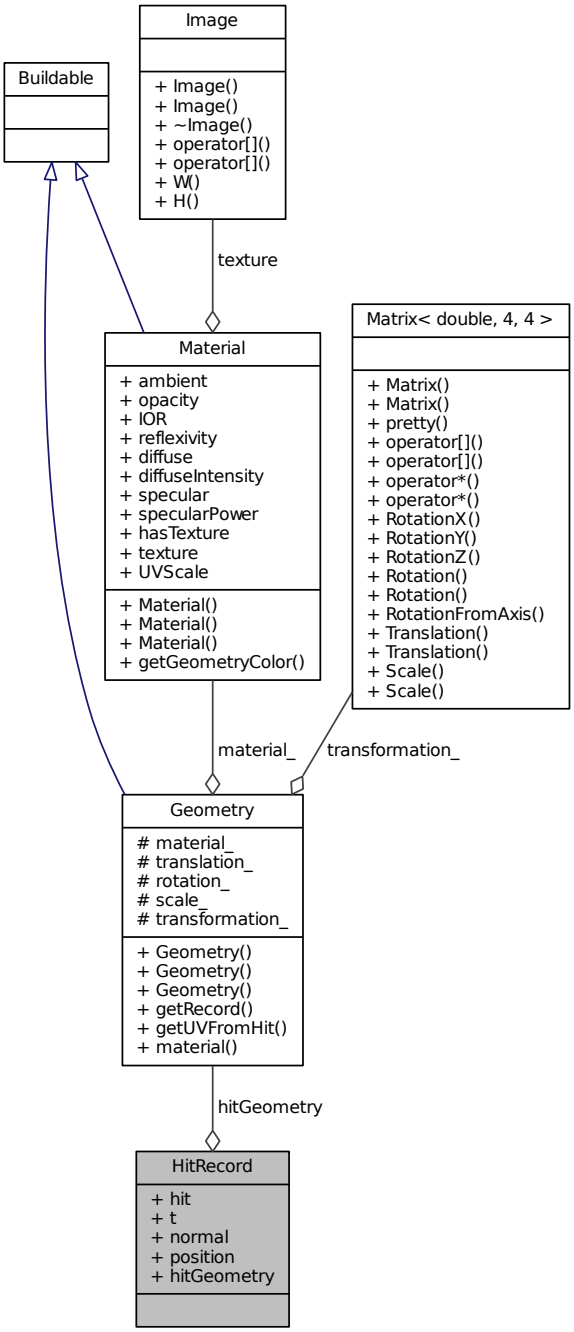
Définition à la ligne 64 du fichier Geometry.hpp.

La documentation de cette classe a été générée à partir des fichiers suivants :
– src/Geometry/Geometry.hpp
– src/Geometry/Geometry.cc

4.11 Référence de la structure HitRecord

```
#include <HitRecord.hpp>
```

Graphe de collaboration de HitRecord :



Attributs publics

– bool hit
– double t
– Vector3d normal
– Vector3d position
– Geometry const ∗ hitGeometry

Amis

– std : :ostream & operator<< (std : :ostream &oss, HitRecord const &m)

4.11.1 Description détaillée

Définition à la ligne 19 du fichier HitRecord.hpp.

4.11.2 Documentation des fonctions amies et associées

4.11.2.1 std : :ostream& operator<< (std : :ostream & *oss*, HitRecord const & *m*) [friend]

Définition à la ligne 50 du fichier HitRecord.hpp.

4.11.3 Documentation des données membres

4.11.3.1 bool HitRecord : :hit

Indique si le rayon a touché une géométrie.

Définition à la ligne 23 du fichier HitRecord.hpp.

4.11.3.2 Geometry const∗ HitRecord : :hitGeometry

Indique la géométrie touchée.

Remarques

Valable si seulement une géométrie à été touchée.

Définition à la ligne 48 du fichier HitRecord.hpp.

4.11.3.3 Vector3d HitRecord : :normal

Indique la normale à l'endroit de l'intersection.

Remarques

Valable si seulement une géométrie à été touchée.

Définition à la ligne 36 du fichier HitRecord.hpp.

4.11.3.4 Vector3d HitRecord : :position

Indique l'endroit de l'intersection.

Remarques

Valable si seulement une géométrie à été touchée.

Définition à la ligne 42 du fichier HitRecord.hpp.

4.11.3.5 double HitRecord : :t

Indique à quel endroit du rayon la géométrie à été touchée.

Remarques

Valable si seulement une géométrie à été touchée.

Définition à la ligne 30 du fichier HitRecord.hpp.

La documentation de cette structure a été générée à partir du fichier suivant :
– src/HitRecord.hpp

4.12 Référence de la classe Image

`#include <Image.hpp>`

Fonctions membres publiques

– Image (unsigned int width, unsigned int height)
– Image (Color< double > **const &pixelsData)
– ~Image ()
– Color< double > * operator[] (unsigned int i)
– Color< double > * operator[] (unsigned int i) const
– unsigned int W () const
– unsigned int H () const

4.12.1 Description détaillée

Définition à la ligne 17 du fichier Image.hpp.

4.12.2 Documentation des constructeurs et destructeur

4.12.2.1 Image : :Image (unsigned int *width,* unsigned int *height*)

Paramètres

width	Largeur de l'image.
height	Hauteur de l'image.

Définition à la ligne 8 du fichier Image.cc.

Voici le graphe d'appel pour cette fonction :

4.12.2.2 Image : :Image (Color< double > ∗∗const & *pixelsData*)

Paramètres

pixelsData	Tableau contenant l'ensemble des pixels avec pour origine (0,0) le coin supérieur gauche de l'image.

4.12.2.3 Image : :∼Image ( )

Définition à la ligne 21 du fichier Image.cc.

4.12.3 Documentation des fonctions membres

4.12.3.1 unsigned int Image : :H ( ) const

Renvoie

La hauteur de l'image.

Définition à la ligne 37 du fichier Image.cc.

4.12.3.2 Color< double > ∗ Image : :operator[ ] (unsigned int *i*)

Définition à la ligne 27 du fichier Image.cc.

4.12.3.3 Color< double > * Image : :operator[ ] (unsigned int *i*) const

Définition à la ligne 31 du fichier Image.cc.

4.12.3.4 unsigned int Image : :W ( ) const

Renvoie

La largeur de l'image.

Définition à la ligne 36 du fichier Image.cc.

La documentation de cette classe a été générée à partir des fichiers suivants :
– src/Graphics/Image.hpp
– src/Graphics/Image.cc

4.13 Référence de la classe ImageFactory

`#include <ImageFactory.hpp>`

Fonctions membres publiques statiques

– static void addHandler (ImageHandler *handler)
– static void Save (Image const &img, string const &filename)
– static Image * Load (std : :string const &filename)

4.13.1 Description détaillée

Définition à la ligne 19 du fichier ImageFactory.hpp.

4.13.2 Documentation des fonctions membres

4.13.2.1 void ImageFactory : :addHandler (ImageHandler * *handler*) [static]

Ajoute un handler à la liste des handlers disponibles.

Paramètres

Handler	à ajouter.

Définition à la ligne 8 du fichier ImageFactory.cc.

4.13.2.2 Image * ImageFactory : :Load (std : :string const & *filename*) [static]

Charge une image

Paramètres

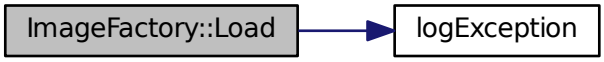
filename	Chemin de l'image à charger.

Renvoie

L'image nouvellement crée ou un pointeur nulle si le format est invalide ou l'image inexistante.

Définition à la ligne 25 du fichier ImageFactory.cc.

Voici le graphe d'appel pour cette fonction :



4.13.2.3 void ImageFactory : :Save (Image const & *img*, string const & *filename*) [static]

Sauvergarde l'image.

Paramètres

img	Image à sauvegarder.
filename	Destination de la nouvelle image.

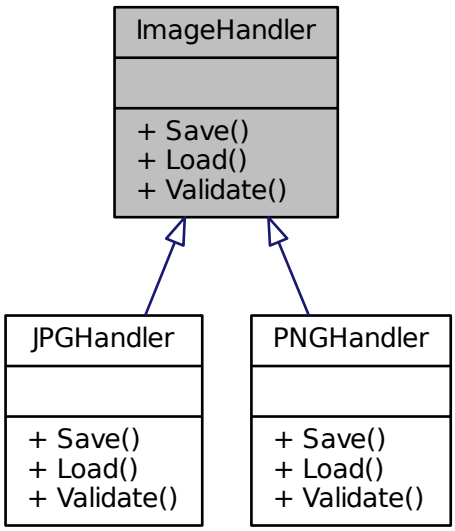
Définition à la ligne 12 du fichier ImageFactory.cc.

La documentation de cette classe a été générée à partir des fichiers suivants :
– src/Graphics/ImageFactory.hpp
– src/Graphics/ImageFactory.cc

4.14 Référence de la classe ImageHandler

```
#include <ImageHandler.hpp>
```

Graphe d'héritage de ImageHandler :



Fonctions membres publiques

– virtual void Save (Image const &img, std : :string const &filename)=0
– virtual Image ∗ Load (std : :string const &filename)=0
– virtual bool Validate (std : :string const &filename)=0

4.14.1 Description détaillée

Définition à la ligne 12 du fichier ImageHandler.hpp.

4.14.2 Documentation des fonctions membres

4.14.2.1 virtual Image∗ ImageHandler : :Load (std : :string const & *filename*) [pure virtual]

Charge une image

Paramètres

filename	Chemin de l'image à charger.

Renvoie

L'image nouvellement crée ou un pointeur nulle si le format est invalide ou l'image inexistante.

Implémenté dans JPGHandler, et PNGHandler.

4.14.2.2 virtual void ImageHandler : :Save (Image const & *img,* std : :string const & *filename*) [pure virtual]

Sauvergarde l'image.

Paramètres

img	Image à sauvegarder.
filename	Destination de la nouvelle image.

4.14.2.3 virtual bool ImageHandler : :Validate (std : :string const & *filename*) [pure virtual]

Renvoie

true si la handler est capable de gérer ce format, false sinon.

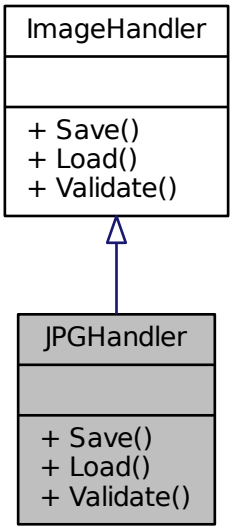
Implémenté dans JPGHandler, et PNGHandler.

La documentation de cette classe a été générée à partir du fichier suivant :
– src/Graphics/ImageHandler.hpp

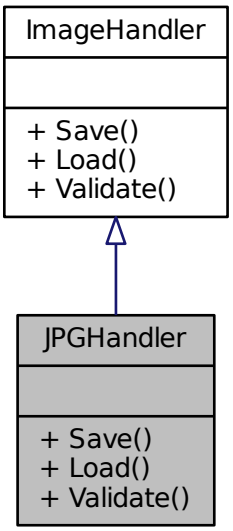
4.15 Référence de la classe JPGHandler

```
#include <JPGHandler.hpp>
```

Graphe d'héritage de JPGHandler :



Graphe de collaboration de JPGHandler :



Fonctions membres publiques

– void Save (Image const &img, string const &filename)
– Image ∗ Load (std : :string const &filename)
– bool Validate (std : :string const &filename)

4.15.1 Description détaillée

Définition à la ligne 7 du fichier JPGHandler.hpp.

4.15.2 Documentation des fonctions membres

4.15.2.1 Image∗ JPGHandler : :Load (std : :string const & *filename*) [virtual]

Charge une image

Paramètres

filename	Chemin de l'image à charger.

Renvoie

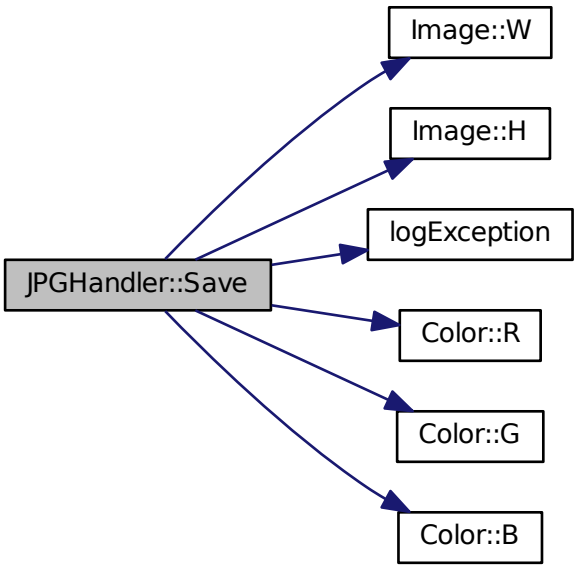
L'image nouvellement crée ou un pointeur nulle si le format est invalide ou l'image inexistante.

Implémente ImageHandler.

4.15.2.2 void JPGHandler : :Save (Image const & *img*, string const & *filename*)

Définition à la ligne 12 du fichier JPGHandler.cc.

Voici le graphe d'appel pour cette fonction :



4.15.2.3 bool JPGHandler : :Validate (std : :string const & *filename*) [virtual]

Renvoie

true si la handler est capable de gérer ce format, false sinon.

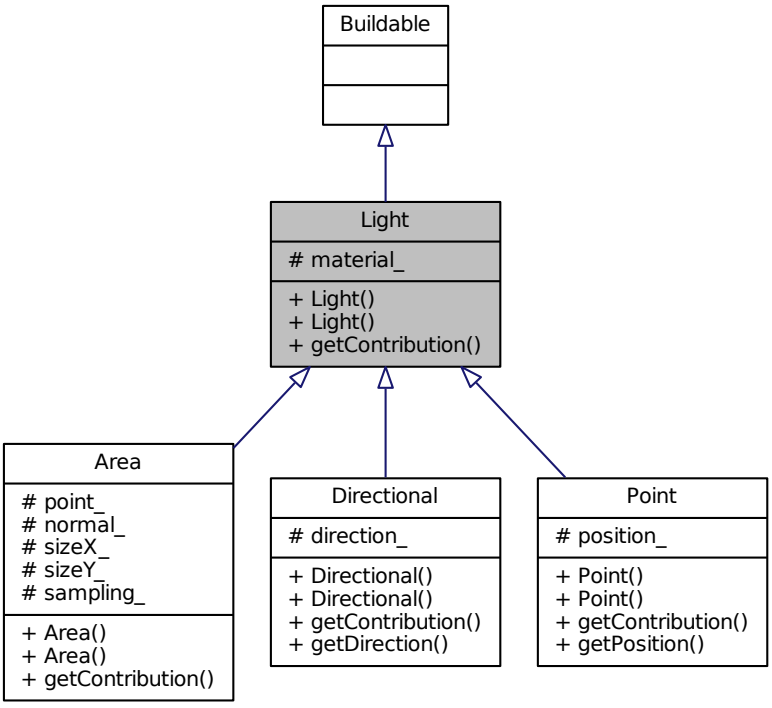
Implémente ImageHandler.

La documentation de cette classe a été générée à partir des fichiers suivants :
- src/Graphics/JPGHandler.hpp
- src/Graphics/JPGHandler.cc

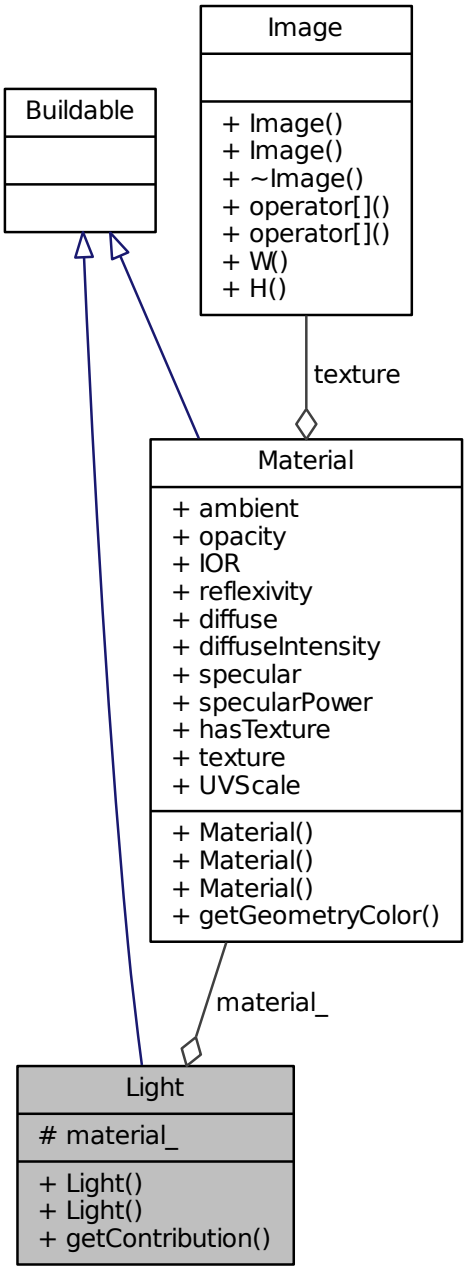
4.16 Référence de la classe Light

`#include <Light.hpp>`

Graphe d'héritage de Light :



Graphe de collaboration de Light :



Fonctions membres publiques

- Light ()
- Light (Material ∗material)
- virtual Color_d getContribution (Camera ∗camera, std : :vector< Geometry ∗ > const &geometries, HitRecord const &record) const =0

Attributs protégés

- Material ∗ material_

4.16.1 Description détaillée

Définition à la ligne 22 du fichier Light.hpp.

4.16.2 Documentation des constructeurs et destructeur

4.16.2.1 Light : :Light ( ) [inline]

Définition à la ligne 24 du fichier Light.hpp.

4.16.2.2 Light : :Light (Material ∗ *material*) [inline]

Paramètres

material	Matériau à associer à la lumière.

Définition à la ligne 29 du fichier Light.hpp.

4.16.3 Documentation des fonctions membres

4.16.3.1 virtual Color_d Light : :getContribution (Camera ∗ *camera,* std : :vector< Geometry ∗ > const & *geometries,* HitRecord const & *record*) const [pure virtual]

Renvoie

La contribution, c'est à dire l'ajout de lumière de la source à la lumière totale du point considéré.

Paramètres

camera	La caméra à considérer.
geometries	Listes des géométries de la scène.
record	Enregistrement de l'intersection.

Implémenté dans Area, Point, et Directional.

4.16.4 Documentation des données membres

4.16.4.1 Material* Light : :material_ [protected]

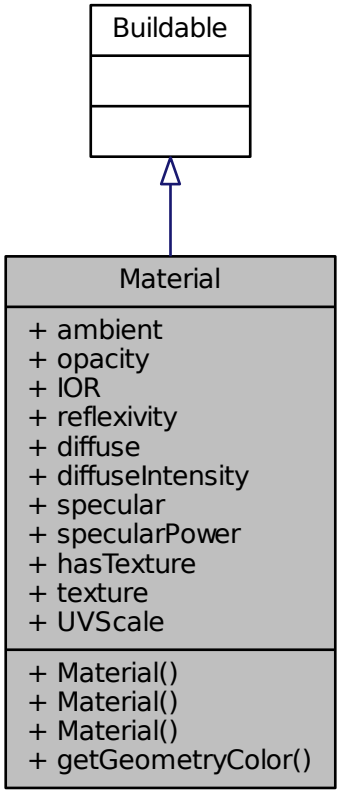
Définition à la ligne 46 du fichier Light.hpp.

La documentation de cette classe a été générée à partir du fichier suivant :
– src/Lights/Light.hpp

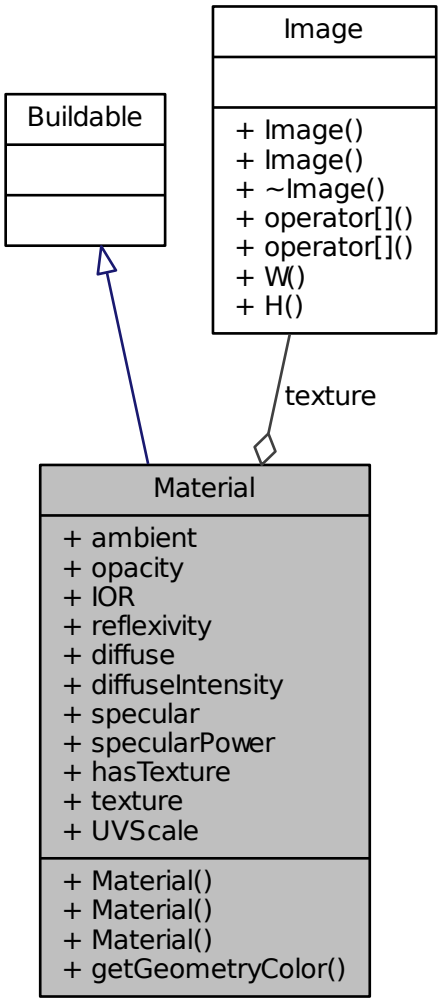
4.17 Référence de la classe Material

`#include <Material.hpp>`

Graphe d'héritage de Material :



Graphe de collaboration de Material :



Fonctions membres publiques

– Material ()
– Material (Color_d diffuse)
– Material (Color_d diffuse, Color_d ambient, Color_d specular)

Fonctions membres publiques statiques

– static Color_d getGeometryColor (HitRecord const &record)

Attributs publics

– Color_d ambient
– double opacity
– double IOR
  Indice de réfraction du matériaux.
– double reflexivity
  Reflexivité du matériaux.
– Color_d diffuse
  Couleur diffuse du matériaux.
– double diffuseIntensity
– Color_d specular
  Couleur spéculaire du matériaux.
– double specularPower
  Puissance de spécularitée.
– bool hasTexture
  Indique si le matériau est composé d'une texture.
– Image ∗ texture
  La texture en question.
– double UVScale
  La mise à l'échelle à appliquer à la texture.

4.17.1 Description détaillée

Définition à la ligne 31 du fichier Material.hpp.

4.17.2 Documentation des constructeurs et destructeur

4.17.2.1 Material : :Material ( )

Définition à la ligne 5 du fichier Material.cc.

4.17.2.2 Material : :Material (Color_d *diffuse*)

Définition à la ligne 16 du fichier Material.cc.

4.17.2.3 Material : :Material (Color_d *diffuse,* Color_d *ambient,* Color_d *specular*)

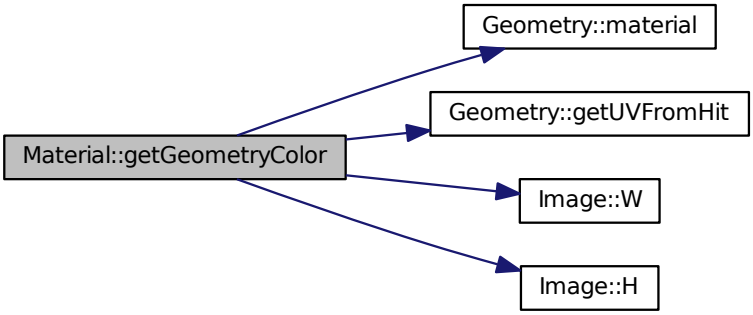
Définition à la ligne 28 du fichier Material.cc.

4.17.3 Documentation des fonctions membres

4.17.3.1 Color_d Material : :getGeometryColor (HitRecord const & *record*) [static]

Définition à la ligne 39 du fichier Material.cc.

Voici le graphe d'appel pour cette fonction :



4.17.4 Documentation des données membres

4.17.4.1 Color_d Material : :ambient

Définition à la ligne 62 du fichier Material.hpp.

4.17.4.2 Color_d Material : :diffuse

Couleur diffuse du matériaux.

Définition à la ligne 78 du fichier Material.hpp.

4.17.4.3 double Material : :diffuseIntensity

Intensité de la couleur diffuse, i.e. le coefficient multiplicateur de la couleur diffuse.

Définition à la ligne 84 du fichier Material.hpp.

4.17.4.4 bool Material : :hasTexture

Indique si le matériau est composé d'une texture.

Définition à la ligne 93 du fichier Material.hpp.

4.17.4.5 double Material : :IOR

Indice de réfraction du matériaux.

Définition à la ligne 72 du fichier Material.hpp.

4.17.4.6 double Material : :opacity

Définition à la ligne 69 du fichier Material.hpp.

4.17.4.7 double Material : :reflexivity

Reflexivité du matériaux.

Définition à la ligne 75 du fichier Material.hpp.

4.17.4.8 Color_d Material : :specular

Couleur spéculaire du matériaux.

Définition à la ligne 87 du fichier Material.hpp.

4.17.4.9 double Material : :specularPower

Puissance de spécularitée.

Définition à la ligne 90 du fichier Material.hpp.

4.17.4.10 Image* Material : :texture

La texture en question.

Définition à la ligne 96 du fichier Material.hpp.

4.17.4.11 double Material : :UVScale

La mise à l'échelle à appliquer à la texture.

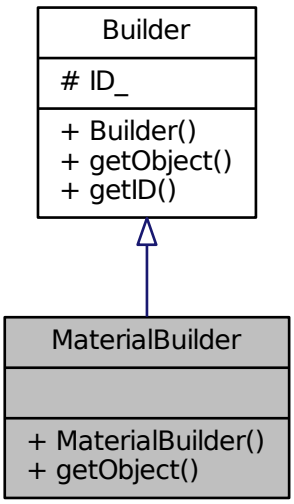
Définition à la ligne 99 du fichier Material.hpp.

La documentation de cette classe a été générée à partir des fichiers suivants :
– src/Material.hpp
– src/Material.cc

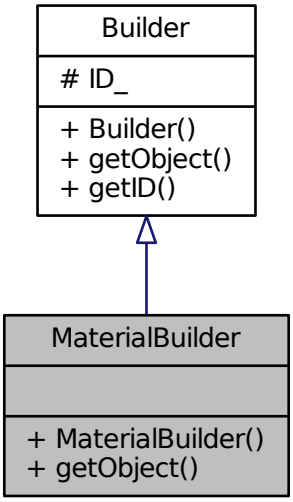
4.18 Référence de la classe MaterialBuilder

```
#include <MaterialBuilder.hpp>
```

Graphe d'héritage de MaterialBuilder :



Graphe de collaboration de MaterialBuilder :



Fonctions membres publiques

– MaterialBuilder ()
– Buildable ∗ getObject (boost : :property_tree : :ptree pt, Material ∗material) const

4.18.1 Description détaillée

Définition à la ligne 17 du fichier MaterialBuilder.hpp.

4.18.2 Documentation des constructeurs et destructeur

4.18.2.1 MaterialBuilder : :MaterialBuilder ( )

Définition à la ligne 8 du fichier MaterialBuilder.cc.

4.18.3 Documentation des fonctions membres

4.18.3.1 Buildable ∗ MaterialBuilder : :getObject (boost : :property_tree : :ptree *pt,* Material ∗ *material*) const [virtual]

Paramètres

pt	L'arbre DOM contenant les paramètres du fichier de configuration de la scène,
material	Matériau associé.

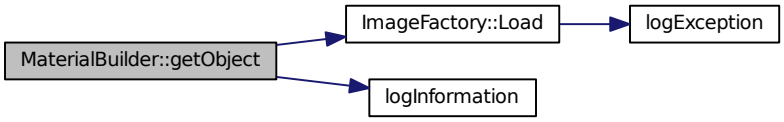
Renvoie

Un pointeur sur l'objet construit.

Implémente Builder.

Définition à la ligne 12 du fichier MaterialBuilder.cc.

Voici le graphe d'appel pour cette fonction :



La documentation de cette classe a été générée à partir des fichiers suivants :
– src/Builders/MaterialBuilder.hpp
– src/Builders/MaterialBuilder.cc

4.19 Référence du modèle de la classe Matrix< P, N, M >

`#include <Matrix.hpp>`

Fonctions membres publiques

– Matrix (MATRIX_TYPE type)
– Matrix (const Matrix< P, N, M > &m)
– string pretty () const
– P * operator[] (unsigned int i)
– const P * operator[] (unsigned int i) const
– Matrix< P, N, M > operator* (const Matrix< P, N, M > &m) const
– Vector< P, M > operator* (Vector< P, M > const &v) const

Fonctions membres publiques statiques

– static Matrix< P, 4, 4 > RotationX (double angle)
– static Matrix< P, 4, 4 > RotationY (double angle)
– static Matrix< P, 4, 4 > RotationZ (double angle)
– static Matrix< P, 4, 4 > Rotation (double Xangle, double Yangle, double Zangle)
– static Matrix< P, 4, 4 > Rotation (Vector3d const &angle)

– static Matrix< P, 4, 4 > RotationFromAxis (Vector< P, 3 > const &axis, double angle)
– static Matrix< P, 4, 4 > Translation (double X, double Y, double Z)
– static Matrix< P, 4, 4 > Translation (Vector3d const &translation)
– static Matrix< P, 4, 4 > Scale (double X, double Y, double Z)
– static Matrix< P, 4, 4 > Scale (Vector3d const &scaleFactor)

Amis

– ostream & operator<< (ostream &oss, const Matrix< P, N, M > &m)

4.19.1 Description détaillée

template<typename P, int N, int M>class Matrix< P, N, M >

Définition à la ligne 34 du fichier Matrix.hpp.

4.19.2 Documentation des constructeurs et destructeur

4.19.2.1 template<typename P, int N, int M> Matrix< P, N, M > : :Matrix (MATRIX_TYPE *type*)

Paramètres

type	Si ZERO, construit une matrice nulle, si IDENTITY, construit une matrice identité

4.19.2.2 template<typename P, int N, int M> Matrix< P, N, M > : :Matrix (const Matrix< P, N, M > & *m*)

4.19.3 Documentation des fonctions membres

4.19.3.1 template<typename P, int N, int M> Matrix< P, N, M > Matrix< P, N, M > : :operator∗ (const Matrix< P, N, M > & *m*) const

4.19.3.2 template<typename P, int N, int M> Vector< P, M > Matrix< P, N, M > : :operator∗ (Vector< P, M > const & *v*) const

4.19.3.3 template<typename P, int N, int M> P∗ Matrix< P, N, M > : :operator[] (unsigned int *i*)

4.19.3.4 template<typename P, int N, int M> const P∗ Matrix< P, N, M > : :operator[] (unsigned int *i*) const

4.19.3.5 template<typename P, int N, int M> string Matrix< P, N, M > : :pretty ( ) const

4.19.3.6 template<typename P, int N, int M> static Matrix< P, 4, 4 > Matrix< P, N, M > : :Rotation (double *Xangle,* double *Yangle,* double *Zangle*) [static]

Construit une matrice homogène de rotation autours des 3 axes de la base orthogonale main droite.

Paramètres

Xangle	Angle en radian de la rotation autours de X.
Yangle	Angle en radian de la rotation autours de Y.
Zangle	Angle en radian de la rotation autours de Z.

Renvoie

La matrice homogène construite en calculant le produit des trois matrices de rotations.

4.19.3.7 template<typename P, int N, int M> static Matrix< P, 4, 4 > Matrix< P, N, M > : :Rotation (Vector3d const & *angle*) [static]

Construit une matrice homogène de rotation autours des 3 axes de la base orthogonale main droite.

Paramètres

angle	Angle de rotation selon X, Y et Z.

4.19.3.8 template<typename P, int N, int M> static Matrix< P, 4, 4 > Matrix< P, N, M > : :RotationFromAxis (Vector< P, 3 > const & *axis,* double *angle*) [static]

4.19.3.9 template<typename P, int N, int M> static Matrix< P, 4, 4 > Matrix< P, N, M > : :RotationX (double *angle*) [static]

Construit une matrice homogène de rotation autours de l'axe des X (repère main droite).

Paramètres

angle	Angle en radian de la rotation.

Renvoie

La matrice homogène construite.

4.19.3.10 template<typename P, int N, int M> static Matrix< P, 4, 4 > Matrix< P, N, M > : :RotationY (double *angle*) [static]

Construit une matrice homogène de rotation autours de l'axe des Y (repère main droite).

Paramètres

angle	Angle en radian de la rotation.

Renvoie

La matrice homogène construite.

4.19.3.11 template<typename P, int N, int M> static Matrix< P, 4, 4 > Matrix< P, N, M > : :RotationZ (double *angle*) [static]

Construit une matrice homogène de rotation autours de l'axe des Z (repère main droite).

Paramètres

angle	Angle en radian de la rotation.

Renvoie

La matrice homogène construite.

4.19.3.12 template<typename P, int N, int M> static Matrix< P, 4, 4 > Matrix< P, N, M > : :Scale (double *X,* double *Y,* double *Z*) [static]

Construit la matrice homogène de mise à l'échelle.

Paramètres

X	Mise à l'échelle par rapport à l'axe des X (en repère main droite).
Y	Mise à l'échelle par rapport à l'axe des Y (en repère main droite).
Z	Mise à l'échelle par rapport à l'axe des Z (en repère main droite).

Renvoie

La matrice homogène construite.

4.19.3.13 template<typename P, int N, int M> static Matrix< P, 4, 4 > Matrix< P, N, M > : :Scale (Vector3d const & *scaleFactor*) [static]

4.19.3.14 template<typename P, int N, int M> static Matrix< P, 4, 4 > Matrix< P, N, M > : :Translation (double *X,* double *Y,* double *Z*) [static]

Construit la matrice homogène de translation.

Paramètres

X	Translation par rapport à l'axe des X (en repère main droite).
Y	Translation par rapport à l'axe des Y (en repère main droite).
Z	Translation par rapport à l'axe des Z (en repère main droite).

Renvoie

La matrice homogène construite.

4.19.3.15 template<typename P, int N, int M> static Matrix< P, 4, 4 > Matrix< P, N, M > : :Translation (Vector3d const & *translation*) [static]

4.19.4 Documentation des fonctions amies et associées

4.19.4.1 template<typename P, int N, int M> ostream& operator<< (ostream & *oss,* const Matrix< P, N, M > & *m*) [friend]

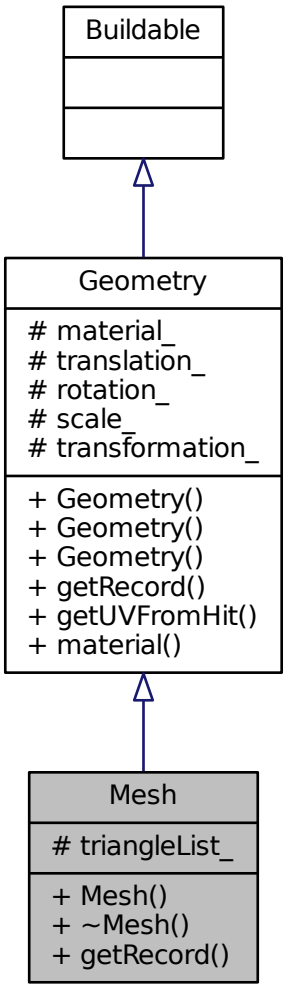
Définition à la ligne 139 du fichier Matrix.hpp.

La documentation de cette classe a été générée à partir du fichier suivant :
– src/Maths/Matrix.hpp

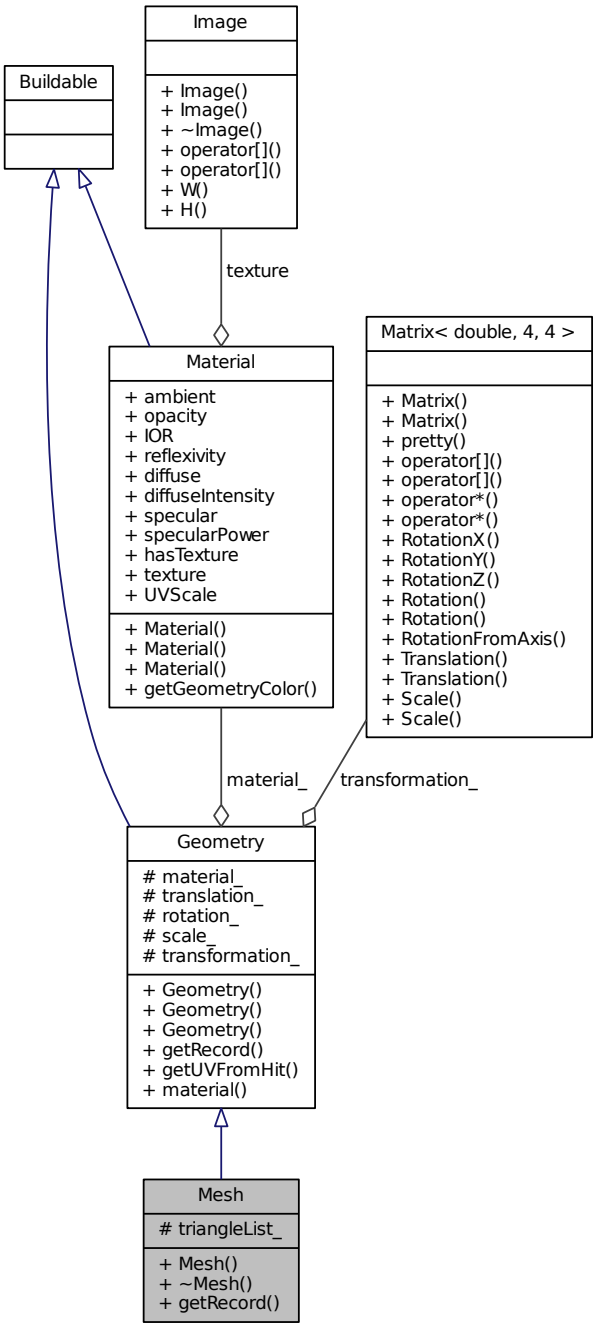
4.20 Référence de la classe Mesh

```
#include <Mesh.hpp>
```

Graphe d'héritage de Mesh :



Graphe de collaboration de Mesh :



Fonctions membres publiques

- Mesh (std : :vector< Triangle * > const &triangleList, Material *material, Vector3d const &translation, Vector3d const &rotation, Vector3d const &scale)
- ~Mesh ()
- HitRecord getRecord (Ray const &ray) const

Attributs protégés

- std : :vector< Triangle * > triangleList_

4.20.1 Description détaillée

Définition à la ligne 15 du fichier Mesh.hpp.

4.20.2 Documentation des constructeurs et destructeur

4.20.2.1 Mesh : :Mesh (std : :vector< Triangle * > const & *triangleList,* Material * *material,* Vector3d const & *translation,* Vector3d const & *rotation,* Vector3d const & *scale*)

Définition à la ligne 8 du fichier Mesh.cc.

4.20.2.2 Mesh : :~Mesh ( )

Définition à la ligne 20 du fichier Mesh.cc.

4.20.3 Documentation des fonctions membres

4.20.3.1 HitRecord Mesh : :getRecord (Ray const & *ray*) const [virtual]

Renvoie

L'enregistrement lié à la collision du rayon *ray* et la géométrie.

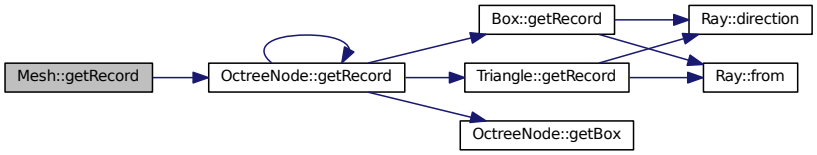
Paramètres

ray	Rayon à intersecter.

Implémente Geometry.

Définition à la ligne 47 du fichier Mesh.cc.

Voici le graphe d'appel pour cette fonction :



4.20.4 Documentation des données membres

4.20.4.1 std : :vector<Triangle*> Mesh : :triangleList_ [protected]

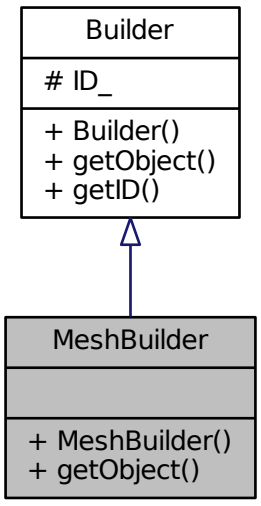
Définition à la ligne 33 du fichier Mesh.hpp.

La documentation de cette classe a été générée à partir des fichiers suivants :
– src/Geometry/Mesh.hpp
– src/Geometry/Mesh.cc

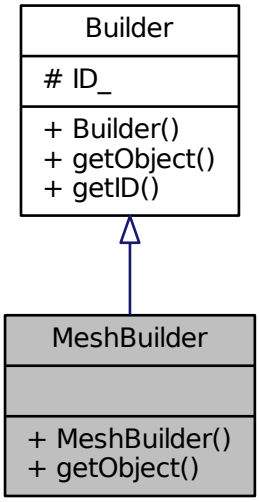
4.21 Référence de la classe MeshBuilder

```
#include <MeshBuilder.hpp>
```

Graphe d'héritage de MeshBuilder :



Graphe de collaboration de MeshBuilder :



Fonctions membres publiques

– MeshBuilder ()
– Buildable ∗ getObject (boost : :property_tree : :ptree pt, Material ∗material) const

4.21.1 Description détaillée

Définition à la ligne 17 du fichier MeshBuilder.hpp.

4.21.2 Documentation des constructeurs et destructeur

4.21.2.1 MeshBuilder : :MeshBuilder ( )

Définition à la ligne 8 du fichier MeshBuilder.cc.

4.21.3 Documentation des fonctions membres

4.21.3.1 Buildable ∗ MeshBuilder : :getObject (boost : :property_tree : :ptree *pt,* Material ∗ *material*) const [virtual]

Paramètres

pt	L'arbre DOM contenant les paramètres du fichier de configuration de la scène,
material	Matériau associé.

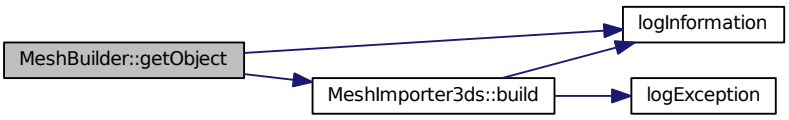
Renvoie

Un pointeur sur l'objet construit.

Implémente Builder.

Définition à la ligne 12 du fichier MeshBuilder.cc.

Voici le graphe d'appel pour cette fonction :

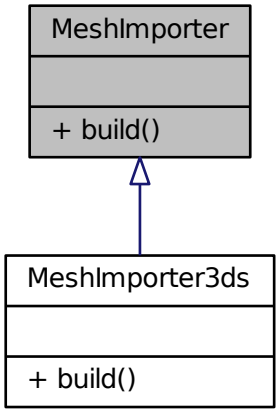


La documentation de cette classe a été générée à partir des fichiers suivants :
– src/Builders/MeshBuilder.hpp
– src/Builders/MeshBuilder.cc

4.22 Référence de la classe MeshImporter

`#include <MeshImporter.hpp>`

Graphe d'héritage de MeshImporter :



Fonctions membres publiques

– virtual Mesh ∗ build (std : :string const &filename, Material ∗material, Vector3d const &translation, Vector3d const &rotation, Vector3d const &scale) const =0

4.22.1 Description détaillée

Définition à la ligne 9 du fichier MeshImporter.hpp.

4.22.2 Documentation des fonctions membres

4.22.2.1 virtual Mesh∗ MeshImporter : :build (std : :string const & *filename,* Material ∗ *material,* Vector3d const & *translation,* Vector3d const & *rotation,* Vector3d const & *scale*) const [pure virtual]

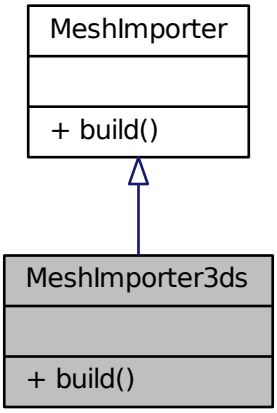
Implémenté dans MeshImporter3ds.

La documentation de cette classe a été générée à partir du fichier suivant :
– src/MeshImporters/MeshImporter.hpp

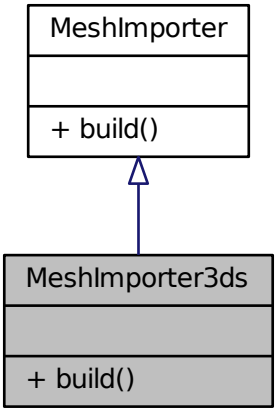
4.23 Référence de la classe MeshImporter3ds

```
#include <MeshImporter3ds.hpp>
```

Graphe d'héritage de MeshImporter3ds :



Graphe de collaboration de MeshImporter3ds :



Fonctions membres publiques

– Mesh ∗ build (std : :string const &filename, Material ∗material, Vector3d const &translation, Vector3d const &rotation, Vector3d const &scale) const

4.23.1 Description détaillée

Définition à la ligne 5 du fichier MeshImporter3ds.hpp.

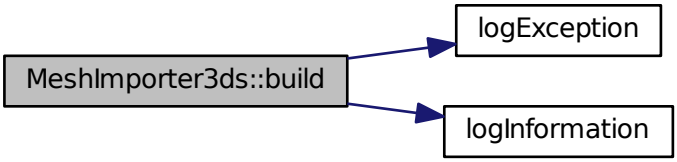
4.23.2 Documentation des fonctions membres

4.23.2.1 Mesh ∗ MeshImporter3ds : :build (std : :string const & *filename,* Material ∗ *material,* Vector3d const & *translation,* Vector3d const & *rotation,* Vector3d const & *scale*) const [virtual]

Implémente MeshImporter.

Définition à la ligne 13 du fichier MeshImporter3ds.cc.

Voici le graphe d'appel pour cette fonction :

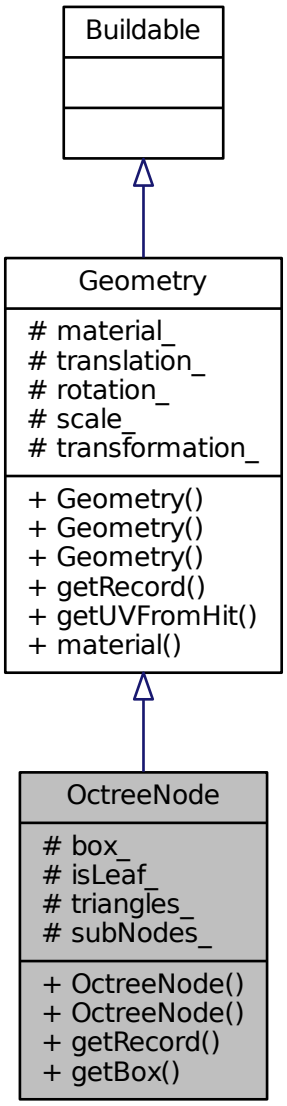


La documentation de cette classe a été générée à partir des fichiers suivants :
– src/MeshImporters/MeshImporter3ds.hpp
– src/MeshImporters/MeshImporter3ds.cc

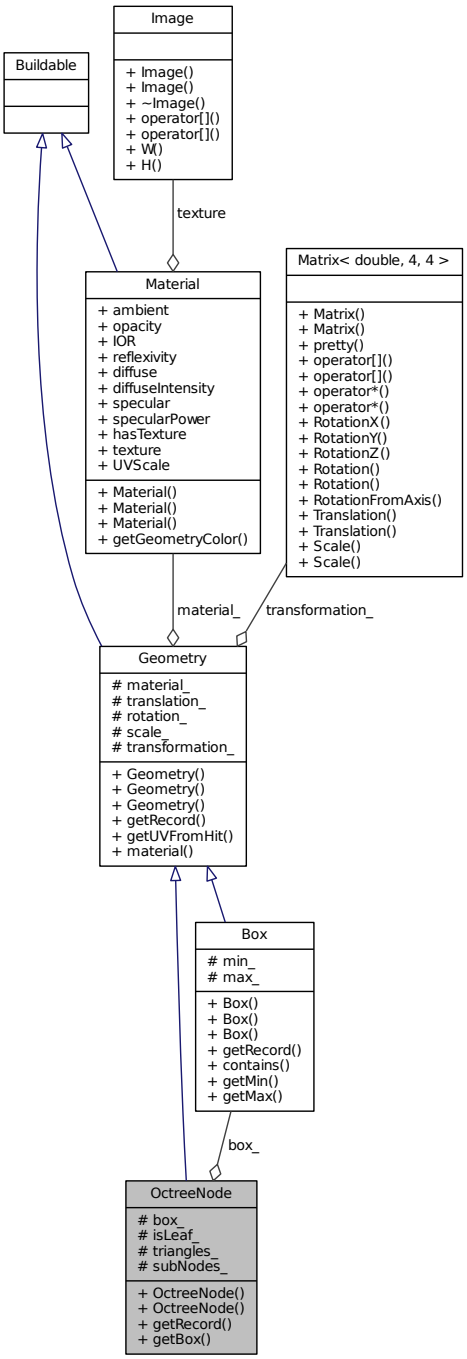
4.24 Référence de la classe OctreeNode

```
#include <OctreeNode.hpp>
```

Graphe d'héritage de OctreeNode :



Graphe de collaboration de OctreeNode :



Fonctions membres publiques

- OctreeNode ()
- OctreeNode (Box const &box, vector< Triangle * > const &triangles, unsigned int minTriangles)
- HitRecord getRecord (Ray const &ray) const
- Box getBox () const

Attributs protégés

- Box box_
- bool isLeaf_
- vector< Triangle * > triangles_
- vector< OctreeNode * > subNodes_

4.24.1 Description détaillée

Définition à la ligne 9 du fichier OctreeNode.hpp.

4.24.2 Documentation des constructeurs et destructeur

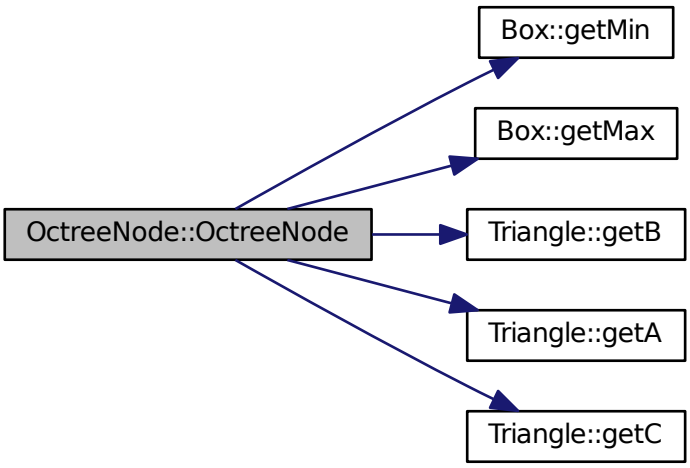
4.24.2.1 OctreeNode : :OctreeNode ( )

Définition à la ligne 6 du fichier OctreeNode.cc.

4.24.2.2 OctreeNode : :OctreeNode (Box const & *box,* vector< Triangle * > const & *triangles,* unsigned int *minTriangles*)

Définition à la ligne 10 du fichier OctreeNode.cc.

Voici le graphe d'appel pour cette fonction :



4.24.3 Documentation des fonctions membres

4.24.3.1 Box OctreeNode : :getBox ( ) const [inline]

Définition à la ligne 19 du fichier OctreeNode.hpp.

4.24.3.2 HitRecord OctreeNode : :getRecord (Ray const & *ray*) const [virtual]

Renvoie

L'enregistrement lié à la collision du rayon *ray* et la géométrie.

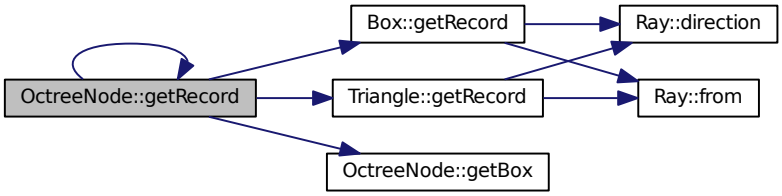
Paramètres

ray	Rayon à intersecter.

Implémente Geometry.

Définition à la ligne 72 du fichier OctreeNode.cc.

Voici le graphe d'appel pour cette fonction :



4.24.4 Documentation des données membres

4.24.4.1 Box OctreeNode : :box_ [protected]

Définition à la ligne 25 du fichier OctreeNode.hpp.

4.24.4.2 bool OctreeNode : :isLeaf_ [protected]

Définition à la ligne 26 du fichier OctreeNode.hpp.

4.24.4.3 vector< OctreeNode* > OctreeNode : :subNodes_ [protected]

Définition à la ligne 29 du fichier OctreeNode.hpp.

4.24.4.4 vector< Triangle* > OctreeNode : :triangles_ [protected]

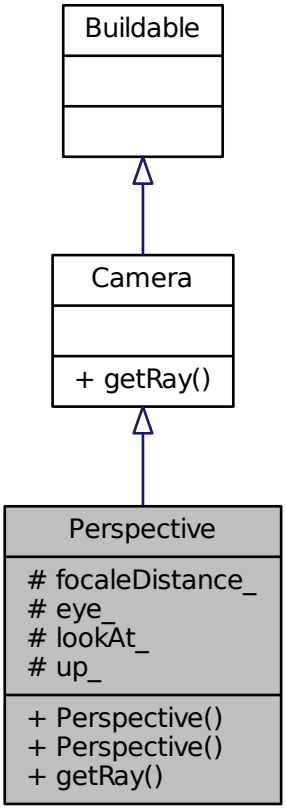
Définition à la ligne 27 du fichier OctreeNode.hpp.

La documentation de cette classe a été générée à partir des fichiers suivants :
– src/Optimization/OctreeNode.hpp
– src/Optimization/OctreeNode.cc

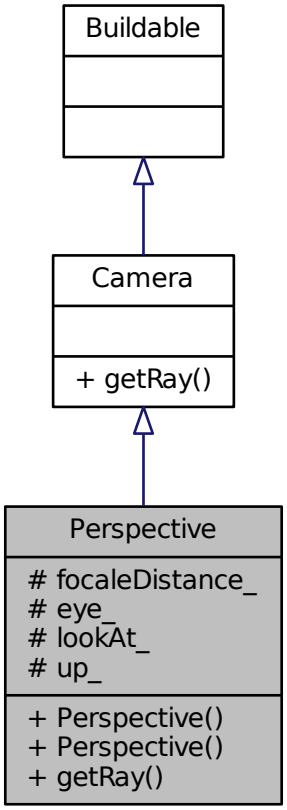
4.25 Référence de la classe Perspective

```
#include <Perspective.hpp>
```

Graphe d'héritage de Perspective :



Graphe de collaboration de Perspective :



Fonctions membres publiques

– Perspective (Vector3d const &eye, Vector3d const &lookAt, Vector3d const &up)
– Perspective (double focaleDistance, Vector3d const &eye, Vector3d const &lookAt, Vector3d const &up)
– std : :vector< Ray > getRay (double u, double v) const

Attributs protégés

– double focaleDistance_
– Vector3d eye_
– Vector3d lookAt_
– Vector3d up_

4.25.1 Description détaillée

Définition à la ligne 21 du fichier Perspective.hpp.

4.25.2 Documentation des constructeurs et destructeur

4.25.2.1 Perspective : :Perspective (Vector3d const & *eye,* Vector3d const & *lookAt,* Vector3d const & *up*)

Paramètres

eye	Position de la caméra.
lookAt	Endroit où la caméra pointe.
up	Vecteur indiquant où doit pointer le haut de l'image.

Remarques

Par défaut, la distance focale de la caméra est 1.0.

Définition à la ligne 6 du fichier Perspective.cc.

4.25.2.2 Perspective : :Perspective (double *focaleDistance,* Vector3d const & *eye,* Vector3d const & *lookAt,* Vector3d const & *up*)

Paramètres

focale-Distance	Distance focale de la caméra.
eye	Position de la caméra.
lookAt	Endroit où la caméra pointe.
up	Vecteur indiquant où doit pointer le haut de l'image.

Définition à la ligne 18 du fichier Perspective.cc.

4.25.3 Documentation des fonctions membres

4.25.3.1 std : :vector< Ray > Perspective : :getRay (double *u,* double *v*) const [virtual]

Renvoie

L'ensemble des rayons associés à la coordonnée (u,v)

Implémente Camera.

Définition à la ligne 45 du fichier Perspective.cc.

4.25.4 Documentation des données membres

4.25.4.1 Vector3d Perspective : :eye_ [protected]

Définition à la ligne 55 du fichier Perspective.hpp.

4.25.4.2 double Perspective : :focaleDistance_ [protected]

Définition à la ligne 54 du fichier Perspective.hpp.

4.25.4.3 Vector3d Perspective : :lookAt_ [protected]

Définition à la ligne 56 du fichier Perspective.hpp.

4.25.4.4 Vector3d Perspective : :up_ [protected]

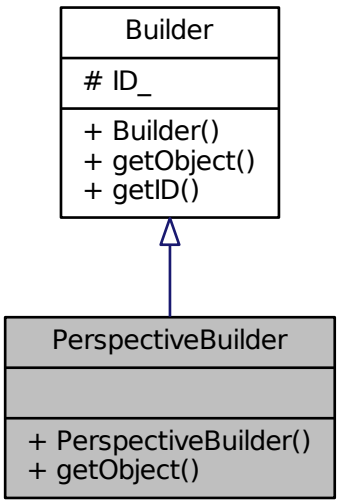
Définition à la ligne 57 du fichier Perspective.hpp.

La documentation de cette classe a été générée à partir des fichiers suivants :
– src/Cameras/Perspective.hpp
– src/Cameras/Perspective.cc

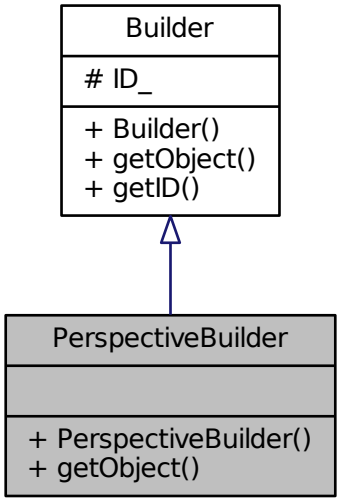
4.26 Référence de la classe PerspectiveBuilder

```
#include <PerspectiveBuilder.hpp>
```

Graphe d'héritage de PerspectiveBuilder :



Graphe de collaboration de PerspectiveBuilder :



Fonctions membres publiques

– PerspectiveBuilder ()
– Buildable ∗ getObject (boost : :property_tree : :ptree pt, Material ∗material) const

4.26.1 Description détaillée

Définition à la ligne 17 du fichier PerspectiveBuilder.hpp.

4.26.2 Documentation des constructeurs et destructeur

4.26.2.1 PerspectiveBuilder : :PerspectiveBuilder ( )

Définition à la ligne 6 du fichier PerspectiveBuilder.cc.

4.26.3 Documentation des fonctions membres

4.26.3.1 Buildable ∗ PerspectiveBuilder : :getObject (boost : :property_tree : :ptree *pt,* Material ∗ *material*) const [virtual]

Paramètres

pt	L'arbre DOM contenant les paramètres du fichier de configuration de la scène,
material	Matériau associé.

Renvoie

Un pointeur sur l'objet construit.

Implémente Builder.

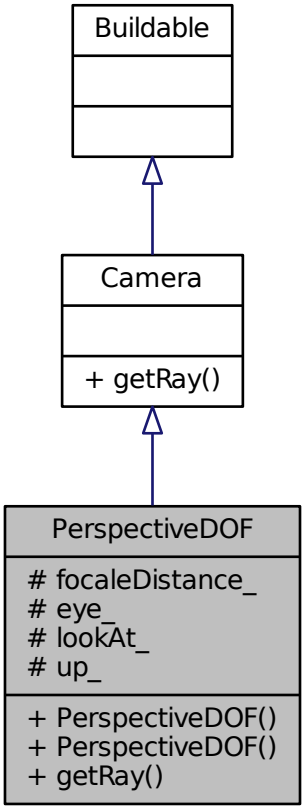
Définition à la ligne 10 du fichier PerspectiveBuilder.cc.

La documentation de cette classe a été générée à partir des fichiers suivants :
– src/Builders/PerspectiveBuilder.hpp
– src/Builders/PerspectiveBuilder.cc

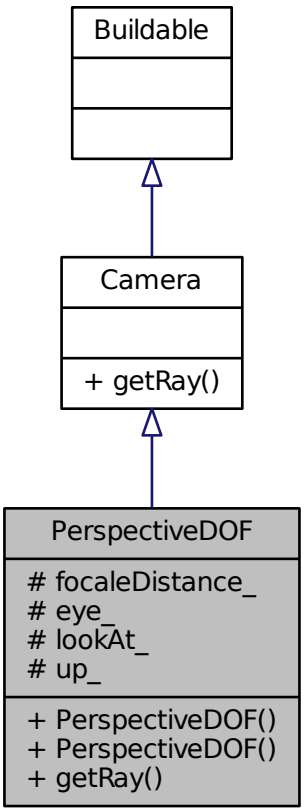
4.27 Référence de la classe PerspectiveDOF

```
#include <PerspectiveDOF.hpp>
```

Graphe d'héritage de PerspectiveDOF :



Graphe de collaboration de PerspectiveDOF :



Fonctions membres publiques

- PerspectiveDOF (Vector3d const &eye, Vector3d const &lookAt, Vector3d const &up, Vector3d const &focusPoint, unsigned int sampling, double aperture)
- PerspectiveDOF (double focaleDistance, Vector3d const &eye, Vector3d const &lookAt, Vector3d const &up, Vector3d const &focusPoint, unsigned int sampling, double aperture)
- std : :vector< Ray > getRay (double u, double v) const

Attributs protégés

- double focaleDistance_
- Vector3d eye_
- Vector3d lookAt_

– Vector3d up_

4.27.1 Description détaillée

Définition à la ligne 24 du fichier PerspectiveDOF.hpp.

4.27.2 Documentation des constructeurs et destructeur

4.27.2.1 PerspectiveDOF : :PerspectiveDOF (Vector3d const & *eye,* Vector3d const & *lookAt,* Vector3d const & *up,* Vector3d const & *focusPoint,* unsigned int *sampling,* double *aperture*)

Paramètres

eye	Position de la caméra.
lookAt	Endroit où la caméra pointe.
up	Vecteur indiquant où doit pointer le haut de l'image.
focusPoint	Point net.
sampling	Échantillonnage de l'effet.
aperture	"Ouverture" de l'objectif (correspond à la disturbance appliquée aux rayons.)

Remarques

Par défaut, la distance focale de la caméra est 1.0.

Définition à la ligne 7 du fichier PerspectiveDOF.cc.

4.27.2.2 PerspectiveDOF : :PerspectiveDOF (double *focaleDistance,* Vector3d const & *eye,* Vector3d const & *lookAt,* Vector3d const & *up,* Vector3d const & *focusPoint,* unsigned int *sampling,* double *aperture*)

Paramètres

focale-Distance	Distance focale de la caméra.
eye	Position de la caméra.
lookAt	Endroit où la caméra pointe.
up	Vecteur indiquant où doit pointer le haut de l'image.
focusPoint	Point net.
sampling	Échantillonnage de l'effet.
focusPoint	Point net.
sampling	Échantillonnage de l'effet.
aperture	"Ouverture" de l'objectif (correspond à la disturbance appliquée aux rayons.)
aperture	"Ouverture" de l'objectif (correspond à la disturbance appliquée aux rayons.)

Définition à la ligne 22 du fichier PerspectiveDOF.cc.

4.27.3 Documentation des fonctions membres

4.27.3.1 std : :vector< Ray > PerspectiveDOF : :getRay (double *u,* double *v*) const [virtual]

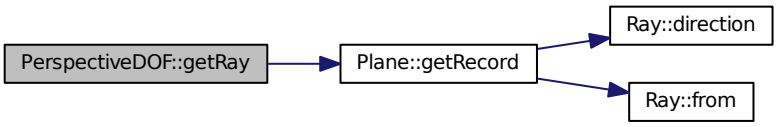
Renvoie

L'ensemble des rayons associés à la coordonnée (u,v)

Implémente Camera.

Définition à la ligne 58 du fichier PerspectiveDOF.cc.

Voici le graphe d'appel pour cette fonction :



4.27.4 Documentation des données membres

4.27.4.1 Vector3d PerspectiveDOF : :eye_ [protected]

Définition à la ligne 77 du fichier PerspectiveDOF.hpp.

4.27.4.2 double PerspectiveDOF : :focaleDistance_ [protected]

Définition à la ligne 76 du fichier PerspectiveDOF.hpp.

4.27.4.3 Vector3d PerspectiveDOF : :lookAt_ [protected]

Définition à la ligne 78 du fichier PerspectiveDOF.hpp.

4.27.4.4 Vector3d PerspectiveDOF : :up_ [protected]

Définition à la ligne 79 du fichier PerspectiveDOF.hpp.

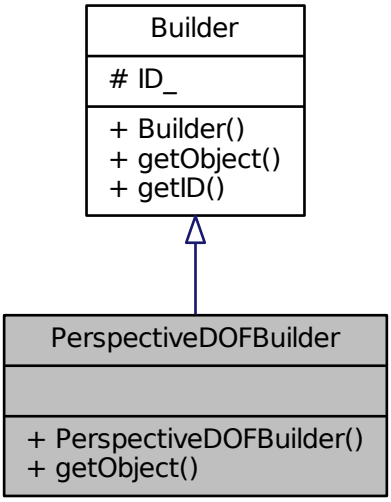
La documentation de cette classe a été générée à partir des fichiers suivants :
– src/Cameras/PerspectiveDOF.hpp

– src/Cameras/PerspectiveDOF.cc

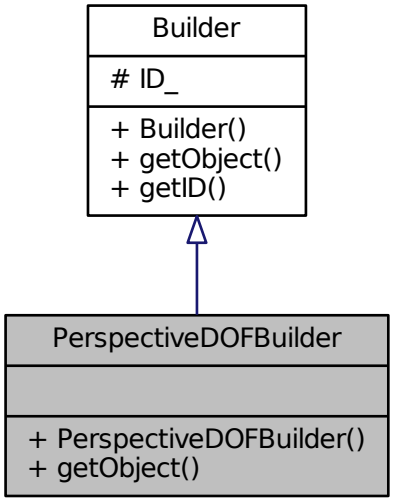
4.28 Référence de la classe PerspectiveDOFBuilder

`#include <PerspectiveDOFBuilder.hpp>`

Graphe d'héritage de PerspectiveDOFBuilder :



Graphe de collaboration de PerspectiveDOFBuilder :



Fonctions membres publiques

– PerspectiveDOFBuilder ()
– Buildable ∗ getObject (boost : :property_tree : :ptree pt, Material ∗material) const

4.28.1 Description détaillée

Définition à la ligne 17 du fichier PerspectiveDOFBuilder.hpp.

4.28.2 Documentation des constructeurs et destructeur

4.28.2.1 PerspectiveDOFBuilder : :PerspectiveDOFBuilder ( )

Définition à la ligne 6 du fichier PerspectiveDOFBuilder.cc.

4.28.3 Documentation des fonctions membres

4.28.3.1 Buildable ∗ PerspectiveDOFBuilder : :getObject (boost : :property_tree : :ptree *pt,* Material ∗ *material*) const [virtual]

Paramètres

pt	L'arbre DOM contenant les paramètres du fichier de configuration de la scène,
material	Matériau associé.

Renvoie

Un pointeur sur l'objet construit.

Implémente Builder.

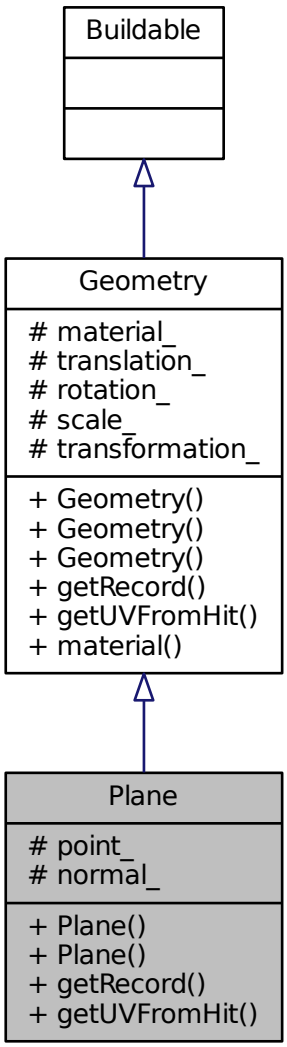
Définition à la ligne 10 du fichier PerspectiveDOFBuilder.cc.

La documentation de cette classe a été générée à partir des fichiers suivants :
- src/Builders/PerspectiveDOFBuilder.hpp
- src/Builders/PerspectiveDOFBuilder.cc

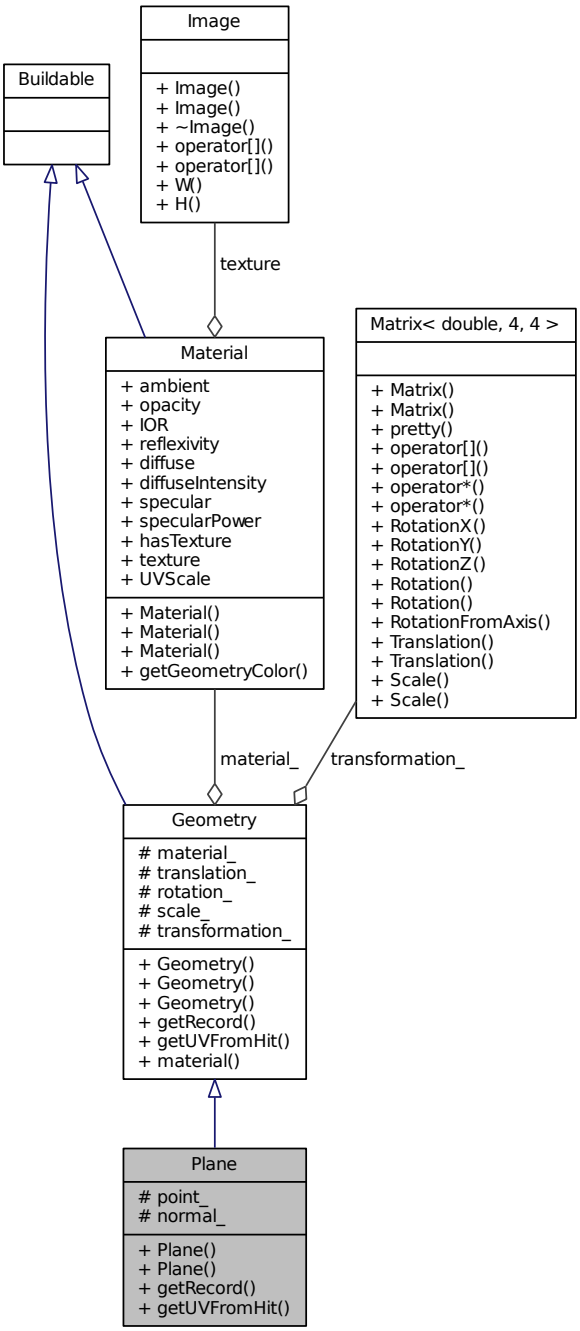
4.29 Référence de la classe Plane

```
#include <Plane.hpp>
```

Graphe d'héritage de Plane :



Graphe de collaboration de Plane :



Fonctions membres publiques

– Plane (Vector3d const &point, Vector3d const &normal)
– Plane (Vector3d const &point, Vector3d const &normal, Material ∗material)
– HitRecord getRecord (Ray const &ray) const
– Vector< double, 2 > getUVFromHit (HitRecord const &record) const

Attributs protégés

– Vector3d point_
– Vector3d normal_

4.29.1 Description détaillée

Définition à la ligne 7 du fichier Plane.hpp.

4.29.2 Documentation des constructeurs et destructeur

4.29.2.1 Plane : :Plane (Vector3d const & *point,* Vector3d const & *normal*)

Définition à la ligne 6 du fichier Plane.cc.

4.29.2.2 Plane : :Plane (Vector3d const & *point,* Vector3d const & *normal,* Material ∗ *material*)

Définition à la ligne 14 du fichier Plane.cc.

4.29.3 Documentation des fonctions membres

4.29.3.1 HitRecord Plane : :getRecord (Ray const & *ray*) const [virtual]

Renvoie

L'enregistrement lié à la collision du rayon *ray* et la géométrie.

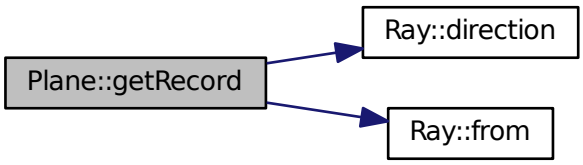
Paramètres

ray	Rayon à intersecter.

Implémente Geometry.

Définition à la ligne 30 du fichier Plane.cc.

Voici le graphe d'appel pour cette fonction :



4.29.3.2 Vector< double, 2 > Plane : :getUVFromHit (HitRecord const & *record*) const [virtual]

Renvoie

La coordonnées de texture correspondant à l'intersection enregistrée

Paramètres

record	Enregistrement de l'intersection.

Réimplémentée à partir de Geometry.

Définition à la ligne 51 du fichier Plane.cc.

4.29.4 Documentation des données membres

4.29.4.1 Vector3d Plane : :normal_ [protected]

Définition à la ligne 18 du fichier Plane.hpp.

4.29.4.2 Vector3d Plane : :point_ [protected]

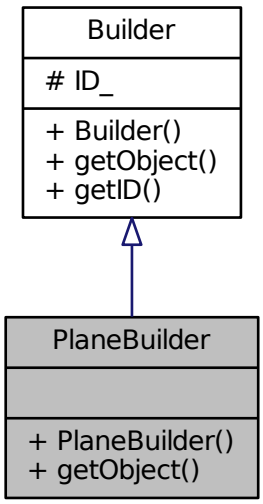
Définition à la ligne 17 du fichier Plane.hpp.

La documentation de cette classe a été générée à partir des fichiers suivants :
- src/Geometry/Plane.hpp
- src/Geometry/Plane.cc

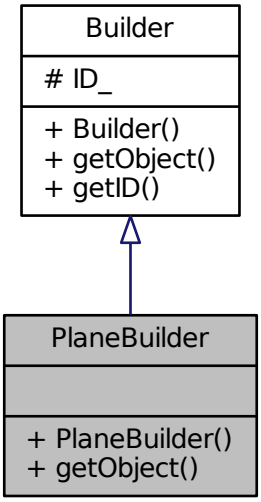
4.30 Référence de la classe PlaneBuilder

`#include <PlaneBuilder.hpp>`

Graphe d'héritage de PlaneBuilder :



Graphe de collaboration de PlaneBuilder :



Fonctions membres publiques

– PlaneBuilder ()
– Buildable ∗ getObject (boost : :property_tree : :ptree pt, Material ∗material) const

4.30.1 Description détaillée

Définition à la ligne 17 du fichier PlaneBuilder.hpp.

4.30.2 Documentation des constructeurs et destructeur

4.30.2.1 PlaneBuilder : :PlaneBuilder ( )

Définition à la ligne 6 du fichier PlaneBuilder.cc.

4.30.3 Documentation des fonctions membres

4.30.3.1 Buildable ∗ PlaneBuilder : :getObject (boost : :property_tree : :ptree *pt,* Material ∗ *material*) const [virtual]

Paramètres

pt	L'arbre DOM contenant les paramètres du fichier de configuration de la scène,
material	Matériau associé.

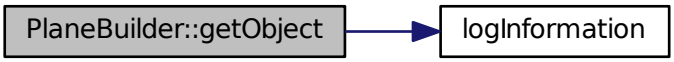
Renvoie

Un pointeur sur l'objet construit.

Implémente Builder.

Définition à la ligne 10 du fichier PlaneBuilder.cc.

Voici le graphe d'appel pour cette fonction :

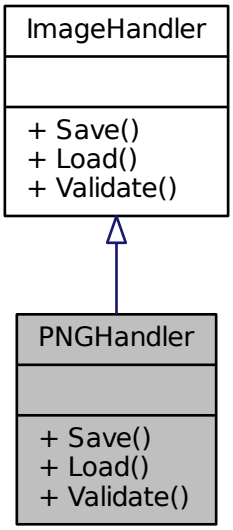


La documentation de cette classe a été générée à partir des fichiers suivants :
– src/Builders/PlaneBuilder.hpp
– src/Builders/PlaneBuilder.cc

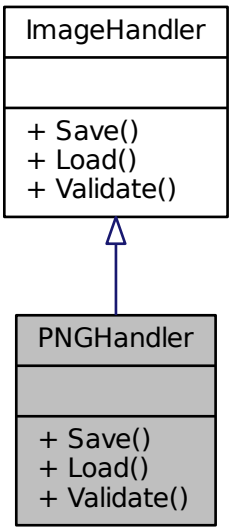
4.31 Référence de la classe PNGHandler

```
#include <PNGHandler.hpp>
```

Graphe d'héritage de PNGHandler :



Graphe de collaboration de PNGHandler :



Fonctions membres publiques

– void Save (Image const &img, string const &filename)
– Image ∗ Load (std : :string const &filename)
– bool Validate (std : :string const &filename)

4.31.1 Description détaillée

Définition à la ligne 7 du fichier PNGHandler.hpp.

4.31.2 Documentation des fonctions membres

4.31.2.1 Image ∗ PNGHandler : :Load (std : :string const & *filename*) [virtual]

Charge une image

Paramètres

filename	Chemin de l'image à charger.

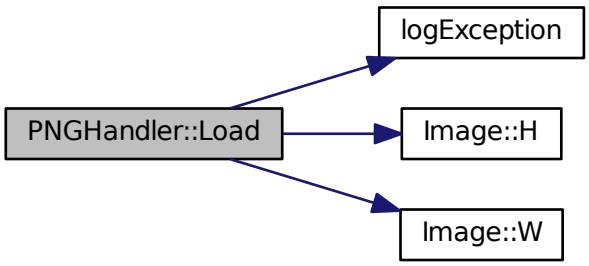
Renvoie

L'image nouvellement crée ou un pointeur nulle si le format est invalide ou l'image inexistante.

Implémente ImageHandler.

Définition à la ligne 70 du fichier PNGHandler.cc.

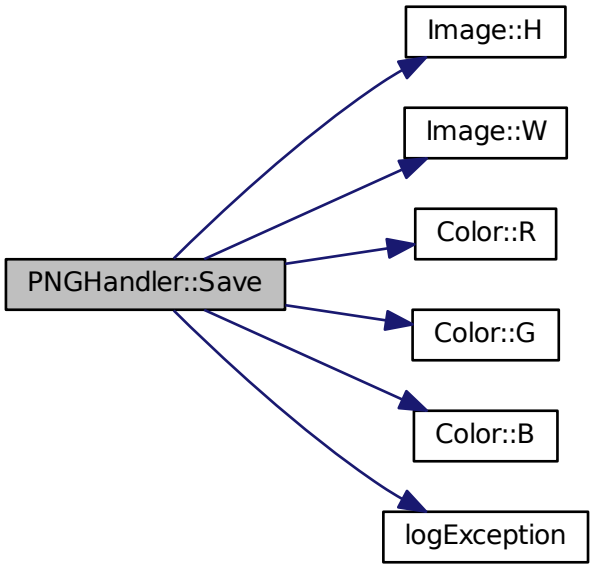
Voici le graphe d'appel pour cette fonction :



4.31.2.2 void PNGHandler : :Save (Image const & *img,* string const & *filename*)

Définition à la ligne 16 du fichier PNGHandler.cc.

Voici le graphe d'appel pour cette fonction :



4.31.2.3 bool PNGHandler : :Validate (std : :string const & *filename*) [virtual]

Renvoie

true si la handler est capable de gérer ce format, false sinon.

Implémente ImageHandler.

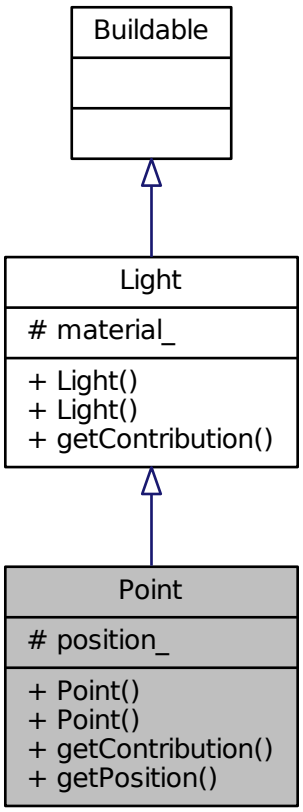
Définition à la ligne 64 du fichier PNGHandler.cc.

La documentation de cette classe a été générée à partir des fichiers suivants :
- src/Graphics/PNGHandler.hpp
- src/Graphics/PNGHandler.cc

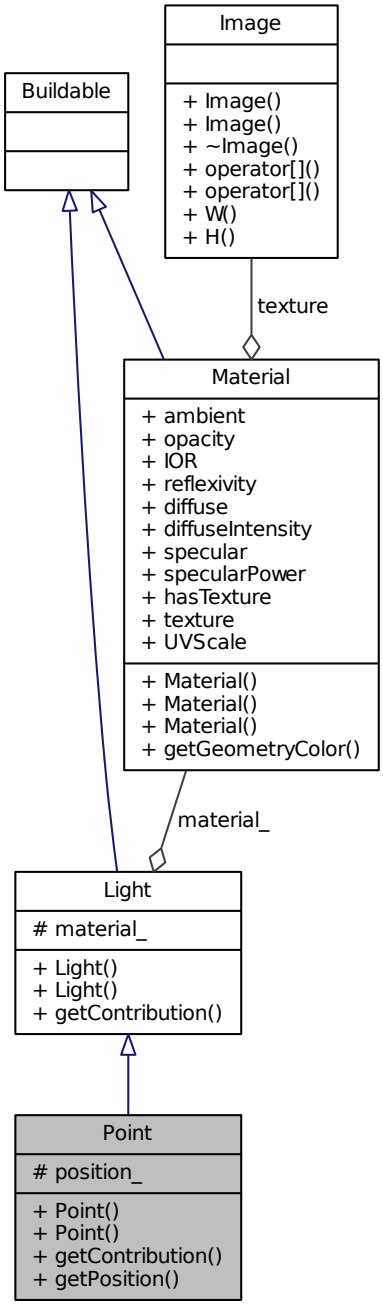
4.32 Référence de la classe Point

```
#include <Point.hpp>
```

Graphe d'héritage de Point :



Graphe de collaboration de Point :



Fonctions membres publiques

- Point (Vector3d const &position)
- Point (Vector3d const &position, Material ∗material)
- Color_d getContribution (Camera ∗camera, std : :vector< Geometry ∗ > const &geometries, HitRecord const &record) const
- Vector3d getPosition () const

Attributs protégés

- Vector3d position_

4.32.1 Description détaillée

Définition à la ligne 9 du fichier Point.hpp.

4.32.2 Documentation des constructeurs et destructeur

4.32.2.1 Point : :Point (Vector3d const & *position*)

Définition à la ligne 8 du fichier Point.cc.

4.32.2.2 Point : :Point (Vector3d const & *position,* Material ∗ *material*)

Définition à la ligne 13 du fichier Point.cc.

4.32.3 Documentation des fonctions membres

4.32.3.1 Color_d Point : :getContribution (Camera ∗ *camera,* std : :vector< Geometry ∗ > const & *geometries,* HitRecord const & *record*) const [virtual]

Renvoie

La contribution, c'est à dire l'ajout de lumière de la source à la lumière totale du point considéré.

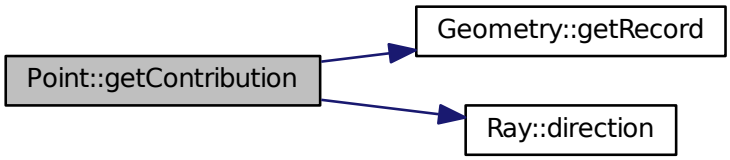
Paramètres

camera	La caméra à considérer.
geometries	Listes des géométries de la scène.
record	Enregistrement de l'intersection.

Implémente Light.

Définition à la ligne 19 du fichier Point.cc.

Voici le graphe d'appel pour cette fonction :



4.32.3.2 Vector3d Point : :getPosition ( ) const [inline]

Définition à la ligne 20 du fichier Point.hpp.

4.32.4 Documentation des données membres

4.32.4.1 Vector3d Point : :position_ [protected]

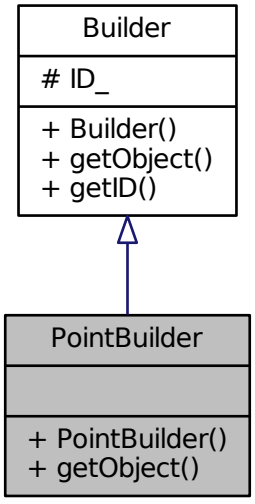
Définition à la ligne 23 du fichier Point.hpp.

La documentation de cette classe a été générée à partir des fichiers suivants :
– src/Lights/Point.hpp
– src/Lights/Point.cc

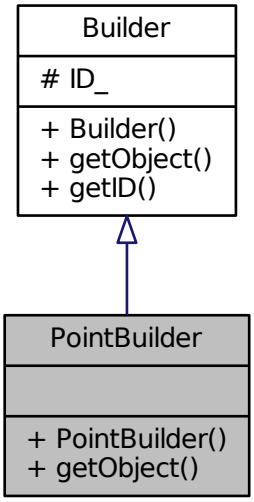
4.33 Référence de la classe PointBuilder

```
#include <PointBuilder.hpp>
```

Graphe d'héritage de PointBuilder :



Graphe de collaboration de PointBuilder :



Fonctions membres publiques

– PointBuilder ()
– Buildable ∗ getObject (boost : :property_tree : :ptree pt, Material ∗material) const

4.33.1 Description détaillée

Définition à la ligne 17 du fichier PointBuilder.hpp.

4.33.2 Documentation des constructeurs et destructeur

4.33.2.1 PointBuilder : :PointBuilder ( )

Définition à la ligne 6 du fichier PointBuilder.cc.

4.33.3 Documentation des fonctions membres

4.33.3.1 Buildable ∗ PointBuilder : :getObject (boost : :property_tree : :ptree *pt,* Material ∗ *material*) const [virtual]

Paramètres

pt	L'arbre DOM contenant les paramètres du fichier de configuration de la scène,
material	Matériau associé.

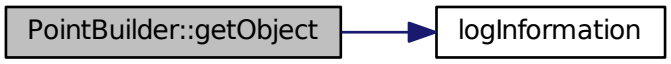
Renvoie

Un pointeur sur l'objet construit.

Implémente Builder.

Définition à la ligne 10 du fichier PointBuilder.cc.

Voici le graphe d'appel pour cette fonction :



La documentation de cette classe a été générée à partir des fichiers suivants :
– src/Builders/PointBuilder.hpp
– src/Builders/PointBuilder.cc

4.34 Référence de la classe Ray

`#include <Ray.hpp>`

Fonctions membres publiques

– Ray (Vector3d const &from, Vector3d const &direction)
– string pretty () const
– Vector3d from () const
– Vector3d direction () const

Amis

– std : :ostream & operator<< (std : :ostream &oss, const Ray &r)

4.34.1 Description détaillée

Définition à la ligne 13 du fichier Ray.hpp.

4.34.2 Documentation des constructeurs et destructeur

4.34.2.1 Ray : :Ray (Vector3d const & *from,* Vector3d const & *direction*)

Paramètres

from	Point de départ du rayon.
direction	Direction que suivra le rayon lumineux.

Remarques

Le paramètre direction sera automatiquement normalisé.

Définition à la ligne 7 du fichier Ray.cc.

4.34.3 Documentation des fonctions membres

4.34.3.1 Vector3d Ray : :direction ( ) const

Renvoie

La direction normalisée du rayon.

Définition à la ligne 24 du fichier Ray.cc.

4.34.3.2 Vector3d Ray : :from ( ) const

Renvoie

Le point de départ du rayon

Définition à la ligne 23 du fichier Ray.cc.

4.34.3.3 string Ray : :pretty ( ) const

Définition à la ligne 15 du fichier Ray.cc.

4.34.4 Documentation des fonctions amies et associées

4.34.4.1 std : :ostream& operator<< (std : :ostream & *oss,* const Ray & *r*) [friend]

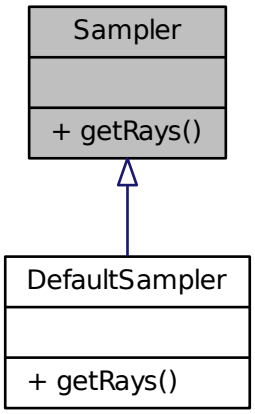
Définition à la ligne 25 du fichier Ray.hpp.

La documentation de cette classe a été générée à partir des fichiers suivants :
– src/Ray.hpp
– src/Ray.cc

4.35 Référence de la classe Sampler

`#include <Sampler.hpp>`

Graphe d'héritage de Sampler :



Fonctions membres publiques

– virtual std : :vector< Ray > getRays (Scene const &scene, unsigned int X, unsigned int Y)=0

4.35.1 Description détaillée

Définition à la ligne 16 du fichier Sampler.hpp.

4.35.2 Documentation des fonctions membres

4.35.2.1 virtual std : :vector<Ray> Sampler : :getRays (Scene const & *scene,* unsigned int *X,* unsigned int *Y*) [pure virtual]

Renvoie

Ensemble des rayons à lancer pour la scène et les coordonnées passés en paramètres.

Implémenté dans DefaultSampler.

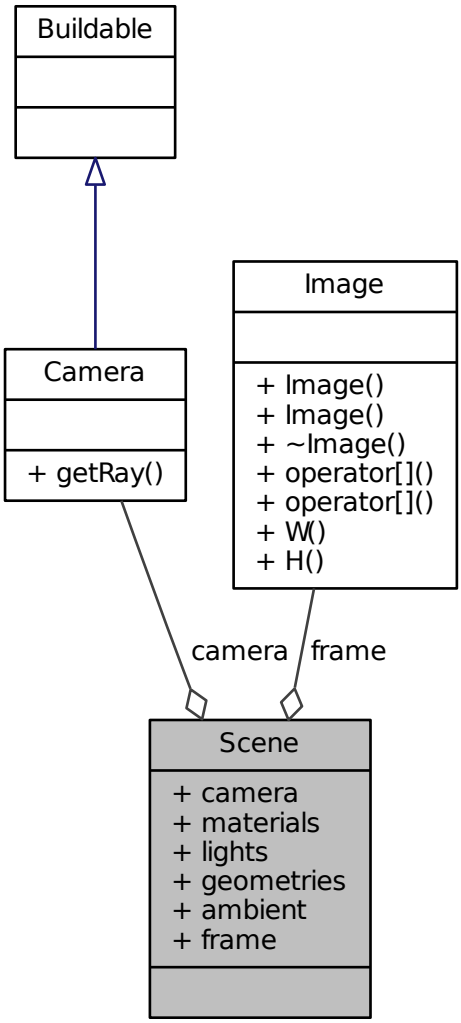
La documentation de cette classe a été générée à partir du fichier suivant :

– src/Samplers/Sampler.hpp

4.36 Référence de la structure Scene

`#include <Scene.hpp>`

Graphe de collaboration de Scene :



Attributs publics

– Camera ∗ camera
– std : :map< std : :string, Material ∗ > materials
– std : :vector< Light ∗ > lights
– std : :vector< Geometry ∗ > geometries
– Color_d ambient
– Image ∗ frame

4.36.1 Description détaillée

Définition à la ligne 27 du fichier Scene.hpp.

4.36.2 Documentation des données membres

4.36.2.1 Color_d Scene : :ambient

Définition à la ligne 33 du fichier Scene.hpp.

4.36.2.2 Camera∗ Scene : :camera

Définition à la ligne 28 du fichier Scene.hpp.

4.36.2.3 Image∗ Scene : :frame

Définition à la ligne 35 du fichier Scene.hpp.

4.36.2.4 std : :vector<Geometry∗> Scene : :geometries

Définition à la ligne 31 du fichier Scene.hpp.

4.36.2.5 std : :vector<Light∗> Scene : :lights

Définition à la ligne 30 du fichier Scene.hpp.

4.36.2.6 std : :map< std : :string, Material∗ > Scene : :materials

Définition à la ligne 29 du fichier Scene.hpp.

La documentation de cette structure a été générée à partir du fichier suivant :
– src/Scene/Scene.hpp

4.37 Référence de la classe SceneReader

```
#include <SceneReader.hpp>
```

Fonctions membres publiques

– void addBuilder (Builder ∗builder)
– Scene read (std : :string const &filename) const

Attributs protégés

– std : :map< std : :string, Builder ∗ > builders_

4.37.1 Description détaillée

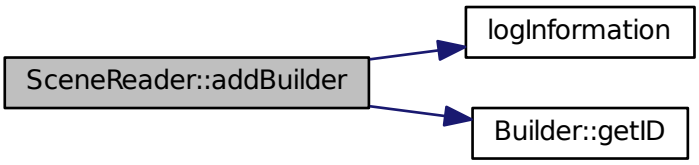
Définition à la ligne 9 du fichier SceneReader.hpp.

4.37.2 Documentation des fonctions membres

4.37.2.1 void SceneReader : :addBuilder (Builder ∗ *builder*)

Définition à la ligne 26 du fichier SceneReader.cc.

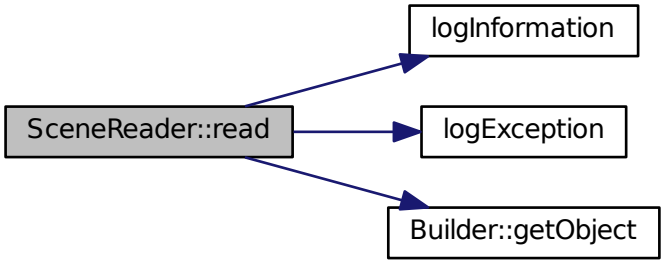
Voici le graphe d'appel pour cette fonction :



4.37.2.2 Scene SceneReader : :read (std : :string const & *filename*) const

Définition à la ligne 34 du fichier SceneReader.cc.

Voici le graphe d'appel pour cette fonction :



4.37.3 Documentation des données membres

4.37.3.1 std : :map<std : :string, Builder*> SceneReader : :builders_ [protected]

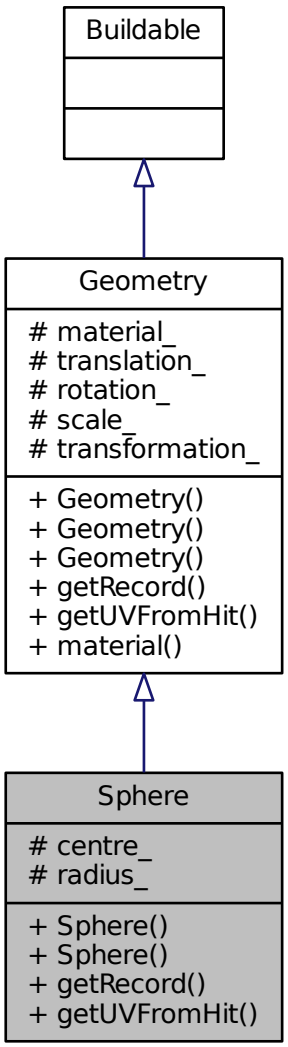
Définition à la ligne 18 du fichier SceneReader.hpp.

La documentation de cette classe a été générée à partir des fichiers suivants :
– src/Scene/SceneReader.hpp
– src/Scene/SceneReader.cc

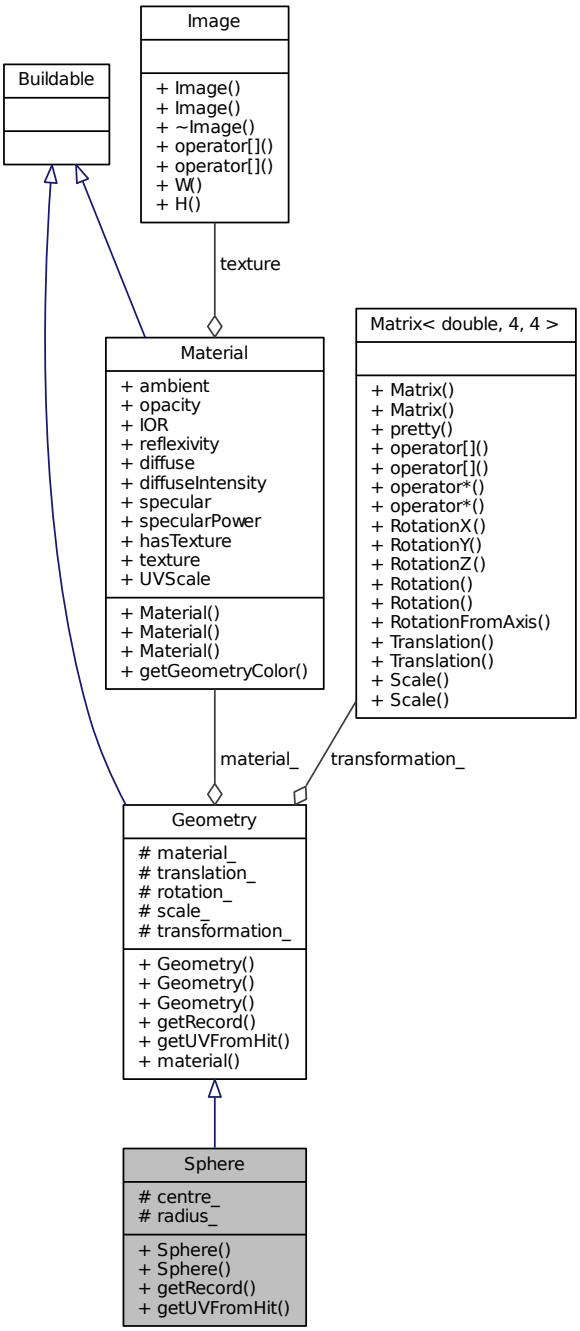
4.38 Référence de la classe Sphere

```
#include <Sphere.hpp>
```

Graphe d'héritage de Sphere :



Graphe de collaboration de Sphere :



Fonctions membres publiques

- Sphere (Vector3d const &centre, double radius)
- Sphere (Vector3d const &centre, double radius, Material *material)
- HitRecord getRecord (Ray const &ray) const
- Vector< double, 2 > getUVFromHit (HitRecord const &record) const

Attributs protégés

- Vector3d centre_
- double radius_

4.38.1 Description détaillée

Définition à la ligne 7 du fichier Sphere.hpp.

4.38.2 Documentation des constructeurs et destructeur

4.38.2.1 Sphere : :Sphere (Vector3d const & *centre,* double *radius*)

Paramètres

centre	Position du centre de la sphère.
radius	Rayon de la sphère.

Définition à la ligne 3 du fichier Sphere.cc.

4.38.2.2 Sphere : :Sphere (Vector3d const & *centre,* double *radius,* Material * *material*)

Paramètres

centre	Position du centre de la sphère.
radius	Rayon de la sphère.
material	Matériaux de la sphère.

Définition à la ligne 9 du fichier Sphere.cc.

4.38.3 Documentation des fonctions membres

4.38.3.1 HitRecord Sphere : :getRecord (Ray const & *ray*) const [virtual]

Renvoie

L'enregistrement lié à la collision du rayon *ray* et la géométrie.

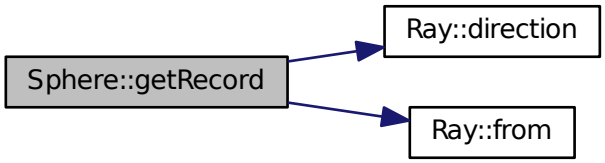
Paramètres

ray	Rayon à intersecter.

Implémente Geometry.

Définition à la ligne 15 du fichier Sphere.cc.

Voici le graphe d'appel pour cette fonction :



4.38.3.2 Vector< double, 2 > Sphere : :getUVFromHit (HitRecord const & *record*) const [virtual]

Renvoie

La coordonnées de texture correspondant à l'intersection enregistrée

Paramètres

record	Enregistrement de l'intersection.

Réimplémentée à partir de Geometry.

Définition à la ligne 50 du fichier Sphere.cc.

4.38.4 Documentation des données membres

4.38.4.1 Vector3d Sphere : :centre_ [protected]

Définition à la ligne 28 du fichier Sphere.hpp.

4.38.4.2 double Sphere : :radius_ [protected]

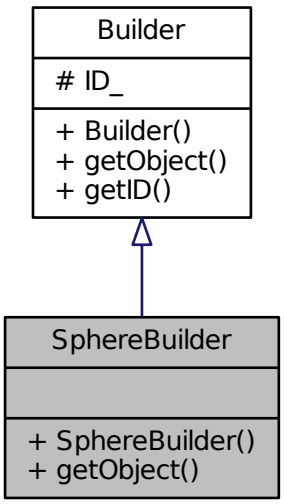
Définition à la ligne 29 du fichier Sphere.hpp.

La documentation de cette classe a été générée à partir des fichiers suivants :
- src/Geometry/Sphere.hpp
- src/Geometry/Sphere.cc

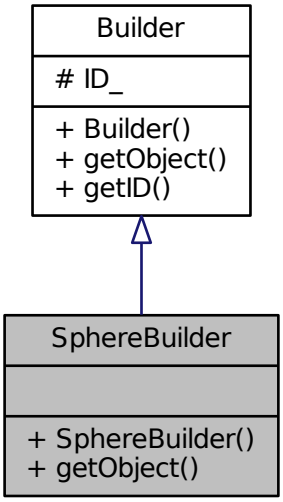
4.39 Référence de la classe SphereBuilder

`#include <SphereBuilder.hpp>`

Graphe d'héritage de SphereBuilder :



Graphe de collaboration de SphereBuilder :



Fonctions membres publiques

– SphereBuilder ()
– Buildable ∗ getObject (boost : :property_tree : :ptree pt, Material ∗material) const

4.39.1 Description détaillée

Définition à la ligne 17 du fichier SphereBuilder.hpp.

4.39.2 Documentation des constructeurs et destructeur

4.39.2.1 SphereBuilder : :SphereBuilder ( )

Définition à la ligne 6 du fichier SphereBuilder.cc.

4.39.3 Documentation des fonctions membres

4.39.3.1 Buildable ∗ SphereBuilder : :getObject (boost : :property_tree : :ptree *pt,* Material ∗ *material*) const [virtual]

Paramètres

pt	L'arbre DOM contenant les paramètres du fichier de configuration de la scène,
material	Matériau associé.

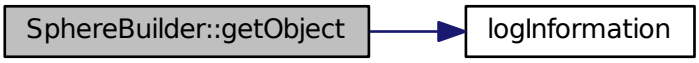
Renvoie

Un pointeur sur l'objet construit.

Implémente Builder.

Définition à la ligne 10 du fichier SphereBuilder.cc.

Voici le graphe d'appel pour cette fonction :

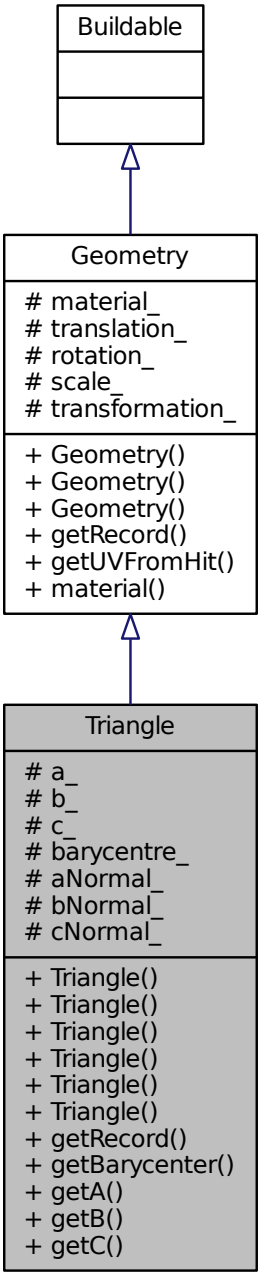


La documentation de cette classe a été générée à partir des fichiers suivants :
– src/Builders/SphereBuilder.hpp
– src/Builders/SphereBuilder.cc

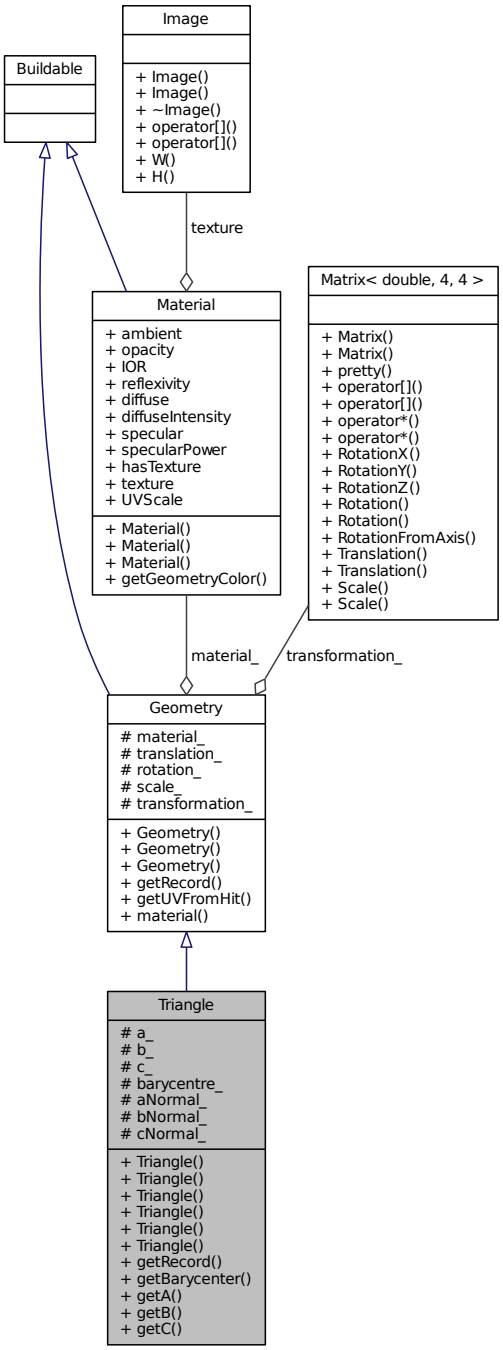
4.40 Référence de la classe Triangle

```
#include <Triangle.hpp>
```

Graphe d'héritage de Triangle :



Graphe de collaboration de Triangle :



Fonctions membres publiques

- Triangle (Vector3d const &a, Vector3d const &b, Vector3d const &c)
- Triangle (Vector3d const &a, Vector3d const &b, Vector3d const &c, Material ∗material)
- Triangle (Vector3d const &a, Vector3d const &b, Vector3d const &c, Vector3d const &normal)
- Triangle (Vector3d const &a, Vector3d const &b, Vector3d const &c, Vector3d const &normal, Material ∗material)
- Triangle (Vector3d const &a, Vector3d const &b, Vector3d const &c, Vector3d const &aNormal, Vector3d const &bNormal, Vector3d const &cNormal)
- Triangle (Vector3d const &a, Vector3d const &b, Vector3d const &c, Vector3d const &aNormal, Vector3d const &bNormal, Vector3d const &cNormal, Material ∗material)
- HitRecord getRecord (Ray const &ray) const
- Vector3d getBarycenter () const
- Vector3d getA () const
- Vector3d getB () const
- Vector3d getC () const

Attributs protégés

- Vector3d a_
- Vector3d b_
- Vector3d c_
- Vector3d barycentre_
- Vector3d aNormal_
- Vector3d bNormal_
- Vector3d cNormal_

4.40.1 Description détaillée

Définition à la ligne 7 du fichier Triangle.hpp.

4.40.2 Documentation des constructeurs et destructeur

4.40.2.1 Triangle : :Triangle (Vector3d const & *a,* Vector3d const & *b,* Vector3d const & *c*)

Définition à la ligne 4 du fichier Triangle.cc.

4.40.2.2 Triangle : :Triangle (Vector3d const & *a,* Vector3d const & *b,* Vector3d const & *c,* Material ∗ *material*)

Définition à la ligne 14 du fichier Triangle.cc.

4.40.2.3 Triangle : :Triangle (Vector3d const & *a,* Vector3d const & *b,* Vector3d const & *c,* Vector3d const & *normal*)

Définition à la ligne 26 du fichier Triangle.cc.

4.40.2.4 Triangle : :Triangle (Vector3d const & *a,* Vector3d const & *b,* Vector3d const & *c,* Vector3d const & *normal,* Material ∗ *material*)

Définition à la ligne 38 du fichier Triangle.cc.

4.40.2.5 Triangle : :Triangle (Vector3d const & *a,* Vector3d const & *b,* Vector3d const & *c,* Vector3d const & *aNormal,* Vector3d const & *bNormal,* Vector3d const & *cNormal*)

Définition à la ligne 51 du fichier Triangle.cc.

4.40.2.6 Triangle : :Triangle (Vector3d const & *a,* Vector3d const & *b,* Vector3d const & *c,* Vector3d const & *aNormal,* Vector3d const & *bNormal,* Vector3d const & *cNormal,* Material ∗ *material*)

4.40.3 Documentation des fonctions membres

4.40.3.1 Vector3d Triangle : :getA ( ) const

Définition à la ligne 69 du fichier Triangle.cc.

4.40.3.2 Vector3d Triangle : :getB ( ) const

Définition à la ligne 70 du fichier Triangle.cc.

4.40.3.3 Vector3d Triangle : :getBarycenter ( ) const

Définition à la ligne 65 du fichier Triangle.cc.

4.40.3.4 Vector3d Triangle : :getC ( ) const

Définition à la ligne 71 du fichier Triangle.cc.

4.40.3.5 HitRecord Triangle : :getRecord (Ray const & *ray*) const [virtual]

Renvoie

L'enregistrement lié à la collision du rayon *ray* et la géométrie.

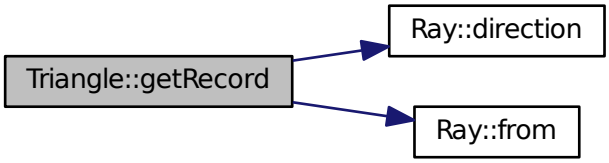
Paramètres

ray	Rayon à intersecter.

Implémente Geometry.

Définition à la ligne 74 du fichier Triangle.cc.

Voici le graphe d'appel pour cette fonction :



4.40.4 Documentation des données membres

4.40.4.1 Vector3d Triangle : :a_ [protected]

Définition à la ligne 42 du fichier Triangle.hpp.

4.40.4.2 Vector3d Triangle : :aNormal_ [protected]

Définition à la ligne 47 du fichier Triangle.hpp.

4.40.4.3 Vector3d Triangle : :b_ [protected]

Définition à la ligne 43 du fichier Triangle.hpp.

4.40.4.4 Vector3d Triangle : :barycentre_ [protected]

Définition à la ligne 45 du fichier Triangle.hpp.

4.40.4.5 Vector3d Triangle : :bNormal_ [protected]

Définition à la ligne 48 du fichier Triangle.hpp.

4.40.4.6 Vector3d Triangle : :c_ [protected]

Définition à la ligne 44 du fichier Triangle.hpp.

4.40.4.7 Vector3d Triangle : :cNormal_ [protected]

Définition à la ligne 49 du fichier Triangle.hpp.

La documentation de cette classe a été générée à partir des fichiers suivants :
– src/Geometry/Triangle.hpp
– src/Geometry/Triangle.cc

4.41 Référence du modèle de la classe Vector< P, N >

```
#include <Vector.hpp>
```

Fonctions membres publiques

– Vector ()
– Vector (P v)
– Vector (const Vector< P, N > &v)
– Vector (P const &X, P const &Y, P const &Z)
– P Length () const
– P SquaredLength () const
– Vector< P, N > Normalized () const
– Vector< P, 4 > Homogenous () const
– Vector< P, N > Project (Vector< P, N > const &v2) const
– string pretty () const
– bool operator== (Vector< P, N > const &v2) const
– bool operator != (Vector< P, N > const &v2) const
– Vector< P, N > operator+ (const Vector< P, N > &v2) const
– Vector< P, N > & operator+= (const Vector< P, N > &v2)
– Vector< P, N > operator- () const
– Vector< P, N > operator- (const Vector< P, N > &v2) const
– Vector< P, N > operator∗ (double const &d) const
– Vector< P, N > operator/ (double const &d) const
– Vector< P, N > & operator/= (double const &d)
– P & operator[] (unsigned int i)
– P const & operator[] (unsigned int i) const
– P X () const
– P Y () const
– P Z () const
– P T () const

Fonctions membres publiques statiques

– static P Dot (const Vector< P, N > &v1, const Vector< P, N > &v2)
– static Vector< P, 3 > Cross (const Vector< P, 3 > &v1, const Vector< P, 3 > &v2)

Attributs protégés

– P _values [N]

Amis

– Vector< P, N > operator∗ (double const &d, Vector< P, N > const &v)
– ostream & operator<< (ostream &oss, const Vector< P, N > &v)

4.41.1 Description détaillée

template<typename P, int N>class Vector< P, N >

Définition à la ligne 29 du fichier Vector.hpp.

4.41.2 Documentation des constructeurs et destructeur

4.41.2.1 template<typename P, int N> Vector< P, N > : :Vector ( )

Constructeur par défaut, initialise toutes les composantes à 0 ;

4.41.2.2 template<typename P, int N> Vector< P, N > : :Vector (P *v*)

Initialise toutes les composantes à v

Paramètres

v	Valeur de toutes les composantes.

4.41.2.3 template<typename P, int N> Vector< P, N > : :Vector (const Vector< P, N > & *v*)

Constructeur de copie, initialise toutes les composantes à la même valeur que le vecteur de même taille passé en paramètre.

Paramètres

v	Vecteur à copier.

4.41.2.4 template<typename P, int N> Vector< P, N > : :Vector (P const & *X,* P const & *Y,* P const & *Z*)

4.41.3 Documentation des fonctions membres

4.41.3.1 template<typename P, int N> static Vector<P, 3> Vector< P, N > : :Cross (const Vector< P, 3 > & *v1,* const Vector< P, 3 > & *v2*) [static]

Renvoie le produit vectoriel de deux vecteur de dimension 3

Paramètres

v1	Membre de gauche du produit vectoriel.
v2	Membre de droite du produit vectoriel.

Renvoie

Résultat du produit vectoriel.

4.41.3.2 template<typename P, int N> static P Vector< P, N > : :Dot (const Vector< P, N > & *v1,* const Vector< P, N > & *v2*) [static]

Renvoie la produit scalaire des deux vecteurs passés en paramètre.

Paramètres

v1	Membre de gauche du produit scalaire.
v2	Membre de droite du produit scalaire.

Renvoie

Résultat du produit scalaire.

4.41.3.3 template<typename P, int N> Vector<P, 4> Vector< P, N > : :Homogenous ( ) const

Renvoie

Le vecteur en coordonnées homogènes.

Remarques

Attention, les composantes d'indices > 3 seront perdues !

4.41.3.4 template<typename P, int N> P Vector< P, N > : :Length ( ) const

Renvoie la taille du vecteur calculé comme la racine du produit scalaire du vecteur avec lui même.

Renvoie

Taille du vecteur.

4.41.3.5 template<typename P, int N> Vector<P, N> Vector< P, N > : :Normalized ( ) const

Renvoie la versione normalisée du vecteur (i.e. de taille unitaire).

Renvoie

Vecteur normalisé.

4.41.3.6 template<typename P, int N> bool Vector< P, N > : :operator != (Vector< P, N > const & *v2*) const

4.41.3.7 template<typename P, int N> Vector<P, N> Vector< P, N > : :operator* (double const & *d*) const

4.41.3.8 template<typename P, int N> Vector<P, N> Vector< P, N > : :operator+ (const Vector< P, N > & *v2*) const

4.41.3.9 template<typename P, int N> Vector<P, N>& Vector< P, N > : :operator+= (const Vector< P, N > & *v2*)

4.41.3.10 template<typename P, int N> Vector<P, N> Vector< P, N > : :operator- ( ) const

4.41.3.11 template<typename P, int N> Vector<P, N> Vector< P, N > : :operator- (const Vector< P, N > & *v2*) const

4.41.3.12 template<typename P, int N> Vector<P, N> Vector< P, N > : :operator/ (double const & *d*) const

4.41.3.13 template<typename P, int N> Vector<P, N>& Vector< P, N > : :operator/= (double const & *d*)

4.41.3.14 template<typename P, int N> bool Vector< P, N > : :operator== (Vector< P, N > const & *v2*) const

4.41.3.15 template<typename P, int N> P& Vector< P, N > : :operator[] (unsigned int *i*)

4.41.3.16 template<typename P, int N> P const& Vector< P, N > : :operator[] (unsigned int *i*) const

4.41.3.17 template<typename P, int N> string Vector< P, N > : :pretty ( ) const

4.41.3.18 template<typename P, int N> Vector<P, N> Vector< P, N > : :Project (Vector< P, N > const & *v2*) const

4.41.3.19 template<typename P, int N> P Vector< P, N > : :SquaredLength ( ) const

Renvoie la taille du vecteur au carré.

Renvoie

Taille du vecteur au carré.

4.41.3.20 template<typename P, int N> P Vector< P, N > : :T ( ) const

Renvoie la valeur de la quatrième composante si elle existe. Si la composante n'existe pas, la compilation est interrompue.

4.41.3.21 template<typename P, int N> P Vector< P, N > : :X ( ) const

Renvoie la valeur de la première composante

4.41.3.22 template<typename P, int N> P Vector< P, N > : :Y ( ) const

Renvoie la valeur de la secondes composante si elle existe. Si la composante n'existe pas, la compilation est interrompue.

4.41.3.23 template<typename P, int N> P Vector< P, N > : :Z ( ) const

Renvoie la valeur de la troisième composante si elle existe. Si la composante n'existe pas, la compilation est interrompue.

4.41.4 Documentation des fonctions amies et associées

4.41.4.1 template<typename P, int N> Vector<P, N> operator* (double const & *d*, Vector< P, N > const & *v*) [friend]

Définition à la ligne 118 du fichier Vector.hpp.

4.41.4.2 template<typename P, int N> ostream& operator<< (ostream & *oss*, const Vector< P, N > & *v*) [friend]

Définition à la ligne 127 du fichier Vector.hpp.

4.41.5 Documentation des données membres

4.41.5.1 template<typename P, int N> P Vector< P, N > : :_values[N] [protected]

Définition à la ligne 160 du fichier Vector.hpp.

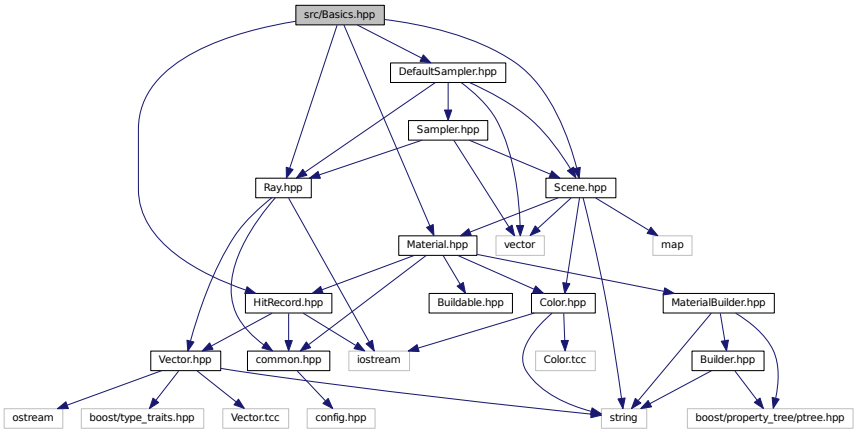
La documentation de cette classe a été générée à partir du fichier suivant :
– src/Maths/Vector.hpp

Chapitre 5

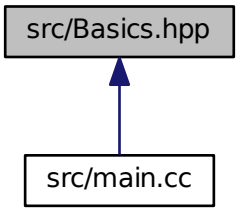
Documentation des fichiers

5.1 Référence du fichier src/Basics.hpp

`#include <Ray.hpp> #include <HitRecord.hpp> #include <Material.hpp> #include <DefaultSampler.hpp> #include <Scene.hpp>` Graphe des dépendances par inclusion de Basics.hpp :



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



5.2 Référence du fichier src/Buildable.hpp

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

Classes

– class Buildable

5.2.1 Description détaillée

Auteur

Maxime Gaudin

Date

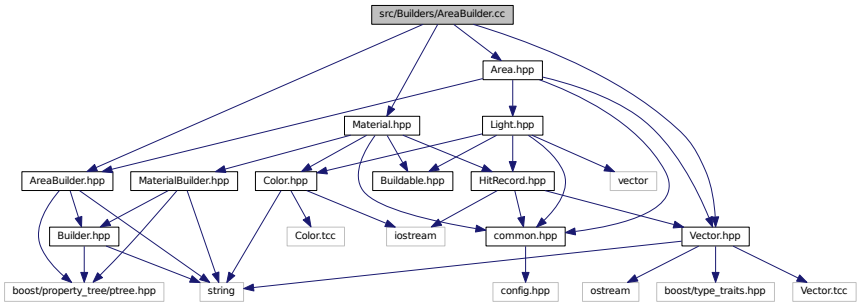
2011

Cette classe est l'interface de toute les classes pouvant être crée via le fichier de description de la scène. Il ne définit aucun membre abstrait et reste purement sémentique.

Définition dans le fichier Buildable.hpp.

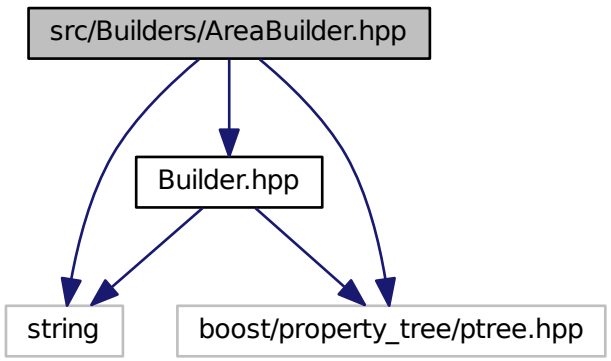
5.3 Référence du fichier src/Builders/AreaBuilder.cc

`#include "AreaBuilder.hpp" #include <Vector.hpp> #include <Area.hpp> #include <Material.hpp>` Graphe des dépendances par inclusion de AreaBuilder.cc :

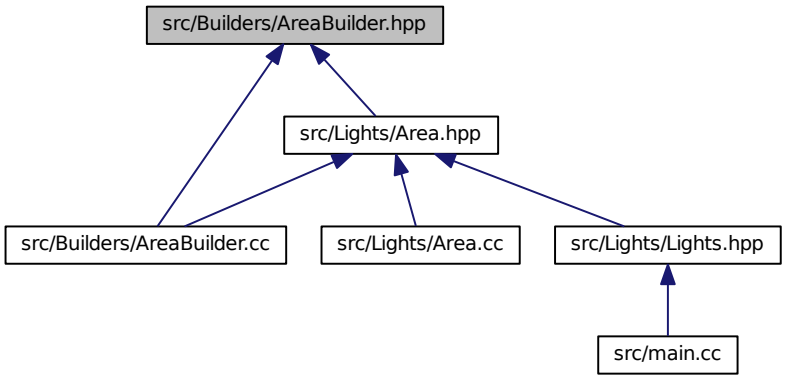


5.4 Référence du fichier src/Builders/AreaBuilder.hpp

`#include <Builder.hpp> #include <string> #include <boost/property-_tree/ptree.hpp>` Graphe des dépendances par inclusion de AreaBuilder.hpp :



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



Classes

– class AreaBuilder

5.4.1 Description détaillée

Auteur

Maxime Gaudin

Date

2011

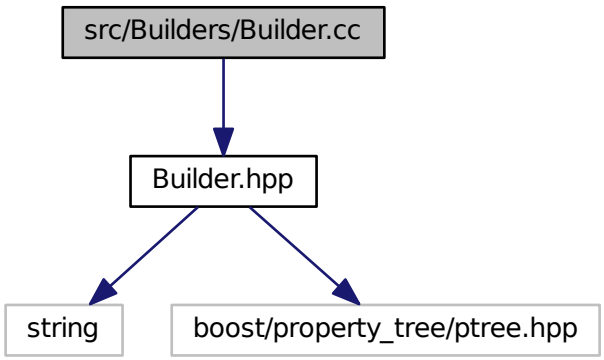
Builder associé à l'objet AreaLight.

Définition dans le fichier AreaBuilder.hpp.

5.5 Référence du fichier src/Builders/Builder.cc

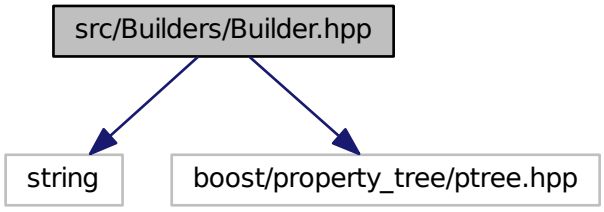
`#include "Builder.hpp"` Graphe des dépendances par inclusion de Builder.cc-

:

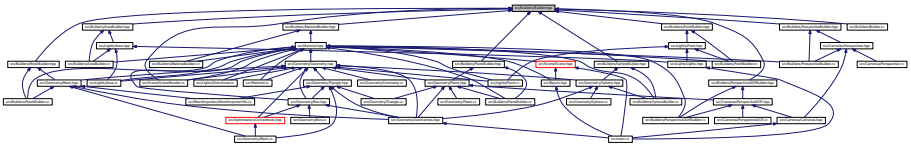


5.6 Référence du fichier src/Builders/Builder.hpp

`#include <string>  #include <boost/property_tree/ptree.-hpp>` Graphe des dépendances par inclusion de Builder.hpp :



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



Classes

– class Builder

5.6.1 Description détaillée

Auteur

Maxime Gaudin

Date

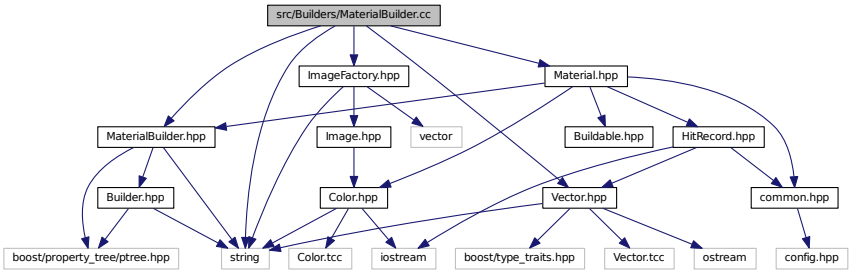
2011

Ce fichier déclare la classe abstraite de tous les constructeurs d'objets (d'interface - Buildable). C'est ces builders qui seronts passés au SceneReader.

Définition dans le fichier Builder.hpp.

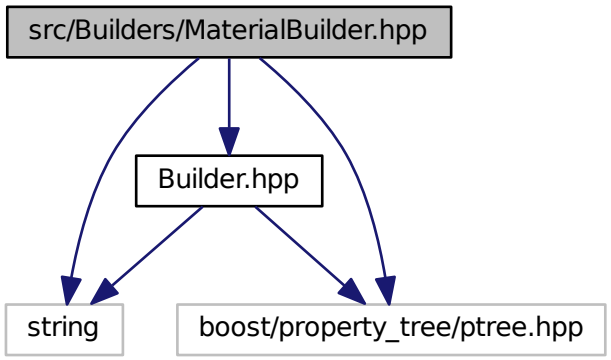
5.7 Référence du fichier src/Builders/MaterialBuilder.cc

`#include "MaterialBuilder.hpp" #include <string> #include <Vector.hpp> #include <Material.hpp> #include <Image-Factory.hpp>` Graphe des dépendances par inclusion de MaterialBuilder.cc :

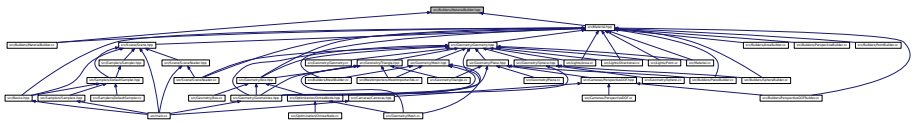


5.8 Référence du fichier src/Builders/MaterialBuilder.hpp

`#include <Builder.hpp>` `#include <string>` `#include <boost/property_tree/ptree.hpp>` Graphe des dépendances par inclusion de MaterialBuilder.hpp :



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



Classes

– class MaterialBuilder

5.8.1 Description détaillée

Auteur

Maxime Gaudin

Date

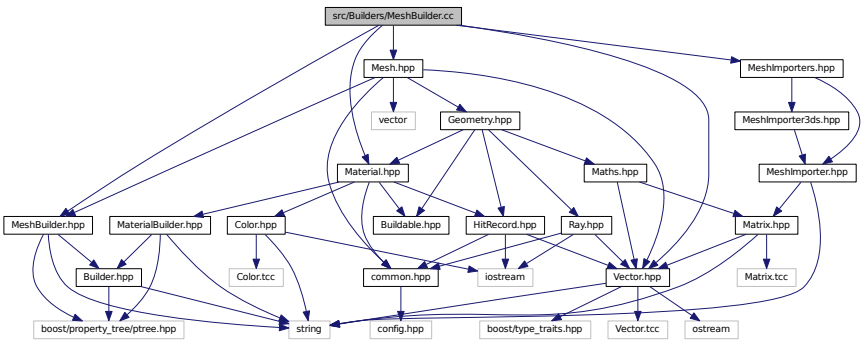
2011

Builder associé à l'objet Material.

Définition dans le fichier MaterialBuilder.hpp.

5.9 Référence du fichier src/Builders/MeshBuilder.cc

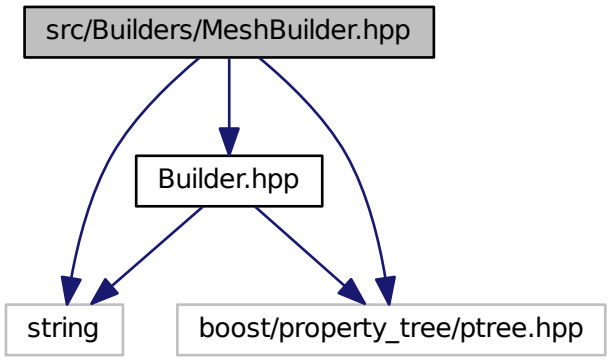
`#include "MeshBuilder.hpp" #include <Vector.hpp> #include <Mesh.hpp> #include <Material.hpp> #include <MeshImporters.-hpp>` Graphe des dépendances par inclusion de MeshBuilder.cc :



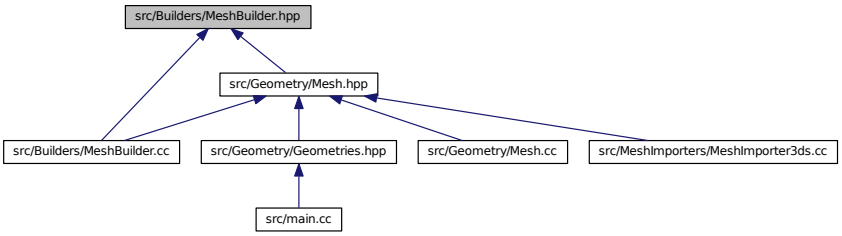
5.10 Référence du fichier src/Builders/MeshBuilder.hpp

`#include <Builder.hpp> #include <string> #include <boost/property-`

_tree/ptree.hpp> Graphe des dépendances par inclusion de MeshBuilder.hpp :



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



Classes

– class MeshBuilder

5.10.1 Description détaillée

Auteur

Maxime Gaudin

Date

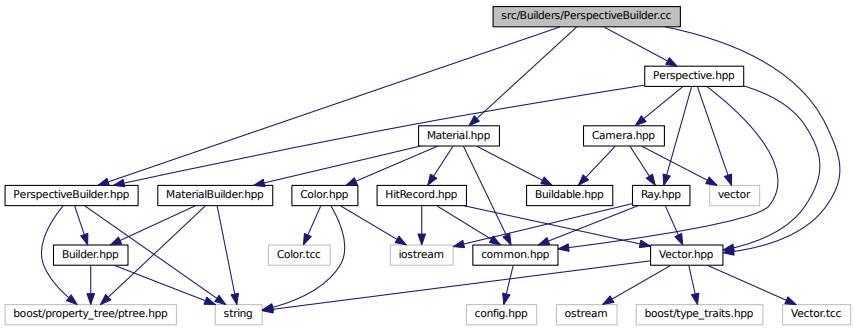
2011

Builder associé à l'objet Mesh.

Définition dans le fichier MeshBuilder.hpp.

5.11 Référence du fichier src/Builders/PerspectiveBuilder.cc

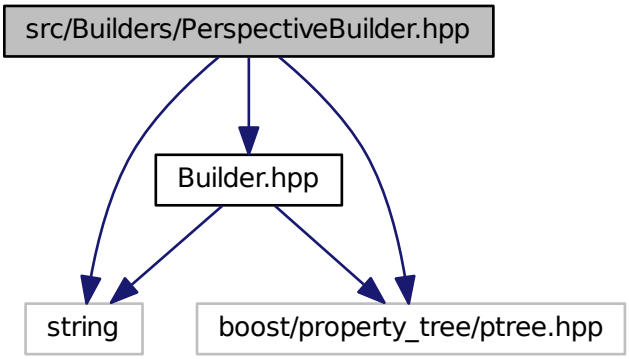
`#include "PerspectiveBuilder.hpp" #include <Vector.hpp>` × `#include <Perspective.hpp> #include <Material.hpp>` Graphe des dépendances par inclusion de PerspectiveBuilder.cc :



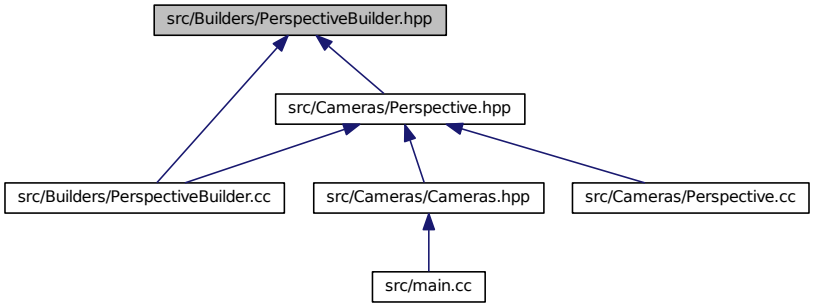
5.12 Référence du fichier src/Builders/PerspectiveBuilder.hpp

`#include <Builder.hpp>` `#include <string>` `#include <boost/property-_tree/ptree.hpp>` Graphe des dépendances par inclusion de Perspective-

Builder.hpp :



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



Classes

– class PerspectiveBuilder

5.12.1 Description détaillée

Auteur

Maxime Gaudin

Date

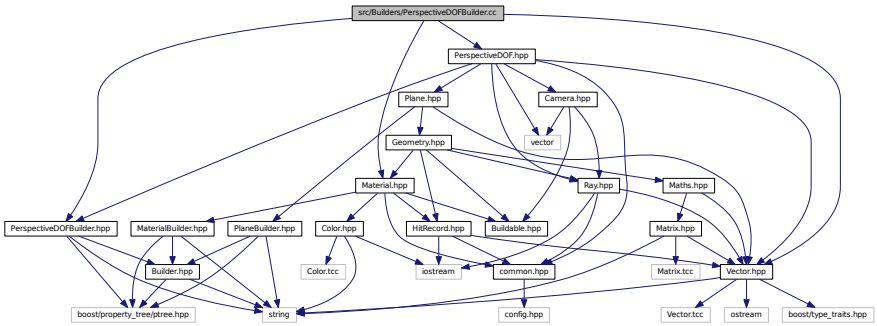
2011

Builder associé à l'objet Perspective.

Définition dans le fichier PerspectiveBuilder.hpp.

5.13 Référence du fichier src/Builders/PerspectiveDOFBuilder.cc

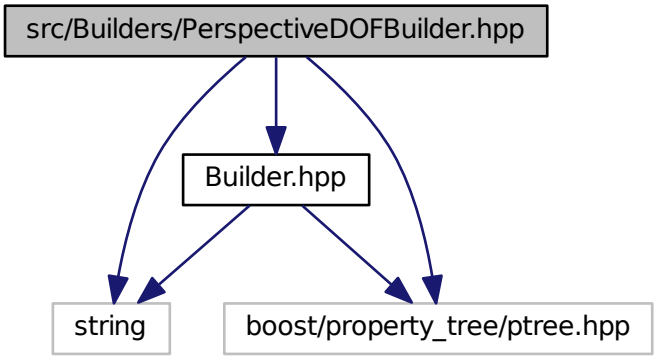
`#include "PerspectiveDOFBuilder.hpp" #include <Vector.hpp> #include <PerspectiveDOF.hpp> #include <Material.hpp>` Graphe des dépendances par inclusion de PerspectiveDOFBuilder.cc :



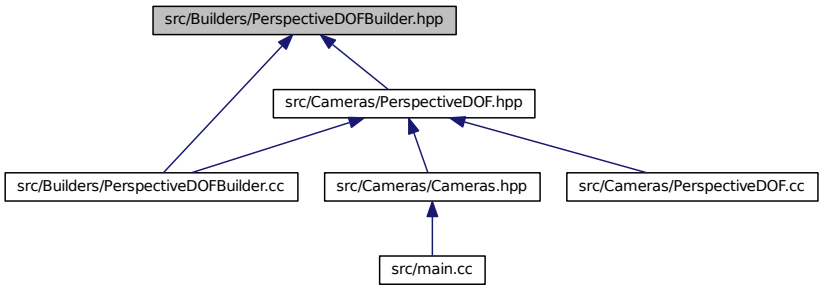
5.14 Référence du fichier src/Builders/PerspectiveDOFBuilder.hpp

`#include <Builder.hpp> #include <string> #include <boost/property_tree/ptree.hpp>` Graphe des dépendances par inclusion de PerspectiveDOF-

Builder.hpp :



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



Classes

– class PerspectiveDOFBuilder

5.14.1 Description détaillée

Auteur

Maxime Gaudin

Date

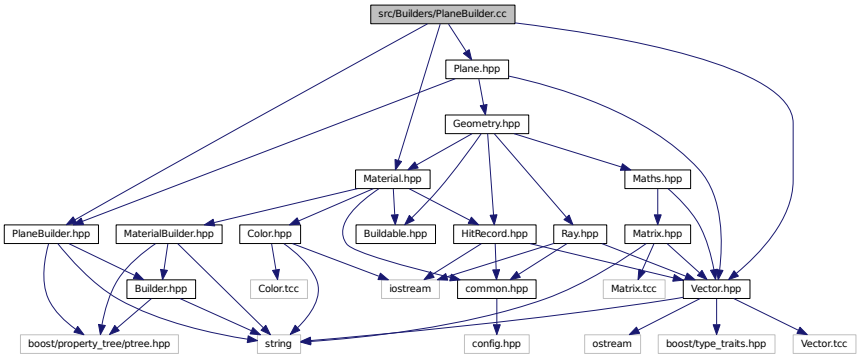
2011

Builder associé à l'objet PerspectiveDOF.

Définition dans le fichier PerspectiveDOFBuilder.hpp.

5.15 Référence du fichier src/Builders/PlaneBuilder.cc

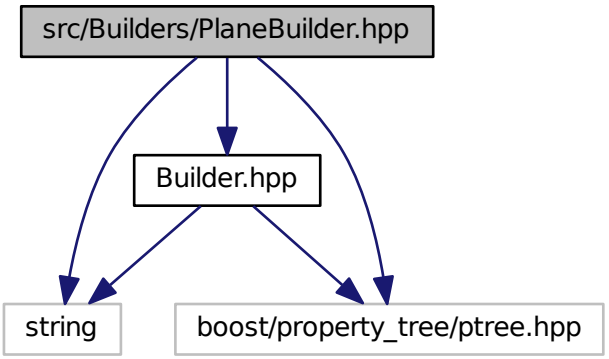
`#include "PlaneBuilder.hpp"` `#include <Vector.hpp>` `#include <Plane.hpp>` `#include <Material.hpp>` Graphe des dépendances par inclusion de PlaneBuilder.cc :



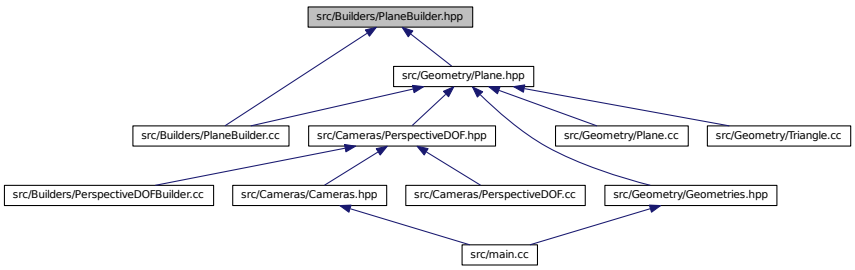
5.16 Référence du fichier src/Builders/PlaneBuilder.hpp

`#include <Builder.hpp>` `#include <string>` `#include <boost/property-`

_tree/ptree.hpp> Graphe des dépendances par inclusion de PlaneBuilder.hpp :



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



Classes

– class PlaneBuilder

5.16.1 Description détaillée

Auteur

Maxime Gaudin

Date

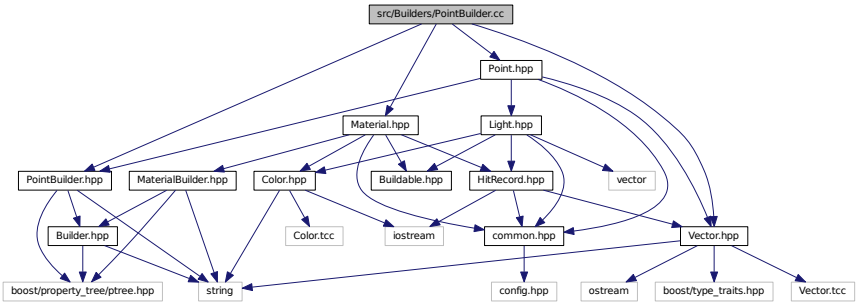
2011

Builder associé à l'objet Plane.

Définition dans le fichier PlaneBuilder.hpp.

5.17 Référence du fichier src/Builders/PointBuilder.cc

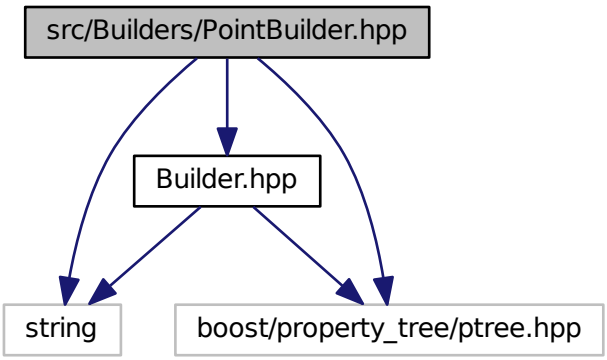
`#include "PointBuilder.hpp"` `#include <Vector.hpp>` `#include <Point.hpp>` `#include <Material.hpp>` Graphe des dépendances par inclusion de PointBuilder.cc :



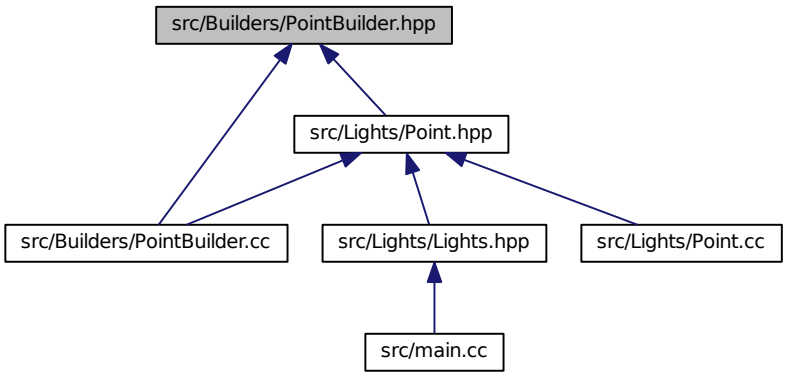
5.18 Référence du fichier src/Builders/PointBuilder.hpp

`#include <Builder.hpp>` `#include <string>` `#include <boost/property-`

_tree/ptree.hpp> Graphe des dépendances par inclusion de PointBuilder.hpp :



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



Classes

– class PointBuilder

5.18.1 Description détaillée

Auteur

Maxime Gaudin

Date

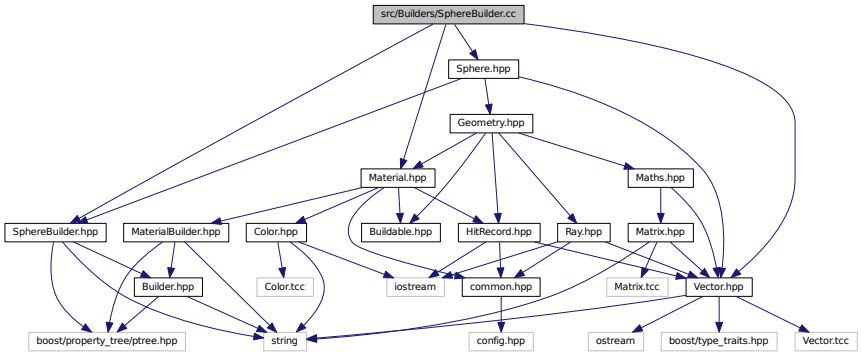
2011

Builder associé à l'objet Point.

Définition dans le fichier PointBuilder.hpp.

5.19 Référence du fichier src/Builders/SphereBuilder.cc

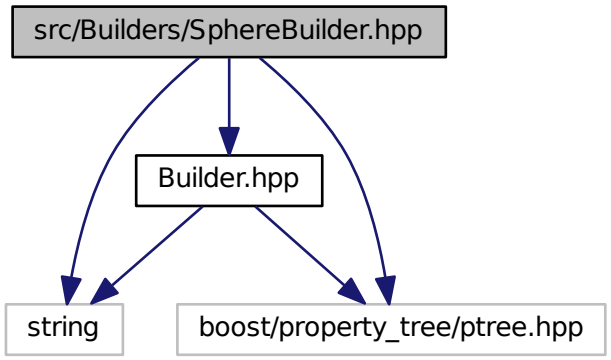
`#include "SphereBuilder.hpp"` `#include <Vector.hpp>` `#include <Sphere.hpp>` `#include <Material.hpp>` Graphe des dépendances par inclusion de SphereBuilder.cc :



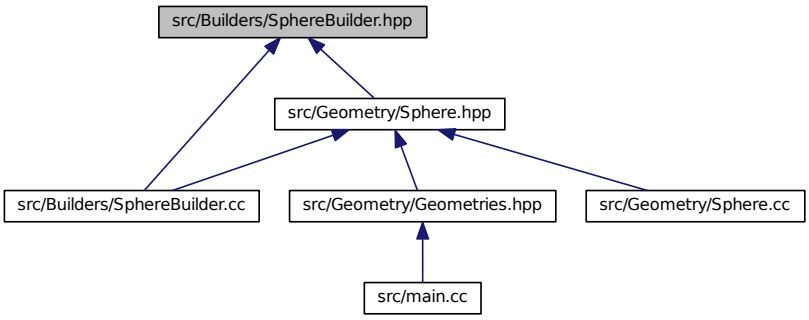
5.20 Référence du fichier src/Builders/SphereBuilder.hpp

`#include <Builder.hpp>` `#include <string>` `#include <boost/property-_tree/ptree.hpp>` Graphe des dépendances par inclusion de SphereBuilder.hpp-

:



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



Classes

– class SphereBuilder

5.20.1 Description détaillée

Auteur

Maxime Gaudin

Date

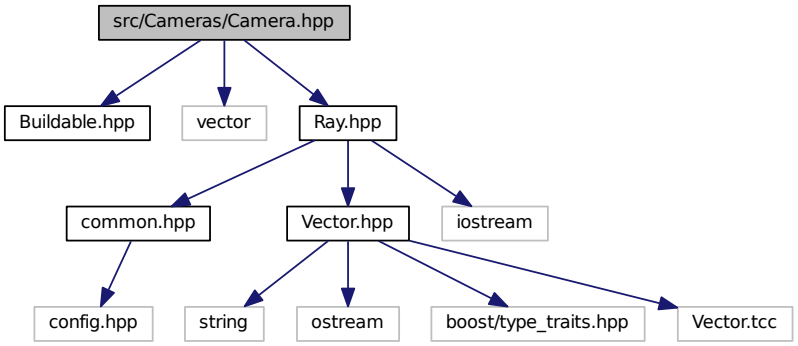
2011

Builder associé à l'objet Sphere.

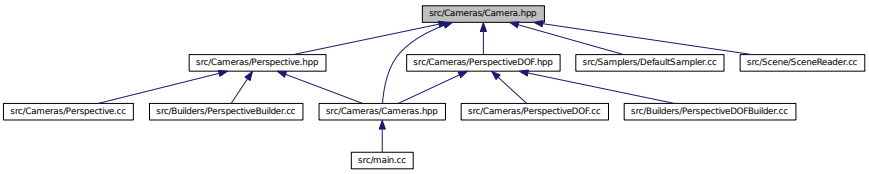
Définition dans le fichier SphereBuilder.hpp.

5.21 Référence du fichier src/Cameras/Camera.hpp

`#include <Buildable.hpp> #include <vector> #include <Ray.hpp>` Graphe des dépendances par inclusion de Camera.hpp :



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



Classes

– class Camera

5.21.1 Description détaillée

Auteur

Maxime Gaudin

Date

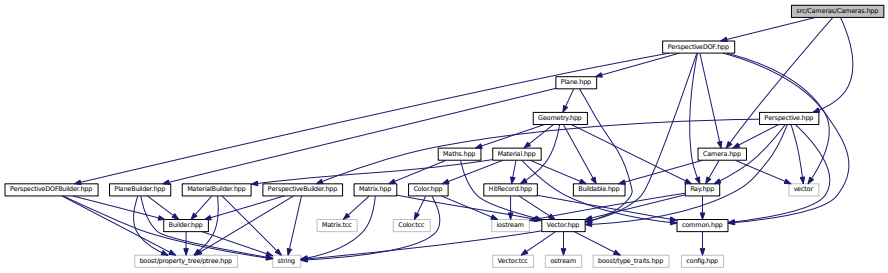
2011

Cette classe est l'interface de toutes les caméras. Les coordonnées utilisées dans cette classe (et donc toutes les classes qui en héritent) doivent utiliser le système de coordonnées UV qui est définit comme suit : (0,1)----------(1,1) | | | | | | | | | | (0,0)----------(1,0)

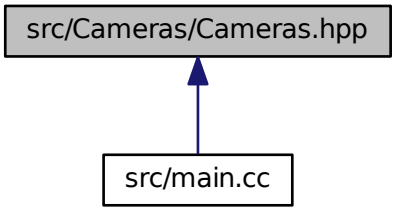
Définition dans le fichier Camera.hpp.

5.22 Référence du fichier src/Cameras/Cameras.hpp

`#include <Camera.hpp>` `#include <Perspective.hpp>` `#include <PerspectiveDOF.hpp>` Graphe des dépendances par inclusion de - Cameras.hpp :

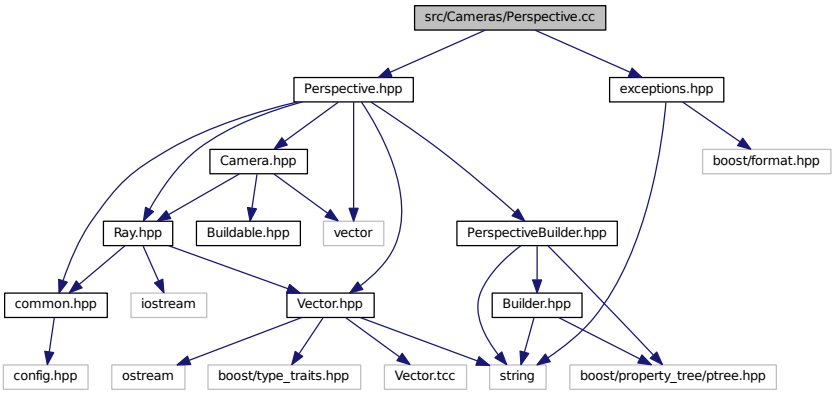


Ce graphe montre quels fichiers incluent directement ou indirectement ce fichier :



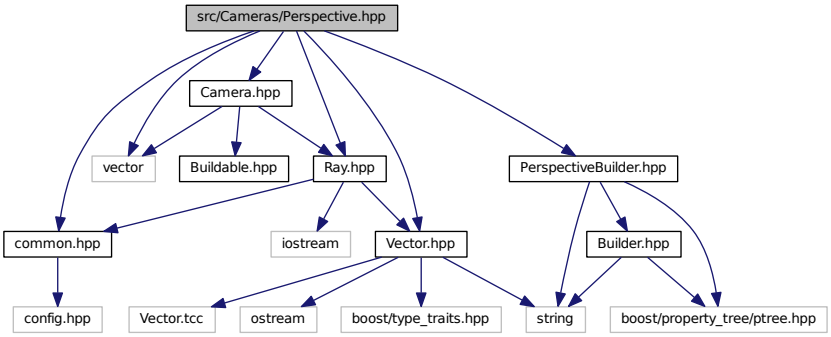
5.23 Référence du fichier src/Cameras/Perspective.cc

`#include "Perspective.hpp"` `#include <exceptions.hpp>` Graphe des dépendances par inclusion de Perspective.cc :

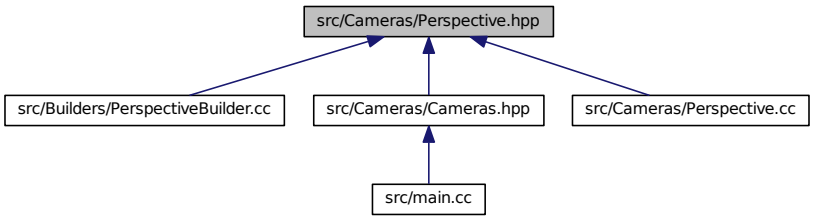


5.24 Référence du fichier src/Cameras/Perspective.hpp

`#include <vector>` `#include <common.hpp>` `#include <Camera.hpp>` `#include <PerspectiveBuilder.hpp>` `#include <Vector.hpp>` `#include <Ray.hpp>` Graphe des dépendances par inclusion de Perspective.hpp :



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



Classes

– class Perspective

5.24.1 Description détaillée

Auteur

Maxime Gaudin

Date

2011

Déclare la caméra Perspective. C'est une caméra "normale" avec projection perspective.

Auteur

Maxime Gaudin

Date

2011

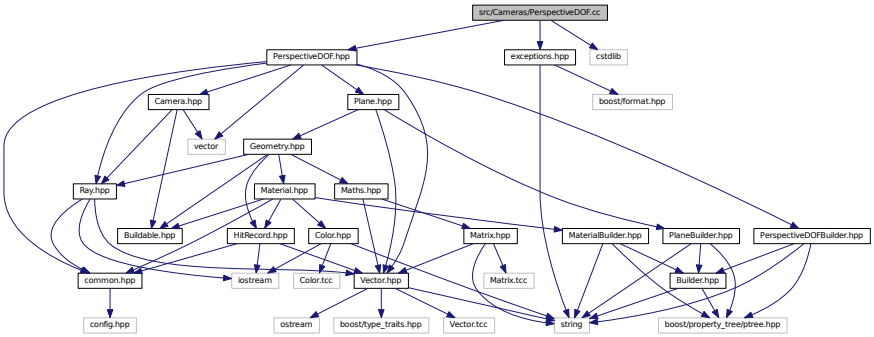
Déclare la caméra Perspective. C'est une caméra "normale" avec projection perspective et effet de profondeur de champ. Le réglage de celui-ci n'est pas trivial et il faut tester plusieurs ouvertures avant d'obtenir un effet réaliste.

Définition dans le fichier Perspective.hpp.

5.25 Référence du fichier src/Cameras/PerspectiveDOF.cc

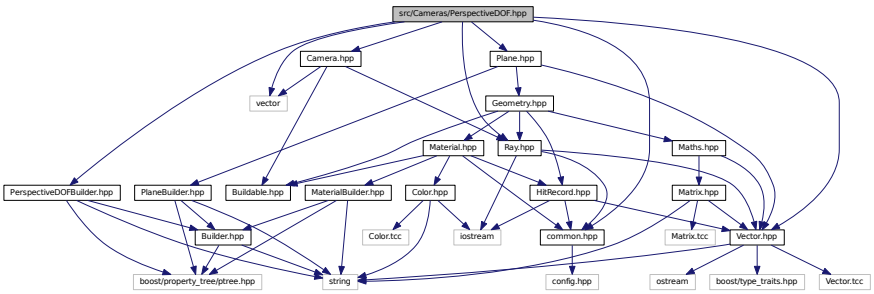
```
#include "PerspectiveDOF.hpp" #include <cstdlib> #include
```

<exceptions.hpp> Graphe des dépendances par inclusion de PerspectiveDOF.cc :

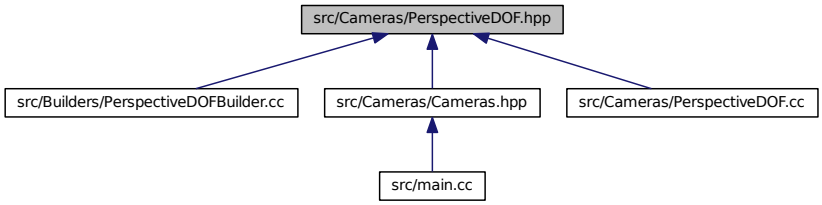


5.26 Référence du fichier src/Cameras/PerspectiveDOF.hpp

#include <vector> #include <PerspectiveDOFBuilder.hpp> × #include <Plane.hpp> #include <common.hpp> #include <- Camera.hpp> #include <Vector.hpp> #include <Ray.hpp> Graphe des dépendances par inclusion de PerspectiveDOF.hpp :



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

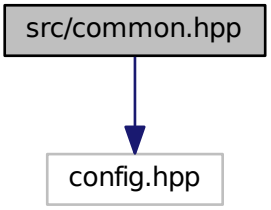


Classes

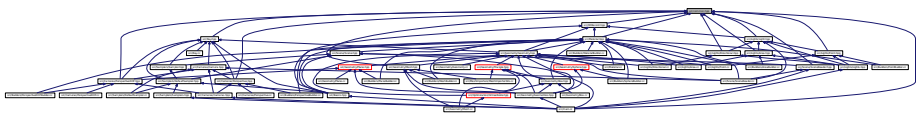
– class PerspectiveDOF

5.27 Référence du fichier src/common.hpp

`#include <config.hpp>` Graphe des dépendances par inclusion de common.hpp :



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



Macros

– #define feach BOOST_FOREACH

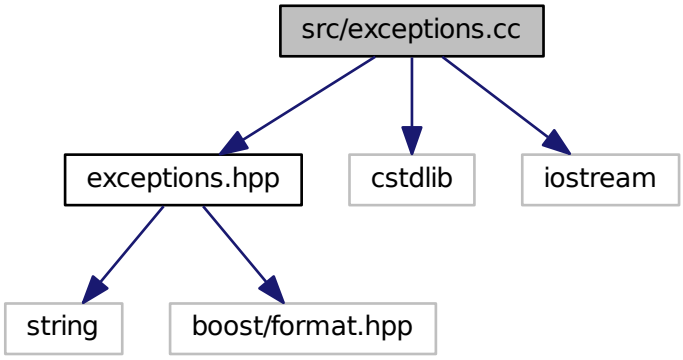
5.27.1 Documentation des macros

5.27.1.1 #define feach BOOST_FOREACH

Définition à la ligne 6 du fichier common.hpp.

5.28 Référence du fichier src/exceptions.cc

`#include "exceptions.hpp"` `#include <cstdlib>` `#include <iostream>` Graphe des dépendances par inclusion de exceptions.cc :



Fonctions

– void logException (std : :string const &message)
– void logException (std : :string const &launcher, std : :string const &message)
– void logInformation (std : :string const &launcher, std : :string const &message)

5.28.1 Documentation des fonctions

5.28.1.1 void logException (std : :string const & *message*)

Définition à la ligne 5 du fichier exceptions.cc.

5.28.1.2 void logException (std : :string const & *launcher,* std : :string const & *message*)

Définition à la ligne 10 du fichier exceptions.cc.

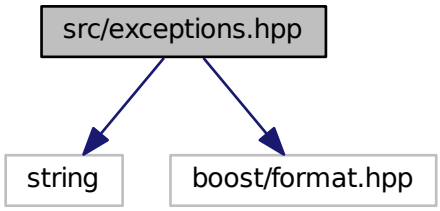
Voici le graphe d'appel pour cette fonction :

5.28.1.3 void logInformation (std : :string const & *launcher,* std : :string const & *message*)

Définition à la ligne 16 du fichier exceptions.cc.

5.29 Référence du fichier src/exceptions.hpp

`#include <string>` `#include <boost/format.hpp>` Graphe des dépendances par inclusion de exceptions.hpp :



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

Fonctions

– void logException (std : :string const &message)
– void logException (std : :string const &launcher, std : :string const &message)
– void logInformation (std : :string const &launcher, std : :string const &message)

5.29.1 Description détaillée

Auteur

Maxime Gaudin

Date

2011

Ce fichier déclare des fonctions d'aide permettant de normaliser l'accès aux mécanisme de journalisation.

Définition dans le fichier exceptions.hpp.

5.29.2 Documentation des fonctions

5.29.2.1 void logException (std : :string const & *message*)

Définition à la ligne 5 du fichier exceptions.cc.

5.29.2.2 void logException (std : :string const & *launcher,* std : :string const & *message*)

Définition à la ligne 10 du fichier exceptions.cc.

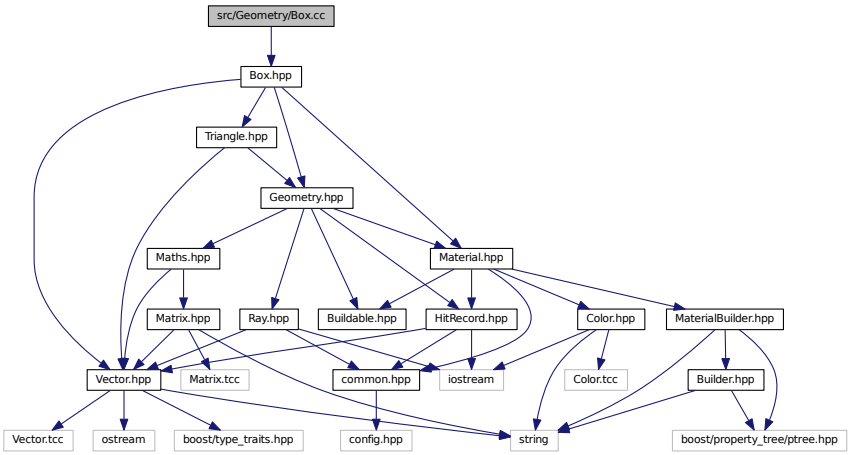
Voici le graphe d'appel pour cette fonction :

5.29.2.3 void logInformation (std : :string const & *launcher,* std : :string const & *message*)

Définition à la ligne 16 du fichier exceptions.cc.

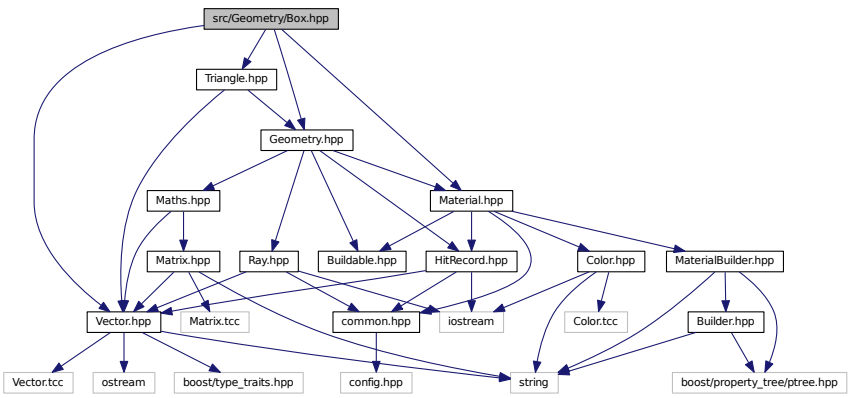
5.30 Référence du fichier src/Geometry/Box.cc

`#include "Box.hpp"` Graphe des dépendances par inclusion de Box.cc :

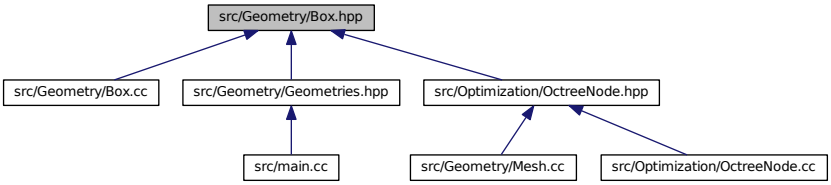


5.31 Référence du fichier src/Geometry/Box.hpp

`#include <Geometry.hpp> #include <Material.hpp> #include <Vector.hpp> #include <Triangle.hpp>` Graphe des dépendances par inclusion de Box.hpp :



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

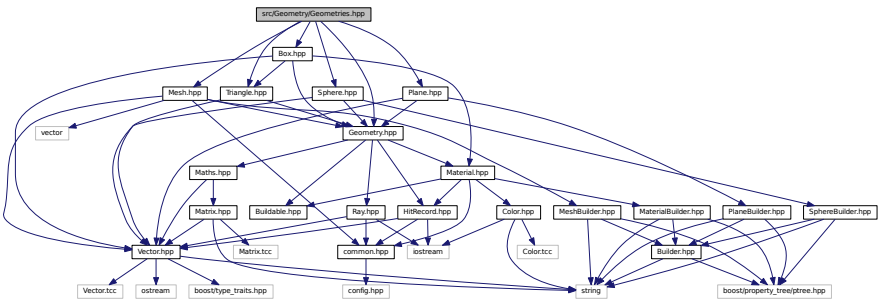


Classes

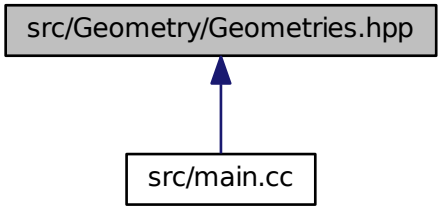
– class Box

5.32 Référence du fichier src/Geometry/Geometries.hpp

`#include <Geometry.hpp> #include <Sphere.hpp> #include <Triangle.hpp> #include <Mesh.hpp> #include <Plane.hpp> #include <Box.hpp>` Graphe des dépendances par inclusion de Geometries.hpp :

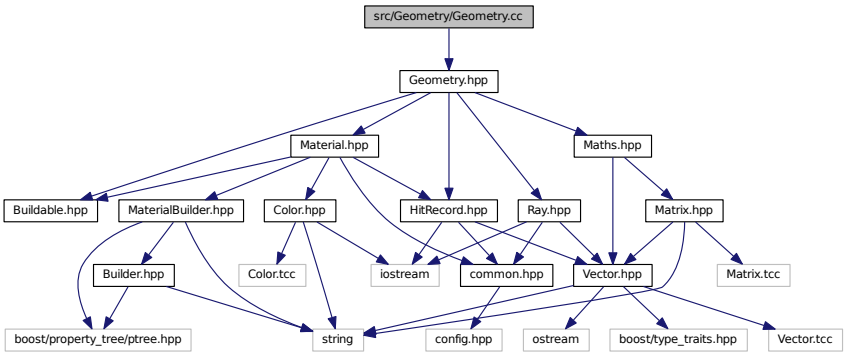


Ce graphe montre quels fichiers incluent directement ou indirectement ce fichier :



5.33 Référence du fichier src/Geometry/Geometry.cc

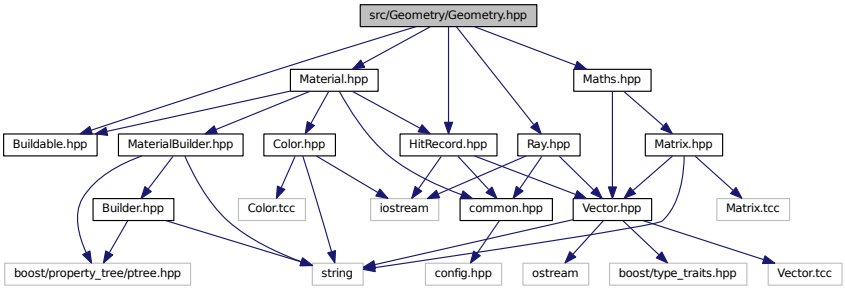
`#include "Geometry.hpp"` Graphe des dépendances par inclusion de - Geometry.cc :



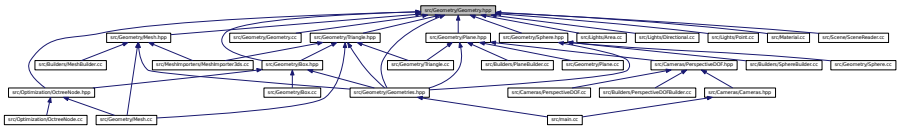
5.34 Référence du fichier src/Geometry/Geometry.hpp

`#include <Buildable.hpp>` `#include <HitRecord.hpp>` `#include <Ray.hpp>` `#include <Material.hpp>` `#include <Maths.hpp>` ×

Graphe des dépendances par inclusion de Geometry.hpp :



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



Classes

– class Geometry

5.34.1 Description détaillée

Auteur

Maxime Gaudin.

Date

2011

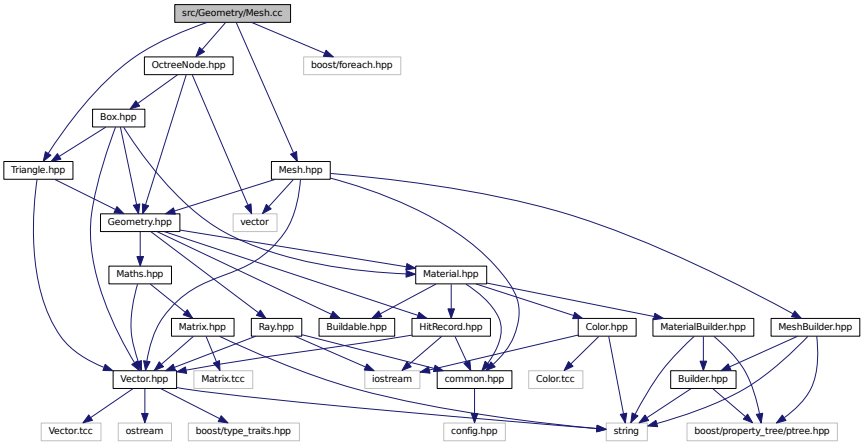
Cette classe est l'interface de tous les objets dont on peut calculer l'intersection avec un ray

Définition dans le fichier Geometry.hpp.

5.35 Référence du fichier src/Geometry/Mesh.cc

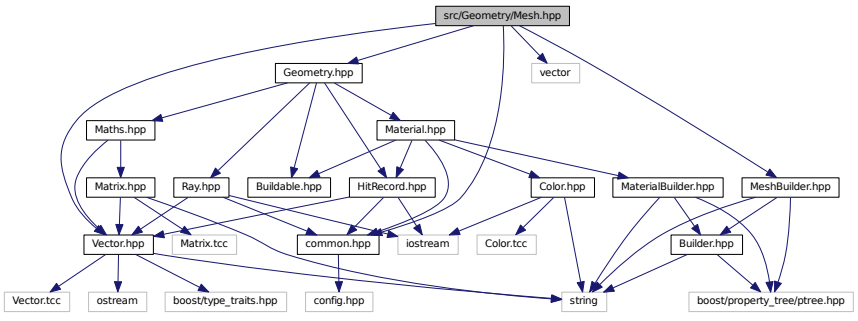
`#include "Mesh.hpp" #include <Triangle.hpp> #include <-OctreeNode.hpp> #include <boost/foreach.hpp>` Graphe des dépen-

dances par inclusion de Mesh.cc :

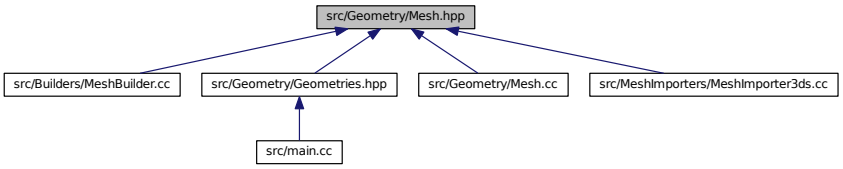


5.36 Référence du fichier src/Geometry/Mesh.hpp

`#include <Geometry.hpp> #include <common.hpp> #include <vector> #include <MeshBuilder.hpp> #include <Vector.hpp>` Graphe des dépendances par inclusion de Mesh.hpp :



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

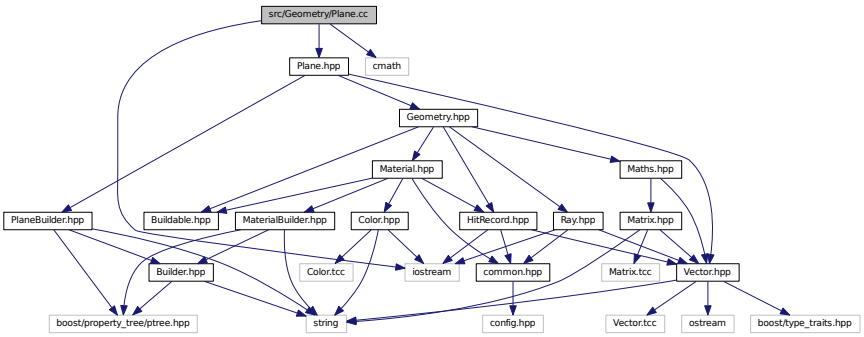


Classes

– class Mesh

5.37 Référence du fichier src/Geometry/Plane.cc

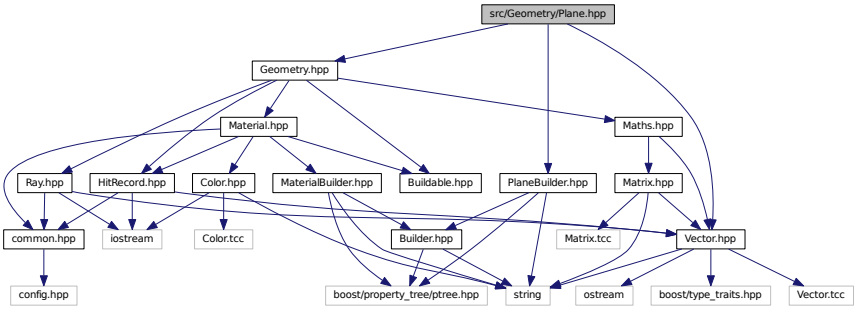
`#include <Plane.hpp> #include <cmath> #include <iostream>` ×
Graphe des dépendances par inclusion de Plane.cc :



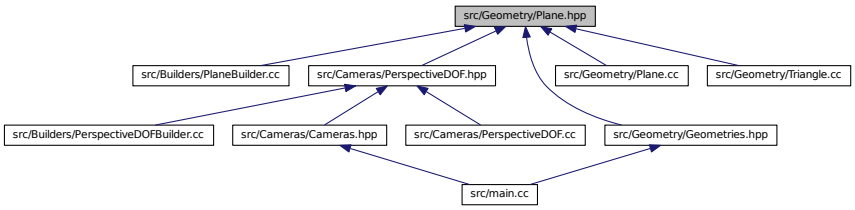
5.38 Référence du fichier src/Geometry/Plane.hpp

`#include <Geometry.hpp> #include <Vector.hpp> #include <-`

PlaneBuilder.hpp> Graphe des dépendances par inclusion de Plane.hpp :



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

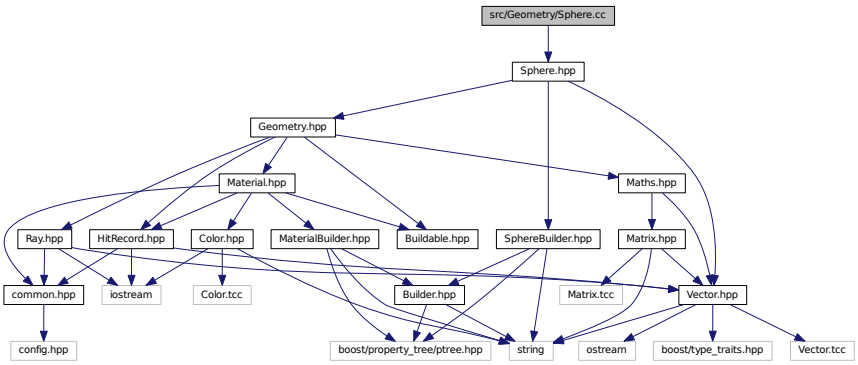


Classes

– class Plane

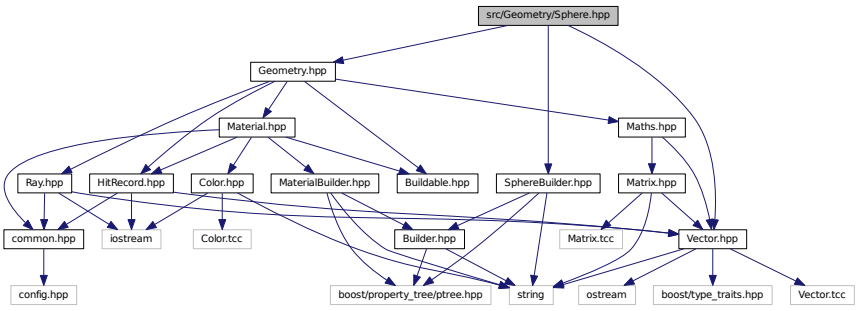
5.39 Référence du fichier src/Geometry/Sphere.cc

`#include "Sphere.hpp"` Graphe des dépendances par inclusion de Sphere.cc :

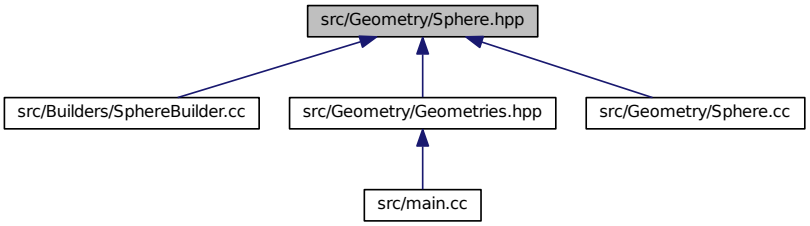


5.40 Référence du fichier src/Geometry/Sphere.hpp

`#include <Geometry.hpp>` `#include <Vector.hpp>` `#include <SphereBuilder.hpp>` Graphe des dépendances par inclusion de Sphere.hpp :



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

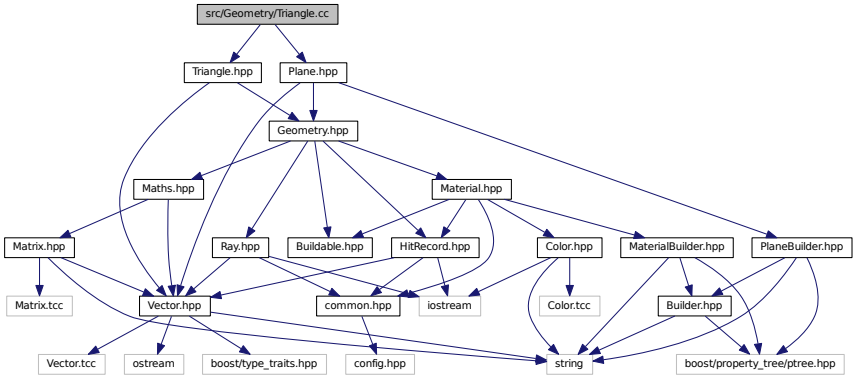


Classes

– class Sphere

5.41 Référence du fichier src/Geometry/Triangle.cc

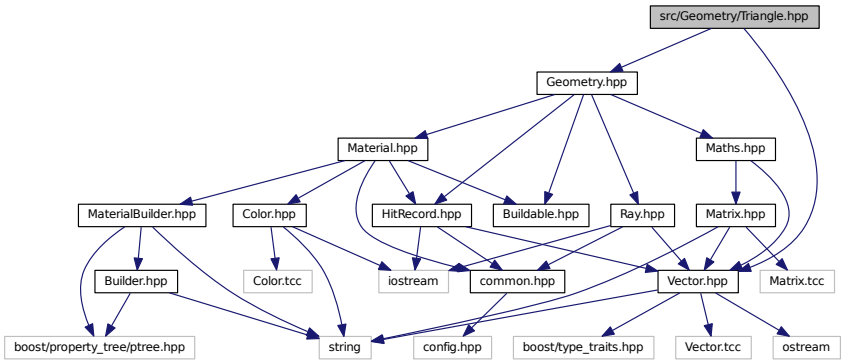
`#include "Triangle.hpp"` `#include <Plane.hpp>` Graphe des dépendances par inclusion de Triangle.cc :



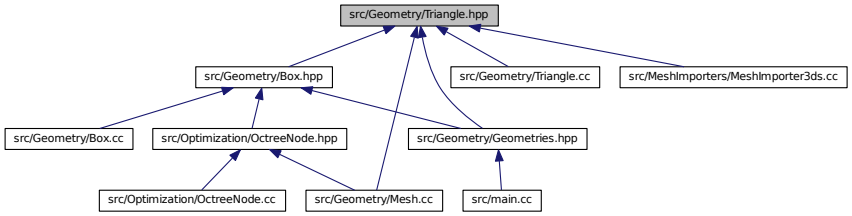
5.42 Référence du fichier src/Geometry/Triangle.hpp

`#include <Geometry.hpp>` `#include <Vector.hpp>` Graphe des dé-

pendances par inclusion de Triangle.hpp :



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



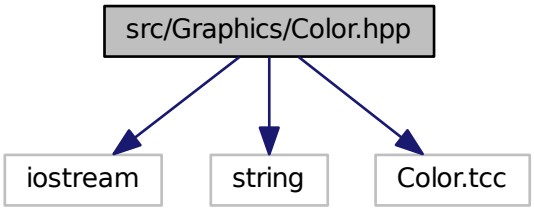
Classes

– class Triangle

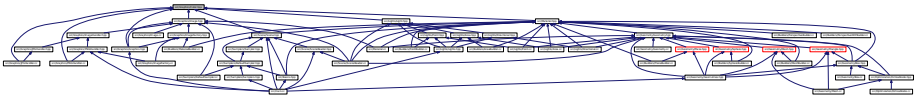
5.43 Référence du fichier src/Graphics/Color.hpp

Définition de la classe Color.

`#include <iostream> #include <string> #include "Color.-tcc"` Graphe des dépendances par inclusion de Color.hpp :



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



Classes

– class Color< P >

5.43.1 Description détaillée

Définition de la classe Color.

Auteur

Maxime Gaudin

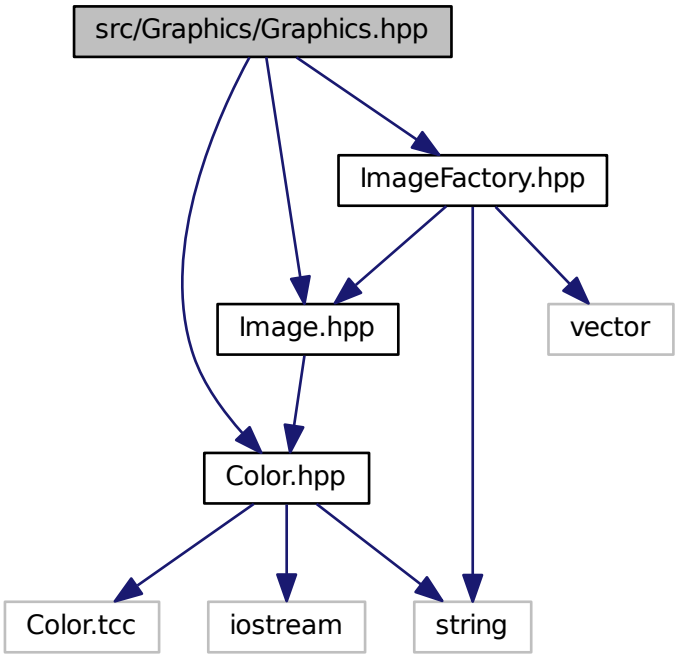
Date

2011

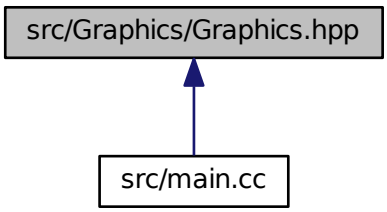
La classe couleur est paramétrée par le type numérique stockant chaque composante.
Les types autorisés sont :
– unsigned char/ char
– unsigned short/ short
– unsigned int/ int
– double
– float
Définition dans le fichier Color.hpp.

5.44 Référence du fichier src/Graphics/Graphics.hpp

`#include <Image.hpp>` `#include <Color.hpp>` `#include <ImageFactory.hpp>` Graphe des dépendances par inclusion de Graphics.hpp :

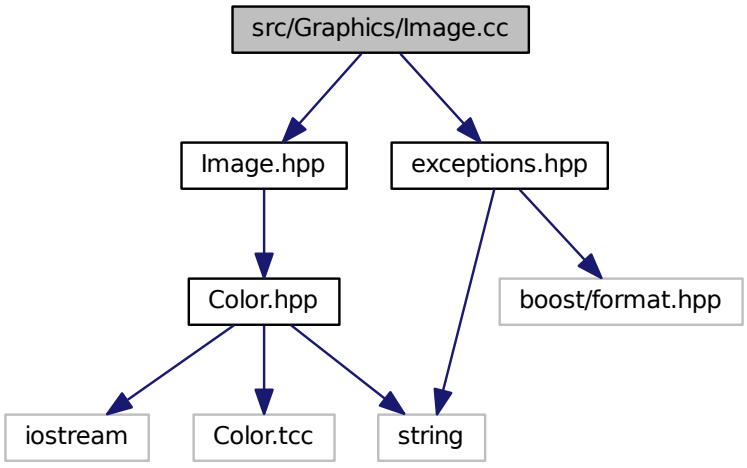


Ce graphe montre quels fichiers incluent directement ou indirectement ce fichier :



5.45 Référence du fichier src/Graphics/Image.cc

`#include "Image.hpp"` `#include <exceptions.hpp>` Graphe des dépendances par inclusion de Image.cc :



Macros

– #define IMAGE_TI_H_

5.45.1 Documentation des macros

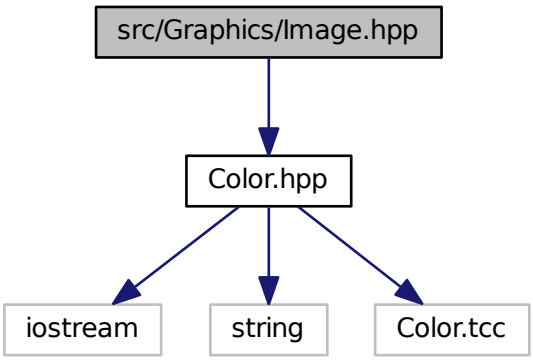
5.45.1.1 #define IMAGE_TI_H_

Définition à la ligne 2 du fichier Image.cc.

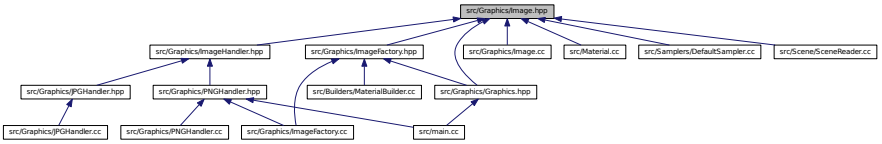
5.46 Référence du fichier src/Graphics/Image.hpp

Ce header contient la déclaration de la classe Image.

`#include <Color.hpp>` Graphe des dépendances par inclusion de Image.hpp :



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



Classes

– class Image

5.46.1 Description détaillée

Ce header contient la déclaration de la classe Image.

Auteur

Maxime Gaudin

Date

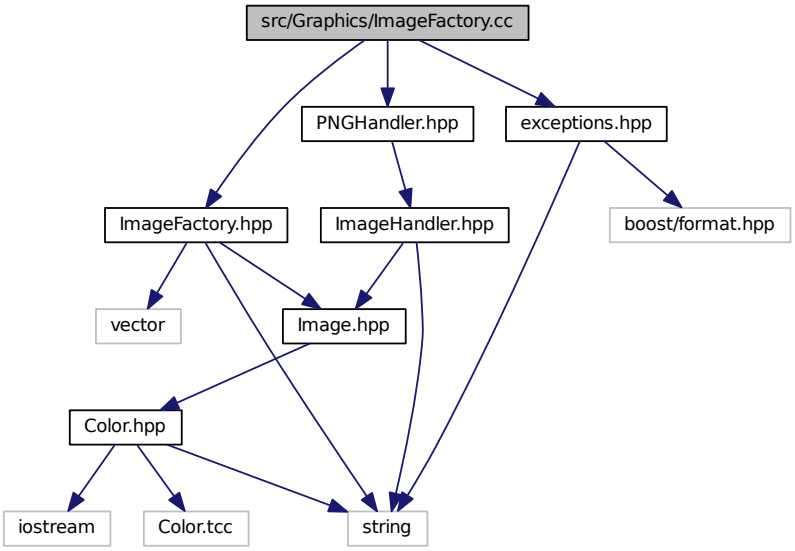
2011

Cette classe permet d'abstraire la notion d'image et de créer un format générique utilisable par tous et notamment par les handlers. La classe Image est paramétré par le type de couleur qu'elle devra contenir et les pixels sont stockés sur des axes partant du coin supérieur gauche.

Définition dans le fichier Image.hpp.

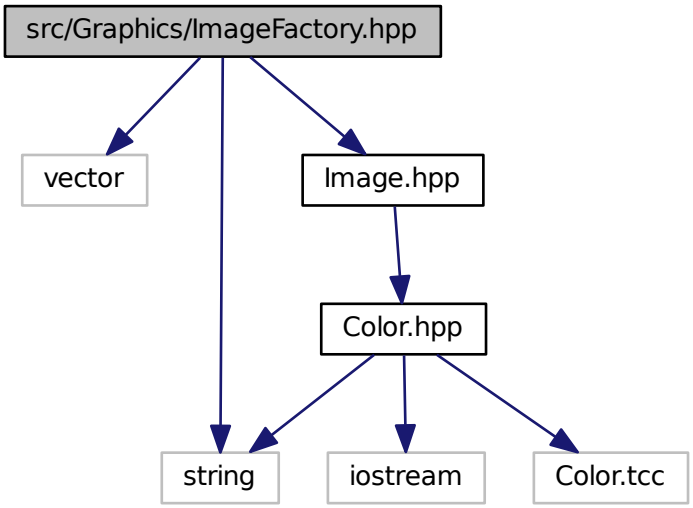
5.47 Référence du fichier src/Graphics/ImageFactory.cc

`#include "ImageFactory.hpp"` `#include <exceptions.hpp>` × `#include <PNGHandler.hpp>` Graphe des dépendances par inclusion de ImageFactory.cc :

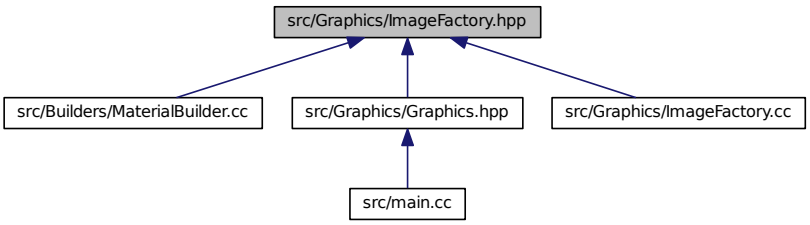


5.48 Référence du fichier src/Graphics/ImageFactory.hpp

`#include <vector>` `#include <string>` `#include <Image.-hpp>` Graphe des dépendances par inclusion de ImageFactory.hpp :



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



Classes

– class ImageFactory

5.48.1 Description détaillée

Auteur

Maxime Gaudin

Date

2011

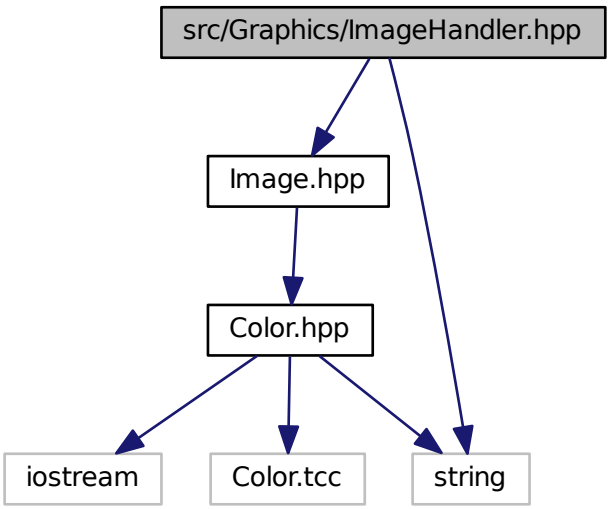
Cette classe est une usine à image. Elle permet, sans avoir à se préocupper du handler, de charger ou sauvergarder une image sous un format supporté par l'application.

Définition dans le fichier ImageFactory.hpp.

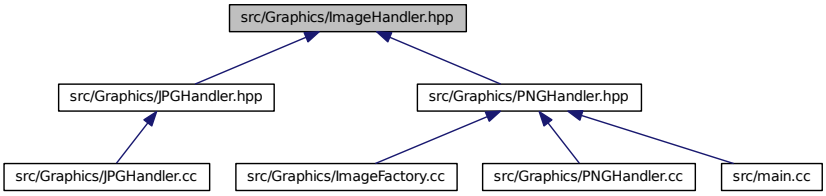
5.49 Référence du fichier src/Graphics/ImageHandler.hpp

Interface d'un exporteur d'image.

`#include <Image.hpp> #include <string>` Graphe des dépendances par inclusion de ImageHandler.hpp :



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



Classes

– class ImageHandler

5.49.1 Description détaillée

Interface d'un exporteur d'image.

Auteur

Maxime Gaudin

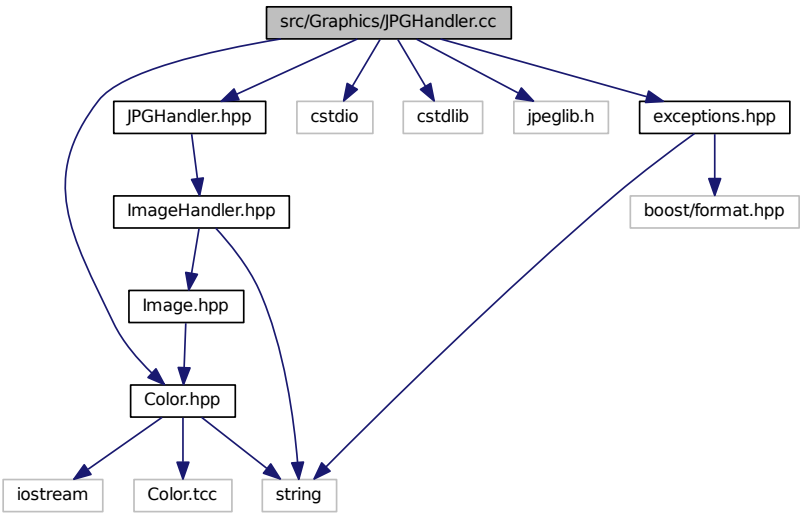
Date

2011

Définition dans le fichier ImageHandler.hpp.

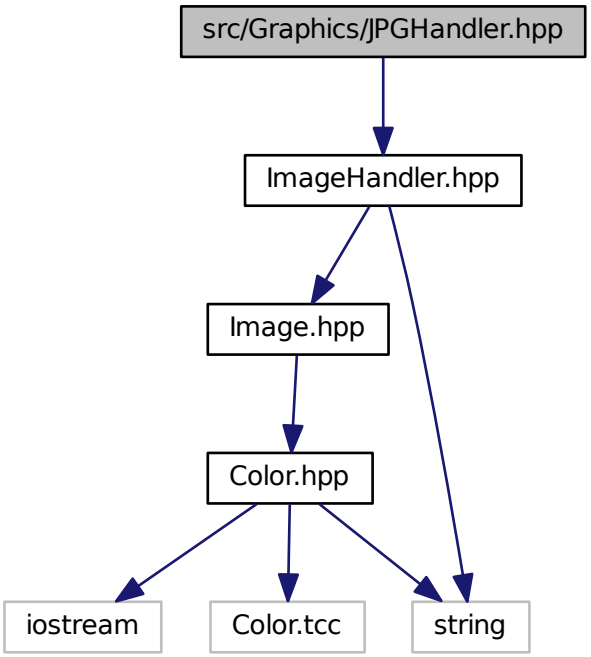
5.50 Référence du fichier src/Graphics/JPGHandler.cc

`#include "JPGHandler.hpp" #include <Color.hpp> #include <cstdio> #include <cstdlib> #include <jpeglib.h> #include <exceptions.hpp>` Graphe des dépendances par inclusion de JPGHandler.cc :

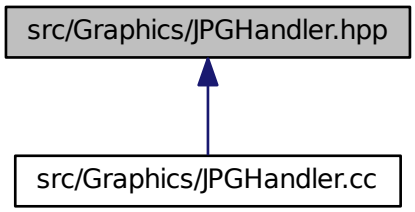


5.51 Référence du fichier src/Graphics/JPGHandler.hpp

`#include "ImageHandler.hpp"` Graphe des dépendances par inclusion de JPGHandler.hpp :



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



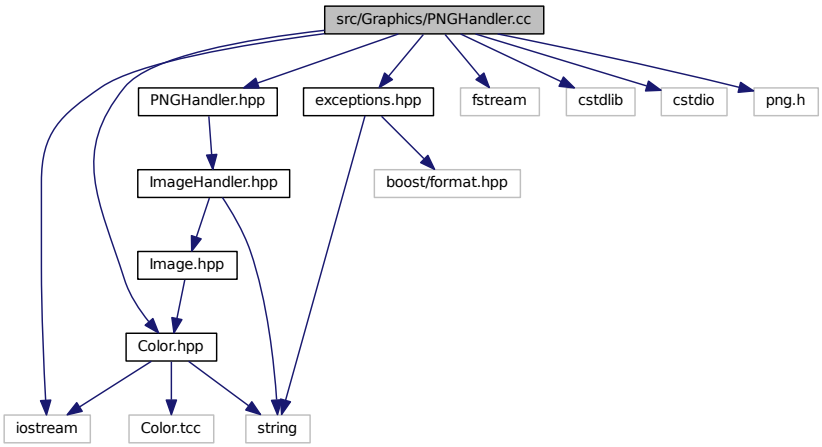
Classes

– class JPGHandler

5.52 Référence du fichier src/Graphics/PNGHandler.cc

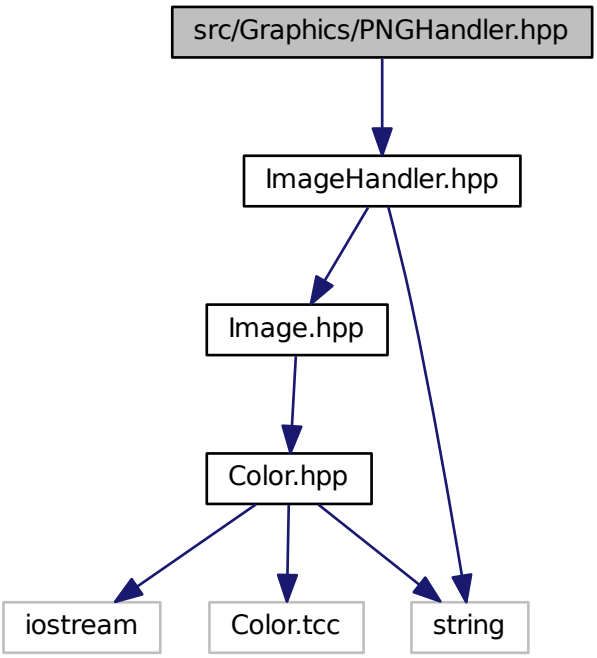
```
#include "PNGHandler.hpp"  #include <iostream>  #include
<fstream> #include <cstdlib> #include <cstdio> #include
<Color.hpp> #include <png.h> #include <exceptions.hpp> ×
```

Graphe des dépendances par inclusion de PNGHandler.cc :

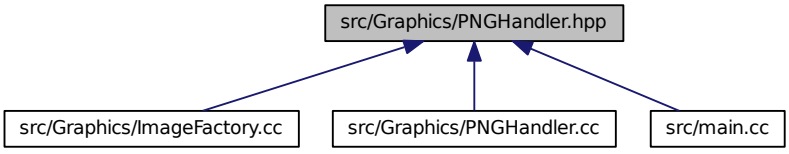


5.53 Référence du fichier src/Graphics/PNGHandler.hpp

`#include "ImageHandler.hpp"` Graphe des dépendances par inclusion de PNGHandler.hpp :



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

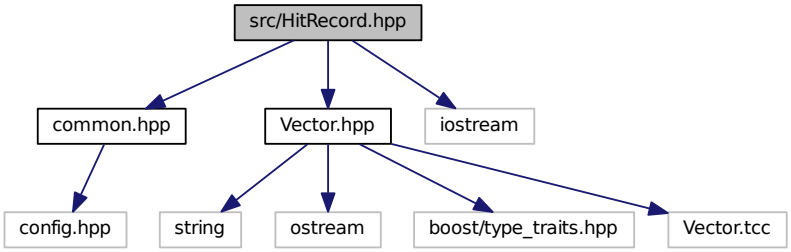


Classes

– class PNGHandler

5.54 Référence du fichier src/HitRecord.hpp

`#include <common.hpp>` `#include <Vector.hpp>` `#include <iostream>` Graphe des dépendances par inclusion de HitRecord.hpp :



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

Classes

– struct HitRecord

5.54.1 Description détaillée

Auteur

Maxime Gaudin

Date

2011

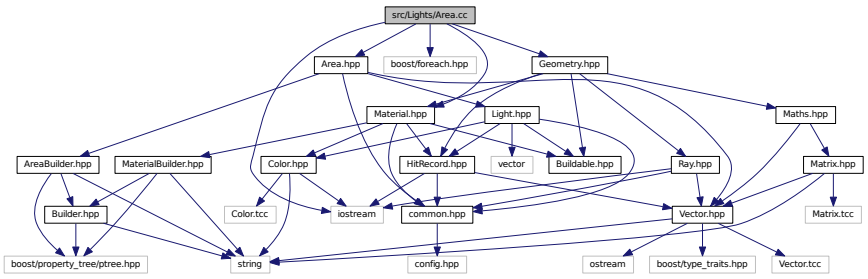
Ce fichier définit la structure permettant de stocker le résultat d'un lancer de rayon. Il permet de mémoriser toutes les informations nécessaires aux différents calcul dans le cas où il y a eu intersection.

Définition dans le fichier HitRecord.hpp.

5.55 Référence du fichier src/Lights/Area.cc

`#include "Area.hpp" #include <boost/foreach.hpp> #include <Material.hpp> #include <Geometry.hpp> #include <iostream>` ×

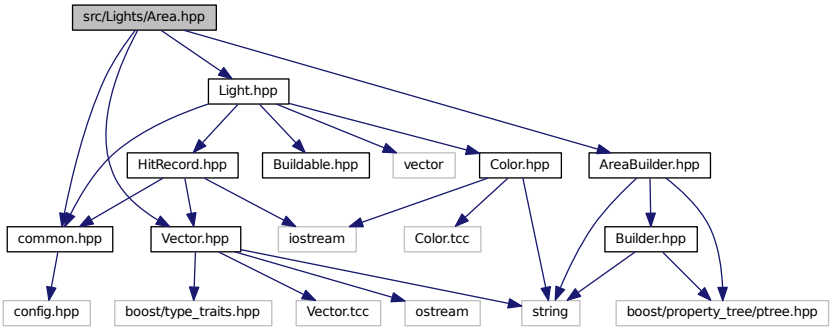
Graphe des dépendances par inclusion de Area.cc :



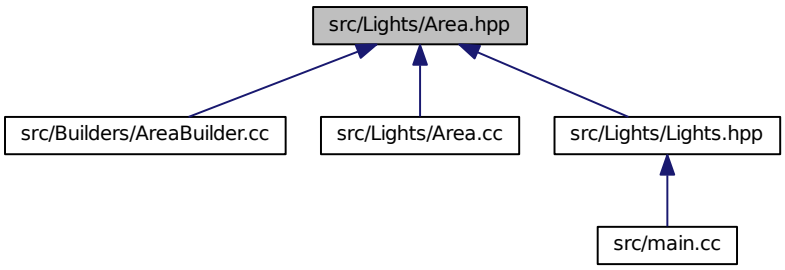
5.56 Référence du fichier src/Lights/Area.hpp

`#include <Light.hpp> #include <common.hpp> #include <AreaBuilder.hpp> #include <Vector.hpp>` Graphe des dépendances

par inclusion de Area.hpp :



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



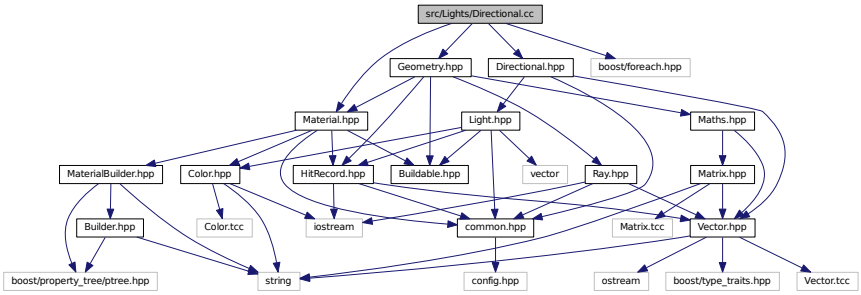
Classes

– class Area

5.57 Référence du fichier src/Lights/Directional.cc

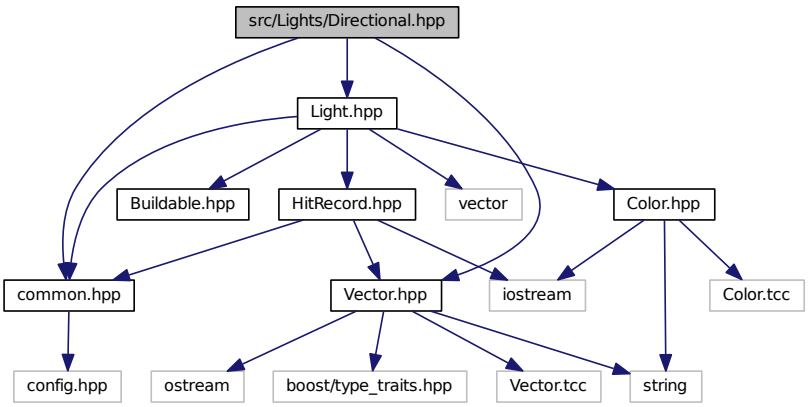
`#include "Directional.hpp" #include <boost/foreach.hpp>` × `#include <Material.hpp> #include <Geometry.hpp>` Graphe des

dépendances par inclusion de Directional.cc :

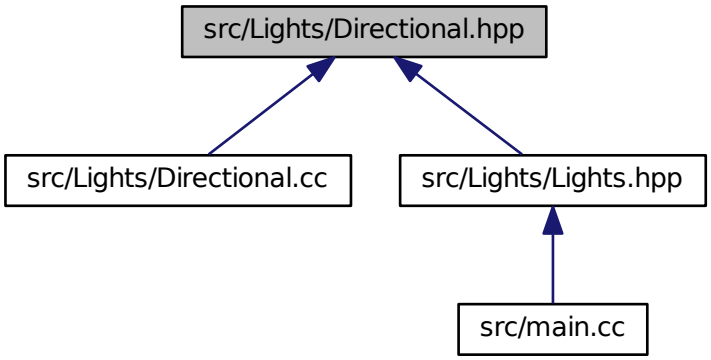


5.58 Référence du fichier src/Lights/Directional.hpp

`#include <Light.hpp>  #include <common.hpp>  #include <-Vector.hpp>` Graphe des dépendances par inclusion de Directional.hpp :



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



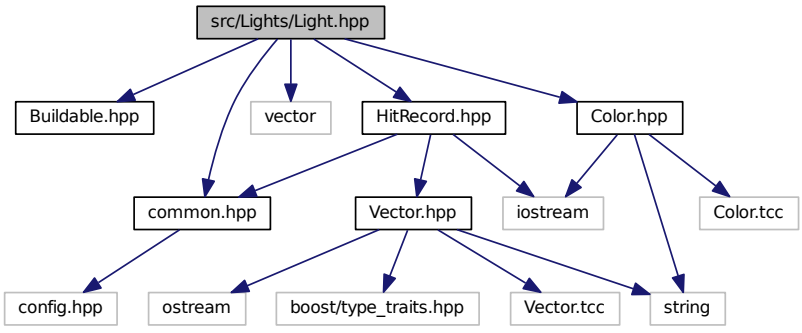
Classes

– class Directional

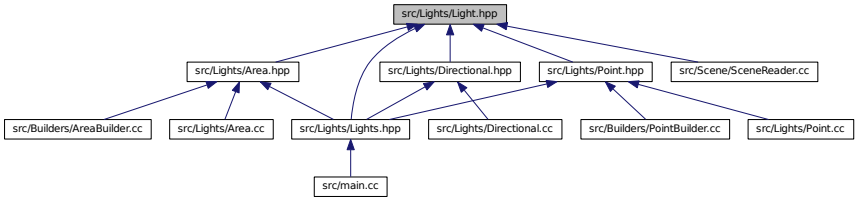
5.59 Référence du fichier src/Lights/Light.hpp

`#include <Buildable.hpp> #include <common.hpp> #include <vector> #include <HitRecord.hpp> #include <Color.hpp>` ×

Graphe des dépendances par inclusion de Light.hpp :



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



Classes

– class Light

5.59.1 Description détaillée

Auteur

Maxime Gaudin

Date

2011

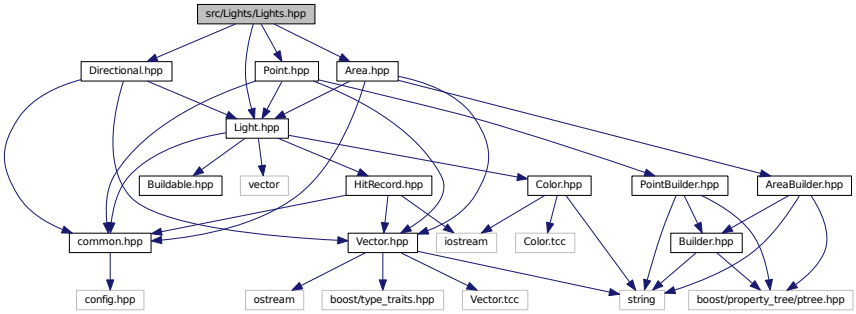
Interface de toutes les lumières.

Définition dans le fichier Light.hpp.

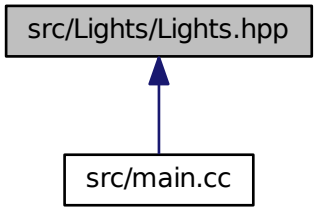
5.60 Référence du fichier src/Lights/Lights.hpp

`#include <Light.hpp> #include <Directional.hpp> #include <Point.hpp> #include <Area.hpp>` Graphe des dépendances par inclu-

sion de Lights.hpp :



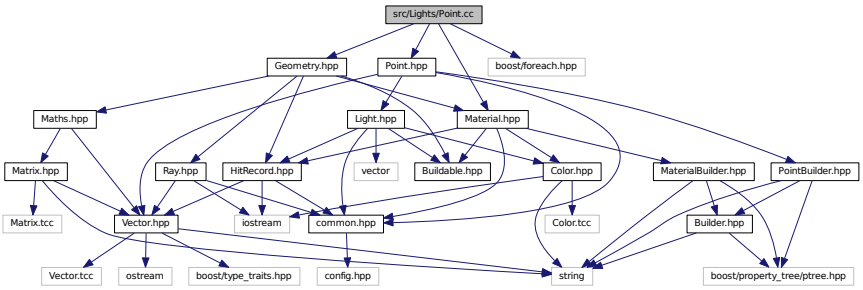
Ce graphe montre quels fichiers incluent directement ou indirectement ce fichier :



5.61 Référence du fichier src/Lights/Point.cc

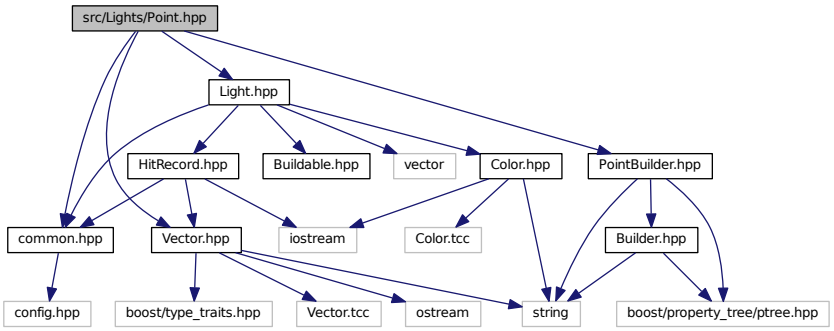
`#include "Point.hpp"` `#include <boost/foreach.hpp>` `#include <Material.hpp>` `#include <Geometry.hpp>` Graphe des dépendances

par inclusion de Point.cc :

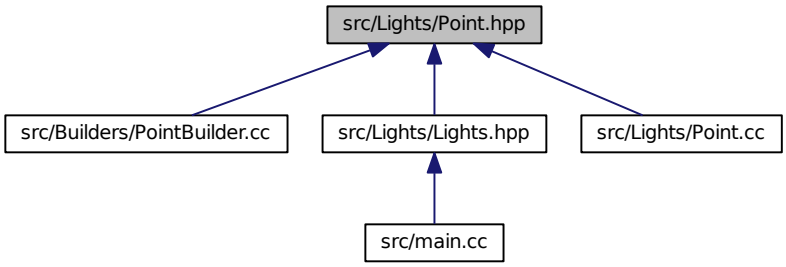


5.62 Référence du fichier src/Lights/Point.hpp

`#include <Light.hpp>  #include <common.hpp>  #include <-PointBuilder.hpp> #include <Vector.hpp>` Graphe des dépendances par inclusion de Point.hpp :



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

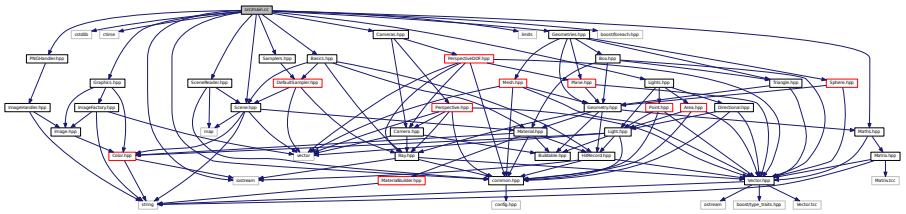


Classes

– class Point

5.63 Référence du fichier src/main.cc

`#include <common.hpp> #include <cstdlib> #include <ctime>` × `#include <Maths.hpp> #include <Basics.hpp> #include <Cameras.hpp> #include <Graphics.hpp> #include <Lights.hpp> #include <Geometries.hpp> #include <Samplers.hpp> #include <iostream> #include <limits> #include <Scene.hpp> #include <SceneReader.hpp> #include <boost/foreach.hpp> #include <PNGHandler.hpp>` Graphe des dépendances par inclusion de main.cc :



Fonctions

– HitRecord getClosestHit (Ray ray, vector< Geometry * > const &geometries)
– Color_d computeDirectLighting (Scene const &scene, HitRecord record)
– void displayProgress (unsigned int Y, unsigned int H)
– Ray getReflectedRay (Ray const &ray, HitRecord const &record)

– Ray getRefractedRay (Ray const &ray, double IOR, HitRecord const &record)
– Color_d getPixel (Scene const &scene, Ray const &ray, double IOR, unsigned char recursionsLevel)
– void Render (Scene &scene, Sampler ∗sampler)
– int main (int argc, char ∗∗argv)

Variables

– const unsigned char MAX_RECURSION = 4

5.63.1 Documentation des fonctions

5.63.1.1 Color_d computeDirectLighting (Scene const & *scene,* HitRecord *record*)

Définition à la ligne 44 du fichier main.cc.

Voici le graphe d'appel pour cette fonction :

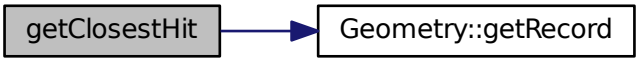
5.63.1.2 void displayProgress (unsigned int *Y,* unsigned int *H*)

Définition à la ligne 53 du fichier main.cc.

5.63.1.3 HitRecord getClosestHit (Ray *ray,* vector< Geometry ∗ > const & *geometries*)

Définition à la ligne 28 du fichier main.cc.

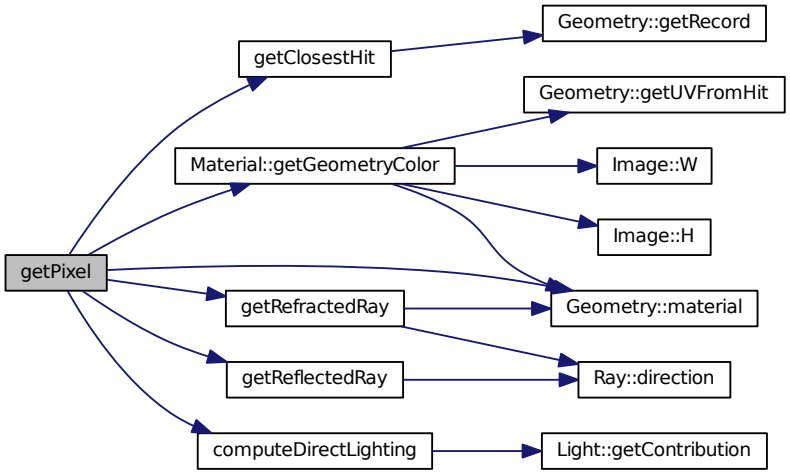
Voici le graphe d'appel pour cette fonction :



5.63.1.4 Color_d getPixel (Scene const & *scene*, Ray const & *ray*, double *IOR*, unsigned char *recursionsLevel*)

Définition à la ligne 83 du fichier main.cc.

Voici le graphe d'appel pour cette fonction :



5.63.1.5 Ray getReflectedRay (Ray const & *ray*, HitRecord const & *record*)

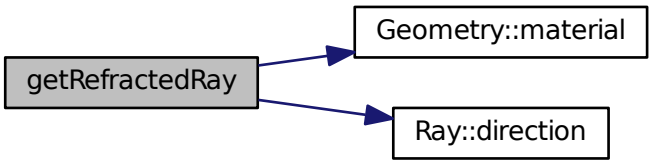
Définition à la ligne 59 du fichier main.cc.

Voici le graphe d'appel pour cette fonction :

5.63.1.6 Ray getRefractedRay (Ray const & *ray*, double *IOR*, HitRecord const & *record*)

Définition à la ligne 64 du fichier main.cc.

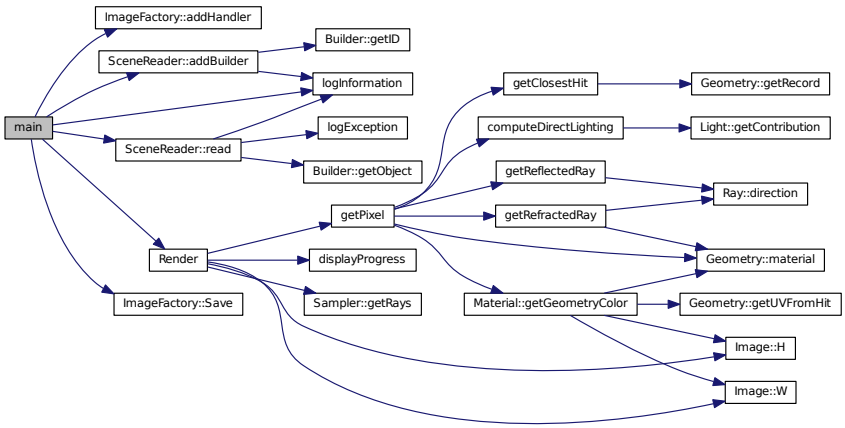
Voici le graphe d'appel pour cette fonction :



5.63.1.7 int main (int *argc,* char ** *argv*)

Définition à la ligne 141 du fichier main.cc.

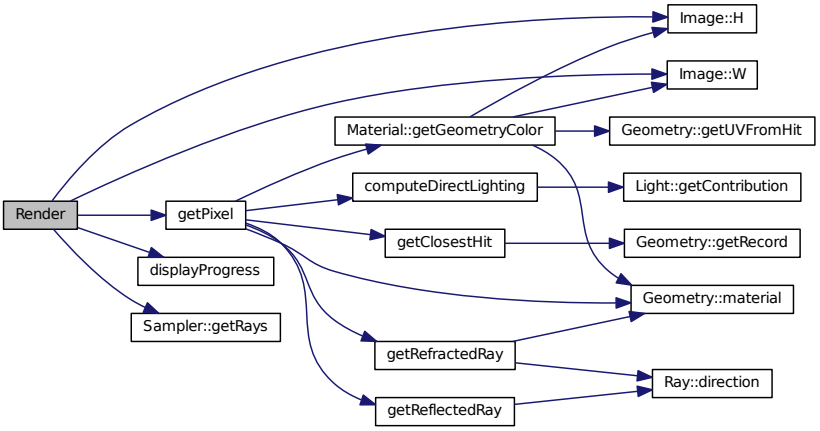
Voici le graphe d'appel pour cette fonction :



5.63.1.8 void Render (Scene & *scene,* Sampler * *sampler*)

Définition à la ligne 119 du fichier main.cc.

Voici le graphe d'appel pour cette fonction :



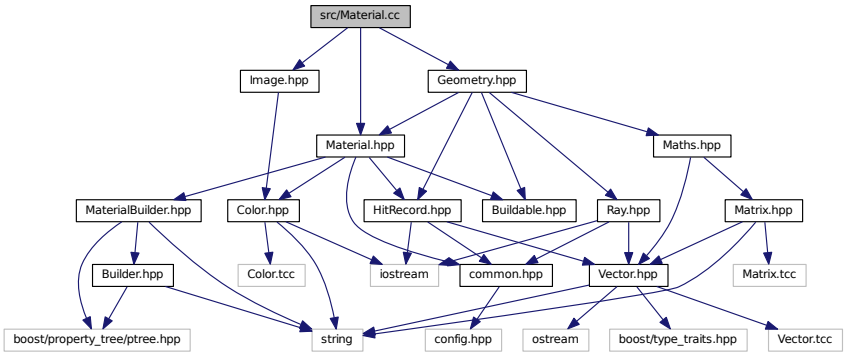
5.63.2 Documentation des variables

5.63.2.1 const unsigned char MAX_RECURSION = 4

Définition à la ligne 26 du fichier main.cc.

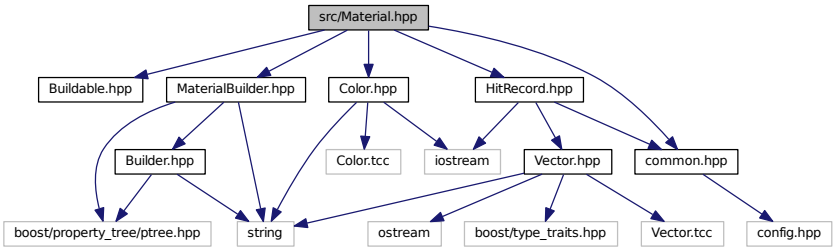
5.64 Référence du fichier src/Material.cc

`#include "Material.hpp" #include <Geometry.hpp> #include <Image.hpp>` Graphe des dépendances par inclusion de Material.cc :



5.65 Référence du fichier src/Material.hpp

`#include <Buildable.hpp>` `#include <MaterialBuilder.hpp>` `#include <HitRecord.hpp>` `#include <common.hpp>` `#include <Color.hpp>` Graphe des dépendances par inclusion de Material.hpp :



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

Classes

– class Material

5.65.1 Description détaillée

/ Material.cc

Auteur

Maxime Gaudin.

Date

2011

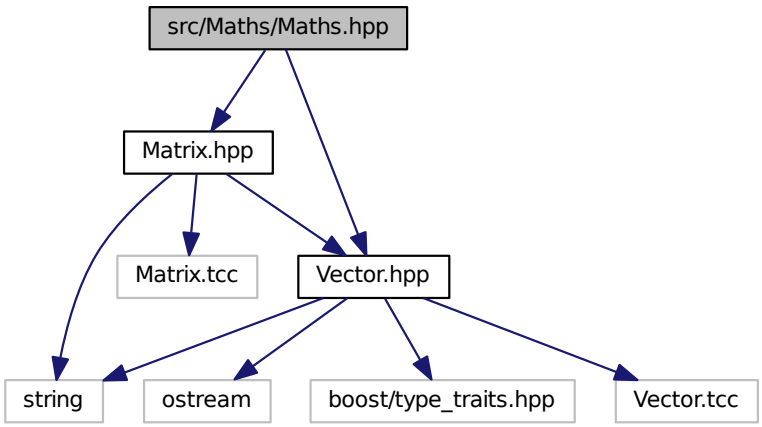
Material définit une classe permettant de stocker les paramètres esthétiques chaque objet comme sa couleur, son opacité, son indice de réfraction, etc.

Les paramètres par défault sont :
– ambient (0.03, 0.03, 0.03)
– opacity (1.0)
– IOR (1.0)

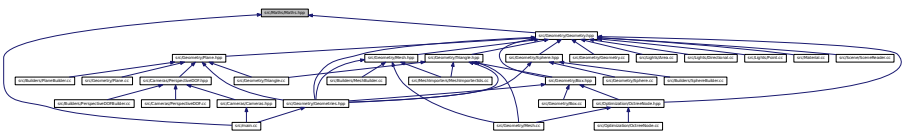
– Reflexivity (0.1)
– Diffuse (0.5, 0.5, 0.5)
– Diffuse Intensity (1.0)
– Specular (1.0, 1.0, 1.0)
– SpecularPower (60)
Définition dans le fichier Material.hpp.

5.66 Référence du fichier src/Maths/Maths.hpp

`#include <Matrix.hpp>` `#include <Vector.hpp>` Graphe des dépendances par inclusion de Maths.hpp :



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



5.66.1 Description détaillée

Auteur

Maxime Gaudin

Date

2011

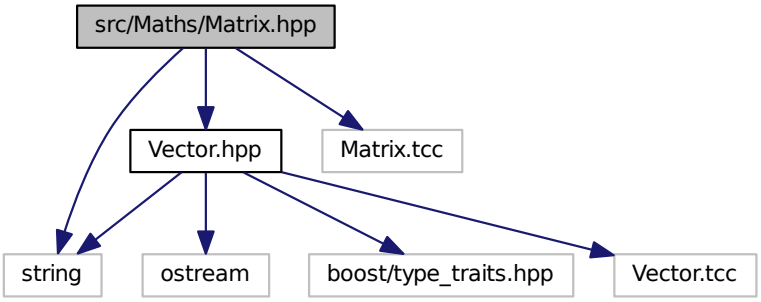
Fichier agrégeant les déclarations des différents templates mathématiques.

Définition dans le fichier Maths.hpp.

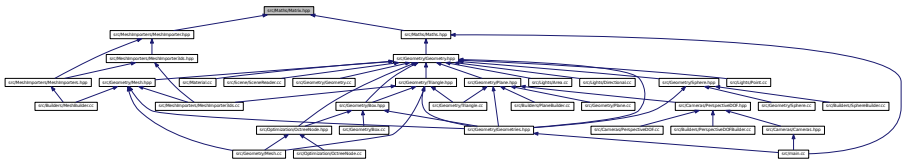
5.67 Référence du fichier src/Maths/Matrix.hpp

Header de la classe Matrix.

`#include <string> #include <Vector.hpp> #include "Matrix.-tcc"` Graphe des dépendances par inclusion de Matrix.hpp :



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



Classes

– class Matrix< P, N, M >

Énumérations

– enum MATRIX_TYPE { ZERO = 0, IDENTITY = 1 }

5.67.1 Description détaillée

Header de la classe Matrix.

Auteur

Maxime Gaudin

Date

2011

La classe Matrix dépend de 3 paramètres template :
– P : Le type numérique à stocker.
– N : Le nombre de lignes de la matrice.
– M : Le nombre de colonne de la matrice.
Les types numériques acceptés sont :
– unsigned char / char
– unsigned short / short
– unsigned int / int
– float
– double
– long
– long X
Définition dans le fichier Matrix.hpp.

5.67.2 Documentation du type de l'énumération

5.67.2.1 enum MATRIX_TYPE

Type de la matrice à construire. Cf. constructeur.

Valeurs énumérées :

ZERO

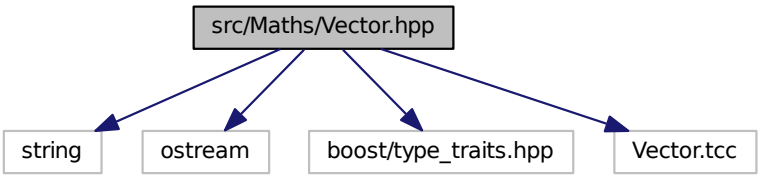
IDENTITY

Définition à la ligne 31 du fichier Matrix.hpp.

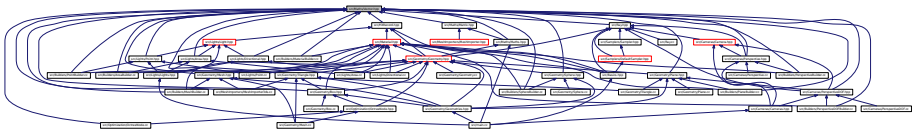
5.68 Référence du fichier src/Maths/Vector.hpp

Header de la classe Vector.

`#include <string>` `#include <ostream>` `#include <boost/type-_traits.hpp>` `#include "Vector.tcc"` Graphe des dépendances par inclusion de Vector.hpp :



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



Classes

– class Vector< P, N >

5.68.1 Description détaillée

Header de la classe Vector.

Auteur

Maxime Gaudin

Date

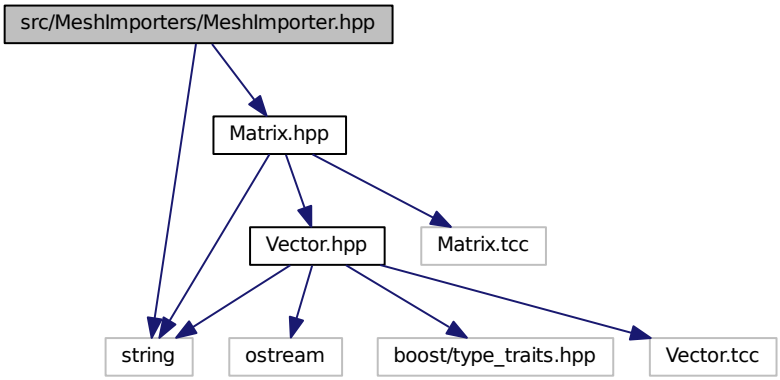
2011

La classe Vector dépend de 2 paramètres template :
– P : Le type numérique à stocker.
– N : Le nombre de lignes du vecteur.
Les types numériques acceptés sont :
– unsigned char / char
– unsigned short / short
– unsigned int / int
– float

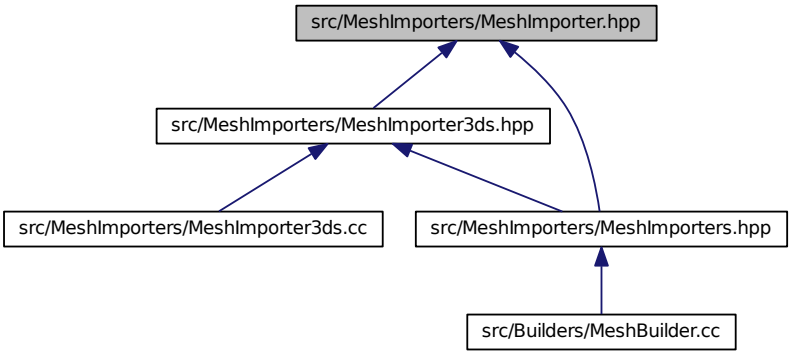
– double
– long
– long X
Définition dans le fichier Vector.hpp.

5.69 Référence du fichier src/MeshImporters/MeshImporter.hpp

`#include <string>` `#include <Matrix.hpp>` Graphe des dépendances par inclusion de MeshImporter.hpp :



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

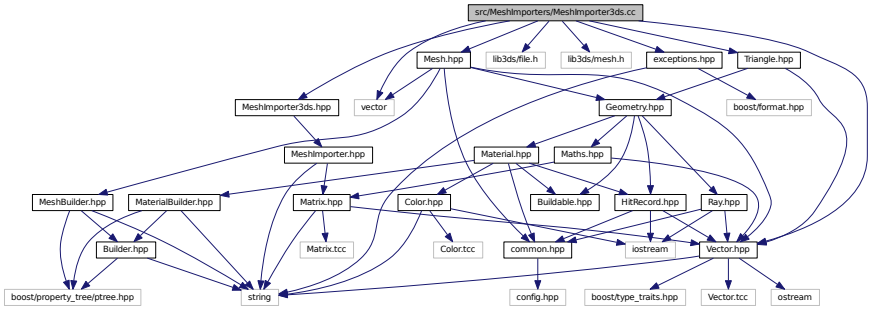


Classes

– class MeshImporter

5.70 Référence du fichier src/MeshImporters/MeshImporter3ds.cc

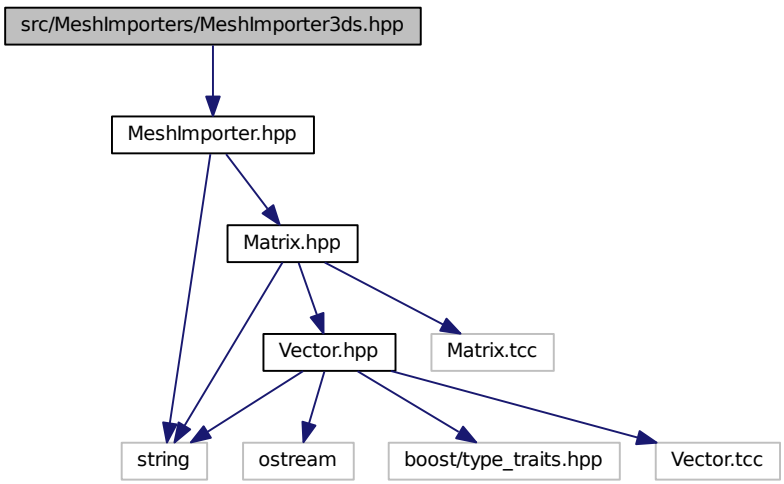
`#include "MeshImporter3ds.hpp" #include <vector> #include <lib3ds/file.h> #include <lib3ds/mesh.h> #include <Triangle.hpp> #include <Vector.hpp> #include <Mesh.hpp> #include <exceptions.hpp>` Graphe des dépendances par inclusion de MeshImporter3ds.cc :



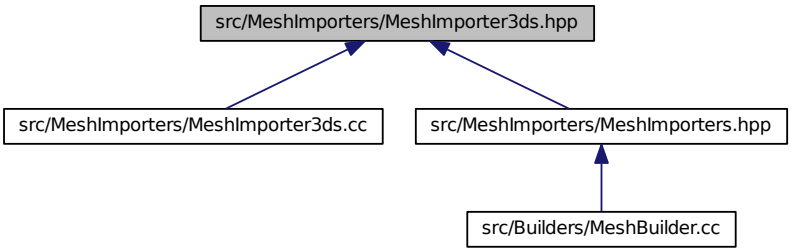
5.71 Référence du fichier src/MeshImporters/MeshImporter3ds.hpp

`#include "MeshImporter.hpp"` Graphe des dépendances par inclusion de -

MeshImporter3ds.hpp :



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



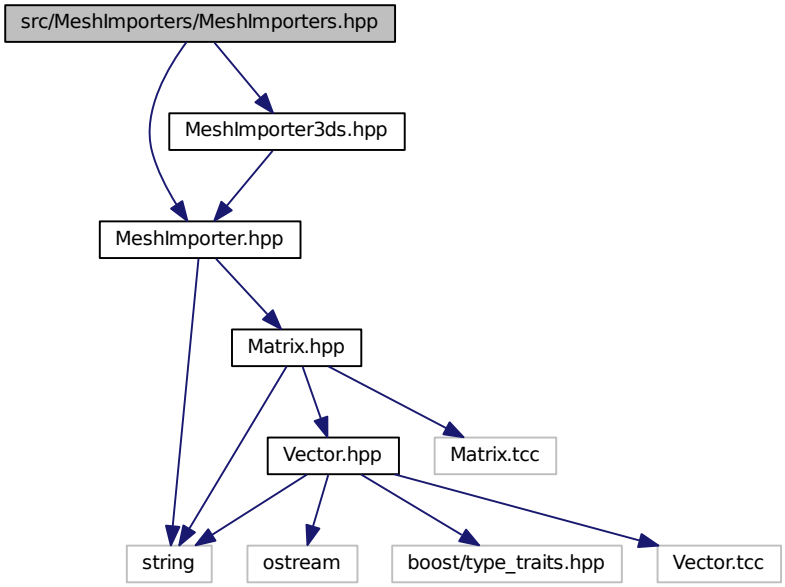
Classes

– class MeshImporter3ds

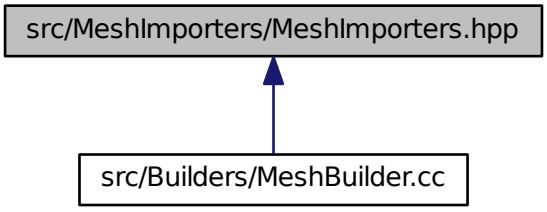
5.72 Référence du fichier src/MeshImporters/MeshImporters.hpp

```
#include <MeshImporter.hpp>   #include <MeshImporter3ds.-
```

hpp> Graphe des dépendances par inclusion de MeshImporters.hpp :



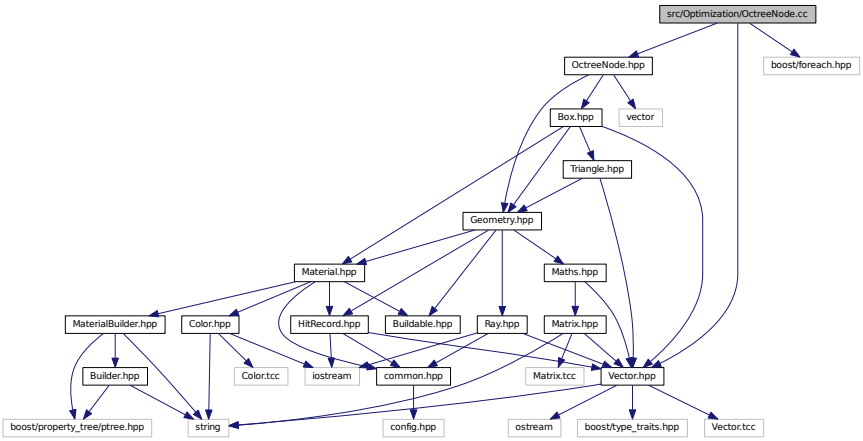
Ce graphe montre quels fichiers incluent directement ou indirectement ce fichier :



5.73 Référence du fichier src/Optimization/OctreeNode.cc

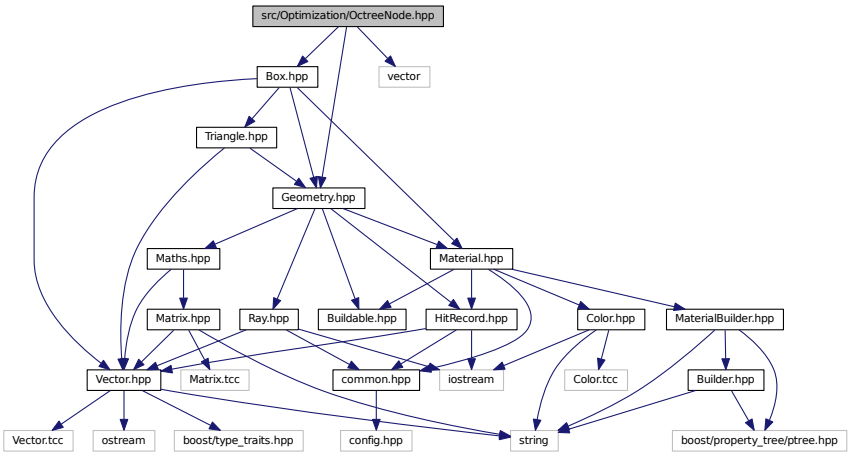
```
#include "OctreeNode.hpp" #include <Vector.hpp> #include
```

<boost/foreach.hpp> Graphe des dépendances par inclusion de OctreeNode.cc :

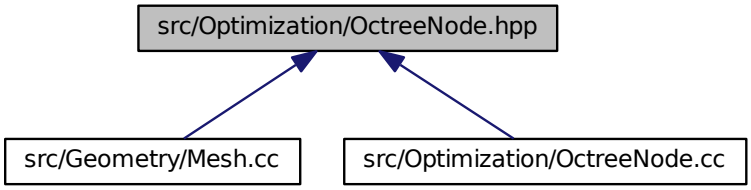


5.74 Référence du fichier src/Optimization/OctreeNode.hpp

#include <Box.hpp> #include <vector> #include <Geometry.-hpp> Graphe des dépendances par inclusion de OctreeNode.hpp :



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

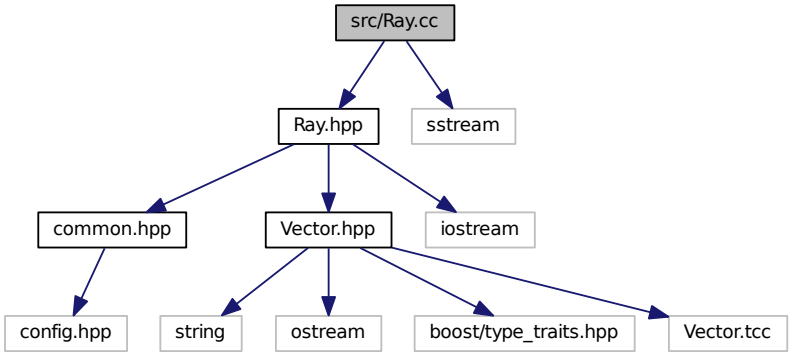


Classes

– class OctreeNode

5.75 Référence du fichier src/Ray.cc

`#include "Ray.hpp"` `#include <sstream>` Graphe des dépendances par inclusion de Ray.cc :



Macros

– #define RAY_TI_H

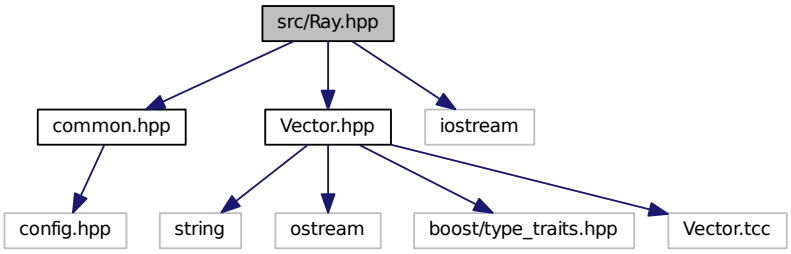
5.75.1 Documentation des macros

5.75.1.1 #define RAY_TI_H

Définition à la ligne 2 du fichier Ray.cc.

5.76 Référence du fichier src/Ray.hpp

`#include <common.hpp>` `#include <Vector.hpp>` `#include <iostream>` Graphe des dépendances par inclusion de Ray.hpp :



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

Classes

– class Ray

5.76.1 Description détaillée

/ Ray.cc

Auteur

Maxime Gaudin.

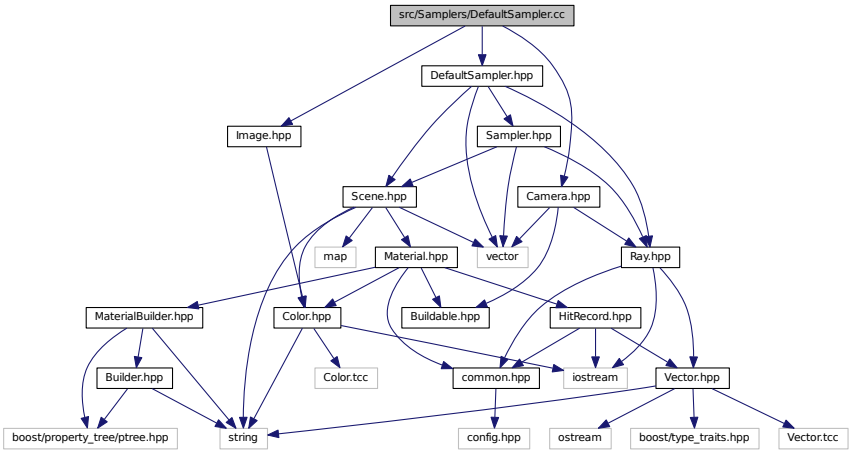
Date

2011

Définition dans le fichier Ray.hpp.

5.77 Référence du fichier src/Samplers/DefaultSampler.cc

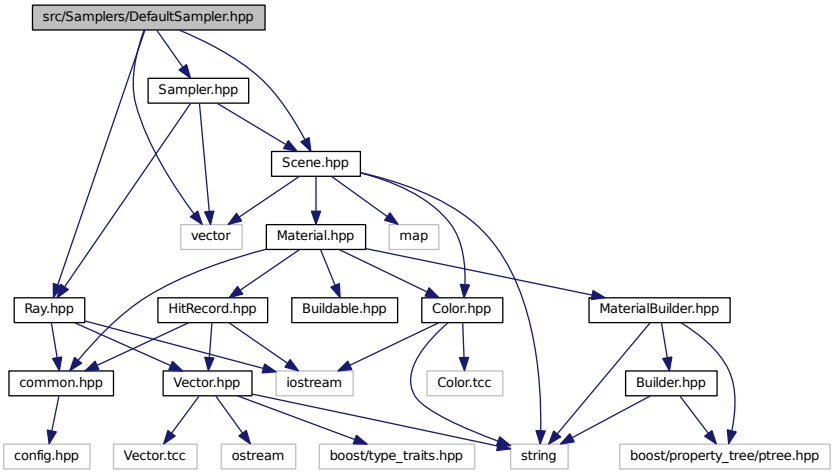
`#include "DefaultSampler.hpp"` `#include <Image.hpp>` `#include <Camera.hpp>` Graphe des dépendances par inclusion de DefaultSampler.cc :



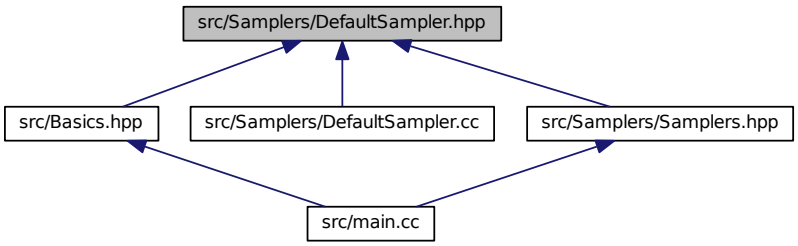
5.78 Référence du fichier src/Samplers/DefaultSampler.hpp

`#include <Sampler.hpp>` `#include <vector>` `#include <Ray.hpp>` `#include <Scene.hpp>` Graphe des dépendances par inclusion de

DefaultSampler.hpp :



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



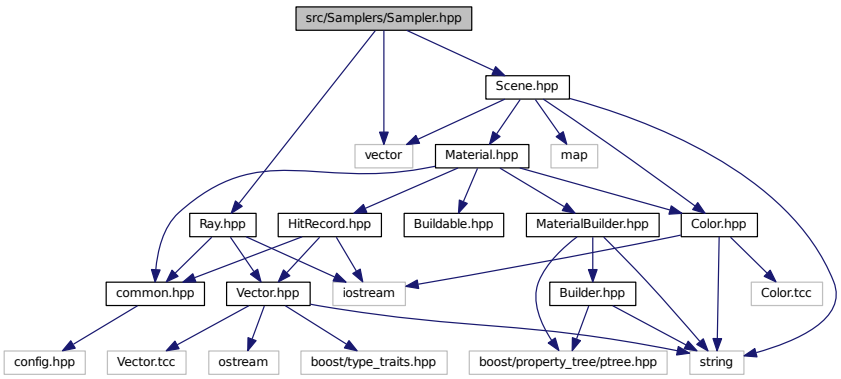
Classes

– class DefaultSampler

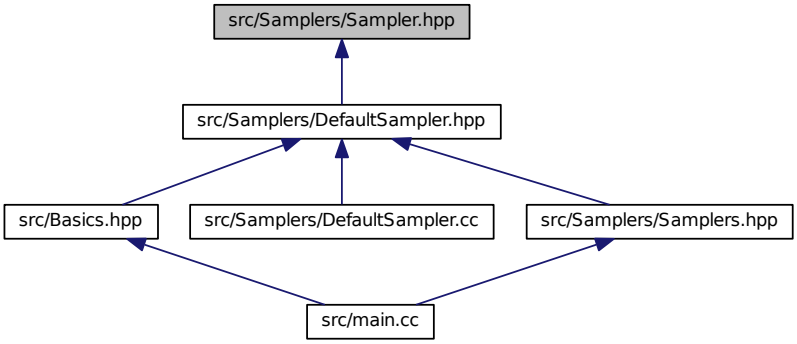
5.79 Référence du fichier src/Samplers/Sampler.hpp

```
#include <vector>  #include <Ray.hpp>  #include <Scene.-
```

hpp> Graphe des dépendances par inclusion de Sampler.hpp :



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



Classes

– class Sampler

5.79.1 Description détaillée

Auteur

Maxime Gaudin

Date

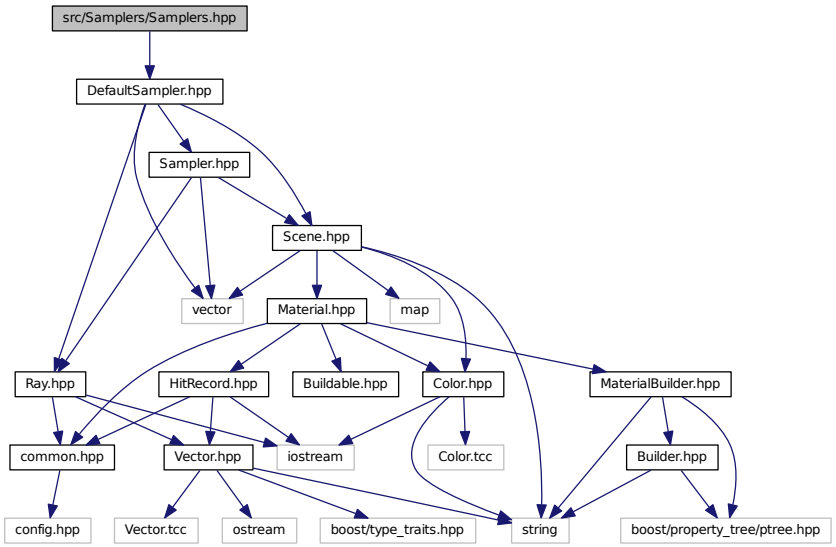
2011

Les samplers ont pour objectif d'échantillonner le nombre de rayons lancés pour chaque pixel.

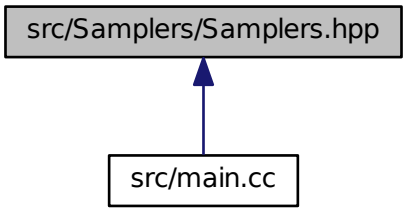
Définition dans le fichier Sampler.hpp.

5.80 Référence du fichier src/Samplers/Samplers.hpp

`#include <DefaultSampler.hpp>` Graphe des dépendances par inclusion de Samplers.hpp :

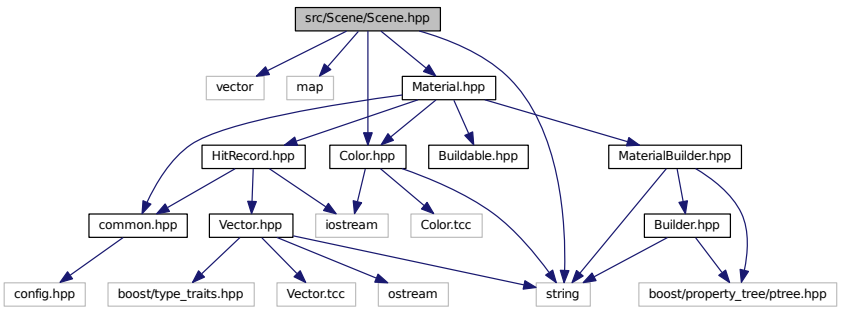


Ce graphe montre quels fichiers incluent directement ou indirectement ce fichier :

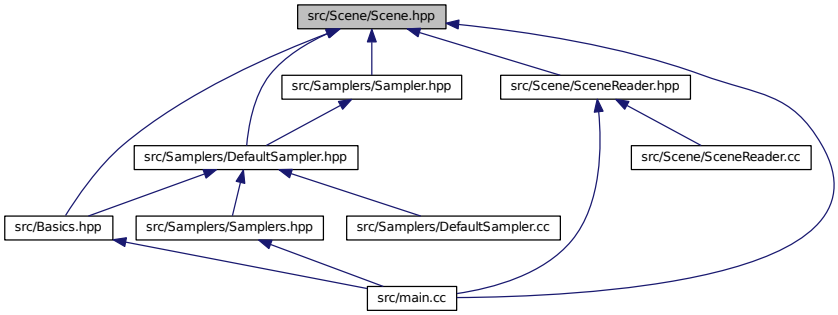


5.81 Référence du fichier src/Scene/Scene.hpp

`#include <vector>` `#include <map>` `#include <string>` × `#include <Color.hpp>` `#include <Material.hpp>` Graphe des dépendances par inclusion de Scene.hpp :



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



Classes

– struct Scene

5.81.1 Description détaillée

Auteur

Maxime Gaudin.

Date

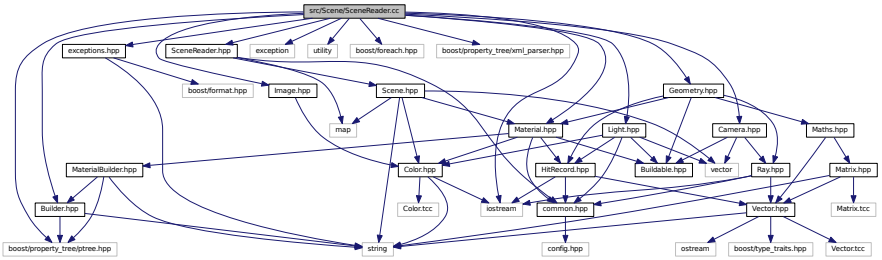
2011

Classe permettant de gérer une scène. Une scène est composée de :
– Une seule caméra,
– un ensemble de lumières,
– un ensemble de géométries,
– une couleur ambiente,
– une image où écrire le rendu.
Définition dans le fichier Scene.hpp.

5.82 Référence du fichier src/Scene/SceneReader.cc

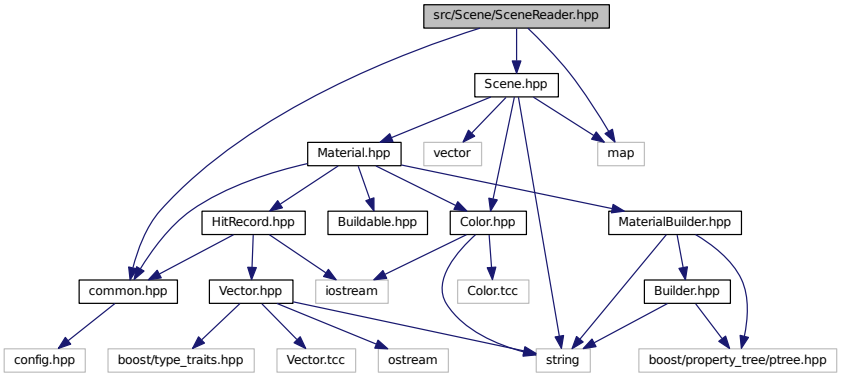
```
#include "SceneReader.hpp" #include <exception> #include
<utility> #include <boost/foreach.hpp> #include <boost/property-
_tree/ptree.hpp> #include <boost/property_tree/xml_parser.-
hpp>  #include <exceptions.hpp>  #include <Image.hpp> ×
#include <Camera.hpp>  #include <Material.hpp>  #include
```

<Light.hpp> #include <Geometry.hpp> #include <Builder.-hpp> #include <iostream> Graphe des dépendances par inclusion de -SceneReader.cc :

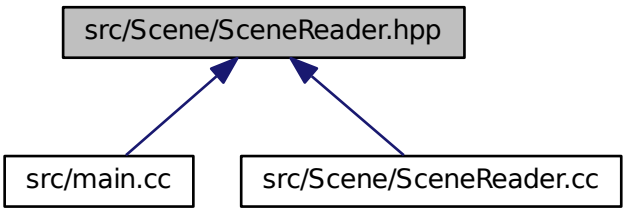


5.83 Référence du fichier src/Scene/SceneReader.hpp

#include <common.hpp> #include <map> #include <Scene.-hpp> Graphe des dépendances par inclusion de SceneReader.hpp :



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



Classes

– class SceneReader