

Moteur 3D - PRAP

Généré par Doxygen 1.15.0



---

<b>1 Topic Index</b>	<b>1</b>
1.1 Topics . . . . .	1
<b>2 Index des espaces de nommage</b>	<b>3</b>
2.1 Liste des espaces de nommage . . . . .	3
<b>3 Index des classes</b>	<b>5</b>
3.1 Liste des classes . . . . .	5
<b>4 Index des fichiers</b>	<b>7</b>
4.1 Liste des fichiers . . . . .	7
<b>5 Topic Documentation</b>	<b>9</b>
5.1 Moteur 3D . . . . .	9
5.1.1 Description détaillée . . . . .	10
<b>6 Documentation des espaces de nommage</b>	<b>11</b>
6.1 Référence de l'espace de nommage Engine3D . . . . .	11
6.1.1 Description détaillée . . . . .	12
<b>7 Documentation des classes</b>	<b>13</b>
7.1 Référence de la classe Engine3D::Camera . . . . .	13
7.1.1 Description détaillée . . . . .	15
7.1.2 Documentation des constructeurs et destructeur . . . . .	15
7.1.2.1 Camera() . . . . .	15
7.1.3 Documentation des fonctions membres . . . . .	16
7.1.3.1 moveForward() . . . . .	16
7.1.3.2 moveLeft() . . . . .	16
7.1.3.3 moveRight() . . . . .	16
7.1.3.4 moveUp() . . . . .	17
7.1.4 Documentation des données membres . . . . .	17
7.1.4.1 direction . . . . .	17
7.1.4.2 fov . . . . .	17
7.1.4.3 nearPlane . . . . .	17
7.1.4.4 position . . . . .	17
7.1.4.5 up . . . . .	18
7.2 Référence de la structure Engine3D::ColoredTriangle . . . . .	18
7.2.1 Description détaillée . . . . .	20
7.2.2 Documentation des données membres . . . . .	20
7.2.2.1 b . . . . .	20
7.2.2.2 g . . . . .	20
7.2.2.3 r . . . . .	20
7.2.2.4 triangle . . . . .	20
7.3 Référence de la classe Engine3D::Pave3d . . . . .	21

---

7.3.1 Description détaillée . . . . .	22
7.3.2 Documentation des constructeurs et destructeur . . . . .	23
7.3.2.1 Pave3d() . . . . .	23
7.3.3 Documentation des fonctions membres . . . . .	23
7.3.3.1 getTriangles() . . . . .	23
7.3.3.2 rotatePoint() . . . . .	24
7.3.3.3 setRotation() . . . . .	24
7.3.4 Documentation des données membres . . . . .	24
7.3.4.1 angleX . . . . .	24
7.3.4.2 angleY . . . . .	24
7.3.4.3 angleZ . . . . .	25
7.3.4.4 center . . . . .	25
7.3.4.5 sizeX . . . . .	25
7.3.4.6 sizeY . . . . .	25
7.3.4.7 sizeZ . . . . .	25
7.4 Référence de la classe Engine3D::Point2d . . . . .	26
7.4.1 Description détaillée . . . . .	27
7.4.2 Documentation des constructeurs et destructeur . . . . .	27
7.4.2.1 Point2d() [1/2] . . . . .	27
7.4.2.2 Point2d() [2/2] . . . . .	27
7.4.3 Documentation des données membres . . . . .	27
7.4.3.1 x . . . . .	27
7.4.3.2 y . . . . .	27
7.4.3.3 z . . . . .	28
7.5 Référence de la classe Engine3D::Point3d . . . . .	28
7.5.1 Description détaillée . . . . .	29
7.5.2 Documentation des constructeurs et destructeur . . . . .	30
7.5.2.1 Point3d() [1/2] . . . . .	30
7.5.2.2 Point3d() [2/2] . . . . .	30
7.5.3 Documentation des fonctions membres . . . . .	30
7.5.3.1 cross() . . . . .	30
7.5.3.2 dot() . . . . .	31
7.5.3.3 normalize() . . . . .	32
7.5.3.4 operator*() . . . . .	32
7.5.3.5 operator+() . . . . .	33
7.5.3.6 operator-() . . . . .	33
7.5.4 Documentation des données membres . . . . .	33
7.5.4.1 x . . . . .	33
7.5.4.2 y . . . . .	34
7.5.4.3 z . . . . .	34
7.6 Référence de la classe Engine3D::Quad3d . . . . .	34
7.6.1 Description détaillée . . . . .	36

---

7.6.2 Documentation des constructeurs et destructeur . . . . .	36
7.6.2.1 Quad3d() [1/2] . . . . .	36
7.6.2.2 Quad3d() [2/2] . . . . .	36
7.6.3 Documentation des fonctions membres . . . . .	37
7.6.3.1 getTriangles() . . . . .	37
7.6.4 Documentation des données membres . . . . .	37
7.6.4.1 t1 . . . . .	37
7.6.4.2 t2 . . . . .	37
7.7 Référence de la classe Engine3D::Scene . . . . .	38
7.7.1 Description détaillée . . . . .	39
7.7.2 Documentation des constructeurs et destructeur . . . . .	40
7.7.2.1 Scene() . . . . .	40
7.7.3 Documentation des fonctions membres . . . . .	40
7.7.3.1 addTriangles() . . . . .	40
7.7.3.2 clearTriangles() . . . . .	40
7.7.3.3 drawScanline() . . . . .	41
7.7.3.4 fillFlatBottom() . . . . .	41
7.7.3.5 fillFlatTop() . . . . .	42
7.7.3.6 getCamera() . . . . .	42
7.7.3.7 isTriangleVisible() . . . . .	43
7.7.3.8 project() . . . . .	43
7.7.3.9 rasterizeTriangle() . . . . .	44
7.7.3.10 render() . . . . .	44
7.7.4 Documentation des données membres . . . . .	45
7.7.4.1 camera . . . . .	45
7.7.4.2 sdl . . . . .	45
7.7.4.3 triangles . . . . .	45
7.8 Référence de la classe Engine3D::Sdl . . . . .	46
7.8.1 Description détaillée . . . . .	47
7.8.2 Documentation des constructeurs et destructeur . . . . .	47
7.8.2.1 Sdl() . . . . .	47
7.8.2.2 ~Sdl() . . . . .	48
7.8.3 Documentation des fonctions membres . . . . .	48
7.8.3.1 clear() . . . . .	48
7.8.3.2 getHeight() . . . . .	48
7.8.3.3 getWidth() . . . . .	49
7.8.3.4 isValid() . . . . .	49
7.8.3.5 present() . . . . .	49
7.8.3.6 setPixel() . . . . .	49
7.8.4 Documentation des données membres . . . . .	50
7.8.4.1 depthBuffer . . . . .	50
7.8.4.2 height . . . . .	50

---

7.8.4.3 pixelBuffer . . . . .	50
7.8.4.4 renderer . . . . .	50
7.8.4.5 texture . . . . .	50
7.8.4.6 width . . . . .	50
7.8.4.7 window . . . . .	51
7.9 Référence de la classe Engine3D::Sphere3d . . . . .	51
7.9.1 Description détaillée . . . . .	53
7.9.2 Documentation des constructeurs et destructeur . . . . .	53
7.9.2.1 Sphere3d() . . . . .	53
7.9.3 Documentation des fonctions membres . . . . .	54
7.9.3.1 getTriangles() . . . . .	54
7.9.3.2 spherePoint() . . . . .	54
7.9.4 Documentation des données membres . . . . .	55
7.9.4.1 center . . . . .	55
7.9.4.2 meridians . . . . .	55
7.9.4.3 parallels . . . . .	55
7.9.4.4 radius . . . . .	55
7.10 Référence de la classe Engine3D::Triangle2d . . . . .	55
7.10.1 Description détaillée . . . . .	57
7.10.2 Documentation des constructeurs et destructeur . . . . .	57
7.10.2.1 Triangle2d() [1/2] . . . . .	57
7.10.2.2 Triangle2d() [2/2] . . . . .	57
7.10.3 Documentation des données membres . . . . .	57
7.10.3.1 p1 . . . . .	57
7.10.3.2 p2 . . . . .	57
7.10.3.3 p3 . . . . .	58
7.11 Référence de la classe Engine3D::Triangle3d . . . . .	58
7.11.1 Description détaillée . . . . .	60
7.11.2 Documentation des constructeurs et destructeur . . . . .	60
7.11.2.1 Triangle3d() [1/2] . . . . .	60
7.11.2.2 Triangle3d() [2/2] . . . . .	60
7.11.3 Documentation des fonctions membres . . . . .	61
7.11.3.1 center() . . . . .	61
7.11.3.2 normal() . . . . .	61
7.11.4 Documentation des données membres . . . . .	62
7.11.4.1 p1 . . . . .	62
7.11.4.2 p2 . . . . .	62
7.11.4.3 p3 . . . . .	62
<b>8 Documentation des fichiers</b>	<b>63</b>
8.1 Référence du fichier Geometry.cpp . . . . .	63
8.1.1 Description détaillée . . . . .	63

---

8.2 Référence du fichier Geometry.hpp . . . . .	64
8.2.1 Description détaillée . . . . .	65
8.3 Geometry.hpp . . . . .	65
8.4 Référence du fichier main.cpp . . . . .	67
8.4.1 Description détaillée . . . . .	68
8.4.2 Documentation des fonctions . . . . .	68
8.4.2.1 addCube() . . . . .	68
8.4.2.2 addSphere() . . . . .	69
8.4.2.3 handleKeyboard() . . . . .	69
8.4.2.4 main() . . . . .	70
8.4.2.5 mainLoop() . . . . .	71
8.4.3 Documentation des variables . . . . .	72
8.4.3.1 CAMERA_SPEED . . . . .	72
8.4.3.2 ROTATION_SPEED . . . . .	72
8.4.3.3 WINDOW_HEIGHT . . . . .	72
8.4.3.4 WINDOW_WIDTH . . . . .	72
8.5 Référence du fichier Scene.cpp . . . . .	73
8.5.1 Description détaillée . . . . .	73
8.6 Référence du fichier Scene.hpp . . . . .	73
8.6.1 Description détaillée . . . . .	75
8.7 Scene.hpp . . . . .	75
8.8 Référence du fichier Sdl.cpp . . . . .	76
8.8.1 Description détaillée . . . . .	76
8.9 Référence du fichier Sdl.hpp . . . . .	77
8.9.1 Description détaillée . . . . .	78
8.10 Sdl.hpp . . . . .	78
<b>Index</b>	<b>79</b>



# **Chapitre 1**

## **Topic Index**

### **1.1 Topics**

Here is a list of all topics with brief descriptions:

Moteur 3D . . . . .	9
---------------------	---



## Chapitre 2

# Index des espaces de nommage

### 2.1 Liste des espaces de nommage

Liste de tous les espaces de nommage avec une brève description:

Engine3D	Espace de noms contenant toutes les classes du moteur 3D . . . . .	11
----------	--	----



# Chapitre 3

## Index des classes

### 3.1 Liste des classes

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

<a href="#">Engine3D::Camera</a>	Représente la caméra (œil) dans la scène . . . . .	13
<a href="#">Engine3D::ColoredTriangle</a>	Triangle avec sa couleur associée . . . . .	18
<a href="#">Engine3D::Pave3d</a>	Pavé composé de 6 faces ( <a href="#">Quad3d</a> ) . . . . .	21
<a href="#">Engine3D::Point2d</a>	Représente un point sur l'écran 2D . . . . .	26
<a href="#">Engine3D::Point3d</a>	Représente un point dans l'espace 3D . . . . .	28
<a href="#">Engine3D::Quad3d</a>	Quadrilatère composé de deux triangles . . . . .	34
<a href="#">Engine3D::Scene</a>	Gère le rendu de la scène 3D . . . . .	38
<a href="#">Engine3D::Sdl</a>	Classe contenant les fonctionnalités SDL pour le rendu graphique . . . . .	46
<a href="#">Engine3D::Sphere3d</a>	Sphère composée d'un maillage de <a href="#">Quad3d</a> . . . . .	51
<a href="#">Engine3D::Triangle2d</a>	Triangle projeté sur l'écran 2D . . . . .	55
<a href="#">Engine3D::Triangle3d</a>	Triangle dans l'espace 3D . . . . .	58



# Chapitre 4

## Index des fichiers

### 4.1 Liste des fichiers

Liste de tous les fichiers avec une brève description :

<a href="#">Geometry.cpp</a>	Implémentation des classes géométriques . . . . .	63
<a href="#">Geometry.hpp</a>	Définition des classes géométriques . . . . .	64
<a href="#">main.cpp</a>	Boucle principale et gestion des événements clavier . . . . .	67
<a href="#">Scene.cpp</a>	Implémentation de la gestion de scène et rasterisation . . . . .	73
<a href="#">Scene.hpp</a>	Gestion de la scène, caméra et rasterisation . . . . .	73
<a href="#">Sdl.cpp</a>	Implémentation de la classe Sdl . . . . .	76
<a href="#">Sdl.hpp</a>	Encapsulation de la bibliothèque SDL2 . . . . .	77



# Chapitre 5

## Topic Documentation

### 5.1 Moteur 3D

Ensemble des classes du moteur de rendu 3D.

#### Espaces de nommage

- namespace [Engine3D](#)

*Espace de noms contenant toutes les classes du moteur 3D.*

#### Classes

- class [Engine3D::Point3d](#)

*Représente un point dans l'espace 3D.*

- class [Engine3D::Point2d](#)

*Représente un point sur l'écran 2D.*

- class [Engine3D::Triangle3d](#)

*Triangle dans l'espace 3D.*

- class [Engine3D::Triangle2d](#)

*Triangle projeté sur l'écran 2D.*

- class [Engine3D::Quad3d](#)

*Quadrilatère composé de deux triangles.*

- class [Engine3D::Pave3d](#)

*Pavé composé de 6 faces ([Quad3d](#)).*

- class [Engine3D::Sphere3d](#)

*Sphère composée d'un maillage de Quad3d.*

- class [Engine3D::Camera](#)

*Représente la caméra (œil) dans la scène.*

- struct [Engine3D::ColoredTriangle](#)

*Triangle avec sa couleur associée.*

- class [Engine3D::Scene](#)

*Gère le rendu de la scène 3D.*

- class [Engine3D::Sdl](#)

*Classe contenant les fonctionnalités SDL pour le rendu graphique.*

### 5.1.1 Description détaillée

Ensemble des classes du moteur de rendu 3D.

# Chapitre 6

## Documentation des espaces de nommage

### 6.1 Référence de l'espace de nommage Engine3D

Espace de noms contenant toutes les classes du moteur 3D.

#### Classes

— class [Point3d](#)

*Représente un point dans l'espace 3D.*

— class [Point2d](#)

*Représente un point sur l'écran 2D.*

— class [Triangle3d](#)

*Triangle dans l'espace 3D.*

— class [Triangle2d](#)

*Triangle projeté sur l'écran 2D.*

— class [Quad3d](#)

*Quadrilatère composé de deux triangles.*

— class [Pave3d](#)

*Pavé composé de 6 faces ([Quad3d](#)).*

— class [Sphere3d](#)

*Sphère composée d'un maillage de [Quad3d](#).*

— class [Camera](#)

*Représente la caméra (œil) dans la scène.*

— struct [ColoredTriangle](#)

*Triangle avec sa couleur associée.*

— class [Scene](#)

*Gère le rendu de la scène 3D.*

— class [Sdl](#)

*Classe contenant les fonctionnalités SDL pour le rendu graphique.*

### 6.1.1 Description détaillée

Espace de noms contenant toutes les classes du moteur 3D.

## **Chapitre 7**

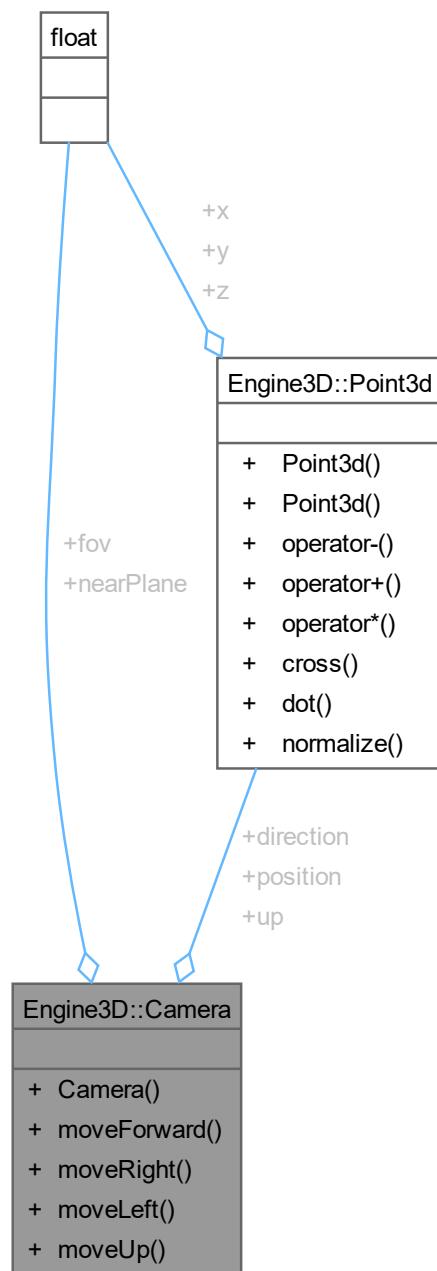
# **Documentation des classes**

### **7.1 Référence de la classe Engine3D::Camera**

Représente la caméra (œil) dans la scène.

```
#include <Scene.hpp>
```

Graphe de collaboration de Engine3D::Camera:



## Fonctions membres publiques

- [Camera \(\)](#)  
*Constructeur par défaut.*
- [void moveForward \(float delta\)](#)

*Déplace la caméra vers l'avant/arrière.*

- void [moveRight](#) (float delta)

*Déplace la caméra vers la droite.*

- void [moveLeft](#) (float delta)

*Déplace la caméra vers la gauche.*

- void [moveUp](#) (float delta)

*Déplace la caméra vers le haut/bas.*

### Attributs publics

- Point3d [position](#)

*Position de l'oeil.*

- Point3d [direction](#)

*Direction de l'oeil.*

- Point3d [up](#)

*Vecteur haut.*

- float [fov](#)

*Champ de vision (field of view).*

- float [nearPlane](#)

*Plan de projection proche.*

### 7.1.1 Description détaillée

Représente la caméra (œil) dans la scène.

### 7.1.2 Documentation des constructeurs et destructeur

#### 7.1.2.1 Camera()

Engine3D::Camera::Camera ()

Constructeur par défaut.

### 7.1.3 Documentation des fonctions membres

#### 7.1.3.1 moveForward()

```
void Engine3D::Camera::moveForward (
    float delta)
```

Déplace la caméra vers l'avant/arrière.

Voici le graphe des appelants de cette fonction :



#### 7.1.3.2 moveLeft()

```
void Engine3D::Camera::moveLeft (
    float delta)
```

Déplace la caméra vers la gauche.

Voici le graphe des appelants de cette fonction :



#### 7.1.3.3 moveRight()

```
void Engine3D::Camera::moveRight (
    float delta)
```

Déplace la caméra vers la droite.

Voici le graphe des appelants de cette fonction :



### 7.1.3.4 moveUp()

```
void Engine3D::Camera::moveUp (
    float delta)
```

Déplace la caméra vers le haut/bas.

Voici le graphe des appelants de cette fonction :



## 7.1.4 Documentation des données membres

### 7.1.4.1 direction

```
Point3d Engine3D::Camera::direction
```

Direction de l'oeil.

### 7.1.4.2 fov

```
float Engine3D::Camera::fov
```

Champ de vision (field of view).

### 7.1.4.3 nearPlane

```
float Engine3D::Camera::nearPlane
```

Plan de projection proche.

### 7.1.4.4 position

```
Point3d Engine3D::Camera::position
```

Position de l'oeil.

#### 7.1.4.5 up

```
Point3d Engine3D::Camera::up
```

Vecteur haut.

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

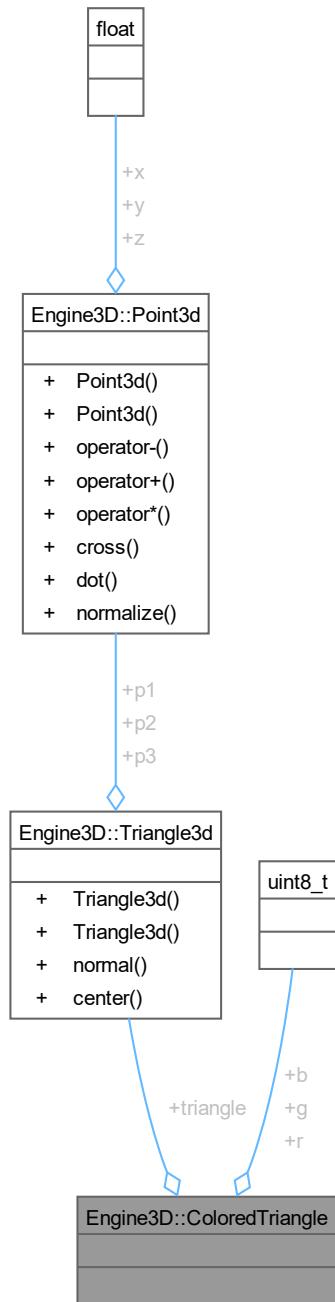
- [Scene.hpp](#)
- [Scene.cpp](#)

## 7.2 Référence de la structure Engine3D::ColoredTriangle

Triangle avec sa couleur associée.

```
#include <Scene.hpp>
```

Graphe de collaboration de Engine3D::ColoredTriangle:



### Attributs publics

- `Triangle3d triangle`

*Le triangle 3D.*

- `uint8_t r`

*Composante rouge.*

— `uint8_t g`

*Composante verte.*

— `uint8_t b`

*Composante bleue.*

### 7.2.1 Description détaillée

Triangle avec sa couleur associée.

### 7.2.2 Documentation des données membres

#### 7.2.2.1 `b`

```
uint8_t Engine3D::ColoredTriangle::b
```

Composante bleue.

#### 7.2.2.2 `g`

```
uint8_t Engine3D::ColoredTriangle::g
```

Composante verte.

#### 7.2.2.3 `r`

```
uint8_t Engine3D::ColoredTriangle::r
```

Composante rouge.

#### 7.2.2.4 `triangle`

`Triangle3d Engine3D::ColoredTriangle::triangle`

Le triangle 3D.

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

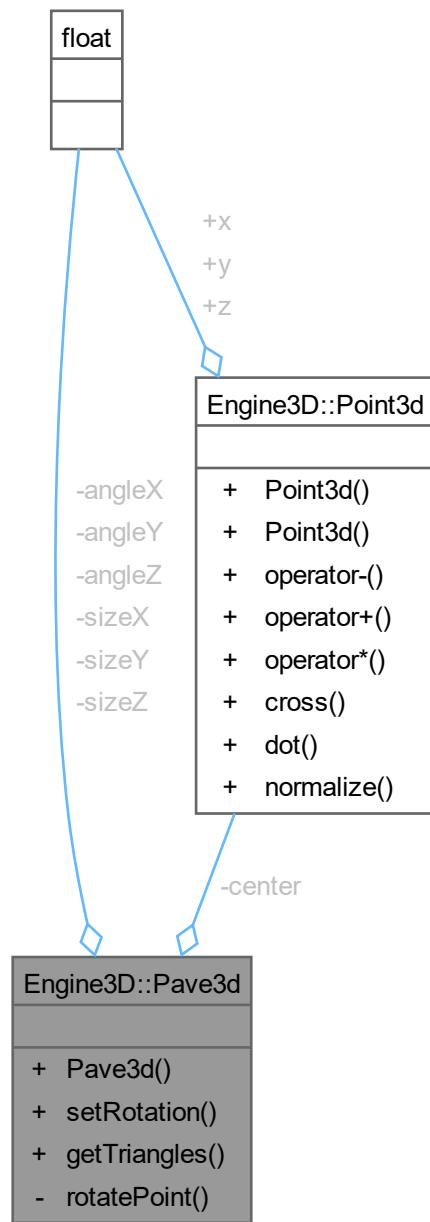
— `Scene.hpp`

### 7.3 Référence de la classe Engine3D::Pave3d

Pavé composé de 6 faces ([Quad3d](#)).

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

Graphe de collaboration de Engine3D::Pave3d:



#### Fonctions membres publiques

- `Pave3d (const Point3d &center, float sizeX, float sizeY, float sizeZ)`

*Constructeur avec paramètres.*

- void `setRotation` (float `angleX`, float `angleY`, float `angleZ`)

*Définit les angles de rotation.*

- std::vector< `Triangle3d` > `getTriangles` () const

*Retourne tous les triangles du pavé*

### Fonctions membres privées

- `Point3d rotatePoint` (const `Point3d` &p) const

*Applique la rotation à un point.*

### Attributs privés

- `Point3d center`

*Centre du pavé*

- float `sizeX`

*Taille en X (largeur).*

- float `sizeY`

*Taille en Y (hauteur).*

- float `sizeZ`

*Taille en Z (profondeur).*

- float `angleX`

*Angle de rotation X.*

- float `angleY`

*Angle de rotation Y.*

- float `angleZ`

*Angle de rotation Z.*

### 7.3.1 Description détaillée

Pavé composé de 6 faces ([Quad3d](#)).

### 7.3.2 Documentation des constructeurs et destructeur

#### 7.3.2.1 Pave3d()

```
Engine3D::Pave3d::Pave3d (
    const Point3d & center,
    float sizeX,
    float sizeY,
    float sizeZ)
```

Constructeur avec paramètres.

#### Paramètres

<i>center</i>	
<i>sizeX</i>	
<i>sizeY</i>	
<i>sizeZ</i>	

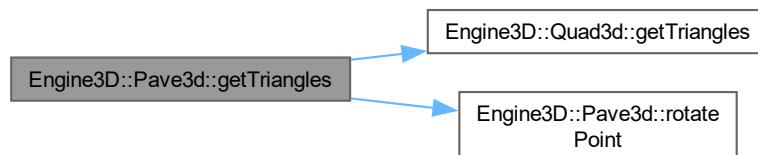
### 7.3.3 Documentation des fonctions membres

#### 7.3.3.1 getTriangles()

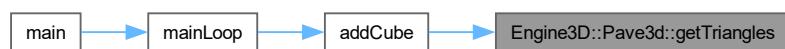
```
std::vector< Triangle3d > Engine3D::Pave3d::getTriangles () const
```

Retourne tous les triangles du pavé

Voici le graphe d'appel pour cette fonction :



Voici le graphe des appelants de cette fonction :

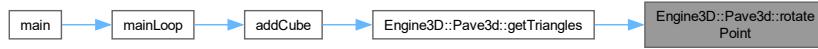


### 7.3.3.2 rotatePoint()

```
Point3d Engine3D::Pave3d::rotatePoint (
    const Point3d & p) const [private]
```

Applique la rotation à un point.

Voici le graphe des appelants de cette fonction :

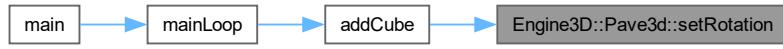


### 7.3.3.3 setRotation()

```
void Engine3D::Pave3d::setRotation (
    float angleX,
    float angleY,
    float angleZ)
```

Définit les angles de rotation.

Voici le graphe des appelants de cette fonction :



## 7.3.4 Documentation des données membres

### 7.3.4.1 angleX

```
float Engine3D::Pave3d::angleX [private]
```

Angle de rotation X.

### 7.3.4.2 angleY

```
float Engine3D::Pave3d::angleY [private]
```

Angle de rotation Y.

### 7.3.4.3 angleZ

```
float Engine3D::Pave3d::angleZ [private]
```

Angle de rotation Z.

### 7.3.4.4 center

```
Point3d Engine3D::Pave3d::center [private]
```

Centre du pavé

### 7.3.4.5 sizeX

```
float Engine3D::Pave3d::sizeX [private]
```

Taille en X (largeur).

### 7.3.4.6 sizeY

```
float Engine3D::Pave3d::sizeY [private]
```

Taille en Y (hauteur).

### 7.3.4.7 sizeZ

```
float Engine3D::Pave3d::sizeZ [private]
```

Taille en Z (profondeur).

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

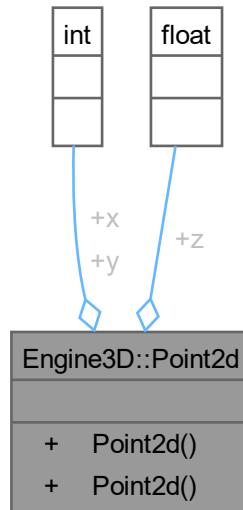
- [Geometry.hpp](#)
- [Geometry.cpp](#)

## 7.4 Référence de la classe Engine3D::Point2d

Représente un point sur l'écran 2D.

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

Graphe de collaboration de Engine3D::Point2d:



### Fonctions membres publiques

- [Point2d \(\)](#)  
*Constructeur par défaut.*
- [Point2d \(int x, int y, float z\)](#)  
*Constructeur avec coordonnées.*

### Attributs publics

- [int x](#)  
*Coordonnée X mais sur l'écran (projétée).*
- [int y](#)  
*Coordonnée Y mais sur l'écran (projétée).*
- [float z](#)  
*Profondeur.*

### 7.4.1 Description détaillée

Représente un point sur l'écran 2D.

### 7.4.2 Documentation des constructeurs et destructeur

#### 7.4.2.1 Point2d() [1/2]

```
Engine3D::Point2d::Point2d ()
```

Constructeur par défaut.

#### 7.4.2.2 Point2d() [2/2]

```
Engine3D::Point2d::Point2d (
    int x,
    int y,
    float z)
```

Constructeur avec coordonnées.

#### Paramètres

$x$	
$y$	
$z$	

### 7.4.3 Documentation des données membres

#### 7.4.3.1 x

```
int Engine3D::Point2d::x
```

Coordonnée X mais sur l'écran (projetée).

#### 7.4.3.2 y

```
int Engine3D::Point2d::y
```

Coordonnée Y mais sur l'écran (projetée).

### 7.4.3.3 z

```
float Engine3D::Point2d::z
```

Profondeur.

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

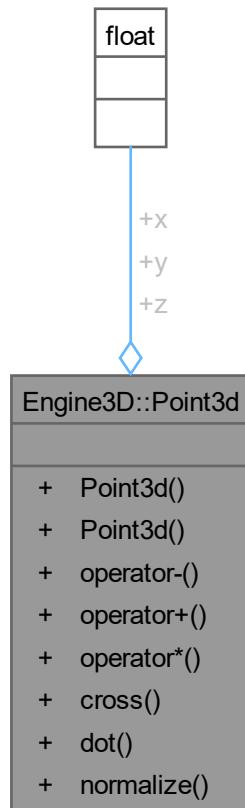
- [Geometry.hpp](#)
- [Geometry.cpp](#)

## 7.5 Référence de la classe Engine3D::Point3d

Représente un point dans l'espace 3D.

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

Graphe de collaboration de Engine3D::Point3d:



**Fonctions membres publiques**

- `Point3d ()`

*Constructeur par défaut.*
- `Point3d (float x, float y, float z)`

*Constructeur avec coordonnées.*
- `Point3d operator- (const Point3d &other) const`

*Soustraction de deux points (retourne un vecteur).*
- `Point3d operator+ (const Point3d &other) const`

*Addition de deux points.*
- `Point3d operator* (float scalar) const`

*Multiplication par un scalaire.*
- `Point3d cross (const Point3d &other) const`

*Produit vectoriel.*
- `float dot (const Point3d &other) const`

*Produit scalaire.*
- `Point3d normalize () const`

*Normalise le vecteur.*

**Attributs publics**

- `float x`

*Coordonnée X.*
- `float y`

*Coordonnée Y.*
- `float z`

*Coordonnée Z.*

**7.5.1 Description détaillée**

Représente un point dans l'espace 3D.

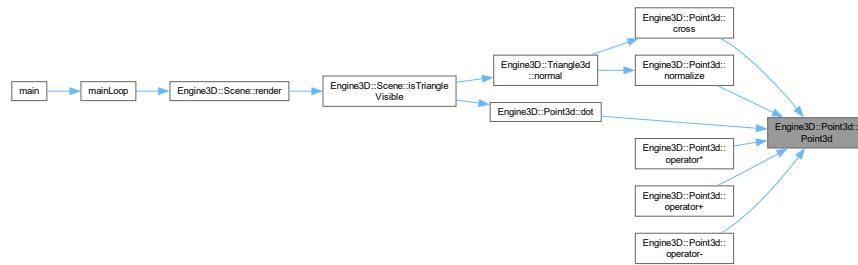
## 7.5.2 Documentation des constructeurs et destructeur

### 7.5.2.1 Point3d() [1/2]

```
Engine3D::Point3d::Point3d ()
```

Constructeur par défaut.

Voici le graphe des appels de cette fonction :



### 7.5.2.2 Point3d() [2/2]

```
Engine3D::Point3d::Point3d (
    float x,
    float y,
    float z)
```

Constructeur avec coordonnées.

#### Paramètres

<i>x</i>	
<i>y</i>	
<i>z</i>	

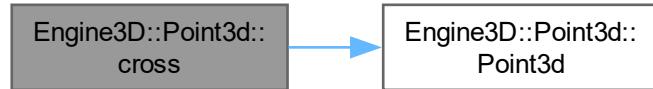
## 7.5.3 Documentation des fonctions membres

### 7.5.3.1 cross()

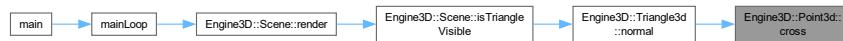
```
Point3d Engine3D::Point3d::cross (
    const Point3d & other) const
```

Produit vectoriel.

Voici le graphe d'appel pour cette fonction :



Voici le graphe des appelants de cette fonction :

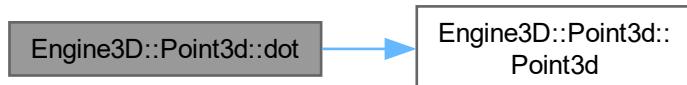


### 7.5.3.2 dot()

```
float Engine3D::Point3d::dot (
    const Point3d & other) const
```

Produit scalaire.

Voici le graphe d'appel pour cette fonction :



Voici le graphe des appelants de cette fonction :

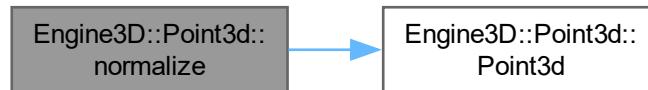


### 7.5.3.3 normalize()

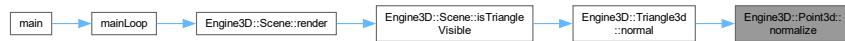
```
Point3d Engine3D::Point3d::normalize () const
```

Normalise le vecteur.

Voici le graphe d'appel pour cette fonction :



Voici le graphe des appels de cette fonction :

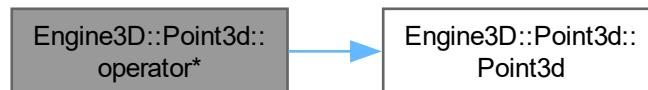


### 7.5.3.4 operator\*()

```
Point3d Engine3D::Point3d::operator* (
    float scalar) const
```

Multiplication par un scalaire.

Voici le graphe d'appel pour cette fonction :

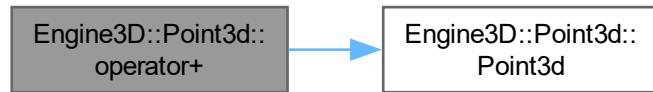


### 7.5.3.5 operator+()

```
Point3d Engine3D::Point3d::operator+ (
    const Point3d & other) const
```

Addition de deux points.

Voici le graphe d'appel pour cette fonction :

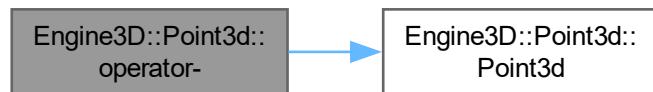


### 7.5.3.6 operator-()

```
Point3d Engine3D::Point3d::operator- (
    const Point3d & other) const
```

Soustraction de deux points (retourne un vecteur).

Voici le graphe d'appel pour cette fonction :



## 7.5.4 Documentation des données membres

### 7.5.4.1 x

```
float Engine3D::Point3d::x
```

Coordonnée X.

#### 7.5.4.2 y

```
float Engine3D::Point3d::y
```

Coordonnée Y.

#### 7.5.4.3 z

```
float Engine3D::Point3d::z
```

Coordonnée Z.

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

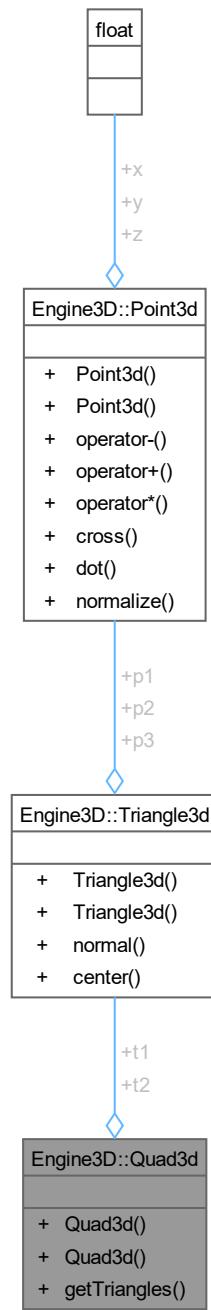
- [Geometry.hpp](#)
- [Geometry.cpp](#)

## 7.6 Référence de la classe Engine3D::Quad3d

Quadrilatère composé de deux triangles.

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

Graphe de collaboration de Engine3D::Quad3d:



## Fonctions membres publiques

### — `Quad3d ()`

*Constructeur par défaut.*

### — `Quad3d (const Point3d &p1, const Point3d &p2, const Point3d &p3, const Point3d &p4)`

*Constructeur avec quatre points (dans le sens trigonométrique).*

- std::vector< [Triangle3d](#) > [getTriangles](#) () const

*Retourne les triangles du quad.*

#### Attributs publics

- [Triangle3d t1](#)

*Premier triangle.*

- [Triangle3d t2](#)

*Deuxième triangle.*

### 7.6.1 Description détaillée

Quadrilatère composé de deux triangles.

### 7.6.2 Documentation des constructeurs et destructeur

#### 7.6.2.1 Quad3d() [1/2]

`Engine3D::Quad3d::Quad3d ()`

Constructeur par défaut.

#### 7.6.2.2 Quad3d() [2/2]

```
Engine3D::Quad3d::Quad3d (
    const Point3d & p1,
    const Point3d & p2,
    const Point3d & p3,
    const Point3d & p4)
```

Constructeur avec quatre points (dans le sens trigonométrique).

#### Paramètres

<i>p1</i>	
<i>p2</i>	
<i>p3</i>	
<i>p4</i>	

### 7.6.3 Documentation des fonctions membres

#### 7.6.3.1 getTriangles()

```
std::vector< Triangle3d > Engine3D::Quad3d::getTriangles () const
```

Retourne les triangles du quad.

Voici le graphe des appels de cette fonction :



### 7.6.4 Documentation des données membres

#### 7.6.4.1 t1

```
Triangle3d Engine3D::Quad3d::t1
```

Premier triangle.

#### 7.6.4.2 t2

```
Triangle3d Engine3D::Quad3d::t2
```

Deuxième triangle.

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

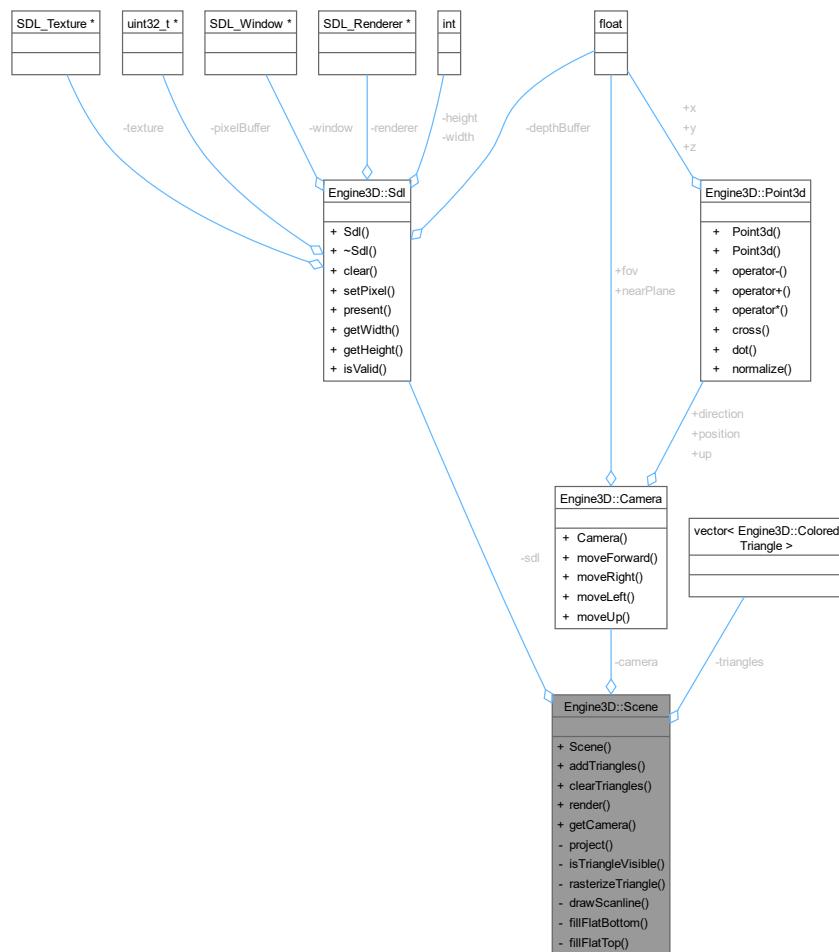
- [Geometry.hpp](#)
- [Geometry.cpp](#)

## 7.7 Référence de la classe Engine3D::Scene

Gère le rendu de la scène 3D.

```
#include <Scene.hpp>
```

Graphe de collaboration de Engine3D::Scene:



### Fonctions membres publiques

- [Scene \(Sdl &sdl\)](#)

*Constructeur.*

- [void addTriangles \(const std::vector< Triangle3d > &tris, uint8\\_t r, uint8\\_t g, uint8\\_t b\)](#)

*Ajoute des triangles à la scène.*

- [void clearTriangles \(\)](#)

*Efface les triangles de la scène.*

- void `render ()`

*Effectue le rendu de la scène.*

- Camera & `getCamera ()`

*Retourne une référence vers la caméra.*

### Fonctions membres privées

- Point2d `project (const Point3d &p) const`

*Projette un point 3D en point 2D.*

- bool `isTriangleVisible (const Triangle3d &triangle) const`

*Vérifie si un triangle est visible.*

- void `rasterizeTriangle (const Triangle2d &triangle, float avgDepth, uint8_t r, uint8_t g, uint8_t b)`

*Rasterise un triangle 2D.*

- void `drawScanline (int y, int x1, int x2, float depth, uint8_t r, uint8_t g, uint8_t b)`

*Dessine une ligne horizontale du triangle via technique scanline.*

- void `fillFlatBottom (const Point2d &p1, const Point2d &p2, const Point2d &p3, float depth, uint8_t r, uint8_t g, uint8_t b)`

*Rasterise la partie plate du triangle (haut ou bas).*

- void `fillFlatTop (const Point2d &p1, const Point2d &p2, const Point2d &p3, float depth, uint8_t r, uint8_t g, uint8_t b)`

*Rasterise la partie plate supérieure du triangle.*

### Attributs privés

- Sdl & `sdl`

*Référence vers SDL.*

- Camera `camera`

*Caméra de la scène.*

- std::vector< ColoredTriangle > `triangles`

*Triangles colorés à render.*

## 7.7.1 Description détaillée

Gère le rendu de la scène 3D.

## 7.7.2 Documentation des constructeurs et destructeur

### 7.7.2.1 Scene()

```
Engine3D::Scene::Scene (
    Sdl & sdl)
```

Constructeur.

#### Paramètres

sdl	Référence vers l'objet <a href="#">Sdl</a>
-----	--

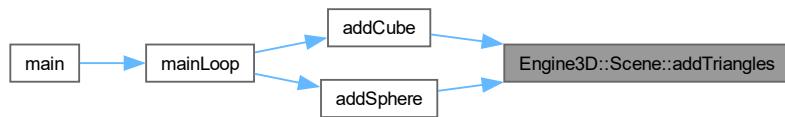
## 7.7.3 Documentation des fonctions membres

### 7.7.3.1 addTriangles()

```
void Engine3D::Scene::addTriangles (
    const std::vector< Triangle3d > & tris,
    uint8_t r,
    uint8_t g,
    uint8_t b)
```

Ajoute des triangles à la scène.

Voici le graphe des appels de cette fonction :



### 7.7.3.2 clearTriangles()

```
void Engine3D::Scene::clearTriangles ()
```

Efface les triangles de la scène.

Voici le graphe des appels de cette fonction :

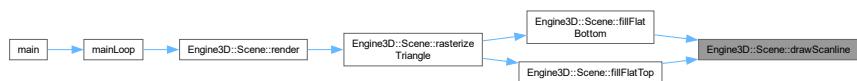


### 7.7.3.3 drawScanline()

```
void Engine3D::Scene::drawScanline (
    int y,
    int x1,
    int x2,
    float depth,
    uint8_t r,
    uint8_t g,
    uint8_t b) [private]
```

Dessine une ligne horizontale du triangle via technique scanline.

Voici le graphe des appelants de cette fonction :



### 7.7.3.4 fillFlatBottom()

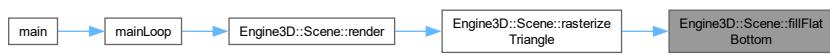
```
void Engine3D::Scene::fillFlatBottom (
    const Point2d & p1,
    const Point2d & p2,
    const Point2d & p3,
    float depth,
    uint8_t r,
    uint8_t g,
    uint8_t b) [private]
```

Rasterise la partie plate du triangle (haut ou bas).

Voici le graphe d'appel pour cette fonction :



Voici le graphe des appelants de cette fonction :



### 7.7.3.5 fillFlatTop()

```
void Engine3D::Scene::fillFlatTop (
    const Point2d & p1,
    const Point2d & p2,
    const Point2d & p3,
    float depth,
    uint8_t r,
    uint8_t g,
    uint8_t b) [private]
```

Rasterise la partie plate supérieure du triangle.

Voici le graphe d'appel pour cette fonction :



Voici le graphe des appels de cette fonction :

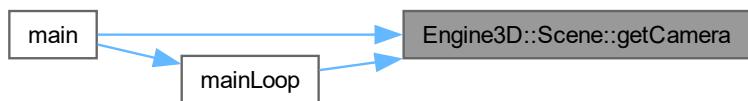


### 7.7.3.6 getCamera()

```
Camera & Engine3D::Scene::getCamera ()
```

Retourne une référence vers la caméra.

Voici le graphe des appels de cette fonction :

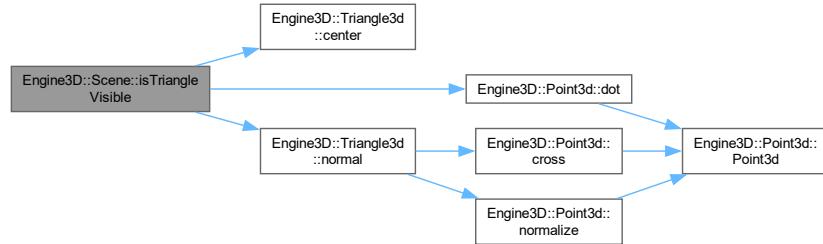


### 7.7.3.7 isTriangleVisible()

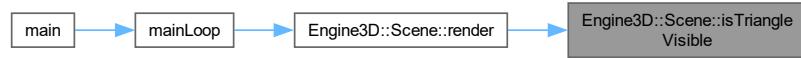
```
bool Engine3D::Scene::isTriangleVisible (
    const Triangle3d & triangle) const [private]
```

Vérifie si un triangle est visible.

Voici le graphe d'appel pour cette fonction :



Voici le graphe des appelants de cette fonction :

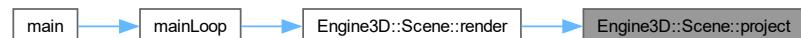


### 7.7.3.8 project()

```
Point2d Engine3D::Scene::project (
    const Point3d & p) const [private]
```

Projette un point 3D en point 2D.

Voici le graphe des appelants de cette fonction :

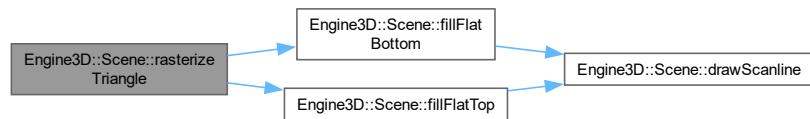


### 7.7.3.9 rasterizeTriangle()

```
void Engine3D::Scene::rasterizeTriangle (
    const Triangle2d & triangle,
    float avgDepth,
    uint8_t r,
    uint8_t g,
    uint8_t b) [private]
```

Rasterise un triangle 2D.

Voici le graphe d'appel pour cette fonction :



Voici le graphe des appels de cette fonction :

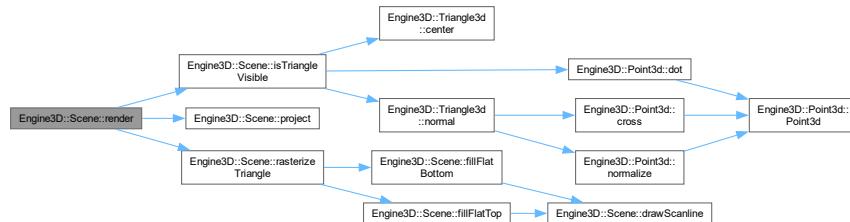


### 7.7.3.10 render()

```
void Engine3D::Scene::render ()
```

Effectue le rendu de la scène.

Voici le graphe d'appel pour cette fonction :



Voici le graphe des appelants de cette fonction :



## 7.7.4 Documentation des données membres

### 7.7.4.1 camera

```
Camera Engine3D::Scene::camera [private]
```

Caméra de la scène.

### 7.7.4.2 sdl

```
Sdl& Engine3D::Scene::sdl [private]
```

Référence vers SDL.

### 7.7.4.3 triangles

```
std::vector<ColoredTriangle> Engine3D::Scene::triangles [private]
```

Triangles colorés à render.

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

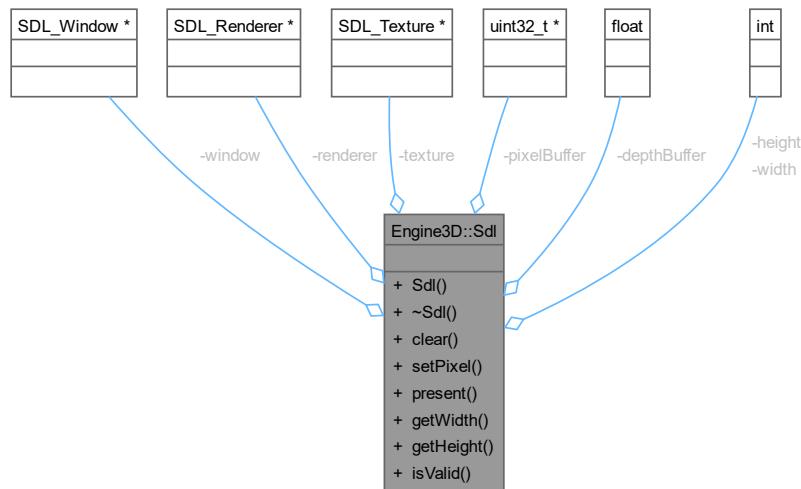
- [Scene.hpp](#)
- [Scene.cpp](#)

## 7.8 Référence de la classe Engine3D::Sdl

Classe contenant les fonctionnalités SDL pour le rendu graphique.

```
#include <Sdl.hpp>
```

Graphe de collaboration de Engine3D::Sdl:



### Fonctions membres publiques

- `Sdl` (const std::string &title, int w, int h)

*Constructeur de la classe Sdl.*

- `~Sdl` ()

*Destructeur de la classe Sdl.*

- void `clear` (uint8\_t r, uint8\_t g, uint8\_t b)

*Efface le tampon de pixels avec une couleur.*

- void `setPixel` (int x, int y, float depth, uint8\_t r, uint8\_t g, uint8\_t b)

*Dessine un pixel dans le tampon.*

- void `present` ()

*Présente le contenu du tampon à l'écran.*

- int `getWidth` () const

*Retourne la largeur de la fenêtre.*

- int `getHeight` () const

*Retourne la hauteur de la fenêtre.*

- bool **isValid** () const

*Vérifie si *SDL* a été initialisé correctement.*

### Attributs privés

- SDL\_Window \* **window**

*Pointeur vers la fenêtre *SDL*.*

- SDL\_Renderer \* **renderer**

*Pointeur vers le renderer *SDL*.*

- SDL\_Texture \* **texture**

*Texture pour le rendu pixel par pixel.*

- uint32\_t \* **pixelBuffer**

*Tampon de pixels.*

- float \* **depthBuffer**

*Tampon de profondeur.*

- int **width**

*Largeur de la fenêtre.*

- int **height**

*Hauteur de la fenêtre.*

### 7.8.1 Description détaillée

Classe contenant les fonctionnalités *SDL* pour le rendu graphique.

### 7.8.2 Documentation des constructeurs et destructeur

#### 7.8.2.1 Sdl()

```
Engine3D::Sdl::Sdl (
    const std::string & title,
    int w,
    int h)
```

Constructeur de la classe **Sdl**.

Constructeur initialisant *SDL*, la fenêtre et les tampons.

## Paramètres

<i>title</i>	Titre de la fenêtre
<i>w</i>	Largeur de la fenêtre
<i>h</i>	Hauteur de la fenêtre

### 7.8.2.2 ~Sdl()

```
Engine3D::Sdl::~Sdl ()
```

Destructeur de la classe [Sdl](#).

Destructeur libérant les ressources SDL.

## 7.8.3 Documentation des fonctions membres

### 7.8.3.1 clear()

```
void Engine3D::Sdl::clear (
    uint8_t r,
    uint8_t g,
    uint8_t b)
```

Efface le tampon de pixels avec une couleur.

Efface les tampons avec la couleur spécifiée.

## Paramètres

<i>r</i>	Composante rouge (0-255)
<i>g</i>	Composante verte (0-255)
<i>b</i>	Composante bleue (0-255)

### 7.8.3.2 getHeight()

```
int Engine3D::Sdl::getHeight () const
```

Retourne la hauteur de la fenêtre.

## Renvoie

Hauteur en pixels

### 7.8.3.3 getWidth()

```
int Engine3D::Sdl::getWidth () const
```

Retourne la largeur de la fenêtre.

Renvoie

Largeur en pixels

### 7.8.3.4 isValid()

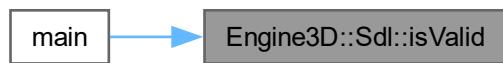
```
bool Engine3D::Sdl::isValid () const
```

Vérifie si SDL a été initialisé correctement.

Renvoie

true si l'initialisation est réussie

Voici le graphe des appels de cette fonction :



### 7.8.3.5 present()

```
void Engine3D::Sdl::present ()
```

Présente le contenu du tampon à l'écran.

Affiche le contenu du tampon à l'écran.

### 7.8.3.6 setPixel()

```
void Engine3D::Sdl::setPixel (
    int x,
    int y,
    float depth,
    uint8_t r,
    uint8_t g,
    uint8_t b)
```

Dessine un pixel dans le tampon.

Généré par Doxygen

Dessine un pixel avec profondeur.

## Paramètres

<i>x</i>	Coordonnée X
<i>y</i>	Coordonnée Y
<i>depth</i>	Profondeur du pixel
<i>r</i>	Composante rouge
<i>g</i>	Composante verte
<i>b</i>	Composante bleue

## 7.8.4 Documentation des données membres

### 7.8.4.1 depthBuffer

```
float* Engine3D::Sdl::depthBuffer [private]
```

Tampon de profondeur.

### 7.8.4.2 height

```
int Engine3D::Sdl::height [private]
```

Hauteur de la fenêtre.

### 7.8.4.3 pixelBuffer

```
uint32_t* Engine3D::Sdl::pixelBuffer [private]
```

Tampon de pixels.

### 7.8.4.4 renderer

```
SDL_Renderer* Engine3D::Sdl::renderer [private]
```

Pointeur vers le renderer SDL.

### 7.8.4.5 texture

```
SDL_Texture* Engine3D::Sdl::texture [private]
```

Texture pour le rendu pixel par pixel.

### 7.8.4.6 width

```
int Engine3D::Sdl::width [private]
```

Largeur de la fenêtre.

#### 7.8.4.7 window

```
SDL_Window* Engine3D::Sdl::window [private]
```

Pointeur vers la fenêtre SDL.

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

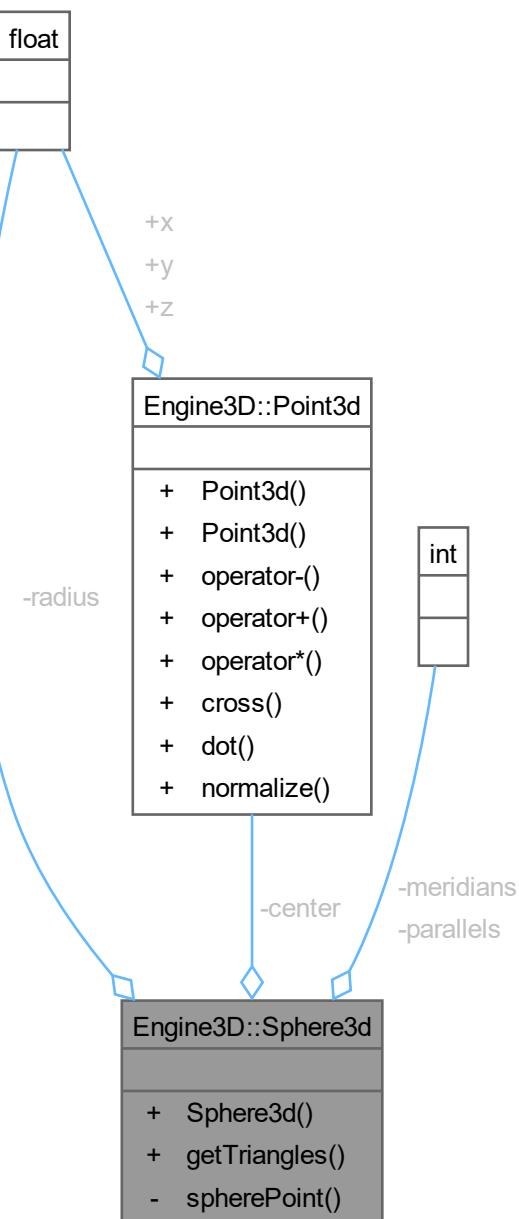
- [Sdl.hpp](#)
- [Sdl.cpp](#)

## 7.9 Référence de la classe Engine3D::Sphere3d

Sphère composée d'un maillage de [Quad3d](#).

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

Graphe de collaboration de Engine3D::Sphere3d:



## Fonctions membres publiques

— `Sphere3d (const Point3d &center, float radius, int meridians, int parallels)`

*Constructeur.*

— `std::vector< Triangle3d > getTriangles () const`

*Retourne tous les triangles de la sphère.*

### Fonctions membres privées

- [Point3d spherePoint](#) (float theta, float phi) const

*Calcule un point sur la sphère.*

### Attributs privés

- [Point3d center](#)

*Centre de la sphère.*

- float [radius](#)

*Rayon de la sphère.*

- int [meridians](#)

*Nombre de méridiens.*

- int [parallels](#)

*Nombre de parallèles.*

### 7.9.1 Description détaillée

Sphère composée d'un maillage de [Quad3d](#).

### 7.9.2 Documentation des constructeurs et destructeur

#### 7.9.2.1 Sphere3d()

```
Engine3D::Sphere3d::Sphere3d (
    const Point3d & center,
    float radius,
    int meridians,
    int parallels)
```

Constructeur.

#### Paramètres

<i>center</i>	
<i>radius</i>	
<i>meridians</i>	
<i>parallels</i>	

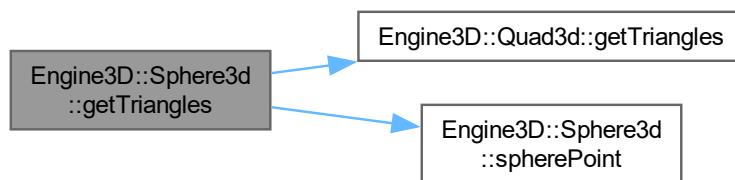
### 7.9.3 Documentation des fonctions membres

#### 7.9.3.1 getTriangles()

```
std::vector< Triangle3d > Engine3D::Sphere3d::getTriangles () const
```

Retourne tous les triangles de la sphère.

Voici le graphe d'appel pour cette fonction :



Voici le graphe des appels de cette fonction :



#### 7.9.3.2 spherePoint()

```
Point3d Engine3D::Sphere3d::spherePoint (
    float theta,
    float phi) const [private]
```

Calcule un point sur la sphère.

Voici le graphe des appels de cette fonction :



## 7.9.4 Documentation des données membres

### 7.9.4.1 center

```
Point3d Engine3D::Sphere3d::center [private]
```

Centre de la sphère.

### 7.9.4.2 meridians

```
int Engine3D::Sphere3d::meridians [private]
```

Nombre de méridiens.

### 7.9.4.3 parallels

```
int Engine3D::Sphere3d::parallels [private]
```

Nombre de parallèles.

### 7.9.4.4 radius

```
float Engine3D::Sphere3d::radius [private]
```

Rayon de la sphère.

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

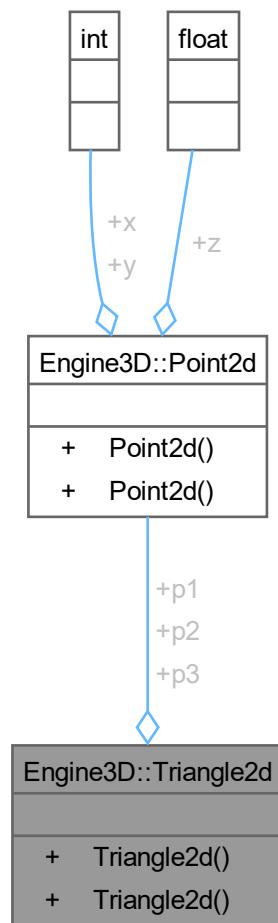
- [Geometry.hpp](#)
- [Geometry.cpp](#)

## 7.10 Référence de la classe Engine3D::Triangle2d

Triangle projeté sur l'écran 2D.

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

Graphe de collaboration de Engine3D::Triangle2d:



## Fonctions membres publiques

- [Triangle2d \(\)](#)

*Constructeur par défaut.*
- [Triangle2d \(const Point2d &p1, const Point2d &p2, const Point2d &p3\)](#)

*Constructeur avec trois points (sommets).*

## Attributs publics

- [Point2d p1](#)

*Premier sommet.*

- [Point2d p2](#)

*Deuxième sommet.*

- [Point2d p3](#)

*Troisième sommet.*

### 7.10.1 Description détaillée

Triangle projeté sur l'écran 2D.

### 7.10.2 Documentation des constructeurs et destructeur

#### 7.10.2.1 Triangle2d() [1/2]

```
Engine3D::Triangle2d::Triangle2d ()
```

Constructeur par défaut.

#### 7.10.2.2 Triangle2d() [2/2]

```
Engine3D::Triangle2d::Triangle2d (const Point2d & p1, const Point2d & p2, const Point2d & p3)
```

Constructeur avec trois points (sommets).

#### Paramètres

<i>p1</i>	
<i>p2</i>	
<i>p3</i>	

### 7.10.3 Documentation des données membres

#### 7.10.3.1 p1

```
Point2d Engine3D::Triangle2d::p1
```

Premier sommet.

#### 7.10.3.2 p2

```
Point2d Engine3D::Triangle2d::p2
```

Deuxième sommet.

### 7.10.3.3 p3

`Point2d Engine3D::Triangle2d::p3`

Troisième sommet.

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

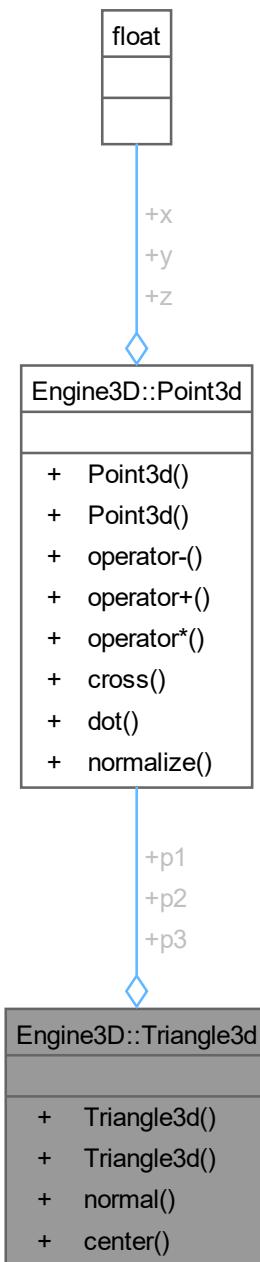
- [Geometry.hpp](#)
- [Geometry.cpp](#)

## 7.11 Référence de la classe Engine3D::Triangle3d

Triangle dans l'espace 3D.

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

Graphe de collaboration de Engine3D::Triangle3d:



## Fonctions membres publiques

- [Triangle3d \(\)](#)  
*Constructeur par défaut.*
- [Triangle3d \(const Point3d &p1, const Point3d &p2, const Point3d &p3\)](#)

*Constructeur avec trois points (sommets).*

- [Point3d normal \(\) const](#)

*Calcule la normale du triangle.*

- [Point3d center \(\) const](#)

*Calcule le centre du triangle.*

## Attributs publics

- [Point3d p1](#)

*Premier sommet.*

- [Point3d p2](#)

*Deuxième sommet.*

- [Point3d p3](#)

*Troisième sommet.*

### 7.11.1 Description détaillée

Triangle dans l'espace 3D.

### 7.11.2 Documentation des constructeurs et destructeur

#### 7.11.2.1 Triangle3d() [1/2]

```
Engine3D::Triangle3d::Triangle3d ()
```

Constructeur par défaut.

#### 7.11.2.2 Triangle3d() [2/2]

```
Engine3D::Triangle3d::Triangle3d (
    const Point3d & p1,
    const Point3d & p2,
    const Point3d & p3)
```

Constructeur avec trois points (sommets).

## Paramètres

<i>p1</i>	
<i>p2</i>	

| p3 |

### 7.11.3 Documentation des fonctions membres

#### 7.11.3.1 center()

```
Point3d Engine3D::Triangle3d::center () const
```

Calcule le centre du triangle.

Voici le graphe des appelants de cette fonction :

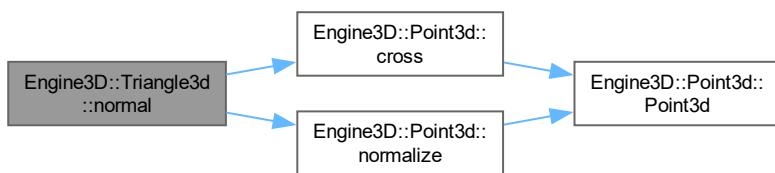


#### 7.11.3.2 normal()

```
Point3d Engine3D::Triangle3d::normal () const
```

Calcule la normale du triangle.

Voici le graphe d'appel pour cette fonction :



Voici le graphe des appelants de cette fonction :



## 7.11.4 Documentation des données membres

### 7.11.4.1 p1

`Point3d` Engine3D::Triangle3d::p1

Premier sommet.

### 7.11.4.2 p2

`Point3d` Engine3D::Triangle3d::p2

Deuxième sommet.

### 7.11.4.3 p3

`Point3d` Engine3D::Triangle3d::p3

Troisième sommet.

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

- [Geometry.hpp](#)
- [Geometry.cpp](#)

# Chapitre 8

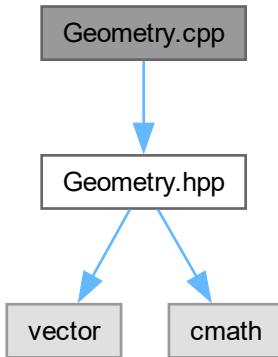
## Documentation des fichiers

### 8.1 Référence du fichier Geometry.cpp

Implémentation des classes géométriques.

```
#include "Geometry.hpp"
```

Graphe des dépendances par inclusion de Geometry.cpp:



#### Espaces de nommage

- namespace [Engine3D](#)

*Espace de noms contenant toutes les classes du moteur 3D.*

#### 8.1.1 Description détaillée

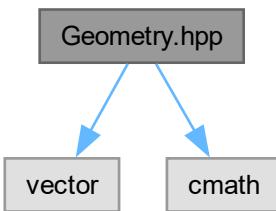
Implémentation des classes géométriques.

## 8.2 Référence du fichier Geometry.hpp

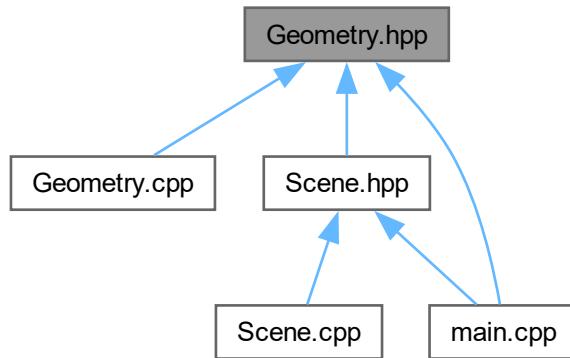
Définition des classes géométriques.

```
#include <vector>
#include <cmath>
```

Graphe des dépendances par inclusion de Geometry.hpp:



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



### Classes

- class [Engine3D::Point3d](#)

*Représente un point dans l'espace 3D.*

- class [Engine3D::Point2d](#)

*Représente un point sur l'écran 2D.*

- class Engine3D::Triangle3d

*Triangle dans l'espace 3D.*

- class Engine3D::Triangle2d

*Triangle projeté sur l'écran 2D.*

- class Engine3D::Quad3d

*Quadrilatère composé de deux triangles.*

- class Engine3D::Pave3d

*Pavé composé de 6 faces (Quad3d).*

- class Engine3D::Sphere3d

*Sphère composée d'un maillage de Quad3d.*

## Espaces de nommage

- namespace Engine3D

*Espace de noms contenant toutes les classes du moteur 3D.*

### 8.2.1 Description détaillée

Définition des classes géométriques.

## 8.3 Geometry.hpp

[Aller à la documentation de ce fichier.](#)

```
00001
00005
00011
00012 #ifndef GEOMETRY_HPP
00013 #define GEOMETRY_HPP
00014
00015 #include <vector>
00016 #include <cmath>
00017
00022 namespace Engine3D {
00023
00029 class Point3d {
00030 public:
00031     float x;
00032     float y;
00033     float z;
00034
00038     Point3d();
00039
00046     Point3d(float x, float y, float z);
00047
00051     Point3d operator-(const Point3d& other) const;
00052
00056     Point3d operator+(const Point3d& other) const;
00057
00061     Point3d operator*(float scalar) const;
00062
00066     Point3d cross(const Point3d& other) const;
00067
```

```
00071     float dot(const Point3d& other) const;
00072
00073     Point3d normalize() const;
00074 };
00075
00076 class Point2d {
00077 public:
00078     int x;
00079     int y;
00080     float z;
00081
00082     Point2d();
00083
00084     Point2d(int x, int y, float z);
00085 };
00086
00087 class Triangle3d {
00088 public:
00089     Point3d p1;
00090     Point3d p2;
00091     Point3d p3;
00092
00093     Triangle3d();
00094
00095     Triangle3d(const Point3d& p1, const Point3d& p2, const Point3d& p3);
00096
00097     Point3d normal() const;
00098
00099     Point3d center() const;
00100 };
00101
00102 class Triangle2d {
00103 public:
00104     Point2d p1;
00105     Point2d p2;
00106     Point2d p3;
00107
00108     Triangle2d();
00109
00110     Triangle2d(const Point2d& p1, const Point2d& p2, const Point2d& p3);
00111 };
00112
00113 class Quad3d {
00114 public:
00115     Triangle3d t1;
00116     Triangle3d t2;
00117
00118     Quad3d();
00119
00120     Quad3d(const Point3d& p1, const Point3d& p2,
00121             const Point3d& p3, const Point3d& p4);
00122
00123     std::vector<Triangle3d> getTriangles() const;
00124 };
00125
00126 class Pave3d {
00127 private:
00128     Point3d center;
00129     float sizeX;
00130     float sizeY;
00131     float sizeZ;
00132     float angleX;
00133     float angleY;
00134     float angleZ;
00135
00136 public:
00137     Pave3d(const Point3d& center, float sizeX, float sizeY, float sizeZ);
00138
00139     void setRotation(float angleX, float angleY, float angleZ);
00140
00141     std::vector<Triangle3d> getTriangles() const;
00142 };
00143
00144 class Sphere3d {
00145 private:
00146     Point3d center;
00147     float radius;
00148     int meridians;
00149     int parallels;
00150
00151 public:
00152     Sphere3d(const Point3d& center, float radius, int meridians, int parallels);
00153
00154     std::vector<Triangle3d> getTriangles() const;
```

```

00263
00264     private:
00265         Point3d spherePoint(float theta, float phi) const;
00266     };
00267 } // namespace Engine3D
00268 // end of Engine3DClasses group
00269
00270
00271 #endif

```

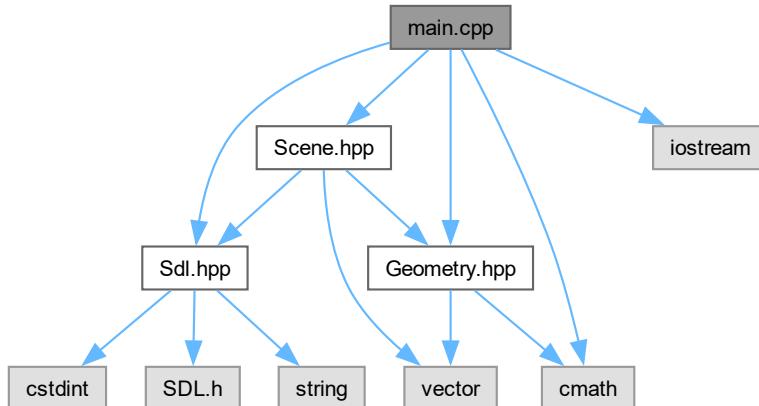
## 8.4 Référence du fichier main.cpp

Boucle principale et gestion des événements clavier.

```

#include "Sdl.hpp"
#include "Geometry.hpp"
#include "Scene.hpp"
#include <cmath>
#include <iostream>
Graphe des dépendances par inclusion de main.cpp:

```



### Fonctions

- void [handleKeyboard](#) ([Camera](#) &camera, const Uint8 \*keys)

*Gère les événements clavier.*

- void [addCube](#) ([Scene](#) &scene, float angle)

*Crée et ajoute le cube à la scène.*

- void [addSphere](#) ([Scene](#) &scene)

*Crée et ajoute la sphère à la scène.*

- int [mainLoop](#) ([Sdl](#) &sdl, [Scene](#) &scene)

*Boucle principale du programme.*

- int `main` (int argc, char \*argv[ ])

*main*

## Variables

- const int `WINDOW_WIDTH` = 1000

*Largeur de la fenêtre en pixels.*

- const int `WINDOW_HEIGHT` = 650

*Hauteur de la fenêtre en pixels.*

- const float `CAMERA_SPEED` = 0.15f

*Vitesse de déplacement de la caméra.*

- const float `ROTATION_SPEED` = 0.02f

*Vitesse de rotation du cube.*

### 8.4.1 Description détaillée

Boucle principale et gestion des événements clavier.

### 8.4.2 Documentation des fonctions

#### 8.4.2.1 addCube()

```
void addCube (
    Scene & scene,
    float angle)
```

Crée et ajoute le cube à la scène.

#### Paramètres

<code>scene</code>	
<code>angle</code>	Angle de rotation actuel

Voici le graphe d'appel pour cette fonction :



Voici le graphe des appelants de cette fonction :



#### 8.4.2.2 addSphere()

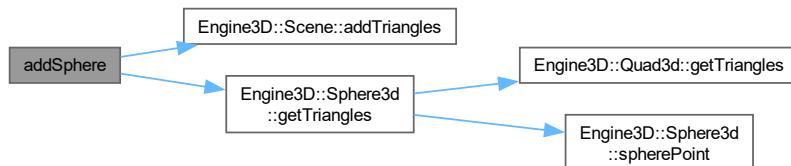
```
void addSphere (
    Scene & scene)
```

Crée et ajoute la sphère à la scène.

##### Paramètres

<code>scene</code>	instance actuelle de <code>Scene</code>
--------------------	---

Voici le graphe d'appel pour cette fonction :



Voici le graphe des appelants de cette fonction :



#### 8.4.2.3 handleKeyboard()

---

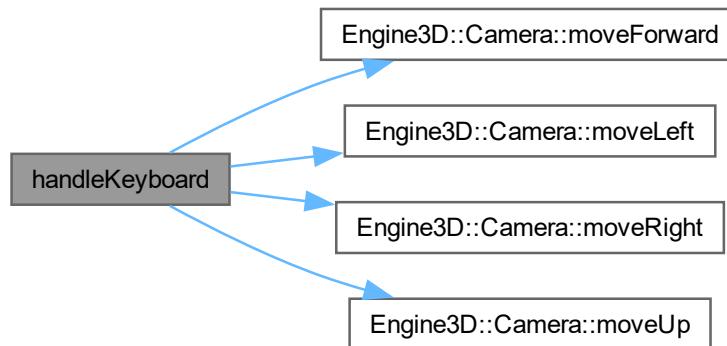
Généré par Doxygen

```
void handleKeyboard (
    Camera & camera,
    const Uint8 * keys)
```

**Paramètres**

<i>camera</i>	instance actuelle de <a href="#">Camera</a>
<i>keys</i>	État des touches (géré par SDL)

Voici le graphe d'appel pour cette fonction :



Voici le graphe des appelants de cette fonction :

**8.4.2.4 main()**

```

int main (
    int argc,
    char * argv[])
}

main
  
```

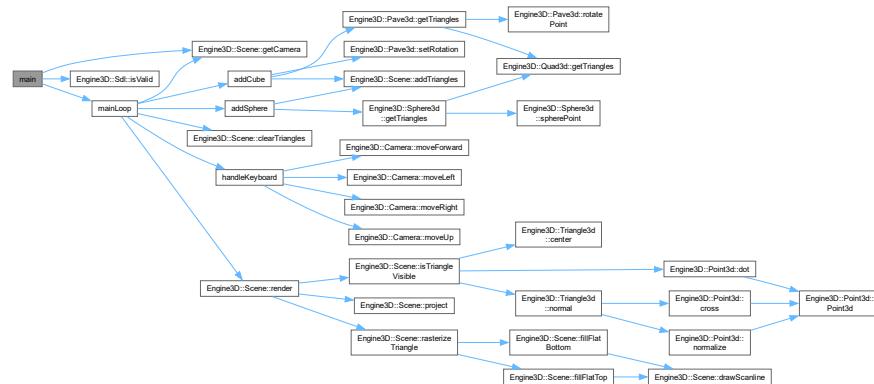
**Paramètres**

<i>argc</i>	
<i>argv</i>	

Renvoie

le code de sortie (0 si succès)

Voici le graphe d'appel pour cette fonction :



#### 8.4.2.5 mainLoop()

```
int mainLoop (
    Sdl & sdl,
    Scene & scene)
```

Boucle principale du programme.

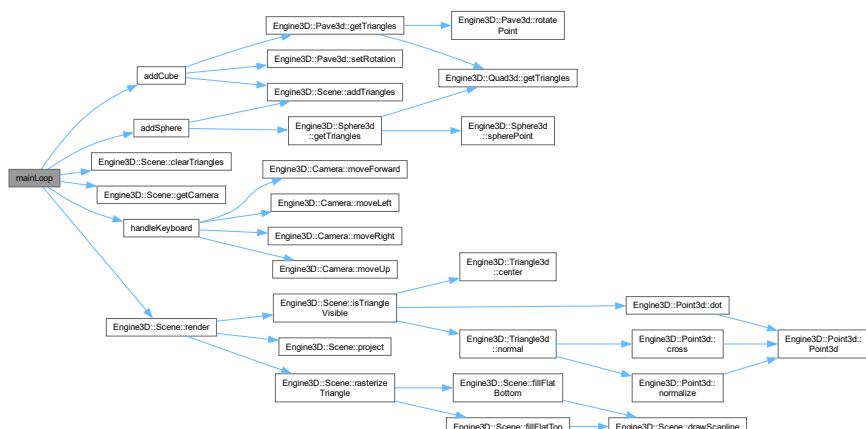
#### Paramètres

sdl	instance actuelle de <a href="#">Sdl</a>
scene	instance actuelle de <a href="#">Scene</a>

Renvoie

le code de sortie (0 si succès)

Voici le graphe d'appel pour cette fonction :



Voici le graphe des appelants de cette fonction :



### 8.4.3 Documentation des variables

#### 8.4.3.1 CAMERA\_SPEED

```
const float CAMERA_SPEED = 0.15f
```

Vitesse de déplacement de la caméra.

#### 8.4.3.2 ROTATION\_SPEED

```
const float ROTATION_SPEED = 0.02f
```

Vitesse de rotation du cube.

#### 8.4.3.3 WINDOW\_HEIGHT

```
const int WINDOW_HEIGHT = 650
```

Hauteur de la fenêtre en pixels.

#### 8.4.3.4 WINDOW\_WIDTH

```
const int WINDOW_WIDTH = 1000
```

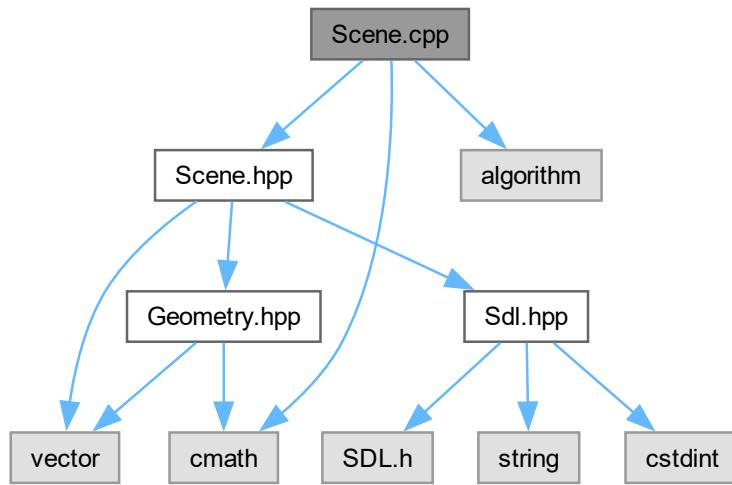
Largeur de la fenêtre en pixels.

## 8.5 Référence du fichier Scene.cpp

Implémentation de la gestion de scène et rasterisation.

```
#include "Scene.hpp"
#include <algorithm>
#include <cmath>
```

Graphe des dépendances par inclusion de Scene.cpp:



### Espaces de nommage

- namespace Engine3D

*Espace de noms contenant toutes les classes du moteur 3D.*

#### 8.5.1 Description détaillée

Implémentation de la gestion de scène et rasterisation.

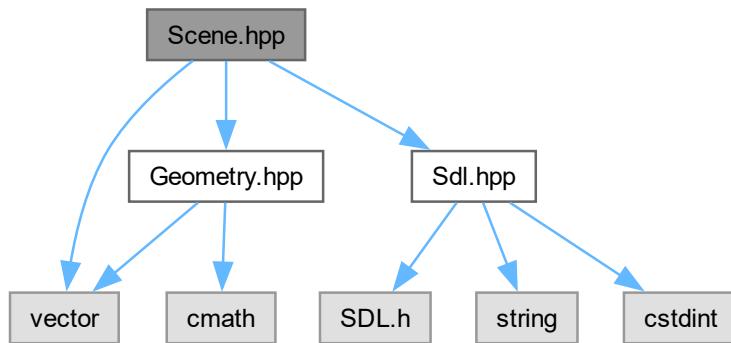
## 8.6 Référence du fichier Scene.hpp

Gestion de la scène, caméra et rasterisation.

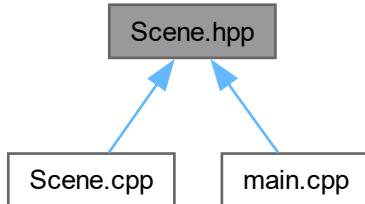
```
#include "Geometry.hpp"
#include "Sdl.hpp"
```

```
#include <vector>
```

Graphe des dépendances par inclusion de Scene.hpp:



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



## Classes

- class [Engine3D::Camera](#)

*Représente la caméra (œil) dans la scène.*

- struct [Engine3D::ColoredTriangle](#)

*Triangle avec sa couleur associée.*

- class [Engine3D::Scene](#)

*Gère le rendu de la scène 3D.*

## Espaces de nommage

- namespace [Engine3D](#)

Espace de noms contenant toutes les classes du moteur 3D.

### 8.6.1 Description détaillée

Gestion de la scène, caméra et rasterisation.

## 8.7 Scene.hpp

[Aller à la documentation de ce fichier.](#)

```
00001
00005
00006 #ifndef SCENE_HPP
00007 #define SCENE_HPP
00008
00009 #include "Geometry.hpp"
00010 #include "Sdl.hpp"
00011 #include <vector>
00012
00013 namespace Engine3D {
00014
00020 class Camera {
00021 public:
00022     Point3d position;
00023     Point3d direction;
00024     Point3d up;
00025     float fov;
00026     float nearPlane;
00027
00031     Camera();
00032
00036     void moveForward(float delta);
00037
00041     void moveRight(float delta);
00042
00046     void moveLeft(float delta);
00047
00051     void moveUp(float delta);
00052 };
00053
00059 struct ColoredTriangle {
00060     Triangle3d triangle;
00061     uint8_t r;
00062     uint8_t g;
00063     uint8_t b;
00064 };
00065
00071 class Scene {
00072 private:
00073     Sdl& sdl;
00074     Camera camera;
00075     std::vector<ColoredTriangle> triangles;
00076
00077 public:
00082     Scene(Sdl& sdl);
00083
00087     void addTriangles(const std::vector<Triangle3d>& tris,
00088                         uint8_t r, uint8_t g, uint8_t b);
00089
00093     void clearTriangles();
00094
00098     void render();
00099
00103     Camera& getCamera();
00104
00105 private:
00109     Point2d project(const Point3d& p) const;
00110
00114     bool isTriangleVisible(const Triangle3d& triangle) const;
00115
00119     void rasterizeTriangle(const Triangle2d& triangle, float avgDepth,
00120                           uint8_t r, uint8_t g, uint8_t b);
00121
00125     void drawScanline(int y, int x1, int x2, float depth,
00126                       uint8_t r, uint8_t g, uint8_t b);
00127
00131     void fillFlatBottom(const Point2d& p1, const Point2d& p2,
```

```

00132             const Point2d& p3, float depth,
00133                 uint8_t r, uint8_t g, uint8_t b);
00134
00138     void fillFlatTop(const Point2d& p1, const Point2d& p2,
00139                     const Point2d& p3, float depth,
00140                     uint8_t r, uint8_t g, uint8_t b);
00141 }
00142
00143 } // namespace Engine3D
00144
00145 #endif

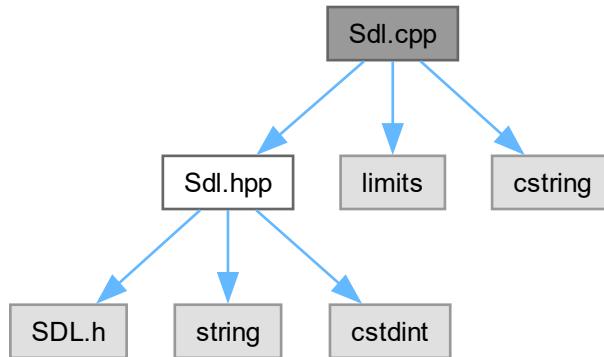
```

## 8.8 Référence du fichier Sdl.cpp

Implémentation de la classe Sdl.

```
#include "Sdl.hpp"
#include <limits>
#include <cstring>
```

Graphe des dépendances par inclusion de Sdl.cpp:



### Espaces de nommage

— namespace Engine3D

*Espace de noms contenant toutes les classes du moteur 3D.*

#### 8.8.1 Description détaillée

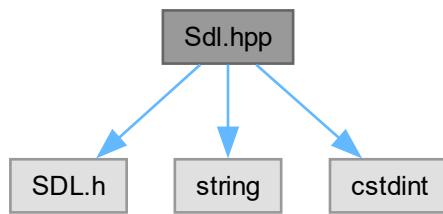
Implémentation de la classe Sdl.

## 8.9 Référence du fichier Sdl.hpp

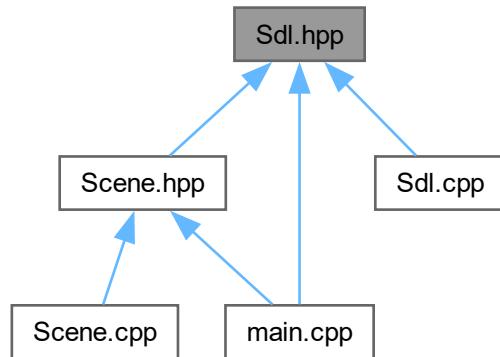
Encapsulation de la bibliothèque SDL2.

```
#include <SDL.h>
#include <string>
#include <cstdint>
```

Graphe des dépendances par inclusion de Sdl.hpp:



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



### Classes

— class [Engine3D::Sdl](#)

*Classe contenant les fonctionnalités SDL pour le rendu graphique.*

## Espaces de nommage

### — namespace Engine3D

*Espace de noms contenant toutes les classes du moteur 3D.*

### 8.9.1 Description détaillée

Encapsulation de la bibliothèque SDL2.

## 8.10 Sdl.hpp

[Aller à la documentation de ce fichier.](#)

```
00001
00005
00006 #ifndef SDL_HPP
00007 #define SDL_HPP
00008
00009 #include <SDL.h>
00010 #include <string>
00011 #include <cstdint>
00012
00017 namespace Engine3D {
00018
00024 class Sdl {
00025 private:
00026     SDL_Window* window;
00027     SDL_Renderer* renderer;
00028     SDL_Texture* texture;
00029     uint32_t* pixelBuffer;
00030     float* depthBuffer;
00031     int width;
00032     int height;
00033
00034 public:
00041     Sdl(const std::string& title, int w, int h);
00042
00046     ~Sdl();
00047
00054     void clear(uint8_t r, uint8_t g, uint8_t b);
00055
00065     void setPixel(int x, int y, float depth, uint8_t r, uint8_t g, uint8_t b);
00066
00070     void present();
00071
00076     int getWidth() const;
00077
00082     int getHeight() const;
00083
00088     bool isValid() const;
00089 };
00090
00091 } // namespace Engine3D
00092
00093 #endif
```

# Index

~Sdl  
    Engine3D::Sdl, 48

addCube  
    main.cpp, 68

addSphere  
    main.cpp, 69

addTriangles  
    Engine3D::Scene, 40

angleX  
    Engine3D::Pave3d, 24

angleY  
    Engine3D::Pave3d, 24

angleZ  
    Engine3D::Pave3d, 24

b  
    Engine3D::ColoredTriangle, 20

Camera  
    Engine3D::Camera, 15

camera  
    Engine3D::Scene, 45

CAMERA\_SPEED  
    main.cpp, 72

center  
    Engine3D::Pave3d, 25  
    Engine3D::Sphere3d, 55  
    Engine3D::Triangle3d, 61

clear  
    Engine3D::Sdl, 48

clearTriangles  
    Engine3D::Scene, 40

cross  
    Engine3D::Point3d, 30

depthBuffer  
    Engine3D::Sdl, 50

direction  
    Engine3D::Camera, 17

dot  
    Engine3D::Point3d, 31

drawScanline  
    Engine3D::Scene, 40

Engine3D, 11

Engine3D::Camera, 13  
    Camera, 15  
    direction, 17  
    fov, 17  
    moveForward, 16

    moveLeft, 16  
    moveRight, 16  
    moveUp, 16  
    nearPlane, 17  
    position, 17  
    up, 17

    Engine3D::ColoredTriangle, 18  
        b, 20  
        g, 20  
        r, 20  
        triangle, 20

    Engine3D::Pave3d, 21  
        angleX, 24  
        angleY, 24  
        angleZ, 24  
        center, 25  
        getTriangles, 23  
        Pave3d, 23  
        rotatePoint, 23  
        setRotation, 24  
        sizeX, 25  
        sizeY, 25  
        sizeZ, 25

    Engine3D::Point2d, 26  
        Point2d, 27  
            x, 27  
            y, 27  
            z, 27

    Engine3D::Point3d, 28  
        cross, 30  
        dot, 31  
        normalize, 31  
        operator+, 32  
        operator-, 33  
        operator\*, 32  
        Point3d, 30  
            x, 33  
            y, 33  
            z, 34

    Engine3D::Quad3d, 34  
        getTriangles, 37  
        Quad3d, 36  
            t1, 37  
            t2, 37

    Engine3D::Scene, 38  
        addTriangles, 40  
        camera, 45  
        clearTriangles, 40  
        drawScanline, 40

fillFlatBottom, 41  
 fillFlatTop, 41  
 getCamera, 42  
 isTriangleVisible, 42  
 project, 43  
 rasterizeTriangle, 43  
 render, 44  
 Scene, 40  
 sdl, 45  
 triangles, 45  
**Engine3D::Sdl**, 46  
 ~Sdl, 48  
 clear, 48  
 depthBuffer, 50  
 getHeight, 48  
 getWidth, 48  
 height, 50  
 isValid, 49  
 pixelBuffer, 50  
 present, 49  
 renderer, 50  
 Sdl, 47  
 setPixel, 49  
 texture, 50  
 width, 50  
 window, 50  
**Engine3D::Sphere3d**, 51  
 center, 55  
 getTriangles, 54  
 meridians, 55  
 parallels, 55  
 radius, 55  
 Sphere3d, 53  
 spherePoint, 54  
**Engine3D::Triangle2d**, 55  
 p1, 57  
 p2, 57  
 p3, 57  
 Triangle2d, 57  
**Engine3D::Triangle3d**, 58  
 center, 61  
 normal, 61  
 p1, 62  
 p2, 62  
 p3, 62  
 Triangle3d, 60  
  
 fillFlatBottom  
     Engine3D::Scene, 41  
 fillFlatTop  
     Engine3D::Scene, 41  
 fov  
     Engine3D::Camera, 17  
  
 g  
     Engine3D::ColoredTriangle, 20  
**Geometry.cpp**, 63  
**Geometry.hpp**, 64  
 getCamera

Engine3D::Scene, 42  
 getHeight  
     Engine3D::Sdl, 48  
 getTriangles  
     Engine3D::Pave3d, 23  
     Engine3D::Quad3d, 37  
     Engine3D::Sphere3d, 54  
 getWidth  
     Engine3D::Sdl, 48  
  
 handleKeyboard  
     main.cpp, 69  
 height  
     Engine3D::Sdl, 50  
  
 isTriangleVisible  
     Engine3D::Scene, 42  
 isValid  
     Engine3D::Sdl, 49  
  
 main  
     main.cpp, 70  
 main.cpp, 67  
     addCube, 68  
     addSphere, 69  
     CAMERA\_SPEED, 72  
     handleKeyboard, 69  
     main, 70  
     mainLoop, 71  
     ROTATION\_SPEED, 72  
     WINDOW\_HEIGHT, 72  
     WINDOW\_WIDTH, 72  
 mainLoop  
     main.cpp, 71  
 meridians  
     Engine3D::Sphere3d, 55  
 Moteur 3D, 9  
 moveForward  
     Engine3D::Camera, 16  
 moveLeft  
     Engine3D::Camera, 16  
 moveRight  
     Engine3D::Camera, 16  
 moveUp  
     Engine3D::Camera, 16  
  
 nearPlane  
     Engine3D::Camera, 17  
 normal  
     Engine3D::Triangle3d, 61  
 normalize  
     Engine3D::Point3d, 31  
  
 operator+  
     Engine3D::Point3d, 32  
 operator-  
     Engine3D::Point3d, 33  
 operator\*  
     Engine3D::Point3d, 32

p1  
    Engine3D::Triangle2d, 57  
    Engine3D::Triangle3d, 62

p2  
    Engine3D::Triangle2d, 57  
    Engine3D::Triangle3d, 62

p3  
    Engine3D::Triangle2d, 57  
    Engine3D::Triangle3d, 62

parallels  
    Engine3D::Sphere3d, 55

Pave3d  
    Engine3D::Pave3d, 23

pixelBuffer  
    Engine3D::Sdl, 50

Point2d  
    Engine3D::Point2d, 27

Point3d  
    Engine3D::Point3d, 30

position  
    Engine3D::Camera, 17

present  
    Engine3D::Sdl, 49

project  
    Engine3D::Scene, 43

Quad3d  
    Engine3D::Quad3d, 36

r  
    Engine3D::ColoredTriangle, 20

radius  
    Engine3D::Sphere3d, 55

rasterizeTriangle  
    Engine3D::Scene, 43

render  
    Engine3D::Scene, 44

renderer  
    Engine3D::Sdl, 50

rotatePoint  
    Engine3D::Pave3d, 23

ROTATION\_SPEED  
    main.cpp, 72

Scene  
    Engine3D::Scene, 40

Scene.cpp, 73

Scene.hpp, 73

Sdl  
    Engine3D::Sdl, 47

sdl  
    Engine3D::Scene, 45

Sdl.cpp, 76

Sdl.hpp, 77

setPixel  
    Engine3D::Sdl, 49

setRotation  
    Engine3D::Pave3d, 24

sizeX  
    Engine3D::Pave3d, 25

sizeY  
    Engine3D::Pave3d, 25

sizeZ  
    Engine3D::Pave3d, 25

Sphere3d  
    Engine3D::Sphere3d, 53

spherePoint  
    Engine3D::Sphere3d, 54

t1  
    Engine3D::Quad3d, 37

t2  
    Engine3D::Quad3d, 37

texture  
    Engine3D::Sdl, 50

triangle  
    Engine3D::ColoredTriangle, 20

Triangle2d  
    Engine3D::Triangle2d, 57

Triangle3d  
    Engine3D::Triangle3d, 60

triangles  
    Engine3D::Scene, 45

up  
    Engine3D::Camera, 17

width  
    Engine3D::Sdl, 50

window  
    Engine3D::Sdl, 50

WINDOW\_HEIGHT  
    main.cpp, 72

WINDOW\_WIDTH  
    main.cpp, 72

x  
    Engine3D::Point2d, 27  
    Engine3D::Point3d, 33

y  
    Engine3D::Point2d, 27  
    Engine3D::Point3d, 33

z  
    Engine3D::Point2d, 27  
    Engine3D::Point3d, 34