

Moteur 3D - PRAP

Generated by Doxygen 1.15.0



<b>1 Class Index</b>	<b>1</b>
1.1 Class List	1
<b>2 File Index</b>	<b>3</b>
2.1 File List	3
<b>3 Class Documentation</b>	<b>5</b>
3.1 Camera Class Reference	5
3.1.1 Detailed Description	6
3.1.2 Constructor & Destructor Documentation	6
3.1.2.1 Camera()	6
3.1.3 Member Function Documentation	6
3.1.3.1 moveForward()	6
3.1.3.2 moveLeft()	6
3.1.3.3 moveRight()	6
3.1.3.4 moveUp()	6
3.1.4 Member Data Documentation	7
3.1.4.1 direction	7
3.1.4.2 fov	7
3.1.4.3 nearPlane	7
3.1.4.4 position	7
3.1.4.5 up	7
3.2 ColoredTriangle Struct Reference	7
3.2.1 Detailed Description	8
3.2.2 Member Data Documentation	8
3.2.2.1 b	8
3.2.2.2 g	8
3.2.2.3 r	8
3.2.2.4 triangle	8
3.3 Pave3d Class Reference	9
3.3.1 Detailed Description	9
3.3.2 Constructor & Destructor Documentation	9
3.3.2.1 Pave3d()	9
3.3.3 Member Function Documentation	10
3.3.3.1 getTriangles()	10
3.3.3.2 rotatePoint()	10
3.3.3.3 setRotation()	10
3.3.4 Member Data Documentation	10
3.3.4.1 angleX	10
3.3.4.2 angleY	10
3.3.4.3 angleZ	11
3.3.4.4 center	11
3.3.4.5 sizeX	11

3.3.4.6 sizeY	11
3.3.4.7 sizeZ	11
3.4 Point2d Class Reference	11
3.4.1 Detailed Description	12
3.4.2 Constructor & Destructor Documentation	12
3.4.2.1 Point2d() [1/2]	12
3.4.2.2 Point2d() [2/2]	12
3.4.3 Member Data Documentation	12
3.4.3.1 x	12
3.4.3.2 y	13
3.4.3.3 z	13
3.5 Point3d Class Reference	13
3.5.1 Detailed Description	14
3.5.2 Constructor & Destructor Documentation	14
3.5.2.1 Point3d() [1/2]	14
3.5.2.2 Point3d() [2/2]	14
3.5.3 Member Function Documentation	14
3.5.3.1 cross()	14
3.5.3.2 dot()	14
3.5.3.3 normalize()	15
3.5.3.4 operator*()	15
3.5.3.5 operator+()	15
3.5.3.6 operator-()	15
3.5.4 Member Data Documentation	15
3.5.4.1 x	15
3.5.4.2 y	15
3.5.4.3 z	15
3.6 Quad3d Class Reference	16
3.6.1 Detailed Description	16
3.6.2 Constructor & Destructor Documentation	16
3.6.2.1 Quad3d() [1/2]	16
3.6.2.2 Quad3d() [2/2]	16
3.6.3 Member Function Documentation	17
3.6.3.1 getTriangles()	17
3.6.4 Member Data Documentation	17
3.6.4.1 t1	17
3.6.4.2 t2	17
3.7 Scene Class Reference	17
3.7.1 Detailed Description	18
3.7.2 Constructor & Destructor Documentation	18
3.7.2.1 Scene()	18
3.7.3 Member Function Documentation	19

3.7.3.1 addTriangles()	19
3.7.3.2 clearTriangles()	19
3.7.3.3 drawScanline()	19
3.7.3.4 fillFlatBottom()	19
3.7.3.5 fillFlatTop()	20
3.7.3.6 getCamera()	20
3.7.3.7 isTriangleVisible()	20
3.7.3.8 project()	20
3.7.3.9 rasterizeTriangle()	20
3.7.3.10 render()	20
3.7.4 Member Data Documentation	21
3.7.4.1 camera	21
3.7.4.2 sdl	21
3.7.4.3 triangles	21
3.8 Sdl Class Reference	21
3.8.1 Detailed Description	22
3.8.2 Constructor & Destructor Documentation	22
3.8.2.1 Sdl()	22
3.8.2.2 ~Sdl()	22
3.8.3 Member Function Documentation	23
3.8.3.1 clear()	23
3.8.3.2 getHeight()	23
3.8.3.3 getWidth()	23
3.8.3.4 isValid()	23
3.8.3.5 present()	24
3.8.3.6 setPixel()	24
3.8.4 Member Data Documentation	24
3.8.4.1 depthBuffer	24
3.8.4.2 height	24
3.8.4.3 pixelBuffer	25
3.8.4.4 renderer	25
3.8.4.5 texture	25
3.8.4.6 width	25
3.8.4.7 window	25
3.9 Sphere3d Class Reference	25
3.9.1 Detailed Description	26
3.9.2 Constructor & Destructor Documentation	26
3.9.2.1 Sphere3d()	26
3.9.3 Member Function Documentation	26
3.9.3.1 getTriangles()	26
3.9.3.2 spherePoint()	27
3.9.4 Member Data Documentation	27

3.9.4.1 center	27
3.9.4.2 meridians	27
3.9.4.3 parallels	27
3.9.4.4 radius	27
3.10 Triangle2d Class Reference	27
3.10.1 Detailed Description	28
3.10.2 Constructor & Destructor Documentation	28
3.10.2.1 Triangle2d() [1/2]	28
3.10.2.2 Triangle2d() [2/2]	28
3.10.3 Member Data Documentation	29
3.10.3.1 p1	29
3.10.3.2 p2	29
3.10.3.3 p3	29
3.11 Triangle3d Class Reference	29
3.11.1 Detailed Description	30
3.11.2 Constructor & Destructor Documentation	30
3.11.2.1 Triangle3d() [1/2]	30
3.11.2.2 Triangle3d() [2/2]	30
3.11.3 Member Function Documentation	30
3.11.3.1 center()	30
3.11.3.2 normal()	31
3.11.4 Member Data Documentation	31
3.11.4.1 p1	31
3.11.4.2 p2	31
3.11.4.3 p3	31
<b>4 File Documentation</b>	<b>33</b>
4.1 Geometry.cpp File Reference	33
4.1.1 Detailed Description	33
4.2 Geometry.hpp File Reference	33
4.2.1 Detailed Description	34
4.3 Geometry.hpp	34
4.4 main.cpp File Reference	35
4.4.1 Detailed Description	36
4.4.2 Function Documentation	36
4.4.2.1 addCube()	36
4.4.2.2 addSphere()	36
4.4.2.3 handleKeyboard()	36
4.4.2.4 main()	37
4.4.2.5 mainLoop()	37
4.4.3 Variable Documentation	37
4.4.3.1 CAMERA_SPEED	37

---

4.4.3.2 ROTATION_SPEED . . . . .	38
4.4.3.3 WINDOW_HEIGHT . . . . .	38
4.4.3.4 WINDOW_WIDTH . . . . .	38
4.5 Scene.cpp File Reference . . . . .	38
4.5.1 Detailed Description . . . . .	38
4.6 Scene.hpp File Reference . . . . .	38
4.6.1 Detailed Description . . . . .	39
4.7 Scene.hpp . . . . .	39
4.8 Sdl.cpp File Reference . . . . .	40
4.8.1 Detailed Description . . . . .	40
4.9 Sdl.hpp File Reference . . . . .	40
4.9.1 Detailed Description . . . . .	40
4.10 Sdl.hpp . . . . .	40
<b>Index</b>	<b>43</b>





# Chapter 1

## Class Index

### 1.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

<a href="#">Camera</a>	Représente la caméra (œil) dans la scène . . . . .	5
<a href="#">ColoredTriangle</a>	Triangle avec sa couleur associée . . . . .	7
<a href="#">Pave3d</a>	Pavé composé de 6 faces ( <a href="#">Quad3d</a> ) . . . . .	9
<a href="#">Point2d</a>	Représente un point sur l'écran 2D . . . . .	11
<a href="#">Point3d</a>	Représente un point dans l'espace 3D . . . . .	13
<a href="#">Quad3d</a>	Quadrilatère composé de deux triangles . . . . .	16
<a href="#">Scene</a>	Gère le rendu de la scène 3D . . . . .	17
<a href="#">Sdl</a>	Classe contenant les fonctionnalités SDL pour le rendu graphique . . . . .	21
<a href="#">Sphere3d</a>	Sphère composée d'un maillage de <a href="#">Quad3d</a> . . . . .	25
<a href="#">Triangle2d</a>	Triangle projeté sur l'écran 2D . . . . .	27
<a href="#">Triangle3d</a>	Triangle dans l'espace 3D . . . . .	29



## Chapter 2

# File Index

### 2.1 File List

Here is a list of all files with brief descriptions:

<a href="#">Geometry.cpp</a>	Implémentation des classes géométriques . . . . .	33
<a href="#">Geometry.hpp</a>	Définition des classes géométriques . . . . .	33
<a href="#">main.cpp</a>	Boucle principale et gestion des événements clavier . . . . .	35
<a href="#">Scene.cpp</a>	Implémentation de la gestion de scène et rasterisation . . . . .	38
<a href="#">Scene.hpp</a>	Gestion de la scène, caméra et rasterisation . . . . .	38
<a href="#">Sdl.cpp</a>	Implémentation de la classe <a href="#">Sdl</a> . . . . .	40
<a href="#">Sdl.hpp</a>	Encapsulation de la bibliothèque SDL2 . . . . .	40



## Chapter 3

# Class Documentation

### 3.1 Camera Class Reference

Représente la caméra (œil) dans la scène.

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

#### Public Member Functions

- [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.*

#### Public Attributes

- [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.*

### 3.1.1 Detailed Description

Représente la caméra (œil) dans la scène.

### 3.1.2 Constructor & Destructor Documentation

#### 3.1.2.1 Camera()

```
Camera::Camera ()
```

Constructeur par défaut.

### 3.1.3 Member Function Documentation

#### 3.1.3.1 moveForward()

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

Déplace la caméra vers l'avant/arrière.

#### 3.1.3.2 moveLeft()

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

Déplace la caméra vers la gauche.

#### 3.1.3.3 moveRight()

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

Déplace la caméra vers la droite.

#### 3.1.3.4 moveUp()

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

Déplace la caméra vers le haut/bas.

### 3.1.4 Member Data Documentation

#### 3.1.4.1 direction

`Point3d` Camera::direction

Direction de l'oeil.

#### 3.1.4.2 fov

`float` Camera::fov

Champ de vision (field of view).

#### 3.1.4.3 nearPlane

`float` Camera::nearPlane

Plan de projection proche.

#### 3.1.4.4 position

`Point3d` Camera::position

Position de l'oeil.

#### 3.1.4.5 up

`Point3d` Camera::up

Vecteur haut.

The documentation for this class was generated from the following files:

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

## 3.2 ColoredTriangle Struct Reference

Triangle avec sa couleur associée.

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

## Public Attributes

- [Triangle3d triangle](#)  
*Le triangle 3D.*
- `uint8_t r`  
*Composante rouge.*
- `uint8_t g`  
*Composante verte.*
- `uint8_t b`  
*Composante bleue.*

## 3.2.1 Detailed Description

Triangle avec sa couleur associée.

## 3.2.2 Member Data Documentation

### 3.2.2.1 b

```
uint8_t ColoredTriangle::b
```

Composante bleue.

### 3.2.2.2 g

```
uint8_t ColoredTriangle::g
```

Composante verte.

### 3.2.2.3 r

```
uint8_t ColoredTriangle::r
```

Composante rouge.

### 3.2.2.4 triangle

```
Triangle3d ColoredTriangle::triangle
```

Le triangle 3D.

The documentation for this struct was generated from the following file:

- [Scene.hpp](#)



## 3.3 Pave3d Class Reference

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

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

### Public Member Functions

- [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*

### Private Member Functions

- [Point3d](#) [rotatePoint](#) (const [Point3d](#) &p) const  
*Applique la rotation à un point.*

### Private Attributes

- [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.*

### 3.3.1 Detailed Description

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

### 3.3.2 Constructor & Destructor Documentation

#### 3.3.2.1 Pave3d()

```
Pave3d::Pave3d (
```

```
const Point3d & center,  
    float sizeX,  
    float sizeY,  
    float sizeZ)
```

**Parameters**

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

**3.3.3 Member Function Documentation****3.3.3.1 getTriangles()**

```
std::vector< Triangle3d > Pav3d::getTriangles () const
```

Retourne tous les triangles du pavé

**3.3.3.2 rotatePoint()**

```
Point3d Pav3d::rotatePoint (  
    const Point3d & p) const [private]
```

Applique la rotation à un point.

**3.3.3.3 setRotation()**

```
void Pav3d::setRotation (  
    float angleX,  
    float angleY,  
    float angleZ)
```

Définit les angles de rotation.

**3.3.4 Member Data Documentation****3.3.4.1 angleX**

```
float Pav3d::angleX [private]
```

Angle de rotation X.

**3.3.4.2 angleY**

```
float Pav3d::angleY [private]
```

Angle de rotation Y.

### 3.3.4.3 angleZ

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

Angle de rotation Z.

### 3.3.4.4 center

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

Centre du pavé

### 3.3.4.5 sizeX

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

Taille en X (largeur).

### 3.3.4.6 sizeY

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

Taille en Y (hauteur).

### 3.3.4.7 sizeZ

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

Taille en Z (profondeur).

The documentation for this class was generated from the following files:

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

## 3.4 Point2d Class Reference

Représente un point sur l'écran 2D.

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

### Public Member Functions

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

## Public Attributes

- int `x`  
*Coordonnée X mais sur l'écran (projetée).*
- int `y`  
*Coordonnée Y mais sur l'écran (projetée).*
- float `z`  
*Profondeur.*

### 3.4.1 Detailed Description

Représente un point sur l'écran 2D.

### 3.4.2 Constructor & Destructor Documentation

#### 3.4.2.1 `Point2d()` [1/2]

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

Constructeur par défaut.

#### 3.4.2.2 `Point2d()` [2/2]

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

Constructeur avec coordonnées.

## Parameters

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

### 3.4.3 Member Data Documentation

#### 3.4.3.1 `x`

```
int Point2d::x
```

Coordonnée X mais sur l'écran (projetée).

### 3.4.3.2 y

```
int Point2d::y
```

Coordonnée Y mais sur l'écran (projetée).

### 3.4.3.3 z

```
float Point2d::z
```

Profondeur.

The documentation for this class was generated from the following files:

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

## 3.5 Point3d Class Reference

Représente un point dans l'espace 3D.

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

### Public Member Functions

- [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.*

### Public Attributes

- float [x](#)  
*Coordonnée X.*
- float [y](#)  
*Coordonnée Y.*
- float [z](#)  
*Coordonnée Z.*

### 3.5.1 Detailed Description

Représente un point dans l'espace 3D.

### 3.5.2 Constructor & Destructor Documentation

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

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

Constructeur par défaut.

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

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

Constructeur avec coordonnées.

#### Parameters

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

### 3.5.3 Member Function Documentation

#### 3.5.3.1 cross()

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

Produit vectoriel.

#### 3.5.3.2 dot()

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

Produit scalaire.

### 3.5.3.3 normalize()

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

Normalise le vecteur.

### 3.5.3.4 operator\*()

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

Multiplication par un scalaire.

### 3.5.3.5 operator+()

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

Addition de deux points.

### 3.5.3.6 operator-()

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

Soustraction de deux points (retourne un vecteur).

## 3.5.4 Member Data Documentation

### 3.5.4.1 x

```
float Point3d::x
```

Coordonnée X.

### 3.5.4.2 y

```
float Point3d::y
```

Coordonnée Y.

### 3.5.4.3 z

```
float Point3d::z
```

Coordonnée Z.

The documentation for this class was generated from the following files:

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

## 3.6 Quad3d Class Reference

Quadrilatère composé de deux triangles.

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

### Public Member Functions

- [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.*

### Public Attributes

- [Triangle3d](#) t1  
*Premier triangle.*
- [Triangle3d](#) t2  
*Deuxième triangle.*

### 3.6.1 Detailed Description

Quadrilatère composé de deux triangles.

### 3.6.2 Constructor & Destructor Documentation

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

```
Quad3d::Quad3d ()
```

Constructeur par défaut.

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

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

Constructeur avec quatre points (dans le sens trigonométrique).



### Parameters

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

## 3.6.3 Member Function Documentation

### 3.6.3.1 getTriangles()

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

Retourne les triangles du quad.

## 3.6.4 Member Data Documentation

### 3.6.4.1 t1

```
Triangle3d Quad3d::t1
```

Premier triangle.

### 3.6.4.2 t2

```
Triangle3d Quad3d::t2
```

Deuxième triangle.

The documentation for this class was generated from the following files:

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

## 3.7 Scene Class Reference

Gère le rendu de la scène 3D.

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

## Public Member Functions

- [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.*

## Private Member Functions

- [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.*

## Private Attributes

- [Sdl](#) & [sdl](#)  
*Référence vers SDL.*
- [Camera](#) [camera](#)  
*Caméra de la scène.*
- std::vector< [ColoredTriangle](#) > [triangles](#)  
*Triangles colorés à render.*

### 3.7.1 Detailed Description

Gère le rendu de la scène 3D.

### 3.7.2 Constructor & Destructor Documentation

#### 3.7.2.1 Scene()

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

Constructeur.

## Parameters

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

## 3.7.3 Member Function Documentation

### 3.7.3.1 addTriangles()

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

Ajoute des triangles à la scène.

### 3.7.3.2 clearTriangles()

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

Efface les triangles de la scène.

### 3.7.3.3 drawScanline()

```
void 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.

### 3.7.3.4 fillFlatBottom()

```
void 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).

### 3.7.3.5 fillFlatTop()

```
void 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.

### 3.7.3.6 getCamera()

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

Retourne une référence vers la caméra.

### 3.7.3.7 isTriangleVisible()

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

Vérifie si un triangle est visible.

### 3.7.3.8 project()

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

Projette un point 3D en point 2D.

### 3.7.3.9 rasterizeTriangle()

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

Rasterise un triangle 2D.

### 3.7.3.10 render()

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

Effectue le rendu de la scène.

## 3.7.4 Member Data Documentation

### 3.7.4.1 camera

`Camera` Scene::camera [private]

Caméra de la scène.

### 3.7.4.2 sdl

`Sdl&` Scene::sdl [private]

Référence vers SDL.

### 3.7.4.3 triangles

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

Triangles colorés à render.

The documentation for this class was generated from the following files:

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

## 3.8 Sdl Class Reference

Classe contenant les fonctionnalités SDL pour le rendu graphique.

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

### Public Member Functions

- `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.*

### Private Attributes

- `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.*

## 3.8.1 Detailed Description

Classe contenant les fonctionnalités SDL pour le rendu graphique.

## 3.8.2 Constructor & Destructor Documentation

### 3.8.2.1 Sdl()

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

Constructeur de la classe [Sdl](#).

Constructeur initialisant SDL, la fenêtre et les tampons.

#### Parameters

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

### 3.8.2.2 ~Sdl()

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

Destructeur de la classe [Sdl](#).

Destructeur libérant les ressources SDL.

### 3.8.3 Member Function Documentation

#### 3.8.3.1 clear()

```
void 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.

##### Parameters

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

#### 3.8.3.2 getHeight()

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

Retourne la hauteur de la fenêtre.

##### Returns

Hauteur en pixels

#### 3.8.3.3 getWidth()

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

Retourne la largeur de la fenêtre.

##### Returns

Largeur en pixels

#### 3.8.3.4 isValid()

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

Vérifie si SDL a été initialisé correctement.

##### Returns

true si l'initialisation est réussie

### 3.8.3.5 present()

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

Présente le contenu du tampon à l'écran.

Affiche le contenu du tampon à l'écran.

### 3.8.3.6 setPixel()

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

Dessine un pixel dans le tampon.

Dessine un pixel avec profondeur.

#### Parameters

<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

## 3.8.4 Member Data Documentation

### 3.8.4.1 depthBuffer

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

Tampon de profondeur.

### 3.8.4.2 height

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

Hauteur de la fenêtre.



#### 3.8.4.3 pixelBuffer

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

Tampon de pixels.

#### 3.8.4.4 renderer

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

Pointeur vers le renderer SDL.

#### 3.8.4.5 texture

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

Texture pour le rendu pixel par pixel.

#### 3.8.4.6 width

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

Largeur de la fenêtre.

#### 3.8.4.7 window

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

Pointeur vers la fenêtre SDL.

The documentation for this class was generated from the following files:

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

## 3.9 Sphere3d Class Reference

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

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

### Public Member Functions

- [Sphere3d](#) (const [Point3d](#) &center, float radius, int meridians, int parallels)  
*Constructeur.*
- std::vector< [Triangle3d](#) > [getTriangles](#) () const  
*Retourne tous les triangles de la sphère.*

## Private Member Functions

- [Point3d spherePoint](#) (float theta, float phi) const  
*Calcule un point sur la sphère.*

## Private Attributes

- [Point3d center](#)  
*Centre de la sphère.*
- float [radius](#)  
*Rayon de la sphère.*
- int [meridians](#)  
*Nombre de méridiens.*
- int [parallels](#)  
*Nombre de parallèles.*

## 3.9.1 Detailed Description

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

## 3.9.2 Constructor & Destructor Documentation

### 3.9.2.1 Sphere3d()

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

Constructeur.

#### Parameters

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

## 3.9.3 Member Function Documentation

### 3.9.3.1 getTriangles()

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

Retourne tous les triangles de la sphère.

### 3.9.3.2 spherePoint()

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

Calcule un point sur la sphère.

## 3.9.4 Member Data Documentation

### 3.9.4.1 center

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

Centre de la sphère.

### 3.9.4.2 meridians

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

Nombre de méridiens.

### 3.9.4.3 parallels

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

Nombre de parallèles.

### 3.9.4.4 radius

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

Rayon de la sphère.

The documentation for this class was generated from the following files:

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

## 3.10 Triangle2d Class Reference

Triangle projeté sur l'écran 2D.

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

## Public Member Functions

- [Triangle2d](#) ()  
*Constructeur par défaut.*
- [Triangle2d](#) (const [Point2d](#) &p1, const [Point2d](#) &p2, const [Point2d](#) &p3)  
*Constructeur avec trois points (sommets).*

## Public Attributes

- [Point2d](#) p1  
*Premier sommet.*
- [Point2d](#) p2  
*Deuxième sommet.*
- [Point2d](#) p3  
*Troisième sommet.*

### 3.10.1 Detailed Description

Triangle projeté sur l'écran 2D.

### 3.10.2 Constructor & Destructor Documentation

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

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

Constructeur par défaut.

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

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

Constructeur avec trois points (sommets).

## Parameters

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

### 3.10.3 Member Data Documentation

#### 3.10.3.1 p1

`Point2d Triangle2d::p1`

Premier sommet.

#### 3.10.3.2 p2

`Point2d Triangle2d::p2`

Deuxième sommet.

#### 3.10.3.3 p3

`Point2d Triangle2d::p3`

Troisième sommet.

The documentation for this class was generated from the following files:

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

## 3.11 Triangle3d Class Reference

Triangle dans l'espace 3D.

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

### Public Member Functions

- [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.*

## Public Attributes

- [Point3d p1](#)  
*Premier sommet.*
- [Point3d p2](#)  
*Deuxième sommet.*
- [Point3d p3](#)  
*Troisième sommet.*

### 3.11.1 Detailed Description

Triangle dans l'espace 3D.

### 3.11.2 Constructor & Destructor Documentation

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

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

Constructeur par défaut.

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

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

Constructeur avec trois points (sommets).

## Parameters

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

### 3.11.3 Member Function Documentation

#### 3.11.3.1 center()

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

Calcule le centre du triangle.

### 3.11.3.2 normal()

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

Calcule la normale du triangle.

## 3.11.4 Member Data Documentation

### 3.11.4.1 p1

```
Point3d Triangle3d::p1
```

Premier sommet.

### 3.11.4.2 p2

```
Point3d Triangle3d::p2
```

Deuxième sommet.

### 3.11.4.3 p3

```
Point3d Triangle3d::p3
```

Troisième sommet.

The documentation for this class was generated from the following files:

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





## Chapter 4

# File Documentation

### 4.1 Geometry.cpp File Reference

Implémentation des classes géométriques.

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

#### 4.1.1 Detailed Description

Implémentation des classes géométriques.

### 4.2 Geometry.hpp File Reference

Définition des classes géométriques.

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

#### 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](#).*

## 4.2.1 Detailed Description

Définition des classes géométriques.

## 4.3 Geometry.hpp

[Go to the documentation of this file.](#)

```

00001
00005
00006 #ifndef GEOMETRY_HPP
00007 #define GEOMETRY_HPP
00008
00009 #include <vector>
00010 #include <cmath>
00011
00016 class Point3d {
00017 public:
00018     float x;
00019     float y;
00020     float z;
00021
00025     Point3d();
00026
00033     Point3d(float x, float y, float z);
00034
00038     Point3d operator-(const Point3d& other) const;
00039
00043     Point3d operator+(const Point3d& other) const;
00044
00048     Point3d operator*(float scalar) const;
00049
00053     Point3d cross(const Point3d& other) const;
00054
00058     float dot(const Point3d& other) const;
00059
00063     Point3d normalize() const;
00064 };
00065
00070 class Point2d {
00071 public:
00072     int x;
00073     int y;
00074     float z;
00075
00079     Point2d();
00080
00087     Point2d(int x, int y, float z);
00088 };
00089
00094 class Triangle3d {
00095 public:
00096     Point3d p1;
00097     Point3d p2;
00098     Point3d p3;
00099
00103     Triangle3d();
00104
00111     Triangle3d(const Point3d& p1, const Point3d& p2, const Point3d& p3);
00112
00116     Point3d normal() const;
00117
00121     Point3d center() const;
00122 };
00123
00128 class Triangle2d {
00129 public:
00130     Point2d p1;
00131     Point2d p2;
00132     Point2d p3;
00133
00137     Triangle2d();
00138
00145     Triangle2d(const Point2d& p1, const Point2d& p2, const Point2d& p3);
00146 };
00147
00152 class Quad3d {
00153 public:
00154     Triangle3d t1;
00155     Triangle3d t2;

```

```

00156
00160     Quad3d();
00161
00169     Quad3d(const Point3d& p1, const Point3d& p2,
00170           const Point3d& p3, const Point3d& p4);
00171
00175     std::vector<Triangle3d> getTriangles() const;
00176 };
00177
00182 class Pave3d {
00183 private:
00184     Point3d center;
00185     float sizeX;
00186     float sizeY;
00187     float sizeZ;
00188     float angleX;
00189     float angleY;
00190     float angleZ;
00191
00192 public:
00200     Pave3d(const Point3d& center, float sizeX, float sizeY, float sizeZ);
00201
00205     void setRotation(float angleX, float angleY, float angleZ);
00206
00210     std::vector<Triangle3d> getTriangles() const;
00211
00212 private:
00216     Point3d rotatePoint(const Point3d& p) const;
00217 };
00218
00223 class Sphere3d {
00224 private:
00225     Point3d center;
00226     float radius;
00227     int meridians;
00228     int parallels;
00229
00230 public:
00238     Sphere3d(const Point3d& center, float radius, int meridians, int parallels);
00239
00243     std::vector<Triangle3d> getTriangles() const;
00244
00245 private:
00249     Point3d spherePoint(float theta, float phi) const;
00250 };
00251
00252 #endif

```

## 4.4 main.cpp File Reference

Boucle principale et gestion des événements clavier.

```

#include "Sdl.hpp"
#include "Geometry.hpp"
#include "Scene.hpp"
#include <cmath>
#include <iostream>

```

### Functions

- 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.*

### 4.4.1 Detailed Description

Boucle principale et gestion des événements clavier.

### 4.4.2 Function Documentation

#### 4.4.2.1 addCube()

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

Crée et ajoute le cube à la scène.

#### Parameters

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

#### 4.4.2.2 addSphere()

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

Crée et ajoute la sphère à la scène.

#### Parameters

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

#### 4.4.2.3 handleKeyboard()

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

Gère les événements clavier.

### Parameters

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

#### 4.4.2.4 main()

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

main

### Parameters

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

### Returns

le code de sortie (0 si succès)

#### 4.4.2.5 mainLoop()

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

Boucle principale du programme.

### Parameters

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

### Returns

le code de sortie (0 si succès)

### 4.4.3 Variable Documentation

#### 4.4.3.1 CAMERA\_SPEED

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

Vitesse de déplacement de la caméra.

#### 4.4.3.2 ROTATION\_SPEED

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

Vitesse de rotation du cube.

#### 4.4.3.3 WINDOW\_HEIGHT

```
const int WINDOW_HEIGHT = 650
```

Hauteur de la fenêtre en pixels.

#### 4.4.3.4 WINDOW\_WIDTH

```
const int WINDOW_WIDTH = 1000
```

Largeur de la fenêtre en pixels.

### 4.5 Scene.cpp File Reference

Implémentation de la gestion de scène et rasterisation.

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

#### 4.5.1 Detailed Description

Implémentation de la gestion de scène et rasterisation.

### 4.6 Scene.hpp File Reference

Gestion de la scène, caméra et rasterisation.

```
#include "Geometry.hpp"  
#include "Sdl.hpp"  
#include <vector>
```

#### Classes

- 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.*

### 4.6.1 Detailed Description

Gestion de la scène, caméra et rasterisation.

## 4.7 Scene.hpp

[Go to the documentation of this file.](#)

```

00001
00005
00006 #ifndef SCENE_HPP
00007 #define SCENE_HPP
00008
00009 #include "Geometry.hpp"
00010 #include "Sdl.hpp"
00011 #include <vector>
00012
00017 class Camera {
00018 public:
00019     Point3d position;
00020     Point3d direction;
00021     Point3d up;
00022     float fov;
00023     float nearPlane;
00024
00028     Camera();
00029
00033     void moveForward(float delta);
00034
00038     void moveRight(float delta);
00039
00043     void moveLeft(float delta);
00044
00048     void moveUp(float delta);
00049 };
00050
00055 struct ColoredTriangle {
00056     Triangle3d triangle;
00057     uint8_t r;
00058     uint8_t g;
00059     uint8_t b;
00060 };
00061
00066 class Scene {
00067 private:
00068     Sdl& sdl;
00069     Camera camera;
00070     std::vector<ColoredTriangle> triangles;
00071
00072 public:
00077     Scene(Sdl& sdl);
00078
00082     void addTriangles(const std::vector<Triangle3d>& tris,
00083                     uint8_t r, uint8_t g, uint8_t b);
00084
00088     void clearTriangles();
00089
00093     void render();
00094
00098     Camera& getCamera();
00099
00100 private:
00104     Point2d project(const Point3d& p) const;
00105
00109     bool isTriangleVisible(const Triangle3d& triangle) const;
00110
00114     void rasterizeTriangle(const Triangle2d& triangle, float avgDepth,
00115                          uint8_t r, uint8_t g, uint8_t b);
00116
00120     void drawScanline(int y, int x1, int x2, float depth,
00121                    uint8_t r, uint8_t g, uint8_t b);
00122
00126     void fillFlatBottom(const Point2d& p1, const Point2d& p2,
00127                      const Point2d& p3, float depth,
00128                      uint8_t r, uint8_t g, uint8_t b);
00129
00133     void fillFlatTop(const Point2d& p1, const Point2d& p2,
00134                   const Point2d& p3, float depth,
00135                   uint8_t r, uint8_t g, uint8_t b);
00136 };
00137
00138 #endif

```

## 4.8 Sdl.cpp File Reference

Implémentation de la classe [Sdl](#).

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

### 4.8.1 Detailed Description

Implémentation de la classe [Sdl](#).

## 4.9 Sdl.hpp File Reference

Encapsulation de la bibliothèque SDL2.

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

### Classes

- class [Sdl](#)

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

### 4.9.1 Detailed Description

Encapsulation de la bibliothèque SDL2.

## 4.10 Sdl.hpp

[Go to the documentation of this file.](#)

```
00001
00005
00006 #ifndef SDL_HPP
00007 #define SDL_HPP
00008
00009 #include <SDL.h>
00010 #include <string>
00011 #include <cstdint>
00012
00017 class Sdl {
00018 private:
00019     SDL_Window* window;
00020     SDL_Renderer* renderer;
00021     SDL_Texture* texture;
00022     uint32_t* pixelBuffer;
00023     float* depthBuffer;
00024     int width;
00025     int height;
00026
```



```
00027 public:
00034     Sdl(const std::string& title, int w, int h);
00035
00039     ~Sdl();
00040
00047     void clear(uint8_t r, uint8_t g, uint8_t b);
00048
00058     void setPixel(int x, int y, float depth, uint8_t r, uint8_t g, uint8_t b);
00059
00063     void present();
00064
00069     int getWidth() const;
00070
00075     int getHeight() const;
00076
00081     bool isValid() const;
00082 };
00083
00084 #endif
```



# Index

- ~Sdl
  - Sdl, [22](#)
- addCube
  - main.cpp, [36](#)
- addSphere
  - main.cpp, [36](#)
- addTriangles
  - Scene, [19](#)
- angleX
  - Pave3d, [10](#)
- angleY
  - Pave3d, [10](#)
- angleZ
  - Pave3d, [10](#)
- b
  - ColoredTriangle, [8](#)
- Camera, [5](#)
  - Camera, [6](#)
  - direction, [7](#)
  - fov, [7](#)
  - moveForward, [6](#)
  - moveLeft, [6](#)
  - moveRight, [6](#)
  - moveUp, [6](#)
  - nearPlane, [7](#)
  - position, [7](#)
  - up, [7](#)
- camera
  - Scene, [21](#)
- CAMERA\_SPEED
  - main.cpp, [37](#)
- center
  - Pave3d, [11](#)
  - Sphere3d, [27](#)
  - Triangle3d, [30](#)
- clear
  - Sdl, [23](#)
- clearTriangles
  - Scene, [19](#)
- ColoredTriangle, [7](#)
  - b, [8](#)
  - g, [8](#)
  - r, [8](#)
  - triangle, [8](#)
- cross
  - Point3d, [14](#)
- depthBuffer
  - Sdl, [24](#)
- direction
  - Camera, [7](#)
- dot
  - Point3d, [14](#)
- drawScanline
  - Scene, [19](#)
- fillFlatBottom
  - Scene, [19](#)
- fillFlatTop
  - Scene, [19](#)
- fov
  - Camera, [7](#)
- g
  - ColoredTriangle, [8](#)
- Geometry.cpp, [33](#)
- Geometry.hpp, [33](#)
- getCamera
  - Scene, [20](#)
- getHeight
  - Sdl, [23](#)
- getTriangles
  - Pave3d, [10](#)
  - Quad3d, [17](#)
  - Sphere3d, [26](#)
- getWidth
  - Sdl, [23](#)
- handleKeyboard
  - main.cpp, [36](#)
- height
  - Sdl, [24](#)
- isTriangleVisible
  - Scene, [20](#)
- isValid
  - Sdl, [23](#)
- main
  - main.cpp, [37](#)
- main.cpp, [35](#)
  - addCube, [36](#)
  - addSphere, [36](#)
  - CAMERA\_SPEED, [37](#)
  - handleKeyboard, [36](#)
  - main, [37](#)
  - mainLoop, [37](#)
  - ROTATION\_SPEED, [37](#)
  - WINDOW\_HEIGHT, [38](#)

- WINDOW\_WIDTH, 38
- mainLoop
  - main.cpp, 37
- meridians
  - Sphere3d, 27
- moveForward
  - Camera, 6
- moveLeft
  - Camera, 6
- moveRight
  - Camera, 6
- moveUp
  - Camera, 6
- nearPlane
  - Camera, 7
- normal
  - Triangle3d, 30
- normalize
  - Point3d, 14
- operator+
  - Point3d, 15
- operator-
  - Point3d, 15
- operator\*
  - Point3d, 15
- p1
  - Triangle2d, 29
  - Triangle3d, 31
- p2
  - Triangle2d, 29
  - Triangle3d, 31
- p3
  - Triangle2d, 29
  - Triangle3d, 31
- parallels
  - Sphere3d, 27
- Pave3d, 9
  - angleX, 10
  - angleY, 10
  - angleZ, 10
  - center, 11
  - getTriangles, 10
  - Pave3d, 9
  - rotatePoint, 10
  - setRotation, 10
  - sizeX, 11
  - sizeY, 11
  - sizeZ, 11
- pixelBuffer
  - Sdl, 24
- Point2d, 11
  - Point2d, 12
  - x, 12
  - y, 12
  - z, 13
- Point3d, 13
  - cross, 14
  - dot, 14
  - normalize, 14
  - operator+, 15
  - operator-, 15
  - operator\*, 15
  - Point3d, 14
  - x, 15
  - y, 15
  - z, 15
- position
  - Camera, 7
- present
  - Sdl, 23
- project
  - Scene, 20
- Quad3d, 16
  - getTriangles, 17
  - Quad3d, 16
  - t1, 17
  - t2, 17
- r
  - ColoredTriangle, 8
- radius
  - Sphere3d, 27
- rasterizeTriangle
  - Scene, 20
- render
  - Scene, 20
- renderer
  - Sdl, 25
- rotatePoint
  - Pave3d, 10
- ROTATION\_SPEED
  - main.cpp, 37
- Scene, 17
  - addTriangles, 19
  - camera, 21
  - clearTriangles, 19
  - drawScanline, 19
  - fillFlatBottom, 19
  - fillFlatTop, 19
  - getCamera, 20
  - isTriangleVisible, 20
  - project, 20
  - rasterizeTriangle, 20
  - render, 20
  - Scene, 18
  - sdl, 21
  - triangles, 21
- Scene.cpp, 38
- Scene.hpp, 38
- Sdl, 21
  - ~Sdl, 22
  - clear, 23
  - depthBuffer, 24

- getHeight, [23](#)
- getWidth, [23](#)
- height, [24](#)
- isValid, [23](#)
- pixelBuffer, [24](#)
- present, [23](#)
- renderer, [25](#)
- Sdl, [22](#)
- setPixel, [24](#)
- texture, [25](#)
- width, [25](#)
- window, [25](#)
- sdl
  - Scene, [21](#)
- Sdl.cpp, [40](#)
- Sdl.hpp, [40](#)
- setPixel
  - Sdl, [24](#)
- setRotation
  - Pave3d, [10](#)
- sizeX
  - Pave3d, [11](#)
- sizeY
  - Pave3d, [11](#)
- sizeZ
  - Pave3d, [11](#)
- Sphere3d, [25](#)
  - center, [27](#)
  - getTriangles, [26](#)
  - meridians, [27](#)
  - parallels, [27](#)
  - radius, [27](#)
  - Sphere3d, [26](#)
  - spherePoint, [26](#)
- spherePoint
  - Sphere3d, [26](#)
- t1
  - Quad3d, [17](#)
- t2
  - Quad3d, [17](#)
- texture
  - Sdl, [25](#)
- triangle
  - ColoredTriangle, [8](#)
- Triangle2d, [27](#)
  - p1, [29](#)
  - p2, [29](#)
  - p3, [29](#)
  - Triangle2d, [28](#)
- Triangle3d, [29](#)
  - center, [30](#)
  - normal, [30](#)
  - p1, [31](#)
  - p2, [31](#)
  - p3, [31](#)
  - Triangle3d, [30](#)
- triangles
  - Scene, [21](#)
- up
  - Camera, [7](#)
- width
  - Sdl, [25](#)
- window
  - Sdl, [25](#)
- WINDOW\_HEIGHT
  - main.cpp, [38](#)
- WINDOW\_WIDTH
  - main.cpp, [38](#)
- x
  - Point2d, [12](#)
  - Point3d, [15](#)
- y
  - Point2d, [12](#)
  - Point3d, [15](#)
- z
  - Point2d, [13](#)
  - Point3d, [15](#)