

Raytracer

Generated by Doxygen 1.9.1



<b>1 Hierarchical Index</b>	<b>1</b>
1.1 Class Hierarchy	1
<b>2 Class Index</b>	<b>3</b>
2.1 Class List	3
<b>3 Class Documentation</b>	<b>5</b>
3.1 Light::Ambient Class Reference	5
3.1.1 Constructor & Destructor Documentation	5
3.1.1.1 Ambient()	6
3.1.2 Member Function Documentation	6
3.1.2.1 computeColor()	6
3.1.2.2 getColor()	6
3.1.2.3 getDiffuseMultiplier()	7
3.1.2.4 getMultiplier()	7
3.1.2.5 getType()	7
3.1.2.6 setDiffuseMultiplier()	7
3.1.2.7 setMultiplier()	8
3.2 Raytracer::Camera Class Reference	8
3.2.1 Constructor & Destructor Documentation	9
3.2.1.1 Camera() [1/2]	9
3.2.1.2 Camera() [2/2]	9
3.2.2 Member Function Documentation	9
3.2.2.1 operator=()	10
3.2.2.2 ray()	10
3.2.2.3 setFov()	10
3.2.2.4 setOrigin()	10
3.2.2.5 setResolution()	11
3.2.2.6 setRotation()	11
3.2.2.7 setScreen()	11
3.3 Raytracer::CameraBuilder Class Reference	12
3.3.1 Member Function Documentation	12
3.3.1.1 build()	12
3.3.1.2 setFov()	12
3.3.1.3 setOrigin()	13
3.3.1.4 setResolution()	13
3.3.1.5 setRotation()	14
3.3.1.6 setScreen()	14
3.4 Color Class Reference	14
3.4.1 Constructor & Destructor Documentation	15
3.4.1.1 Color() [1/2]	15
3.4.1.2 Color() [2/2]	15
3.4.2 Member Function Documentation	16

3.4.2.1	getB()	16
3.4.2.2	getG()	16
3.4.2.3	getR()	16
3.4.2.4	isWrongColor()	16
3.4.2.5	setB()	16
3.4.2.6	setG()	17
3.4.2.7	setR()	17
3.5	Primitive::Cone Class Reference	17
3.5.1	Constructor & Destructor Documentation	18
3.5.1.1	Cone()	18
3.5.2	Member Function Documentation	19
3.5.2.1	getAngle()	19
3.5.2.2	getAxis()	19
3.5.2.3	getColliderBox()	19
3.5.2.4	getMaterial()	19
3.5.2.5	getNormal()	19
3.5.2.6	getOrigin()	20
3.5.2.7	hitPoint()	20
3.5.2.8	setAngle()	20
3.5.2.9	setAxis()	21
3.5.2.10	setMaterial()	21
3.5.2.11	setOrigin()	21
3.5.2.12	setRotation()	22
3.6	Optimisation::Cube Class Reference	22
3.6.1	Constructor & Destructor Documentation	23
3.6.1.1	Cube() [1/2]	23
3.6.1.2	Cube() [2/2]	23
3.6.2	Member Function Documentation	24
3.6.2.1	getCollider()	24
3.6.2.2	getMaxX()	24
3.6.2.3	getMaxY()	24
3.6.2.4	getMaxZ()	25
3.6.2.5	getMinX()	25
3.6.2.6	getMinY()	25
3.6.2.7	getMinZ()	25
3.6.2.8	getPrimitivesContainer()	26
3.6.2.9	getPrimitivesHits()	26
3.6.2.10	identifyPrimitives()	26
3.6.2.11	setCollider()	26
3.6.2.12	setMaxX()	27
3.6.2.13	setMaxY()	27
3.6.2.14	setMaxZ()	27

3.6.2.15 setMinX()	27
3.6.2.16 setMinY()	28
3.6.2.17 setMinZ()	28
3.7 Optimisation::cubeCollider Struct Reference	28
3.8 Primitive::Cylinder Class Reference	29
3.8.1 Constructor & Destructor Documentation	29
3.8.1.1 Cylinder()	29
3.8.2 Member Function Documentation	30
3.8.2.1 getColliderBox()	30
3.8.2.2 getMaterial()	30
3.8.2.3 getNormal()	30
3.8.2.4 hitPoint()	31
3.8.2.5 setAxis() [1/2]	31
3.8.2.6 setAxis() [2/2]	31
3.8.2.7 setMaterial()	32
3.8.2.8 setOrigin()	32
3.8.2.9 setRadius()	32
3.8.2.10 setRotation()	33
3.9 Light::Directional Class Reference	33
3.9.1 Constructor & Destructor Documentation	34
3.9.1.1 Directional()	34
3.9.2 Member Function Documentation	34
3.9.2.1 computeColor()	34
3.9.2.2 getColor()	35
3.9.2.3 getDiffuseMultiplier()	35
3.9.2.4 getDirection()	35
3.9.2.5 getPosition()	35
3.9.2.6 getType()	36
3.9.2.7 setColor()	36
3.9.2.8 setDiffuseMultiplier()	36
3.9.2.9 setDirection()	36
3.9.2.10 setPosition()	37
3.10 Raytracer::DLLoader Class Reference	37
3.10.1 Constructor & Destructor Documentation	37
3.10.1.1 DLoader()	37
3.10.2 Member Function Documentation	38
3.10.2.1 getInstance()	38
3.11 Raytracer::Factory Class Reference	38
3.12 FlatColor Class Reference	39
3.13 Light::ILight Class Reference	39
3.13.1 Member Function Documentation	39
3.13.1.1 computeColor()	39

3.13.1.2 getColor()	40
3.13.1.3 getType()	40
3.14 Material::IMaterial Class Reference	40
3.14.1 Member Function Documentation	40
3.14.1.1 getType()	41
3.15 Primitive::IPrimitive Class Reference	41
3.15.1 Member Function Documentation	41
3.15.1.1 getColliderBox()	41
3.15.1.2 getMaterial()	42
3.15.1.3 getNormal()	42
3.15.1.4 hitPoint()	42
3.16 Light::LightsContainer Class Reference	43
3.16.1 Member Function Documentation	43
3.16.1.1 add()	43
3.16.1.2 computeColor()	44
3.16.1.3 getLightsList()	44
3.17 Optimisation::Octree Class Reference	44
3.17.1 Constructor & Destructor Documentation	45
3.17.1.1 Octree()	45
3.17.2 Member Function Documentation	45
3.17.2.1 getPrimitivesHits()	45
3.18 Raytracer::Scene::ParserException Class Reference	45
3.19 Primitive::Plane Class Reference	46
3.19.1 Constructor & Destructor Documentation	46
3.19.1.1 Plane()	47
3.19.2 Member Function Documentation	48
3.19.2.1 getAxis()	48
3.19.2.2 getColliderBox()	48
3.19.2.3 getMaterial()	48
3.19.2.4 getNormal()	48
3.19.2.5 getPosition()	49
3.19.2.6 hitPoint()	49
3.19.2.7 setAxis()	49
3.19.2.8 setMaterial()	50
3.19.2.9 setPosition()	50
3.19.2.10 setRotation()	50
3.20 Light::Point Class Reference	51
3.20.1 Constructor & Destructor Documentation	51
3.20.1.1 Point()	51
3.20.2 Member Function Documentation	52
3.20.2.1 computeColor()	52
3.20.2.2 getColor()	52

3.20.2.3 getDiffuseMultiplier()	52
3.20.2.4 getPosition()	53
3.20.2.5 getType()	53
3.20.2.6 setColor()	53
3.20.2.7 setDiffuseMultiplier()	53
3.20.2.8 setPosition()	54
3.21 Primitive::PrimitivesContainer Class Reference	54
3.21.1 Member Function Documentation	54
3.21.1.1 add()	55
3.21.1.2 computeColor()	56
3.21.1.3 getColorPoint()	56
3.21.1.4 getPrimitivesList()	57
3.22 Raytracer::Ray Class Reference	57
3.22.1 Detailed Description	57
3.22.2 Constructor & Destructor Documentation	57
3.22.2.1 Ray()	57
3.22.3 Member Function Documentation	58
3.22.3.1 at()	58
3.22.3.2 direction()	58
3.22.3.3 origin()	58
3.23 Raytracer::Rectangle3D Class Reference	59
3.23.1 Constructor & Destructor Documentation	59
3.23.1.1 Rectangle3D()	59
3.23.2 Member Function Documentation	59
3.23.2.1 pointAt()	59
3.24 Primitive::RectangularCuboid Class Reference	60
3.24.1 Constructor & Destructor Documentation	61
3.24.1.1 RectangularCuboid()	61
3.24.2 Member Function Documentation	61
3.24.2.1 getColliderBox()	62
3.24.2.2 getMaterial()	62
3.24.2.3 getMaxX()	62
3.24.2.4 getMaxY()	62
3.24.2.5 getMaxZ()	63
3.24.2.6 getMinX()	63
3.24.2.7 getMinY()	63
3.24.2.8 getMinZ()	63
3.24.2.9 getNormal()	63
3.24.2.10 hitPoint()	64
3.24.2.11 setMaterial()	64
3.24.2.12 setMaxX()	64
3.24.2.13 setMaxY()	65

3.24.2.14 setMaxZ()	65
3.24.2.15 setMinX()	65
3.24.2.16 setMinY()	66
3.24.2.17 setMinZ()	66
3.24.2.18 setRotation()	66
3.25 Raytracer::Renderer Class Reference	66
3.25.1 Constructor & Destructor Documentation	67
3.25.1.1 Renderer()	67
3.25.2 Member Function Documentation	67
3.25.2.1 writeColor()	67
3.26 Raytracer::Scene Class Reference	68
3.26.1 Constructor & Destructor Documentation	68
3.26.1.1 Scene()	68
3.26.2 Member Function Documentation	68
3.26.2.1 getCamera()	68
3.26.2.2 getLights()	69
3.26.2.3 getPrimitives()	69
3.27 Primitives::Shadow Class Reference	69
3.27.1 Constructor & Destructor Documentation	69
3.27.1.1 Shadow()	69
3.27.2 Member Function Documentation	70
3.27.2.1 isShadow()	70
3.28 Primitive::Sphere Class Reference	70
3.28.1 Constructor & Destructor Documentation	71
3.28.1.1 Sphere()	71
3.28.2 Member Function Documentation	71
3.28.2.1 getColliderBox()	71
3.28.2.2 getMaterial()	72
3.28.2.3 getNormal()	72
3.28.2.4 getOrigin()	72
3.28.2.5 getRadius()	73
3.28.2.6 hitPoint()	73
3.28.2.7 setMaterial()	73
3.28.2.8 setOrigin()	73
3.28.2.9 setRadius()	75
3.28.2.10 setRotation()	75
3.29 Primitive::Triangle Class Reference	75
3.29.1 Constructor & Destructor Documentation	76
3.29.1.1 Triangle()	76
3.29.2 Member Function Documentation	77
3.29.2.1 getColliderBox()	77
3.29.2.2 getMaterial()	77



3.29.2.3 getNormal()	77
3.29.2.4 getTriangleInverseNormal()	78
3.29.2.5 getTriangleNormal()	78
3.29.2.6 getVertex1()	78
3.29.2.7 getVertex2()	79
3.29.2.8 getVertex3()	79
3.29.2.9 hitPoint()	79
3.29.2.10 setMaterial()	79
3.29.2.11 setRotation()	80
3.29.2.12 setTriangleInverseNormal()	80
3.29.2.13 setTriangleNormal()	80
3.29.2.14 setVertex1()	81
3.29.2.15 setVertex2()	81
3.29.2.16 setVertex3()	81
3.30 Math::Vector3D Class Reference	81
3.30.1 Detailed Description	82
3.30.2 Constructor & Destructor Documentation	83
3.30.2.1 Vector3D()	83
3.30.3 Member Function Documentation	83
3.30.3.1 cross()	83
3.30.3.2 dot()	83
3.30.3.3 length()	84
3.30.3.4 length_squared()	84
3.30.3.5 rotateX()	84
3.30.3.6 rotateY()	85
3.30.3.7 rotateZ()	85
3.30.3.8 translate()	85
3.30.3.9 x()	85
3.30.3.10 y()	86
3.30.3.11 z()	86
<b>Index</b>	<b>87</b>



# Chapter 1

## Hierarchical Index

### 1.1 Class Hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:

Raytracer::Camera	8
Raytracer::CameraBuilder	12
Color	14
Optimisation::Cube	22
Optimisation::cubeCollider	28
Raytracer::DLLoader	37
std::exception	
Raytracer::Scene::ParserException	45
Raytracer::Factory	38
Light::ILight	39
Light::Ambient	5
Light::Directional	33
Light::Point	51
Material::IMaterial	40
FlatColor	39
Primitive::IPrimitive	41
Primitive::Cone	17
Primitive::Cylinder	29
Primitive::Plane	46
Primitive::RectangularCuboid	60
Primitive::Sphere	70
Primitive::Triangle	75
Light::LightsContainer	43
Optimisation::Octree	44
Primitive::PrimitivesContainer	54
Raytracer::Ray	57
Raytracer::Rectangle3D	59
Raytracer::Renderer	66
Raytracer::Scene	68
Primitives::Shadow	69
Math::Vector3D	81



## Chapter 2

# Class Index

### 2.1 Class List

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

Light::Ambient	5
Raytracer::Camera	8
Raytracer::CameraBuilder	12
Color	14
Primitive::Cone	17
Optimisation::Cube	22
Optimisation::cubeCollider	28
Primitive::Cylinder	29
Light::Directional	33
Raytracer::DLLoader	37
Raytracer::Factory	38
FlatColor	39
Light::ILight	39
Material::IMaterial	40
Primitive::IPrimitive	41
Light::LightsContainer	43
Optimisation::Octree	44
Raytracer::Scene::ParserException	45
Primitive::Plane	46
Light::Point	51
Primitive::PrimitivesContainer	54
Raytracer::Ray	
Ray class, (point and direction vectors)	57
Raytracer::Rectangle3D	59
Primitive::RectangularCuboid	60
Raytracer::Rendered	66
Raytracer::Scene	68
Primitives::Shadow	69
Primitive::Sphere	70
Primitive::Triangle	75
Math::Vector3D	
Vector 3D class (x, y, z)	81



## Chapter 3

# Class Documentation

### 3.1 Light::Ambient Class Reference

Inheritance diagram for Light::Ambient:

Collaboration diagram for Light::Ambient:

#### Public Member Functions

- [Ambient](#) ()  
*Construct a new [Ambient](#) object.*
- [Ambient](#) (double multiplier, double diffuseMultiplier)  
*Construct a new [Ambient](#) object.*
- [~Ambient](#) ()=default  
*Destroy the [Ambient](#) object.*
- double [getMultiplier](#) (void) const  
*Get the Multiplier number of ambient light.*
- void [setMultiplier](#) (double multiplier)  
*Set the Multiplier object.*
- double [getDiffuseMultiplier](#) (void) const  
*Get the Diffuse Multiplier number of ambient light.*
- void [setDiffuseMultiplier](#) (double diffuseMultiplier)  
*Set the Diffuse Multiplier object.*
- Light::LightType [getType](#) (void) const override  
*Get type of Light.*
- [Color](#) [getColor](#) (void) const override  
*Get the [Color](#) object.*
- [Color](#) [computeColor](#) (Math::Vector3D primitiveNormal, const [Math::Point3D](#) &hitPoint, [Math::Point3D](#) color, const [Primitives::Shadow](#) &shadow) const override  
*compute the color point with ambient light*

#### 3.1.1 Constructor & Destructor Documentation

### 3.1.1.1 Ambient()

```
Light::Ambient::Ambient (
    double multiplier,
    double diffuseMultiplier )
```

Construct a new [Ambient](#) object.

#### Parameters

<i>multiplier</i>	Multipler of ambient light
<i>diffuseMultiplier</i>	Diffuse Multipler of ambient light

## 3.1.2 Member Function Documentation

### 3.1.2.1 computeColor()

```
Color Light::Ambient::computeColor (
    Math::Vector3D primitiveNormal,
    const Math::Point3D & hitPoint,
    Math::Point3D color,
    const Primitives::Shadow & shadow ) const [override], [virtual]
```

compute the color point with ambient light

#### Parameters

<i>primitiveNormal</i>	normal to the hitpoint
<i>hitPoint</i>	hitpoint
<i>color</i>	color
<i>shadow</i>	Primitive::Shadow class to handle shadows

#### Returns

[Color](#) color

Implements [Light::Light](#).

### 3.1.2.2 getColor()

```
Color Light::Ambient::getColor (
    void ) const [override], [virtual]
```

Get the [Color](#) object.



**Returns**

[Color](#)

Implements [Light::ILight](#).

**3.1.2.3 getDiffuseMultiplier()**

```
double Light::Ambient::getDiffuseMultiplier (
    void ) const
```

Get the Diffuse Multiplier number of ambient light.

**Returns**

double

**3.1.2.4 getMultiplier()**

```
double Light::Ambient::getMultiplier (
    void ) const
```

Get the Multiplier number of ambient light.

**Returns**

double

**3.1.2.5 getType()**

```
Light::LightType Light::Ambient::getType (
    void ) const [override], [virtual]
```

Get type of Light.

**Returns**

The type of the light

Implements [Light::ILight](#).

**3.1.2.6 setDiffuseMultiplier()**

```
void Light::Ambient::setDiffuseMultiplier (
    double diffuseMultiplier )
```

Set the Diffuse Multiplier object.

## Parameters

<i>diffuseMultiplier</i>	New diffuse multiplier to set
--------------------------	-------------------------------

**3.1.2.7 setMultiplier()**

```
void Light::Ambient::setMultiplier (
    double multiplier )
```

Set the Multiplier object.

## Parameters

<i>multiplier</i>	Multipler of ambient light to set
-------------------	-----------------------------------

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

- include/Lights/Ambient.hpp
- src/Plugins/Lights/Ambient/Ambient.cpp

**3.2 Raytracer::Camera Class Reference****Public Member Functions**

- [Camera](#) ()  
*Construct a new [Camera](#) object.*
- [Camera](#) (const [Camera](#) &other)  
*Construct a new [Camera](#) object.*
- [Camera](#) ([Math::Point3D](#) origin, [Rectangle3D](#) screen)  
*Construct a new [Rectangle 3D](#) object.*
- [Camera](#) & [operator=](#) (const [Camera](#) &other)  
*Construct a new [Camera](#) object by another one.*
- [~Camera](#) ()  
*Destructor a [Camera](#) object.*
- [Raytracer::Ray](#) ray (double u, double v)  
*return a ray, going from the camera to the coordinates u and v of the image*
- [Math::Point3D](#) [getOrigin](#) (void) const  
*Get origin of [Camera](#).*
- [Raytracer::Rectangle3D](#) [getScreen](#) (void) const  
*Get Screen of [Camera](#).*
- double [getFov](#) (void) const  
*Get Fov of [Camera](#).*
- [Math::Vector3D](#) [getRotation](#) (void) const  
*Get Rotation of [Camera](#).*
- std::pair< double, double > [getResolution](#) (void) const

Get Resolution of [Camera](#).

- void [setOrigin](#) ([Math::Point3D](#) origin)  
Set the Origin object.
- void [setScreen](#) ([Raytracer::Rectangle3D](#) screen)  
Set the Screen object.
- void [setFov](#) (double fov)  
Set the Fov object.
- void [setRotation](#) ([Math::Vector3D](#) rotation)  
Set the Rotation object.
- void [setResolution](#) (double width, double height)  
Set the Resolution object.

## 3.2.1 Constructor & Destructor Documentation

### 3.2.1.1 Camera() [1/2]

```
Raytracer::Camera::Camera (
    const Camera & other )
```

Construct a new [Camera](#) object.

Parameters

<i>other</i>	other <a href="#">Camera</a> object to copy
--------------	---

### 3.2.1.2 Camera() [2/2]

```
Raytracer::Camera::Camera (
    Math::Point3D origin,
    Rectangle3D screen )
```

Construct a new Rectangle 3D object.

Parameters

<i>origin</i>	camera's origin point
<i>screen</i>	of camera

## 3.2.2 Member Function Documentation

### 3.2.2.1 operator=()

```
Raytracer::Camera & Raytracer::Camera::operator= (
    const Camera & other )
```

Construct a new [Camera](#) object by another one.

#### Parameters

<i>other</i>	<a href="#">Camera</a> object to duplicate
--------------	--

#### Returns

[Camera](#)& The new [Camera](#) object

### 3.2.2.2 ray()

```
Raytracer::Ray Raytracer::Camera::ray (
    double u,
    double v )
```

return a ray, going from the camera to the coordinates u and v of the image

#### Parameters

<i>u</i>	location u in rectangle
<i>v</i>	location v in rectangle

### 3.2.2.3 setFov()

```
void Raytracer::Camera::setFov (
    double fov )
```

Set the Fov object.

#### Parameters

<i>fov</i>	Fov to set
------------	------------

### 3.2.2.4 setOrigin()

```
void Raytracer::Camera::setOrigin (
    Math::Point3D origin )
```

Set the Origin object.

**Parameters**

<i>origin</i>	Origin to set
---------------	---------------

**3.2.2.5 setResolution()**

```
void Raytracer::Camera::setResolution (
    double width,
    double height )
```

Set the Resolution object.

**Parameters**

<i>width</i>	of image
<i>height</i>	of image

**3.2.2.6 setRotation()**

```
void Raytracer::Camera::setRotation (
    Math::Vector3D rotation )
```

Set the Rotation object.

**Parameters**

<i>rotation</i>	Rotation Vector to set
-----------------	------------------------

**3.2.2.7 setScreen()**

```
void Raytracer::Camera::setScreen (
    Raytracer::Rectangle3D screen )
```

Set the Screen object.

**Parameters**

<i>screen</i>	Screen to set
---------------	---------------

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

- include/Camera/Camera.hpp
- src/Core/Camera/Camera.cpp

## 3.3 Raytracer::CameraBuilder Class Reference

### Public Member Functions

- [CameraBuilder](#) ()=default  
*Construct a new [CameraBuilder](#) object.*
- [~CameraBuilder](#) ()=default  
*Destructor a [CameraBuilder](#) object.*
- [CameraBuilder](#) & [setOrigin](#) ([Math::Point3D](#) origin)  
*Set the Origin object.*
- [CameraBuilder](#) & [setScreen](#) ([Raytracer::Rectangle3D](#) screen)  
*Set the Screen object.*
- [CameraBuilder](#) & [setFov](#) (double fov)  
*Set the Fov object.*
- [CameraBuilder](#) & [setRotation](#) ([Math::Vector3D](#) rotation)  
*Set the Rotation object.*
- [CameraBuilder](#) & [setResolution](#) (double width, double height)  
*Set the Resolution object.*
- [Camera](#) [build](#) (void)  
*Build the [Camera](#) object.*

### 3.3.1 Member Function Documentation

#### 3.3.1.1 build()

```
Raytracer::Camera Raytracer::CameraBuilder::build (  
    void )
```

Build the [Camera](#) object.

Returns

[Camera](#) to build

#### 3.3.1.2 setFov()

```
Raytracer::CameraBuilder & Raytracer::CameraBuilder::setFov (  
    double fov )
```

Set the Fov object.

## Parameters

<i>fov</i>	Fov to set
------------	------------

## Returns

[CameraBuilder](#)& object to chain the setter

### 3.3.1.3 setOrigin()

```
Raytracer::CameraBuilder & Raytracer::CameraBuilder::setOrigin (
    Math::Point3D origin )
```

Set the Origin object.

## Parameters

<i>origin</i>	Origin to set
---------------	---------------

## Returns

[CameraBuilder](#)& object to chain the setter

### 3.3.1.4 setResolution()

```
Raytracer::CameraBuilder & Raytracer::CameraBuilder::setResolution (
    double width,
    double height )
```

Set the Resolution object.

## Parameters

<i>width</i>	of image
<i>height</i>	of image

## Returns

[CameraBuilder](#)& object to chain the setter

### 3.3.1.5 setRotation()

```
Raytracer::CameraBuilder & Raytracer::CameraBuilder::setRotation (
    Math::Vector3D rotation )
```

Set the Rotation object.

#### Parameters

<i>rotation</i>	Rotation Vector to set
-----------------	------------------------

#### Returns

[CameraBuilder](#)& object to chain the setter

### 3.3.1.6 setScreen()

```
Raytracer::CameraBuilder & Raytracer::CameraBuilder::setScreen (
    Raytracer::Rectangle3D screen )
```

Set the Screen object.

#### Parameters

<i>screen</i>	Screen to set
---------------	---------------

#### Returns

[CameraBuilder](#)& object to chain the setter

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

- include/Camera/CameraBuilder.hpp
- src/Core/Camera/CameraBuilder.cpp

## 3.4 Color Class Reference

### Public Member Functions

- [Color](#) (double r, double g, double b)  
*Construct a new [Color](#) object.*
- [Color](#) ([Math::Point3D](#) color)  
*Construct a new [Color](#) object form a Point3D.*
- [Color](#) ()=default  
*Construct a new [Color](#) object with default values (black, (0, 0, 0)).*



- `~Color()`=default  
*Destroy the `Color` object.*
- `void setR (double r)`  
*Set the red value.*
- `double getR () const`  
*Get the red value.*
- `void setG (double g)`  
*Set the green value.*
- `double getG () const`  
*Get the green value.*
- `void setB (double b)`  
*Set the blue value.*
- `double getB () const`  
*Get the blue value.*
- `bool isWrongColor () const`  
*Check if the rgb is valid ( $0 \leq rgb \leq 255$ ).*

### 3.4.1 Constructor & Destructor Documentation

#### 3.4.1.1 `Color()` [1/2]

```
Color::Color (
    double r,
    double g,
    double b )
```

Construct a new `Color` object.

##### Parameters

<i>r</i>	red value
<i>g</i>	green value
<i>b</i>	blue value

#### 3.4.1.2 `Color()` [2/2]

```
Color::Color (
    Math::Point3D color )
```

Construct a new `Color` object form a `Point3D`.

##### Parameters

<i>color</i>	
--------------	--

### 3.4.2 Member Function Documentation

#### 3.4.2.1 getB()

```
double Color::getB ( ) const
```

Get the blue value.

**Returns**

double - blue value

#### 3.4.2.2 getG()

```
double Color::getG ( ) const
```

Get the green value.

**Returns**

double - green value

#### 3.4.2.3 getR()

```
double Color::getR ( ) const
```

Get the red value.

**Returns**

double - red value

#### 3.4.2.4 isWrongColor()

```
bool Color::isWrongColor ( ) const
```

Check if the rgb is valid ( $0 \leq \text{rgb} \leq 255$ ).

**Returns**

true

false

#### 3.4.2.5 setB()

```
void Color::setB (
    double b )
```

Set the blue value.

## Parameters

<i>b</i>	blue value
----------	------------

**3.4.2.6 setG()**

```
void Color::setG (  
    double g )
```

Set the green value.

## Parameters

<i>g</i>	green value
----------	-------------

**3.4.2.7 setR()**

```
void Color::setR (  
    double r )
```

Set the red value.

## Parameters

<i>r</i>	red value
----------	-----------

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

- include/Color.hpp
- src/Core/Color.cpp

## 3.5 Primitive::Cone Class Reference

Inheritance diagram for Primitive::Cone:

Collaboration diagram for Primitive::Cone:

**Public Member Functions**

- [Cone](#) ()  
*Construct a new [Cone](#) object.*

- **Cone** (const [Math::Point3D](#) &origin, double radius, Axis axis, std::shared\_ptr< [Material::IMaterial](#) > material)  
*Construct a new [Cone](#) object.*
- **~Cone** ()=default  
*Destroy the [Cone](#) object.*
- **Math::Point3D hitPoint** (const [Raytracer::Ray](#) &ray) const override  
*return the hit point of the [Cone](#).*
- void **setOrigin** ([Math::Point3D](#) origin)  
*Set the Origin object.*
- void **setAngle** (double angle)  
*Set the Angle.*
- void **setAxis** (Axis axis)  
*Set the Axis.*
- void **setMaterial** (std::shared\_ptr< [Material::IMaterial](#) > material)  
*Set the Material.*
- void **setRotation** ([Math::Vector3D](#) rotation)  
*Set the Rotation object.*
- **Math::Point3D getOrigin** () const  
*Get the Origin object.*
- double **getAngle** () const  
*Get the Angle object.*
- Axis **getAxis** () const  
*Get the Axis object.*
- std::shared\_ptr< [Material::IMaterial](#) > **getMaterial** () const  
*Get the Material object.*
- **Math::Vector3D getNormal** (const [Math::Vector3D](#) &hitPoint, const [Raytracer::Ray](#) &ray) const override  
*Get the Normal of the object.*
- **Optimisation::cubeCollider getColliderBox** () const override  
*Get the collider box object.*

### 3.5.1 Constructor & Destructor Documentation

#### 3.5.1.1 Cone()

```
Primitive::Cone::Cone (
    const Math::Point3D & origin,
    double radius,
    Axis axis,
    std::shared_ptr< Material::IMaterial > material )
```

Construct a new [Cone](#) object.

##### Parameters

<i>origin</i>	center of the <a href="#">Cone</a>
<i>radius</i>	of the <a href="#">Cone</a>
<i>axis</i>	of the <a href="#">Cone</a>
<i>material</i>	Material of <a href="#">Cone</a>

## 3.5.2 Member Function Documentation

### 3.5.2.1 getAngle()

```
double Primitive::Cone::getAngle ( ) const
```

Get the Angle object.

Returns

Angle of [Cone](#)

### 3.5.2.2 getAxis()

```
Primitive::Axis Primitive::Cone::getAxis ( ) const
```

Get the Axis object.

Returns

Axis of [Cone](#)

### 3.5.2.3 getColliderBox()

```
Optimisation::cubeCollider Primitive::Cone::getColliderBox ( ) const [override], [virtual]
```

Get the collider box object.

Returns

Octree::cubeCollider

Implements [Primitive::IPrimitive](#).

### 3.5.2.4 getMaterial()

```
std::shared_ptr< Material::IMaterial > Primitive::Cone::getMaterial ( ) const [virtual]
```

Get the Material object.

Returns

Material of [Cone](#)

Implements [Primitive::IPrimitive](#).

### 3.5.2.5 getNormal()

```
Math::Vector3D Primitive::Cone::getNormal (
    const Math::Vector3D & hitPoint,
    const Raytracer::Ray & ray ) const [override], [virtual]
```

Get the Normal of the object.

## Parameters

<i>hitPoint</i>	to have the normal
<i>ray</i>	of the camera

## Returns

[Math::Vector3D](#)

Implements [Primitive::IPrimitive](#).

### 3.5.2.6 getOrigin()

```
Math::Point3D Primitive::Cone::getOrigin (
    void ) const
```

Get the Origin object.

## Returns

Origin of [Cone](#)

### 3.5.2.7 hitPoint()

```
Math::Point3D Primitive::Cone::hitPoint (
    const Raytracer::Ray & ray ) const [override], [virtual]
```

return the hit point of the [Cone](#).

## Parameters

<i>ray</i>	vector3D
------------	----------

## Returns

Point3D

Implements [Primitive::IPrimitive](#).

### 3.5.2.8 setAngle()

```
void Primitive::Cone::setAngle (
    double angle )
```

Set the Angle.

## Parameters

<i>angle</i>	New angle to set
--------------	------------------

**3.5.2.9 setAxis()**

```
void Primitive::Cone::setAxis (
    Axis axis )
```

Set the Axis.

## Parameters

<i>axis</i>	New axis to set
-------------	-----------------

**3.5.2.10 setMaterial()**

```
void Primitive::Cone::setMaterial (
    std::shared_ptr< Material::IMaterial > material )
```

Set the Material.

## Parameters

<i>material</i>	New material to set
-----------------	---------------------

**3.5.2.11 setOrigin()**

```
void Primitive::Cone::setOrigin (
    Math::Point3D origin )
```

Set the Origin object.

## Parameters

<i>origin</i>	New origin to set
---------------	-------------------

### 3.5.2.12 setRotation()

```
void Primitive::Cone::setRotation (
    Math::Vector3D rotation )
```

Set the Rotation object.

#### Parameters

<i>rotation</i>	- Rotation value
-----------------	------------------

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

- include/Primitives/Cone.hpp
- src/Plugins/Primitives/Cone/Cone.cpp

## 3.6 Optimisation::Cube Class Reference

### Public Member Functions

- [Cube](#) (double minX, double minY, double minZ, double maxX, double maxY, double maxZ, [PrimitivesContainer](#) primitives, int nbRecursions)  
*Construct a new [Cube](#) object.*
- [Cube](#) ([cubeCollider](#) collider, [PrimitivesContainer](#) primitives, int nbRecursions)  
*Construct a new [Cube](#) object.*
- [Cube](#) ()=default  
*Construct a new [Cube](#) object.*
- [~Cube](#) ()=default  
*Destroy the [Cube](#) object.*
- void [setMinX](#) (double minX)  
*Set the min X value.*
- void [setMinY](#) (double minY)  
*Set the min Y value.*
- void [setMinZ](#) (double minZ)  
*Set the min Z value.*
- void [setMaxX](#) (double maxX)  
*Set the max X value.*
- void [setMaxY](#) (double maxY)  
*Set the max Y value.*
- void [setMaxZ](#) (double maxZ)  
*Set the max Z value.*
- double [getMinX](#) () const  
*Get the min X value.*
- double [getMinY](#) () const  
*Get the min Y value.*
- double [getMinZ](#) () const  
*Get the min Z value.*
- double [getMaxX](#) () const  
*Get the max X value.*



- double `getMaxY ()` const  
*Get the max Y value.*
- double `getMaxZ ()` const  
*Get the max Z value.*
- void `setCollider (const cubeCollider &collider)`  
*Set the collider of the cube.*
- `cubeCollider getCollider ()` const  
*Get the collider of the cube.*
- `PrimitivesContainer getPrimitivesContainer (void)` const  
*Get the primitives container object.*
- void `identifyPrimitives (PrimitivesContainer primitives)`  
*Identify all primitives in the cube or on a side of the cube.*
- `PrimitivesContainer getPrimitivesHits (const Raytracer::Ray ray)` const  
*Get the primitives to be calculated with a ray hits object.*

### 3.6.1 Constructor & Destructor Documentation

#### 3.6.1.1 Cube() [1/2]

```
Cube::Cube (
    double minX,
    double minY,
    double minZ,
    double maxX,
    double maxY,
    double maxZ,
    PrimitivesContainer primitives,
    int nbRecursions )
```

Construct a new [Cube](#) object.

##### Parameters

<i>minX</i>	minX position
<i>minY</i>	minY position
<i>minZ</i>	minZ position
<i>maxX</i>	maxX position
<i>maxY</i>	maxY position
<i>maxZ</i>	maxZ position
<i>primitives</i>	primitives to check.

#### 3.6.1.2 Cube() [2/2]

```
Cube::Cube (
    cubeCollider collider,
```

```
PrimitivesContainer primitives,  
int nbRecurSIONS )
```

Construct a new [Cube](#) object.

normal

#### Parameters

<i>collider</i>	collider of the cube.
<i>primitives</i>	primitives to check.
<i>nbRecurSIONS</i>	number of recursions.

## 3.6.2 Member Function Documentation

### 3.6.2.1 getCollider()

```
cubeCollider Cube::getCollider ( ) const
```

Get the collider of the cube.

#### Returns

Octree::cubeCollider - collider of the cube.

### 3.6.2.2 getMaxX()

```
double Cube::getMaxX ( ) const
```

Get the max X value.

#### Returns

double

### 3.6.2.3 getMaxY()

```
double Cube::getMaxY ( ) const
```

Get the max Y value.

#### Returns

double

#### 3.6.2.4 getMaxZ()

```
double Cube::getMaxZ ( ) const
```

Get the max Z value.

**Returns**

double

#### 3.6.2.5 getMinX()

```
double Cube::getMinX ( ) const
```

Get the min X value.

**Returns**

double

#### 3.6.2.6 getMinY()

```
double Cube::getMinY ( ) const
```

Get the min Y value.

**Returns**

double

#### 3.6.2.7 getMinZ()

```
double Cube::getMinZ ( ) const
```

Get the min Z value.

**Returns**

double

### 3.6.2.8 getPrimitivesContainer()

```
PrimitivesContainer Cube::getPrimitivesContainer (
    void ) const
```

Get the primitives container object.

#### Returns

PrimitivesContainer

### 3.6.2.9 getPrimitivesHits()

```
PrimitivesContainer Cube::getPrimitivesHits (
    const Raytracer::Ray ray ) const
```

Get the primitives to be calculated with a ray hits object.

#### Parameters

<i>ray</i>	ray to check.
------------	---------------

#### Returns

PrimitivesContainer

### 3.6.2.10 identifyPrimitives()

```
void Cube::identifyPrimitives (
    PrimitivesContainer primitives )
```

Identify all primitives in the cube or on a side of the cube.

#### Parameters

<i>primitives</i>	primitives to check.
-------------------	----------------------

### 3.6.2.11 setCollider()

```
void Cube::setCollider (
    const cubeCollider & collider )
```

Set the collider of the cube.

## Parameters

<i>collider</i>	collider to set.
-----------------	------------------

**3.6.2.12 setMaxX()**

```
void Cube::setMaxX (
    double maxX )
```

Set the max X value.

## Parameters

<i>maxX</i>	
-------------	--

**3.6.2.13 setMaxY()**

```
void Cube::setMaxY (
    double maxY )
```

Set the max Y value.

## Parameters

<i>maxY</i>	
-------------	--

**3.6.2.14 setMaxZ()**

```
void Cube::setMaxZ (
    double maxZ )
```

Set the max Z value.

## Parameters

<i>maxZ</i>	
-------------	--

**3.6.2.15 setMinX()**

```
void Cube::setMinX (
```

```
double minX )
```

Set the min X value.

#### Parameters

<i>minX</i>	
-------------	--

### 3.6.2.16 setMinY()

```
void Cube::setMinY (  
    double minY )
```

Set the min Y value.

#### Parameters

<i>minY</i>	
-------------	--

### 3.6.2.17 setMinZ()

```
void Cube::setMinZ (  
    double minZ )
```

Set the min Z value.

#### Parameters

<i>minZ</i>	
-------------	--

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

- include/Optimisation/Cube.hpp
- src/Core/Optimisation/Cube.cpp

## 3.7 Optimisation::cubeCollider Struct Reference

### Public Attributes

- std::pair< bool, double > **minX**
- std::pair< bool, double > **minY**
- std::pair< bool, double > **minZ**
- std::pair< bool, double > **maxX**

- `std::pair< bool, double > maxY`
- `std::pair< bool, double > maxZ`

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

- `include/Optimisation/OctreeRules.hpp`

## 3.8 Primitive::Cylinder Class Reference

Inheritance diagram for Primitive::Cylinder:

Collaboration diagram for Primitive::Cylinder:

### Public Member Functions

- [Cylinder](#) ()  
*Construct a new [Cylinder](#) object.*
- [Cylinder](#) (const [Math::Point3D](#) &origin, double radius, Primitive::Axis axis)  
*Construct a new [Cylinder](#) object.*
- [~Cylinder](#) ()=default  
*Destroy the [Cylinder](#) object.*
- [Math::Point3D hitPoint](#) (const [Raytracer::Ray](#) &ray) const override  
*Return the hit point of the cylinder.*
- void [setOrigin](#) (const [Math::Point3D](#) &origin)  
*Set origin of cylinder.*
- void [setAxis](#) (double axis)  
*Set axis of cylinder.*
- void [setRadius](#) (double radius)  
*Set radius of cylinder.*
- void [setAxis](#) (const Primitive::Axis &axis)  
*Set the Axis object.*
- void [setRotation](#) ([Math::Vector3D](#) rotation)  
*Set the Rotation object.*
- `std::shared_ptr< Material::IMaterial > getMaterial ()` const  
*Get the Material object.*
- void [setMaterial](#) (std::shared\_ptr< [Material::IMaterial](#) > material)  
*Set the Material object.*
- [Math::Vector3D getNormal](#) (const [Math::Vector3D](#) &hitPoint, const [Raytracer::Ray](#) &ray) const override  
*Get the Normal of the object.*
- [Optimisation::cubeCollider getColliderBox](#) () const override  
*Get the collider box object.*

### 3.8.1 Constructor & Destructor Documentation

#### 3.8.1.1 Cylinder()

```
Primitive::Cylinder::Cylinder (
    const Math::Point3D & origin,
    double radius,
    Primitive::Axis axis )
```

Construct a new [Cylinder](#) object.

## Parameters

<i>origin</i>	center of the cylinder
<i>radius</i>	of the cylinder
<i>axis</i>	of the cylinder

## 3.8.2 Member Function Documentation

### 3.8.2.1 getColliderBox()

```
Optimisation::cubeCollider Primitive::Cylinder::getColliderBox ( ) const [override], [virtual]
```

Get the collider box object.

## Returns

Octree::cubeCollider

Implements [Primitive::IPrimitive](#).

### 3.8.2.2 getMaterial()

```
std::shared_ptr< Material::IMaterial > Primitive::Cylinder::getMaterial ( ) const [virtual]
```

Get the Material object.

## Returns

Material of cylinder

Implements [Primitive::IPrimitive](#).

### 3.8.2.3 getNormal()

```
Math::Vector3D Primitive::Cylinder::getNormal (
    const Math::Vector3D & hitPoint,
    const Raytracer::Ray & ray ) const [override], [virtual]
```

Get the Normal of the object.



## Parameters

<i>hitPoint</i>	to have the normal
<i>ray</i>	of the camera

## Returns

[Math::Vector3D](#)

Implements [Primitive::IPrimitive](#).

### 3.8.2.4 hitPoint()

```
Math::Point3D Primitive::Cylinder::hitPoint (
    const Raytracer::Ray & ray ) const [override], [virtual]
```

Return the hit point of the cylinder.

## Parameters

<i>ray</i>	vector3D
------------	----------

## Returns

Point3D

Implements [Primitive::IPrimitive](#).

### 3.8.2.5 setAxis() [1/2]

```
void Primitive::Cylinder::setAxis (
    const Primitive::Axis & axis )
```

Set the Axis object.

## Parameters

<i>axis</i>	The axis of cylinder Object to set
-------------	------------------------------------

### 3.8.2.6 setAxis() [2/2]

```
void Primitive::Cylinder::setAxis (
```

```
double axis )
```

Set axis of cylinder.

#### Parameters

<i>axis</i>	double
-------------	--------

### 3.8.2.7 setMaterial()

```
void Primitive::Cylinder::setMaterial (
    std::shared_ptr< Material::IMaterial > material )
```

Set the Material object.

#### Parameters

<i>material</i>	Material of cylinder
-----------------	----------------------

### 3.8.2.8 setOrigin()

```
void Primitive::Cylinder::setOrigin (
    const Math::Point3D & origin )
```

Set origin of cylinder.

#### Parameters

<i>hitPoint</i>	Point3D
-----------------	---------

### 3.8.2.9 setRadius()

```
void Primitive::Cylinder::setRadius (
    double radius )
```

Set radius of cylinder.

#### Parameters

<i>radius</i>	double
---------------	--------

### 3.8.2.10 setRotation()

```
void Primitive::Cylinder::setRotation (
    Math::Vector3D rotation )
```

Set the Rotation object.

#### Parameters

<i>rotation</i>	- Rotation value
-----------------	------------------

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

- include/Primitives/Cylinder.hpp
- src/Plugins/Primitives/Cylinder/Cylinder.cpp

## 3.9 Light::Directional Class Reference

Inheritance diagram for Light::Directional:

Collaboration diagram for Light::Directional:

### Public Member Functions

- **Directional** ()  
*Construct a new **Directional** object.*
- **Directional** (Math::Point3D position, Math::Vector3D direction, double diffuseMultiplier)  
*Construct a new **Directional** object.*
- **~Directional** ()=default  
*Destroy the **Directional** object.*
- Math::Point3D **getPosition** (void) const  
*Get the Position number of **Point** light.*
- void **setPosition** (Math::Point3D position)  
*Set the Position object.*
- Math::Vector3D **getDirection** (void) const  
*Get the Direction number of **Point** light.*
- void **setDirection** (Math::Vector3D direction)  
*Set the Direction object.*
- void **setColor** (const Color &rgb)  
*Set the **Color** object.*
- double **getDiffuseMultiplier** (void) const  
*Get the Diffuse Multiplier number of **Point** light.*
- void **setDiffuseMultiplier** (double diffuseMultiplier)  
*Set the Diffuse Multiplier object.*
- Light::LightType **getType** (void) const override  
*Get type of **Light**.*
- Color **getColor** (void) const override  
*Get the **Color** object.*
- Color **computeColor** (Math::Vector3D primitiveNormal, const Math::Point3D &hitPoint, Math::Point3D color, const Primitives::Shadow &shadow) const override  
*compute the color point with directional light*

### 3.9.1 Constructor & Destructor Documentation

#### 3.9.1.1 Directional()

```
Light::Directional::Directional (
    Math::Point3D position,
    Math::Vector3D direction,
    double diffuseMultiplier )
```

Construct a new [Directional](#) object.

##### Parameters

<i>position</i>	Position of Directionnal Light
<i>direction</i>	Direction of Directionnal Light
<i>diffuseMultiplier</i>	Diffuse Multiplier of Directionnal Light

### 3.9.2 Member Function Documentation

#### 3.9.2.1 computeColor()

```
Color Light::Directional::computeColor (
    Math::Vector3D primitiveNormal,
    const Math::Point3D & hitPoint,
    Math::Point3D color,
    const Primitives::Shadow & shadow ) const [override], [virtual]
```

compute the color point with directional light

##### Parameters

<i>primitiveNormal</i>	normal to the hitpoint
<i>hitPoint</i>	hitpoint
<i>color</i>	color
<i>shadow</i>	Primitive::Shadow class to handle shadows

##### Returns

Math::Point3D color

Implements [Light::Light](#).

### 3.9.2.2 getColor()

```
Color Light::Directional::getColor (
    void ) const [override], [virtual]
```

Get the [Color](#) object.

Returns

[Color](#)

Implements [Light::ILight](#).

### 3.9.2.3 getDiffuseMultiplier()

```
double Light::Directional::getDiffuseMultiplier (
    void ) const
```

Get the Diffuse Multiplier number of [Point](#) light.

Returns

double

### 3.9.2.4 getDirection()

```
Math::Vector3D Light::Directional::getDirection (
    void ) const
```

Get the Direction number of [Point](#) light.

Returns

[Math::Vector3D](#) direction

### 3.9.2.5 getPosition()

```
Math::Point3D Light::Directional::getPosition (
    void ) const
```

Get the Position number of [Point](#) light.

Returns

[Math::Point3D](#) position

### 3.9.2.6 getType()

```
Light::LightType Light::Directional::getType (
    void ) const [override], [virtual]
```

Get type of Light.

#### Returns

The type of the light

Implements [Light::ILight](#).

### 3.9.2.7 setColor()

```
void Light::Directional::setColor (
    const Color & rgb )
```

Set the [Color](#) object.

#### Parameters

<i>rgb</i>	color
------------	-------

### 3.9.2.8 setDiffuseMultiplier()

```
void Light::Directional::setDiffuseMultiplier (
    double diffuseMultiplier )
```

Set the Diffuse Multiplier object.

#### Parameters

<i>diffuseMultiplier</i>	New Diffuse Multiplier of <a href="#">Directional</a> Light
--------------------------	---

### 3.9.2.9 setDirection()

```
void Light::Directional::setDirection (
    Math::Vector3D direction )
```

Set the Direction object.

## Parameters

<i>direction</i>	New Direction of <a href="#">Directional</a> Light
------------------	--

**3.9.2.10 setPosition()**

```
void Light::Directional::setPosition (
    Math::Point3D position )
```

Set the Position object.

## Parameters

<i>position</i>	New position of <a href="#">Directional</a> Light
-----------------	---

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

- include/Lights/Directional.hpp
- src/Plugins/Lights/Directional/Directional.cpp

**3.10 Raytracer::DLLoader Class Reference****Public Member Functions**

- [DLLoader](#) (const std::string libraryPath)  
*Construct a new [DLLoader](#) object.*
- [~DLLoader](#) ()  
*Destroy the [DLLoader](#) object.*
- template<typename T >  
[T getInstance](#) (const std::string functionName) const  
*Get the Instance object.*

**Protected Attributes**

- std::string **\_libraryPath**
- void \* **\_libraryInstance**

**3.10.1 Constructor & Destructor Documentation****3.10.1.1 DLLoader()**

```
Raytracer::DLLoader::DLLoader (
    const std::string libraryPath )
```

Construct a new [DLLoader](#) object.

## Parameters

<i>libraryPath</i>	
--------------------	--

### 3.10.2 Member Function Documentation

#### 3.10.2.1 getInstance()

```
template<typename T >
T Raytracer::DLoader::getInstance (
    const std::string functionName ) const [inline]
```

Get the Instance object.

## Parameters

<i>functionName</i>	
---------------------	--

## Returns

T

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

- include/Parser/DLoader.hpp
- src/Core/Parser/DLoader.cpp

## 3.11 Raytracer::Factory Class Reference

### Public Types

- using **PrimitivesCreator** = std::function< std::shared\_ptr< [Primitive::IPrimitive](#) >()>
- using **LightsCreator** = std::function< std::shared\_ptr< [Light::ILight](#) >()>

### Public Member Functions

- [Factory](#) ()  
*Construct a new [Scene](#) object.*
- [~Factory](#) ()=default  
*Destruct a [Scene](#) object.*
- std::shared\_ptr< [Primitive::IPrimitive](#) > **createPrimitivesComponent** (const std::string &type)
- void **registerPrimitivesComponent** (const std::string &type, PrimitivesCreator creator)
- std::shared\_ptr< [Light::ILight](#) > **createLightsComponent** (const std::string &type)
- void **registerLightsComponent** (const std::string &type, LightsCreator creator)

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

- include/Parser/Factory.hpp
- src/Core/Parser/Factory.cpp



## 3.12 FlatColor Class Reference

Inheritance diagram for FlatColor:

## 3.13 Light::ILight Class Reference

Inheritance diagram for Light::ILight:

### Public Member Functions

- virtual [~ILight](#) ()=default  
*Destroy the [ILight](#) object.*
- virtual Light::LightType [getType](#) (void) const =0  
*Get type of Light.*
- virtual [Color](#) [getColor](#) (void) const =0  
*Get the [Color](#) object.*
- virtual [Color](#) [computeColor](#) ([Math::Vector3D](#) primitiveNormal, const [Math::Point3D](#) &hitPoint, [Math::Point3D](#) color, const [Primitives::Shadow](#) &shadow) const =0  
*Compute the color point with lights.*

### 3.13.1 Member Function Documentation

#### 3.13.1.1 computeColor()

```
virtual Color Light::ILight::computeColor (
    Math::Vector3D primitiveNormal,
    const Math::Point3D & hitPoint,
    Math::Point3D color,
    const Primitives::Shadow & shadow ) const    [pure virtual]
```

Compute the color point with lights.

#### Parameters

<i>primitiveNormal</i>	normal to the hitpoint
<i>hitPoint</i>	hitpoint
<i>color</i>	color
<i>shadow</i>	Primitive::Shadow class to handle shadows

#### Returns

[Color](#) color

Implemented in [Light::Point](#), [Light::Directional](#), and [Light::Ambient](#).

### 3.13.1.2 getColor()

```
virtual Color Light::ILight::getColor (
    void ) const [pure virtual]
```

Get the [Color](#) object.

Returns

[Color](#)

Implemented in [Light::Point](#), [Light::Directional](#), and [Light::Ambient](#).

### 3.13.1.3 getType()

```
virtual Light::LightType Light::ILight::getType (
    void ) const [pure virtual]
```

Get type of Light.

Returns

The type of the light

Implemented in [Light::Point](#), [Light::Directional](#), and [Light::Ambient](#).

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

- include/Lights/ILight.hpp

## 3.14 Material::IMaterial Class Reference

Inheritance diagram for Material::IMaterial:

### Public Member Functions

- virtual [~IMaterial](#) ()=default  
*Destroy the [IMaterial](#) object.*
- virtual MaterialType [getType](#) (void) const =0  
*Get type of Material.*

### 3.14.1 Member Function Documentation

### 3.14.1.1 getType()

```
virtual MaterialType Material::IMaterial::getType (
    void ) const [pure virtual]
```

Get type of Material.

#### Returns

The type of the material

Implemented in [FlatColor](#).

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

- include/Materials/IMaterial.hpp

## 3.15 Primitive::IPrimitive Class Reference

Inheritance diagram for Primitive::IPrimitive:

### Public Member Functions

- virtual [~IPrimitive](#) ()=default  
*Destroy the IPrimitive object.*
- virtual [Math::Point3D](#) hitPoint (const [Raytracer::Ray](#) &ray) const =0  
*compute the hit point of a primitive with a ray*
- virtual [Math::Vector3D](#) getNormal (const [Math::Vector3D](#) &hitPoint, const [Raytracer::Ray](#) &ray) const =0  
*Get the Normal of the object.*
- virtual std::shared\_ptr< [Material::IMaterial](#) > getMaterial () const =0  
*Get the Material object.*
- virtual [Optimisation::cubeCollider](#) getColliderBox () const =0  
*Get the collider box object.*

### 3.15.1 Member Function Documentation

#### 3.15.1.1 getColliderBox()

```
virtual Optimisation::cubeCollider Primitive::IPrimitive::getColliderBox ( ) const [pure virtual]
```

Get the collider box object.

#### Returns

Octree::cubeCollider

Implemented in [Primitive::Triangle](#), [Primitive::Sphere](#), [Primitive::RectangularCuboid](#), [Primitive::Plane](#), [Primitive::Cylinder](#), and [Primitive::Cone](#).

### 3.15.1.2 `getMaterial()`

```
virtual std::shared_ptr<Material::IMaterial> Primitive::IPrimitive::getMaterial ( ) const  
[pure virtual]
```

Get the Material object.

#### Returns

`std::shared_ptr<Material::IMaterial>`

Implemented in [Primitive::Triangle](#), [Primitive::Sphere](#), [Primitive::Plane](#), [Primitive::RectangularCuboid](#), [Primitive::Cylinder](#), and [Primitive::Cone](#).

### 3.15.1.3 `getNormal()`

```
virtual Math::Vector3D Primitive::IPrimitive::getNormal (   
    const Math::Vector3D & hitPoint,   
    const Raytracer::Ray & ray ) const [pure virtual]
```

Get the Normal of the object.

#### Parameters

<i>hitPoint</i>	to compute the normal
<i>ray</i>	of the camera

#### Returns

[Math::Vector3D](#)

Implemented in [Primitive::Triangle](#), [Primitive::Sphere](#), [Primitive::RectangularCuboid](#), [Primitive::Plane](#), [Primitive::Cylinder](#), and [Primitive::Cone](#).

### 3.15.1.4 `hitPoint()`

```
virtual Math::Point3D Primitive::IPrimitive::hitPoint (   
    const Raytracer::Ray & ray ) const [pure virtual]
```

compute the hit point of a primitive with a ray

#### Parameters

<i>ray</i>	Vector3D
------------	----------

## Returns

Math::Point3D

Implemented in [Primitive::Triangle](#), [Primitive::Sphere](#), [Primitive::RectangularCuboid](#), [Primitive::Plane](#), [Primitive::Cylinder](#), and [Primitive::Cone](#).

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

- include/Primitives/Primitive.hpp

## 3.16 Light::LightsContainer Class Reference

### Public Member Functions

- [LightsContainer](#) ()=default  
*Construct a new Lights Container object.*
- [~LightsContainer](#) ()=default  
*Destroy the Lights Container object.*
- void [add](#) (std::shared\_ptr< [Light::ILight](#) > Light)  
*Add a Light to the container.*
- void [clear](#) ()  
*Clear the container.*
- std::vector< std::shared\_ptr< [Light::ILight](#) > > [getLightsList](#) (void) const  
*Get the Lights List object.*
- [Color](#) [computeColor](#) ([Math::Vector3D](#) primitiveNormal, const [Math::Point3D](#) &hitPoint, [Math::Point3D](#) color, const [Primitives::Shadow](#) &shadow) const  
*Compute the color point with lights.*

### 3.16.1 Member Function Documentation

#### 3.16.1.1 add()

```
void Light::LightsContainer::add (
    std::shared_ptr< Light::ILight > Light )
```

Add a Light to the container.

#### Parameters

<i>Light</i>	to add
--------------	--------

### 3.16.1.2 computeColor()

```
Color Light::LightsContainer::computeColor (
    Math::Vector3D primitiveNormal,
    const Math::Point3D & hitPoint,
    Math::Point3D color,
    const Primitives::Shadow & shadow ) const
```

Compute the color point with lights.

#### Parameters

<i>primitiveNormal</i>	normal to the hitpoint
<i>hitPoint</i>	hitpoint
<i>color</i>	color
<i>shadow</i>	Primitive::Shadow class to handle shadows

#### Returns

Math::Point3D color  
 Color color

### 3.16.1.3 getLightsList()

```
std::vector< std::shared_ptr< Light::ILight > > Light::LightsContainer::getLightsList (
    void ) const
```

Get the Lights List object.

#### Returns

std::vector<std::shared\_ptr<Light::ILight>>

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

- include/Lights/LightsContainer.hpp
- src/Core/Lights/LightsContainer.cpp

## 3.17 Optimisation::Octree Class Reference

### Public Member Functions

- [Octree](#) ([PrimitivesContainer](#) primitives, [cubeCollider](#) cubeCollider)  
*Construct a new [Octree](#) object with a list of primitives.*
- [Octree](#) ()=default  
*Construct a new [Octree](#) object.*
- [~Octree](#) ()=default  
*Destroy the [Octree](#) object.*
- [PrimitivesContainer](#) getPrimitivesHits (const [Raytracer::Ray](#) &ray) const  
*Get the primitives to be calculated with a ray hits object.*

### 3.17.1 Constructor & Destructor Documentation

#### 3.17.1.1 Octree()

```
Octree::Octree (
    PrimitivesContainer primitives,
    cubeCollider cubeCollider )
```

Construct a new [Octree](#) object with a list of primitives.

##### Parameters

<i>primitives</i>	primitives to study.
<i>cubeCollider</i>	collider of the cube.

### 3.17.2 Member Function Documentation

#### 3.17.2.1 getPrimitivesHits()

```
PrimitivesContainer Octree::getPrimitivesHits (
    const Raytracer::Ray & ray ) const
```

Get the primitives to be calculated with a ray hits object.

##### Parameters

<i>ray</i>	ray to check.
------------	---------------

##### Returns

PrimitivesContainer

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

- include/Optimisation/Octree.hpp
- src/Core/Optimisation/Octree.cpp

## 3.18 Raytracer::Scene::ParserException Class Reference

Inheritance diagram for Raytracer::Scene::ParserException:

Collaboration diagram for Raytracer::Scene::ParserException:

## Public Member Functions

- **ParserException** (const std::string &msg)
- virtual const char \* **what** () const noexcept override

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

- include/Parser/Scene.hpp

## 3.19 Primitive::Plane Class Reference

Inheritance diagram for Primitive::Plane:

Collaboration diagram for Primitive::Plane:

## Public Member Functions

- [Plane](#) ()  
*Construct a new [Plane](#) object.*
- [Plane](#) (Primitive::Axis axis, double position, std::shared\_ptr< [Material::IMaterial](#) > material)  
*Construct a new [Plane](#) object.*
- [~Plane](#) ()=default  
*Destroy the [Plane](#) object.*
- [Math::Point3D hitPoint](#) (const [Raytracer::Ray](#) &ray) const override  
*Return the hit point of the plane.*
- Primitive::Axis [getAxis](#) (void) const  
*Get the Axis object.*
- void [setAxis](#) (const Primitive::Axis &axis)  
*Set the Axis object.*
- void [setRotation](#) ([Math::Vector3D](#) rotation)  
*Set the Rotation object.*
- [Math::Point3D getPosition](#) (void) const  
*Get the Position object plane.*
- void [setPosition](#) ([Math::Point3D](#) position)  
*Set the Position object.*
- std::shared\_ptr< [Material::IMaterial](#) > [getMaterial](#) () const override  
*Get the Material object.*
- void [setMaterial](#) (std::shared\_ptr< [Material::IMaterial](#) > material)  
*Set the Material.*
- [Math::Vector3D getNormal](#) (const [Math::Vector3D](#) &hitPoint, const [Raytracer::Ray](#) &ray) const override  
*Get the normal of the object.*
- [Optimisation::cubeCollider getColliderBox](#) () const override  
*Get the collider box object.*

### 3.19.1 Constructor & Destructor Documentation



### 3.19.1.1 Plane()

```
Primitive::Plane::Plane (
    Primitive::Axis axis,
    double position,
    std::shared_ptr< Material::IMaterial > material )
```

Construct a new [Plane](#) object.

## Parameters

<i>axis</i>	Axis of the plane
<i>position</i>	offset on axis

## 3.19.2 Member Function Documentation

### 3.19.2.1 getAxis()

```
Primitive::Axis Primitive::Plane::getAxis (  
    void ) const
```

Get the Axis object.

## Returns

Axis The axis of [Plane](#) Object

### 3.19.2.2 getColliderBox()

```
Optimisation::cubeCollider Primitive::Plane::getColliderBox ( ) const [override], [virtual]
```

Get the collider box object.

## Returns

Octree::cubeCollider

Implements [Primitive::IPrimitive](#).

### 3.19.2.3 getMaterial()

```
std::shared_ptr< Material::IMaterial > Primitive::Plane::getMaterial ( ) const [override],  
[virtual]
```

Get the Material object.

## Returns

Material of plane

Implements [Primitive::IPrimitive](#).

### 3.19.2.4 getNormal()

```
Math::Vector3D Primitive::Plane::getNormal (  
    const Math::Vector3D & hitPoint,  
    const Raytracer::Ray & ray ) const [override], [virtual]
```

Get the normal of the object.

## Parameters

<i>hitPoint</i>	to have the normal
<i>ray</i>	of the camera

## Returns

[Math::Vector3D](#)

Implements [Primitive::IPrimitive](#).

### 3.19.2.5 getPosition()

```
Math::Point3D Primitive::Plane::getPosition (
    void ) const
```

Get the Position object plane.

## Returns

Math::Point3D Position of [Plane](#) Object

### 3.19.2.6 hitPoint()

```
Math::Point3D Primitive::Plane::hitPoint (
    const Raytracer::Ray & ray ) const [override], [virtual]
```

Return the hit point of the plane.

## Parameters

<i>ray</i>	ray to check vector3D
------------	-----------------------

## Returns

Point3D

Implements [Primitive::IPrimitive](#).

### 3.19.2.7 setAxis()

```
void Primitive::Plane::setAxis (
    const Primitive::Axis & axis )
```

Set the Axis object.

## Parameters

<i>axis</i>	The axis of <a href="#">Plane</a> Object to set
-------------	---

**3.19.2.8 setMaterial()**

```
void Primitive::Plane::setMaterial (
    std::shared_ptr< Material::IMaterial > material )
```

Set the Material.

## Parameters

<i>material</i>	New material to set
-----------------	---------------------

**3.19.2.9 setPosition()**

```
void Primitive::Plane::setPosition (
    Math::Point3D position )
```

Set the Position object.

## Parameters

<i>position</i>	Position to set
-----------------	-----------------

**3.19.2.10 setRotation()**

```
void Primitive::Plane::setRotation (
    Math::Vector3D rotation )
```

Set the Rotation object.

## Parameters

<i>rotation</i>	- Rotation value
-----------------	------------------

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

- include/Primitives/Plane.hpp
- src/Plugins/Primitives/Plane/Plane.cpp

## 3.20 Light::Point Class Reference

Inheritance diagram for Light::Point:

Collaboration diagram for Light::Point:

### Public Member Functions

- [Point](#) ()  
*Construct a new [Point](#) object.*
- [Point](#) ([Math::Point3D](#) position, double diffuseMultiplier)  
*Construct a new [Point](#) object.*
- [~Point](#) ()=default  
*Destroy the [Point](#) object.*
- [Math::Point3D](#) getPosition (void) const  
*Get the Position number of [Point](#) light.*
- void setPosition ([Math::Point3D](#) position)  
*Set the Position object.*
- void setColor (const [Color](#) &rgb)  
*Set the [Color](#) object.*
- double getDiffuseMultiplier (void) const  
*Get the Diffuse Multiplier number of [Point](#) light.*
- void setDiffuseMultiplier (double diffuseMultiplier)  
*Set the Diffuse Multiplier object.*
- Light::LightType getType (void) const override  
*Get type of Light.*
- [Color](#) getColor (void) const override  
*Get the [Color](#) object.*
- [Color](#) computeColor ([Math::Vector3D](#) primitiveNormal, const [Math::Point3D](#) &hitPoint, [Math::Point3D](#) color, const [Primitives::Shadow](#) &shadow) const override  
*Compute the color point with ponctual light.*

### 3.20.1 Constructor & Destructor Documentation

#### 3.20.1.1 Point()

```
Light::Point::Point (
    Math::Point3D position,
    double diffuseMultiplier )
```

Construct a new [Point](#) object.

#### Parameters

<i>position</i>	Position of <a href="#">Point</a> Light
<i>diffuseMultiplier</i>	Diffuse multiplier of <a href="#">Point</a> Light

## 3.20.2 Member Function Documentation

### 3.20.2.1 computeColor()

```
Color Light::Point::computeColor (
    Math::Vector3D primitiveNormal,
    const Math::Point3D & hitPoint,
    Math::Point3D color,
    const Primitives::Shadow & shadow ) const [override], [virtual]
```

Compute the color point with ponctual light.

#### Parameters

<i>primitiveNormal</i>	normal to the hitpoint
<i>hitPoint</i>	hitpoint
<i>color</i>	color
<i>shadow</i>	Primitive::Shadow class to handle shadows

#### Returns

Math::Point3D color

Implements [Light::ILight](#).

### 3.20.2.2 getColor()

```
Color Light::Point::getColor (
    void ) const [override], [virtual]
```

Get the [Color](#) object.

#### Returns

[Color](#)

Implements [Light::ILight](#).

### 3.20.2.3 getDiffuseMultiplier()

```
double Light::Point::getDiffuseMultiplier (
    void ) const
```

Get the Diffuse Multiplier number of [Point](#) light.

#### Returns

double

#### 3.20.2.4 getPosition()

```
Math::Point3D Light::Point::getPosition (
    void ) const
```

Get the Position number of [Point](#) light.

##### Returns

Math::Point3D position

#### 3.20.2.5 getType()

```
Light::LightType Light::Point::getType (
    void ) const [override], [virtual]
```

Get type of Light.

##### Returns

The type of the light

Implements [Light::ILight](#).

#### 3.20.2.6 setColor()

```
void Light::Point::setColor (
    const Color & rgb )
```

Set the [Color](#) object.

##### Parameters

<i>rgb</i>	color
------------	-------

#### 3.20.2.7 setDiffuseMultiplier()

```
void Light::Point::setDiffuseMultiplier (
    double diffuseMultiplier )
```

Set the Diffuse Multiplier object.

## Parameters

<i>diffuseMultiplier</i>	New Diffuse Multiplier of <a href="#">Point</a> Light
--------------------------	---

**3.20.2.8 setPosition()**

```
void Light::Point::setPosition (
    Math::Point3D position )
```

Set the Position object.

## Parameters

<i>position</i>	New position of <a href="#">Point</a> Light
-----------------	---

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

- include/Lights/Point.hpp
- src/Plugins/Lights/Point/Point.cpp

**3.21 Primitive::PrimitivesContainer Class Reference****Public Member Functions**

- [PrimitivesContainer](#) ()=default  
*Construct a new Primitives Container object.*
- [~PrimitivesContainer](#) ()=default  
*Destroy the Primitives Container object.*
- void [add](#) (std::shared\_ptr< [Primitive::IPrimitive](#) > primitive)  
*Add a Primitive to the container.*
- void [clear](#) ()  
*Clear the container.*
- [Color](#) [getColorPoint](#) (const [Raytracer::Ray](#) &ray, const [Light::LightsContainer](#) &lights) const  
*Return the color of hit point of a ray in all the primitives.*
- std::vector< std::shared\_ptr< [Primitive::IPrimitive](#) > > [getPrimitivesList](#) (void) const  
*Get the Primitives List object.*
- [Color](#) [computeColor](#) (const std::shared\_ptr< [Primitive::IPrimitive](#) > &primitive, const [Math::Point3D](#) &hitPoint, const [Light::LightsContainer](#) &lights, const [Raytracer::Ray](#) &ray) const  
*Compute the color pixel of a primitive's hitpoint.*

**3.21.1 Member Function Documentation**



### 3.21.1.1 add()

```
void Primitive::PrimitivesContainer::add (  
    std::shared_ptr< Primitive::IPrimitive > primitive )
```

Add a Primitive to the container.

## Parameters

<i>primitive</i>	to add
------------------	--------

**3.21.1.2 computeColor()**

```
Color Primitive::PrimitivesContainer::computeColor (
    const std::shared_ptr< Primitive::IPrimitive > & primitive,
    const Math::Point3D & hitPoint,
    const Light::LightsContainer & lights,
    const Raytracer::Ray & ray ) const
```

Compute the color pixel of a primitive's hitpoint.

## Parameters

<i>primitive</i>	primitive to compute
<i>hitPoint</i>	to check
<i>lights</i>	list of lights
<i>ray</i>	ray of the camera

## Returns

Math::Point3D

**3.21.1.3 getColorPoint()**

```
Color Primitive::PrimitivesContainer::getColorPoint (
    const Raytracer::Ray & ray,
    const Light::LightsContainer & lights ) const
```

Return the color of hit point of a ray in all the primitives.

## Parameters

<i>ray</i>	<a href="#">Math::Vector3D</a>
<i>lights</i>	list of lights

## Returns

[Color](#)

### 3.21.1.4 getPrimitivesList()

```
std::vector< std::shared_ptr< Primitive::IPrimitive > > Primitive::PrimitivesContainer::get←
PrimitivesList (
    void ) const
```

Get the Primitives List object.

#### Returns

```
std::vector<std::shared_ptr<Primitive::IPrimitive>>
```

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

- include/Primitives/PrimitivesContainer.hpp
- src/Core/Primitives/PrimitivesContainer.cpp

## 3.22 Raytracer::Ray Class Reference

[Ray](#) class, (point and direction vectors)

```
#include <Ray.hpp>
```

### Public Member Functions

- [Ray](#) ()=default  
*Construct a new [Ray](#) object.*
- [Ray](#) (const [Math::Point3D](#) &origin, const [Math::Vector3D](#) &direction)  
*Construct a new [Ray](#) object.*
- [~Ray](#) ()=default  
*Destroy the [Ray](#) object.*
- const [Math::Point3D](#) & origin () const  
*return the origin of the ray*
- const [Math::Vector3D](#) & direction () const  
*return the direction of the ray*
- [Math::Point3D](#) at (double t) const  
*return the point vector of where point the ray at multiply by t*

### 3.22.1 Detailed Description

[Ray](#) class, (point and direction vectors)

### 3.22.2 Constructor & Destructor Documentation

#### 3.22.2.1 Ray()

```
Raytracer::Ray::Ray (
    const Math::Point3D & origin,
    const Math::Vector3D & direction )
```

Construct a new [Ray](#) object.

## Parameters

<i>origin</i>	point vector
<i>direction</i>	vector direction

### 3.22.3 Member Function Documentation

#### 3.22.3.1 at()

```
Math::Point3D Raytracer::Ray::at (
    double t ) const
```

return the point vector of where point the ray at multiply by t

## Parameters

<i>t</i>	multiplication factor
----------	-----------------------

## Returns

Math::Point3D

#### 3.22.3.2 direction()

```
const Math::Vector3D & Raytracer::Ray::direction ( ) const
```

return the direction of the ray

## Returns

const Math::Vector3D&

#### 3.22.3.3 origin()

```
const Math::Point3D & Raytracer::Ray::origin ( ) const
```

return the origin of the ray

## Returns

const Math::Point3D&

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

- include/Ray.hpp
- src/Core/Ray.cpp

## 3.23 Raytracer::Rectangle3D Class Reference

### Public Member Functions

- [Rectangle3D](#) ()  
*Construct a new Rectangle 3D object.*
- [Rectangle3D](#) ([Math::Point3D](#) origin, [Math::Vector3D](#) bottom\_side, [Math::Vector3D](#) left\_side)  
*Construct a new Rectangle 3D object.*
- [~Rectangle3D](#) ()  
*Destructor a Rectangle 3D object.*
- [Math::Point3D](#) pointAt (double u, double v)  
*returns the 3D coordinates of the point at the given location in our rectangle*
- [Math::Point3D](#) getOrigin (void)  
*Get origin of Rectangle 3D.*
- [Math::Vector3D](#) getBottomSide (void)  
*Get bottom side of Rectangle 3D.*
- [Math::Vector3D](#) getLeftSide (void)  
*Get left side of Rectangle 3D.*

### 3.23.1 Constructor & Destructor Documentation

#### 3.23.1.1 Rectangle3D()

```
Raytracer::Rectangle3D::Rectangle3D (
    Math::Point3D origin,
    Math::Vector3D bottom_side,
    Math::Vector3D left_side )
```

Construct a new Rectangle 3D object.

#### Parameters

<i>origin</i>	bottom-left corner of the rectangle
<i>bottom_side</i>	vector from the bottom-left corner of the rectangle
<i>left_side</i>	vector from the bottom-left corner of the rectangle

### 3.23.2 Member Function Documentation

#### 3.23.2.1 pointAt()

```
Math::Point3D Raytracer::Rectangle3D::pointAt (
    double u,
    double v )
```

returns the 3D coordinates of the point at the given location in our rectangle

#### Parameters

<i>u</i>	location u in rectangle
<i>v</i>	location v in rectangle

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

- include/Camera/Rectangle.hpp
- src/Core/Camera/Rectangle.cpp

## 3.24 Primitive::RectangularCuboid Class Reference

Inheritance diagram for Primitive::RectangularCuboid:

Collaboration diagram for Primitive::RectangularCuboid:

### Public Member Functions

- [RectangularCuboid](#) ()  
*Construct a new [RectangularCuboid](#) object.*
- [RectangularCuboid](#) (double maxX, double maxY, double maxZ, double minX, double minY, double minZ, std::shared\_ptr< [Material::IMaterial](#) > material)  
*Construct a new [RectangularCuboid](#) object.*
- [~RectangularCuboid](#) ()=default  
*Destroy the [RectangularCuboid](#) object.*
- [Math::Point3D hitPoint](#) (const [Raytracer::Ray](#) &ray) const override  
*Return the hit point of the rectangular cuboid.*
- double [getMinX](#) () const  
*Get the min x value.*
- double [getMinY](#) () const  
*Get the min y value.*
- double [getMinZ](#) () const  
*Get the min z value.*
- double [getMaxX](#) () const  
*Get the max x value.*
- double [getMaxY](#) () const  
*Get the max y value.*
- double [getMaxZ](#) () const  
*Get the max z value.*
- void [setMinX](#) (double minX)  
*Set the min x value.*
- void [setMinY](#) (double minY)  
*Set the min y value.*
- void [setMinZ](#) (double minZ)  
*Set the min z value.*
- void [setMaxX](#) (double maxX)

- Set the max x value.*
  - void [setMaxY](#) (double maxY)
- Set the max y value.*
  - void [setMaxZ](#) (double maxZ)
- Set the max z value.*
  - std::shared\_ptr< [Material::IMaterial](#) > [getMaterial](#) () const
- Get the Material object.*
  - void [setMaterial](#) (std::shared\_ptr< [Material::IMaterial](#) > material)
- Set the Material object.*
  - [Math::Vector3D](#) [getNormal](#) (const [Math::Vector3D](#) &hitPoint, const [Raytracer::Ray](#) &ray) const override
- Get the Normal of the object.*
  - void [setRotation](#) ([Math::Vector3D](#) rotation)
- Set the Rotation of the object.*
  - [Optimisation::cubeCollider](#) [getColliderBox](#) () const override
- Get the collider box object.*

### 3.24.1 Constructor & Destructor Documentation

#### 3.24.1.1 RectangularCuboid()

```
Primitive::RectangularCuboid::RectangularCuboid (
    double maxX,
    double maxY,
    double maxZ,
    double minX,
    double minY,
    double minZ,
    std::shared_ptr< Material::IMaterial > material )
```

Construct a new [RectangularCuboid](#) object.

##### Parameters

<i>maxX</i>	double max x of the rectangular cuboid
<i>maxY</i>	double max y of the rectangular cuboid
<i>maxZ</i>	double max z of the rectangular cuboid
<i>minX</i>	double min x of the rectangular cuboid
<i>minY</i>	double min y of the rectangular cuboid
<i>minZ</i>	double min z of the rectangular cuboid
<i>material</i>	Material of the rectangular cuboid

### 3.24.2 Member Function Documentation

### 3.24.2.1 getColliderBox()

```
Optimisation::cubeCollider Primitive::RectangularCuboid::getColliderBox ( ) const [override],  
[virtual]
```

Get the collider box object.

#### Returns

Octree::cubeCollider

Implements [Primitive::IPrimitive](#).

### 3.24.2.2 getMaterial()

```
std::shared_ptr< Material::IMaterial > Primitive::RectangularCuboid::getMaterial ( ) const  
[virtual]
```

Get the Material object.

#### Returns

Material of rectangular cuboid

Implements [Primitive::IPrimitive](#).

### 3.24.2.3 getMaxX()

```
double Primitive::RectangularCuboid::getMaxX ( ) const
```

Get the max x value.

#### Returns

double | max x value.

### 3.24.2.4 getMaxY()

```
double Primitive::RectangularCuboid::getMaxY ( ) const
```

Get the max y value.

#### Returns

double | max y value.



### 3.24.2.5 getMaxZ()

```
double Primitive::RectangularCuboid::getMaxZ ( ) const
```

Get the max z value.

#### Returns

double | max z value.

### 3.24.2.6 getMinX()

```
double Primitive::RectangularCuboid::getMinX ( ) const
```

Get the min x value.

#### Returns

double | min x value.

### 3.24.2.7 getMinY()

```
double Primitive::RectangularCuboid::getMinY ( ) const
```

Get the min y value.

#### Returns

double | min y value.

### 3.24.2.8 getMinZ()

```
double Primitive::RectangularCuboid::getMinZ ( ) const
```

Get the min z value.

#### Returns

double | min z value.

### 3.24.2.9 getNormal()

```
Math::Vector3D Primitive::RectangularCuboid::getNormal (
    const Math::Vector3D & hitPoint,
    const Raytracer::Ray & ray ) const [override], [virtual]
```

Get the Normal of the object.

## Parameters

<i>hitPoint</i>	to have the normal
<i>ray</i>	of the camera

## Returns

[Math::Vector3D](#)

Implements [Primitive::IPrimitive](#).

**3.24.2.10 hitPoint()**

```
Math::Point3D Primitive::RectangularCuboid::hitPoint (
    const Raytracer::Ray & ray ) const [override], [virtual]
```

Return the hit point of the rectangular cuboid.

## Parameters

<i>ray</i>	vector3D
------------	----------

## Returns

Point3D

Implements [Primitive::IPrimitive](#).

**3.24.2.11 setMaterial()**

```
void Primitive::RectangularCuboid::setMaterial (
    std::shared_ptr< Material::IMaterial > material )
```

Set the Material object.

## Parameters

<i>material</i>	Material of rectangular cuboid
-----------------	--------------------------------

**3.24.2.12 setMaxX()**

```
void Primitive::RectangularCuboid::setMaxX (
```

```
double maxX )
```

Set the max x value.

#### Parameters

<i>maxX</i>	double max x value.
-------------	---------------------

### 3.24.2.13 setMaxY()

```
void Primitive::RectangularCuboid::setMaxY (
    double maxY )
```

Set the max y value.

#### Parameters

<i>maxY</i>	double max y value.
-------------	---------------------

### 3.24.2.14 setMaxZ()

```
void Primitive::RectangularCuboid::setMaxZ (
    double maxZ )
```

Set the max z value.

#### Parameters

<i>maxZ</i>	double max z value.
-------------	---------------------

### 3.24.2.15 setMinX()

```
void Primitive::RectangularCuboid::setMinX (
    double minX )
```

Set the min x value.

#### Parameters

<i>minX</i>	double min x value.
-------------	---------------------

### 3.24.2.16 setMinY()

```
void Primitive::RectangularCuboid::setMinY (
    double minY )
```

Set the min y value.

#### Parameters

<i>minY</i>	double min y value.
-------------	---------------------

### 3.24.2.17 setMinZ()

```
void Primitive::RectangularCuboid::setMinZ (
    double minZ )
```

Set the min z value.

#### Parameters

<i>minZ</i>	double min z value.
-------------	---------------------

### 3.24.2.18 setRotation()

```
void Primitive::RectangularCuboid::setRotation (
    Math::Vector3D rotation )
```

Set the Rotation of the object.

#### Parameters

<i>rotation</i>	- Rotation value
-----------------	------------------

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

- include/Primitives/RectangularCuboid.hpp
- src/Plugins/Primitives/RectangularCuboid/RectangularCuboid.cpp

## 3.25 Raytracer::Renderer Class Reference

### Public Member Functions

- [Renderer](#) ([Raytracer::Scene](#) scene)

- Construct a new [Renderer](#) object.
- [~Renderer](#) ()=default  
Destroy the [Renderer](#) object.
- void [renderScene](#) ()  
Render the scene in a window with SDL2.
- void [renderFinalScene](#) ()  
Render the final scene in a .ppm file.
- void [writeColor](#) (std::ostream &o, const [Color](#) &color)  
Write a rgb color in a stream.

## 3.25.1 Constructor & Destructor Documentation

### 3.25.1.1 Renderer()

```
Raytracer::Renderer::Renderer (
    Raytracer::Scene scene )
```

Construct a new [Renderer](#) object.

#### Parameters

<i>scene</i>	to render
--------------	-----------

## 3.25.2 Member Function Documentation

### 3.25.2.1 writeColor()

```
void Raytracer::Renderer::writeColor (
    std::ostream & o,
    const Color & color )
```

Write a rgb color in a stream.

#### Parameters

<i>o</i>	stream to write in
<i>color</i>	color to write

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

- include/Renderer.hpp
- src/Core/Renderer.cpp

## 3.26 Raytracer::Scene Class Reference

### Classes

- class [ParserException](#)

### Public Types

- using **PrimitivesCreator** = std::function< std::unique\_ptr< [Primitive::IPrimitive](#) >()>

### Public Member Functions

- [Scene](#) (std::string filePath)  
*Construct a new [Scene](#) object.*
- [~Scene](#) ()=default  
*Destruct a [Scene](#) object.*
- [Raytracer::Camera](#) & [getCamera](#) (void)  
*Get the [Camera](#) object.*
- [Primitive::PrimitivesContainer](#) [getPrimitives](#) (void) const  
*Get the Primitives object.*
- [Light::LightsContainer](#) [getLights](#) (void) const  
*Get the Lights object.*

### 3.26.1 Constructor & Destructor Documentation

#### 3.26.1.1 Scene()

```
Raytracer::Scene::Scene (
    std::string filePath )
```

Construct a new [Scene](#) object.

#### Parameters

<i>filePath</i>	File to parse
-----------------	---------------

### 3.26.2 Member Function Documentation

#### 3.26.2.1 getCamera()

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

Get the [Camera](#) object.

Returns

[Camera](#) Object

### 3.26.2.2 getLights()

```
Light::LightsContainer Raytracer::Scene::getLights (
    void ) const
```

Get the Lights object.

Returns

[Light::LightsContainer](#)

### 3.26.2.3 getPrimitives()

```
Primitive::PrimitivesContainer Raytracer::Scene::getPrimitives (
    void ) const
```

Get the Primitives object.

Returns

[Primitive::PrimitivesContainer](#)

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

- include/Parser/Scene.hpp
- src/Core/Parser/Scene.cpp

## 3.27 Primitives::Shadow Class Reference

### Public Member Functions

- [Shadow](#) (const std::vector< std::shared\_ptr< [Primitive::IPrimitive](#) >> &primitives)  
*Construct a new [Shadow](#) object.*
- [~Shadow](#) ()=default  
*Destroy the [Shadow](#) object.*
- bool [isShadow](#) (const [Math::Vector3D](#) &vectorLightToPoint, const [Math::Point3D](#) &hitPoint) const  
*Return if there is a shadow.*

### 3.27.1 Constructor & Destructor Documentation

#### 3.27.1.1 Shadow()

```
Primitives::Shadow::Shadow (
    const std::vector< std::shared_ptr< Primitive::IPrimitive >> & primitives )
```

Construct a new [Shadow](#) object.

## Parameters

<i>primitives</i>	list of primitives
-------------------	--------------------

### 3.27.2 Member Function Documentation

#### 3.27.2.1 isShadow()

```
bool Primitives::Shadow::isShadow (
    const Math::Vector3D & vectorLightToPoint,
    const Math::Point3D & hitPoint ) const
```

Return if there is a shadow.

## Returns

true  
false

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

- include/Primitives/Shadow.hpp
- src/Core/Primitives/Shadow.cpp

## 3.28 Primitive::Sphere Class Reference

Inheritance diagram for Primitive::Sphere:

Collaboration diagram for Primitive::Sphere:

### Public Member Functions

- [Sphere](#) ()  
*Construct a new [Sphere](#) object.*
- [Sphere](#) (const [Math::Point3D](#) &origin, double radius, std::shared\_ptr< [Material::IMaterial](#) > material)  
*Construct a new [Sphere](#) object.*
- [~Sphere](#) ()=default  
*Destroy the [Sphere](#) object.*
- [Math::Point3D hitPoint](#) (const [Raytracer::Ray](#) &ray) const override  
*Return the hit point of the sphere.*
- void [setOrigin](#) ([Math::Point3D](#) origin)  
*Set the Origin object.*
- void [setRadius](#) (double radius)  
*Set the Radius.*



- void [setMaterial](#) (std::shared\_ptr< [Material::IMaterial](#) > material)  
*Set the Material.*
- void [setRotation](#) ([Math::Vector3D](#) rotation)  
*Set the Rotation object.*
- [Math::Point3D](#) [getOrigin](#) () const  
*Get the Origin object.*
- double [getRadius](#) () const  
*Get the Origin object.*
- std::shared\_ptr< [Material::IMaterial](#) > [getMaterial](#) () const override  
*Get the Material object.*
- [Math::Vector3D](#) [getNormal](#) (const [Math::Vector3D](#) &hitPoint, const [Raytracer::Ray](#) &ray) const override  
*Get the Normal of the object.*
- [Optimisation::cubeCollider](#) [getColliderBox](#) () const override  
*Get the collider box object.*

## 3.28.1 Constructor & Destructor Documentation

### 3.28.1.1 Sphere()

```
Primitive::Sphere::Sphere (
    const Math::Point3D & origin,
    double radius,
    std::shared_ptr< Material::IMaterial > material )
```

Construct a new [Sphere](#) object.

#### Parameters

<i>origin</i>	center of the sphere
<i>radius</i>	of the sphere
<i>material</i>	Material of <a href="#">Sphere</a>

## 3.28.2 Member Function Documentation

### 3.28.2.1 getColliderBox()

```
Optimisation::cubeCollider Primitive::Sphere::getColliderBox ( ) const [override], [virtual]
```

Get the collider box object.

#### Returns

[Octree::cubeCollider](#)

Implements [Primitive::IPrimitive](#).

### 3.28.2.2 `getMaterial()`

```
std::shared_ptr< Material::IMaterial > Primitive::Sphere::getMaterial ( ) const [override],  
[virtual]
```

Get the Material object.

#### Returns

Material of sphere

Implements [Primitive::IPrimitive](#).

### 3.28.2.3 `getNormal()`

```
Math::Vector3D Primitive::Sphere::getNormal (   
    const Math::Vector3D & hitPoint,   
    const Raytracer::Ray & ray ) const [override], [virtual]
```

Get the Normal of the object.

#### Parameters

<i>hitPoint</i>	to compute the normal
<i>ray</i>	of the camera

#### Returns

[Math::Vector3D](#)

Implements [Primitive::IPrimitive](#).

### 3.28.2.4 `getOrigin()`

```
Math::Point3D Primitive::Sphere::getOrigin (   
    void ) const
```

Get the Origin object.

#### Returns

Origin of sphere

### 3.28.2.5 getRadius()

```
double Primitive::Sphere::getRadius ( ) const
```

Get the Origin object.

#### Returns

Radius of sphere

### 3.28.2.6 hitPoint()

```
Math::Point3D Primitive::Sphere::hitPoint (
    const Raytracer::Ray & ray ) const [override], [virtual]
```

Return the hit point of the sphere.

#### Parameters

<i>ray</i>	vector3D
------------	----------

#### Returns

Point3D

Implements [Primitive::IPrimitive](#).

### 3.28.2.7 setMaterial()

```
void Primitive::Sphere::setMaterial (
    std::shared_ptr< Material::IMaterial > material )
```

Set the Material.

#### Parameters

<i>material</i>	New material to set
-----------------	---------------------

### 3.28.2.8 setOrigin()

```
void Primitive::Sphere::setOrigin (
    Math::Point3D origin )
```

Set the Origin object.

## Parameters

<i>origin</i>	New origin to set
---------------	-------------------

**3.28.2.9 setRadius()**

```
void Primitive::Sphere::setRadius (
    double radius )
```

Set the Radius.

## Parameters

<i>radius</i>	New radius to set
---------------	-------------------

**3.28.2.10 setRotation()**

```
void Primitive::Sphere::setRotation (
    Math::Vector3D rotation )
```

Set the Rotation object.

## Parameters

<i>rotation</i>	- Rotation value
-----------------	------------------

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

- include/Primitives/Sphere.hpp
- src/Plugins/Primitives/Sphere/Sphere.cpp

**3.29 Primitive::Triangle Class Reference**

Inheritance diagram for Primitive::Triangle:

Collaboration diagram for Primitive::Triangle:

**Public Member Functions**

- [Triangle](#) ()  
*Construct a new triangle object.*

- **Triangle** (const [Math::Point3D](#) &vertex1, const [Math::Point3D](#) &vertex2, const [Math::Point3D](#) &vertex3, std::shared\_ptr< [Material::IMaterial](#) > material)  
*Construct a new triangle object.*
- **~Triangle** ()=default  
*Destroy the triangle object.*
- **Math::Point3D hitPoint** (const [Raytracer::Ray](#) &ray) const override  
*Return the hit point of the triangle.*
- void **setVertex1** (const [Math::Point3D](#) &vertex1)  
*Set the vertex1 object.*
- void **setVertex2** (const [Math::Point3D](#) &vertex2)  
*Set the vertex2 object.*
- void **setVertex3** (const [Math::Point3D](#) &vertex3)  
*Set the vertex3 object.*
- void **setMaterial** (std::shared\_ptr< [Material::IMaterial](#) > material)  
*Set the material.*
- void **setRotation** ([Math::Vector3D](#) rotation)  
*Set the rotation object.*
- **Math::Point3D getVertex1** () const  
*Get the vertex1 object.*
- **Math::Point3D getVertex2** () const  
*Get the vertex2 object.*
- **Math::Point3D getVertex3** () const  
*Get the vertex3 object.*
- std::shared\_ptr< [Material::IMaterial](#) > **getMaterial** () const override  
*Get the material object.*
- **Math::Vector3D getNormal** (const [Math::Vector3D](#) &hitPoint, const [Raytracer::Ray](#) &ray) const override  
*Get the normal of the object.*
- void **createNormals** ()  
*Create the two different normals of the triangle.*
- **Math::Vector3D getTriangleNormal** () const  
*Get the triangle normal object.*
- **Math::Vector3D getTriangleInverseNormal** () const  
*Get the triangle inverse normal object.*
- void **setTriangleNormal** (const [Math::Vector3D](#) &normal)  
*Set the triangle normal object.*
- void **setTriangleInverseNormal** (const [Math::Vector3D](#) &inverseNormal)  
*Set the triangle inverse normal object.*
- **Optimisation::cubeCollider getColliderBox** () const override  
*Get the collider box object.*

### 3.29.1 Constructor & Destructor Documentation

#### 3.29.1.1 Triangle()

```
Primitive::Triangle::Triangle (
    const Math::Point3D & vertex1,
    const Math::Point3D & vertex2,
    const Math::Point3D & vertex3,
    std::shared_ptr< Material::IMaterial > material )
```

Construct a new triangle object.

## Parameters

<i>vertex1</i>	first edge of the triangle
<i>vertex2</i>	second edge of the triangle
<i>vertex3</i>	third edge of the triangle
<i>material</i>	of the triangle

## 3.29.2 Member Function Documentation

### 3.29.2.1 getColliderBox()

```
Optimisation::cubeCollider Primitive::Triangle::getColliderBox ( ) const [override], [virtual]
```

Get the collider box object.

## Returns

Octree::cubeCollider

Implements [Primitive::IPrimitive](#).

### 3.29.2.2 getMaterial()

```
std::shared_ptr< Material::IMaterial > Primitive::Triangle::getMaterial ( ) const [override], [virtual]
```

Get the material object.

## Returns

Material of triangle

Implements [Primitive::IPrimitive](#).

### 3.29.2.3 getNormal()

```
Math::Vector3D Primitive::Triangle::getNormal (
    const Math::Vector3D & hitPoint,
    const Raytracer::Ray & ray ) const [override], [virtual]
```

Get the normal of the object.

## Parameters

<i>hitPoint</i>	to compute the normal
<i>ray</i>	of the camera

## Returns

[Math::Vector3D](#)

Implements [Primitive::IPrimitive](#).

### 3.29.2.4 getTriangleInverseNormal()

```
Math::Vector3D Primitive::Triangle::getTriangleInverseNormal ( ) const
```

Get the triangle inverse normal object.

## Returns

[Math::Vector3D](#)

### 3.29.2.5 getTriangleNormal()

```
Math::Vector3D Primitive::Triangle::getTriangleNormal ( ) const
```

Get the triangle normal object.

## Returns

[Math::Vector3D](#)

### 3.29.2.6 getVertex1()

```
Math::Point3D Primitive::Triangle::getVertex1 ( ) const
```

Get the vertex1 object.

## Returns

[Math::Point3D](#)



### 3.29.2.7 getVertex2()

```
Math::Point3D Primitive::Triangle::getVertex2 ( ) const
```

Get the vertex2 object.

#### Returns

Math::Point3D

### 3.29.2.8 getVertex3()

```
Math::Point3D Primitive::Triangle::getVertex3 ( ) const
```

Get the vertex3 object.

#### Returns

Math::Point3D

### 3.29.2.9 hitPoint()

```
Math::Point3D Primitive::Triangle::hitPoint (
    const Raytracer::Ray & ray ) const [override], [virtual]
```

Return the hit point of the triangle.

#### Parameters

<i>ray</i>	vector3D
------------	----------

#### Returns

Point3D

Implements [Primitive::IPrimitive](#).

### 3.29.2.10 setMaterial()

```
void Primitive::Triangle::setMaterial (
    std::shared_ptr< Material::IMaterial > material )
```

Set the material.

## Parameters

<i>material</i>	New material to set
-----------------	---------------------

**3.29.2.11 setRotation()**

```
void Primitive::Triangle::setRotation (
    Math::Vector3D rotation )
```

Set the rotation object.

## Parameters

<i>rotation</i>	- Rotation value
-----------------	------------------

**3.29.2.12 setTriangleInverseNormal()**

```
void Primitive::Triangle::setTriangleInverseNormal (
    const Math::Vector3D & inverseNormal )
```

Set the triangle inverse normal object.

## Parameters

<i>inverseNormal</i>	new value to set
----------------------	------------------

**3.29.2.13 setTriangleNormal()**

```
void Primitive::Triangle::setTriangleNormal (
    const Math::Vector3D & normal )
```

Set the triangle normal object.

## Parameters

<i>normal</i>	new value to set
---------------	------------------

### 3.29.2.14 setVertex1()

```
void Primitive::Triangle::setVertex1 (
    const Math::Point3D & vertex1 )
```

Set the vertex1 object.

#### Parameters

<i>vertex1</i>	first edge of the triangle to set
----------------	-----------------------------------

### 3.29.2.15 setVertex2()

```
void Primitive::Triangle::setVertex2 (
    const Math::Point3D & vertex2 )
```

Set the vertex2 object.

#### Parameters

<i>vertex2</i>	second edge of the triangle to set
----------------	------------------------------------

### 3.29.2.16 setVertex3()

```
void Primitive::Triangle::setVertex3 (
    const Math::Point3D & vertex3 )
```

Set the vertex3 object.

#### Parameters

<i>vertex3</i>	third edge of the triangle to set
----------------	-----------------------------------

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

- include/Primitives/Triangle.hpp
- src/Plugins/Primitives/Triangle/Triangle.cpp

## 3.30 Math::Vector3D Class Reference

Vector 3D class (x, y, z)

```
#include <Vector3D.hpp>
```

## Public Member Functions

- [Vector3D](#) ()  
*Construct a new Vector 3D object.*
- [Vector3D](#) (double [x](#), double [y](#), double [z](#))  
*Construct a new Vector 3D object.*
- double [x](#) () const  
*return x coordinate vector*
- double [y](#) () const  
*return y coordinate vector*
- double [z](#) () const  
*return z coordinate vector*
- double [length](#) () const  
*return the lenght of the vector*
- double [length\\_squared](#) () const  
*return the square lenght of the vector*
- double [dot](#) (const [Vector3D](#) &ptr)  
*return the dot product with an other Vector*
- void [translate](#) (const [Vector3D](#) &ptr)  
*translate a vector with an other*
- void [rotateZ](#) (double degrees)  
*Rotate a vector with an angle on the axis Z.*
- void [rotateY](#) (double degrees)  
*Rotate a vector with an angle on the axis Y.*
- void [rotateX](#) (double degrees)  
*Rotate a vector with an angle on the axis X.*
- [Vector3D](#) [cross](#) (const [Vector3D](#) &ptr)  
*Return the cross product with the vector.*
- [Vector3D](#) [operator+](#) (const [Vector3D](#) &ptr)
- [Vector3D](#) & [operator+=](#) (const [Vector3D](#) &ptr)
- [Vector3D](#) [operator-](#) (const [Vector3D](#) &ptr)
- [Vector3D](#) & [operator-=](#) (const [Vector3D](#) &ptr)
- [Vector3D](#) [operator\\*](#) (const [Vector3D](#) &ptr)
- [Vector3D](#) & [operator\\*=](#) (const [Vector3D](#) &ptr)
- [Vector3D](#) [operator/](#) (const [Vector3D](#) &ptr)
- [Vector3D](#) & [operator/=](#) (const [Vector3D](#) &ptr)
- [Vector3D](#) [operator\\*](#) (double n)
- [Vector3D](#) & [operator\\*=](#) (double n)
- [Vector3D](#) [operator/](#) (double n)
- [Vector3D](#) & [operator/=](#) (double n)
- bool [operator==](#) (const [Vector3D](#) &ptr)
- bool [operator!=](#) (const [Vector3D](#) &ptr)
- bool [operator<](#) (const [Vector3D](#) &ptr)
- bool [operator>](#) (const [Vector3D](#) &ptr)

### 3.30.1 Detailed Description

Vector 3D class (x, y, z)

## 3.30.2 Constructor & Destructor Documentation

### 3.30.2.1 Vector3D()

```
Math::Vector3D::Vector3D (
    double x,
    double y,
    double z )
```

Construct a new Vector 3D object.

#### Parameters

<i>x</i>	position vector
<i>y</i>	position vector
<i>z</i>	position vector

## 3.30.3 Member Function Documentation

### 3.30.3.1 cross()

```
Math::Vector3D Math::Vector3D::cross (
    const Vector3D & ptr )
```

Return the cross product with the vector.

#### Parameters

<i>ptr</i>	within do the cross product
------------	-----------------------------

#### Returns

[Vector3D](#) cross product

### 3.30.3.2 dot()

```
double Math::Vector3D::dot (
    const Vector3D & ptr )
```

return the dot product with an other Vector

**Parameters**

<i>ptr</i>	vector to dot with
------------	--------------------

**Returns**

double result of dot product

**3.30.3.3 length()**

```
double Math::Vector3D::length ( ) const
```

return the lenght of the vector

**Returns**

double

**3.30.3.4 length\_squared()**

```
double Math::Vector3D::length_squared ( ) const
```

return the square lenght of the vector

**Returns**

double

**3.30.3.5 rotateX()**

```
void Math::Vector3D::rotateX (
    double degrees )
```

Rotate a vector with an angle on the axis X.

**Parameters**

<i>degrees</i>	- Rotation value
----------------	------------------

### 3.30.3.6 rotateY()

```
void Math::Vector3D::rotateY (
    double degrees )
```

Rotate a vector with an angle on the axis Y.

#### Parameters

<i>degrees</i>	- Rotation value
----------------	------------------

### 3.30.3.7 rotateZ()

```
void Math::Vector3D::rotateZ (
    double degrees )
```

Rotate a vector with an angle on the axis Z.

#### Parameters

<i>degrees</i>	- Rotation value
----------------	------------------

### 3.30.3.8 translate()

```
void Math::Vector3D::translate (
    const Vector3D & ptr )
```

translate a vector with an other

#### Parameters

<i>ptr</i>	vector of translation
------------	-----------------------

### 3.30.3.9 x()

```
double Math::Vector3D::x ( ) const
```

return x coordinate vector

#### Returns

double

**3.30.3.10 y()**

```
double Math::Vector3D::y ( ) const
```

return y coordinate vector

**Returns**

double

**3.30.3.11 z()**

```
double Math::Vector3D::z ( ) const
```

return z coordinate vector

**Returns**

double

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

- include/Math/Vector3D.hpp
- src/Math/Vector3D.cpp



# Index

- add
  - Light::LightsContainer, [43](#)
  - Primitive::PrimitivesContainer, [54](#)
- Ambient
  - Light::Ambient, [5](#)
- at
  - Raytracer::Ray, [58](#)
- build
  - Raytracer::CameraBuilder, [12](#)
- Camera
  - Raytracer::Camera, [9](#)
- Color, [14](#)
  - Color, [15](#)
  - getB, [16](#)
  - getG, [16](#)
  - getR, [16](#)
  - isWrongColor, [16](#)
  - setB, [16](#)
  - setG, [17](#)
  - setR, [17](#)
- computeColor
  - Light::Ambient, [6](#)
  - Light::Directional, [34](#)
  - Light::ILight, [39](#)
  - Light::LightsContainer, [43](#)
  - Light::Point, [52](#)
  - Primitive::PrimitivesContainer, [56](#)
- Cone
  - Primitive::Cone, [18](#)
- cross
  - Math::Vector3D, [83](#)
- Cube
  - Optimisation::Cube, [23](#)
- Cylinder
  - Primitive::Cylinder, [29](#)
- direction
  - Raytracer::Ray, [58](#)
- Directional
  - Light::Directional, [34](#)
- DLLoader
  - Raytracer::DLLoader, [37](#)
- dot
  - Math::Vector3D, [83](#)
- FlatColor, [39](#)
- getAngle
  - Primitive::Cone, [19](#)
- getAxis
  - Primitive::Cone, [19](#)
  - Primitive::Plane, [48](#)
- getB
  - Color, [16](#)
- getCamera
  - Raytracer::Scene, [68](#)
- getCollider
  - Optimisation::Cube, [24](#)
- getColliderBox
  - Primitive::Cone, [19](#)
  - Primitive::Cylinder, [30](#)
  - Primitive::IPrimitive, [41](#)
  - Primitive::Plane, [48](#)
  - Primitive::RectangularCuboid, [61](#)
  - Primitive::Sphere, [71](#)
  - Primitive::Triangle, [77](#)
- getColor
  - Light::Ambient, [6](#)
  - Light::Directional, [34](#)
  - Light::ILight, [39](#)
  - Light::Point, [52](#)
- getColorPoint
  - Primitive::PrimitivesContainer, [56](#)
- getDiffuseMultiplier
  - Light::Ambient, [7](#)
  - Light::Directional, [35](#)
  - Light::Point, [52](#)
- getDirection
  - Light::Directional, [35](#)
- getG
  - Color, [16](#)
- getInstance
  - Raytracer::DLLoader, [38](#)
- getLights
  - Raytracer::Scene, [69](#)
- getLightsList
  - Light::LightsContainer, [44](#)
- getMaterial
  - Primitive::Cone, [19](#)
  - Primitive::Cylinder, [30](#)
  - Primitive::IPrimitive, [41](#)
  - Primitive::Plane, [48](#)
  - Primitive::RectangularCuboid, [62](#)
  - Primitive::Sphere, [71](#)
  - Primitive::Triangle, [77](#)
- getMaxX
  - Optimisation::Cube, [24](#)
  - Primitive::RectangularCuboid, [62](#)

- getMaxY
  - Optimisation::Cube, [24](#)
  - Primitive::RectangularCuboid, [62](#)
- getMaxZ
  - Optimisation::Cube, [24](#)
  - Primitive::RectangularCuboid, [62](#)
- getMinX
  - Optimisation::Cube, [25](#)
  - Primitive::RectangularCuboid, [63](#)
- getMinY
  - Optimisation::Cube, [25](#)
  - Primitive::RectangularCuboid, [63](#)
- getMinZ
  - Optimisation::Cube, [25](#)
  - Primitive::RectangularCuboid, [63](#)
- getMultiplier
  - Light::Ambient, [7](#)
- getNormal
  - Primitive::Cone, [19](#)
  - Primitive::Cylinder, [30](#)
  - Primitive::IPrimitive, [42](#)
  - Primitive::Plane, [48](#)
  - Primitive::RectangularCuboid, [63](#)
  - Primitive::Sphere, [72](#)
  - Primitive::Triangle, [77](#)
- getOrigin
  - Primitive::Cone, [20](#)
  - Primitive::Sphere, [72](#)
- getPosition
  - Light::Directional, [35](#)
  - Light::Point, [52](#)
  - Primitive::Plane, [49](#)
- getPrimitives
  - Raytracer::Scene, [69](#)
- getPrimitivesContainer
  - Optimisation::Cube, [25](#)
- getPrimitivesHits
  - Optimisation::Cube, [26](#)
  - Optimisation::Octree, [45](#)
- getPrimitivesList
  - Primitive::PrimitivesContainer, [56](#)
- getR
  - Color, [16](#)
- getRadius
  - Primitive::Sphere, [72](#)
- getTriangleInverseNormal
  - Primitive::Triangle, [78](#)
- getTriangleNormal
  - Primitive::Triangle, [78](#)
- getType
  - Light::Ambient, [7](#)
  - Light::Directional, [35](#)
  - Light::ILight, [40](#)
  - Light::Point, [53](#)
  - Material::IMaterial, [40](#)
- getVertex1
  - Primitive::Triangle, [78](#)
- getVertex2
  - Primitive::Triangle, [78](#)
- getVertex3
  - Primitive::Triangle, [79](#)
- hitPoint
  - Primitive::Cone, [20](#)
  - Primitive::Cylinder, [31](#)
  - Primitive::IPrimitive, [42](#)
  - Primitive::Plane, [49](#)
  - Primitive::RectangularCuboid, [64](#)
  - Primitive::Sphere, [73](#)
  - Primitive::Triangle, [79](#)
- identifyPrimitives
  - Optimisation::Cube, [26](#)
- isShadow
  - Primitives::Shadow, [70](#)
- isWrongColor
  - Color, [16](#)
- length
  - Math::Vector3D, [84](#)
- length\_squared
  - Math::Vector3D, [84](#)
- Light::Ambient, [5](#)
  - Ambient, [5](#)
  - computeColor, [6](#)
  - getColor, [6](#)
  - getDiffuseMultiplier, [7](#)
  - getMultiplier, [7](#)
  - getType, [7](#)
  - setDiffuseMultiplier, [7](#)
  - setMultiplier, [8](#)
- Light::Directional, [33](#)
  - computeColor, [34](#)
  - Directional, [34](#)
  - getColor, [34](#)
  - getDiffuseMultiplier, [35](#)
  - getDirection, [35](#)
  - getPosition, [35](#)
  - getType, [35](#)
  - setColor, [36](#)
  - setDiffuseMultiplier, [36](#)
  - setDirection, [36](#)
  - setPosition, [37](#)
- Light::ILight, [39](#)
  - computeColor, [39](#)
  - getColor, [39](#)
  - getType, [40](#)
- Light::LightsContainer, [43](#)
  - add, [43](#)
  - computeColor, [43](#)
  - getLightsList, [44](#)
- Light::Point, [51](#)
  - computeColor, [52](#)
  - getColor, [52](#)
  - getDiffuseMultiplier, [52](#)
  - getPosition, [52](#)
  - getType, [53](#)

- Point, 51
- setColor, 53
- setDiffuseMultiplier, 53
- setPosition, 54
- Material::IMaterial, 40
  - getType, 40
- Math::Vector3D, 81
  - cross, 83
  - dot, 83
  - length, 84
  - length\_squared, 84
  - rotateX, 84
  - rotateY, 84
  - rotateZ, 85
  - translate, 85
  - Vector3D, 83
  - x, 85
  - y, 85
  - z, 86
- Octree
  - Optimisation::Octree, 45
- operator=
  - Raytracer::Camera, 9
- Optimisation::Cube, 22
  - Cube, 23
  - getCollider, 24
  - getMaxX, 24
  - getMaxY, 24
  - getMaxZ, 24
  - getMinX, 25
  - getMinY, 25
  - getMinZ, 25
  - getPrimitivesContainer, 25
  - getPrimitivesHits, 26
  - identifyPrimitives, 26
  - setCollider, 26
  - setMaxX, 27
  - setMaxY, 27
  - setMaxZ, 27
  - setMinX, 27
  - setMinY, 28
  - setMinZ, 28
- Optimisation::cubeCollider, 28
- Optimisation::Octree, 44
  - getPrimitivesHits, 45
  - Octree, 45
- origin
  - Raytracer::Ray, 58
- Plane
  - Primitive::Plane, 46
- Point
  - Light::Point, 51
- pointAt
  - Raytracer::Rectangle3D, 59
- Primitive::Cone, 17
  - Cone, 18
  - getAngle, 19
  - getAxis, 19
  - getColliderBox, 19
  - getMaterial, 19
  - getNormal, 19
  - getOrigin, 20
  - hitPoint, 20
  - setAngle, 20
  - setAxis, 21
  - setMaterial, 21
  - setOrigin, 21
  - setRotation, 21
- Primitive::Cylinder, 29
  - Cylinder, 29
  - getColliderBox, 30
  - getMaterial, 30
  - getNormal, 30
  - hitPoint, 31
  - setAxis, 31
  - setMaterial, 32
  - setOrigin, 32
  - setRadius, 32
  - setRotation, 33
- Primitive::IPrimitive, 41
  - getColliderBox, 41
  - getMaterial, 41
  - getNormal, 42
  - hitPoint, 42
- Primitive::Plane, 46
  - getAxis, 48
  - getColliderBox, 48
  - getMaterial, 48
  - getNormal, 48
  - getPosition, 49
  - hitPoint, 49
  - Plane, 46
  - setAxis, 49
  - setMaterial, 50
  - setPosition, 50
  - setRotation, 50
- Primitive::PrimitivesContainer, 54
  - add, 54
  - computeColor, 56
  - getColorPoint, 56
  - getPrimitivesList, 56
- Primitive::RectangularCuboid, 60
  - getColliderBox, 61
  - getMaterial, 62
  - getMaxX, 62
  - getMaxY, 62
  - getMaxZ, 62
  - getMinX, 63
  - getMinY, 63
  - getMinZ, 63
  - getNormal, 63
  - hitPoint, 64
  - RectangularCuboid, 61
  - setMaterial, 64

- setMaxX, 64
  - setMaxY, 65
  - setMaxZ, 65
  - setMinX, 65
  - setMinY, 65
  - setMinZ, 66
  - setRotation, 66
- Primitive::Sphere, 70
  - getColliderBox, 71
  - getMaterial, 71
  - getNormal, 72
  - getOrigin, 72
  - getRadius, 72
  - hitPoint, 73
  - setMaterial, 73
  - setOrigin, 73
  - setRadius, 75
  - setRotation, 75
- Sphere, 71
- Primitive::Triangle, 75
  - getColliderBox, 77
  - getMaterial, 77
  - getNormal, 77
  - getTriangleInverseNormal, 78
  - getTriangleNormal, 78
  - getVertex1, 78
  - getVertex2, 78
  - getVertex3, 79
  - hitPoint, 79
  - setMaterial, 79
  - setRotation, 80
  - setTriangleInverseNormal, 80
  - setTriangleNormal, 80
  - setVertex1, 80
  - setVertex2, 81
  - setVertex3, 81
- Triangle, 76
- Primitives::Shadow, 69
  - isShadow, 70
- Shadow, 69
- Ray
  - Raytracer::Ray, 57
- ray
  - Raytracer::Camera, 10
- Raytracer::Camera, 8
  - Camera, 9
  - operator=, 9
  - ray, 10
  - setFov, 10
  - setOrigin, 10
  - setResolution, 11
  - setRotation, 11
  - setScreen, 11
- Raytracer::CameraBuilder, 12
  - build, 12
  - setFov, 12
  - setOrigin, 13
  - setResolution, 13
  - setRotation, 13
  - setScreen, 14
- Raytracer::DLLoader, 37
  - DLLoader, 37
  - getInstance, 38
- Raytracer::Factory, 38
- Raytracer::Ray, 57
  - at, 58
  - direction, 58
  - origin, 58
  - Ray, 57
- Raytracer::Rectangle3D, 59
  - pointAt, 59
  - Rectangle3D, 59
- Raytracer::Renderer, 66
  - Renderer, 67
  - writeColor, 67
- Raytracer::Scene, 68
  - getCamera, 68
  - getLights, 69
  - getPrimitives, 69
  - Scene, 68
- Raytracer::Scene::ParseException, 45
- Rectangle3D
  - Raytracer::Rectangle3D, 59
- RectangularCuboid
  - Primitive::RectangularCuboid, 61
- Renderer
  - Raytracer::Renderer, 67
- rotateX
  - Math::Vector3D, 84
- rotateY
  - Math::Vector3D, 84
- rotateZ
  - Math::Vector3D, 85
- Scene
  - Raytracer::Scene, 68
- setAngle
  - Primitive::Cone, 20
- setAxis
  - Primitive::Cone, 21
  - Primitive::Cylinder, 31
  - Primitive::Plane, 49
- setB
  - Color, 16
- setCollider
  - Optimisation::Cube, 26
- setColor
  - Light::Directional, 36
  - Light::Point, 53
- setDiffuseMultiplier
  - Light::Ambient, 7
  - Light::Directional, 36
  - Light::Point, 53
- setDirection
  - Light::Directional, 36
- setFov
  - Raytracer::Camera, 10

- Raytracer::CameraBuilder, 12
- setG
  - Color, 17
- setMaterial
  - Primitive::Cone, 21
  - Primitive::Cylinder, 32
  - Primitive::Plane, 50
  - Primitive::RectangularCuboid, 64
  - Primitive::Sphere, 73
  - Primitive::Triangle, 79
- setMaxX
  - Optimisation::Cube, 27
  - Primitive::RectangularCuboid, 64
- setMaxY
  - Optimisation::Cube, 27
  - Primitive::RectangularCuboid, 65
- setMaxZ
  - Optimisation::Cube, 27
  - Primitive::RectangularCuboid, 65
- setMinX
  - Optimisation::Cube, 27
  - Primitive::RectangularCuboid, 65
- setMinY
  - Optimisation::Cube, 28
  - Primitive::RectangularCuboid, 65
- setMinZ
  - Optimisation::Cube, 28
  - Primitive::RectangularCuboid, 66
- setMultiplier
  - Light::Ambient, 8
- setOrigin
  - Primitive::Cone, 21
  - Primitive::Cylinder, 32
  - Primitive::Sphere, 73
  - Raytracer::Camera, 10
  - Raytracer::CameraBuilder, 13
- setPosition
  - Light::Directional, 37
  - Light::Point, 54
  - Primitive::Plane, 50
- setR
  - Color, 17
- setRadius
  - Primitive::Cylinder, 32
  - Primitive::Sphere, 75
- setResolution
  - Raytracer::Camera, 11
  - Raytracer::CameraBuilder, 13
- setRotation
  - Primitive::Cone, 21
  - Primitive::Cylinder, 33
  - Primitive::Plane, 50
  - Primitive::RectangularCuboid, 66
  - Primitive::Sphere, 75
  - Primitive::Triangle, 80
  - Raytracer::Camera, 11
  - Raytracer::CameraBuilder, 13
- setScreen
  - Raytracer::Camera, 11
  - Raytracer::CameraBuilder, 14
- setTriangleInverseNormal
  - Primitive::Triangle, 80
- setTriangleNormal
  - Primitive::Triangle, 80
- setVertex1
  - Primitive::Triangle, 80
- setVertex2
  - Primitive::Triangle, 81
- setVertex3
  - Primitive::Triangle, 81
- Shadow
  - Primitives::Shadow, 69
- Sphere
  - Primitive::Sphere, 71
- translate
  - Math::Vector3D, 85
- Triangle
  - Primitive::Triangle, 76
- Vector3D
  - Math::Vector3D, 83
- writeColor
  - Raytracer::Renderer, 67
- x
  - Math::Vector3D, 85
- y
  - Math::Vector3D, 85
- z
  - Math::Vector3D, 86