

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 ray()	10
3.2.2.2 setFov()	10
3.2.2.3 setOrigin()	10
3.2.2.4 setResolution()	10
3.2.2.5 setRotation()	11
3.2.2.6 setScreen()	11
3.3 Color Class Reference	11
3.3.1 Constructor & Destructor Documentation	12
3.3.1.1 Color() [1/2]	12
3.3.1.2 Color() [2/2]	12
3.3.2 Member Function Documentation	13
3.3.2.1 getB()	13
3.3.2.2 getG()	13
3.3.2.3 getR()	13
3.3.2.4 isWrongColor()	13
3.3.2.5 setB()	13
3.3.2.6 setG()	14
3.3.2.7 setR()	14
3.4 Primitive::Cone Class Reference	14
3.4.1 Constructor & Destructor Documentation	15

3.4.1.1 Cone()	15
3.4.2 Member Function Documentation	16
3.4.2.1 getAngle()	16
3.4.2.2 getAxis()	16
3.4.2.3 getMaterial()	16
3.4.2.4 getNormal()	16
3.4.2.5 getOrigin()	17
3.4.2.6 hitPoint()	17
3.4.2.7 setAngle()	17
3.4.2.8 setAxis()	18
3.4.2.9 setMaterial()	18
3.4.2.10 setOrigin()	18
3.4.2.11 setRotation()	19
3.5 Primitive::Cylinder Class Reference	19
3.5.1 Constructor & Destructor Documentation	20
3.5.1.1 Cylinder()	20
3.5.2 Member Function Documentation	20
3.5.2.1 getMaterial()	20
3.5.2.2 getNormal()	20
3.5.2.3 hitPoint()	21
3.5.2.4 setAxis() [1/2]	21
3.5.2.5 setAxis() [2/2]	21
3.5.2.6 setMaterial()	22
3.5.2.7 setOrigin()	22
3.5.2.8 setRadius()	22
3.5.2.9 setRotation()	23
3.6 Light::Directional Class Reference	23
3.6.1 Constructor & Destructor Documentation	24
3.6.1.1 Directional()	24
3.6.2 Member Function Documentation	24
3.6.2.1 computeColor()	24
3.6.2.2 getColor()	25
3.6.2.3 getDiffuseMultiplier()	25
3.6.2.4 getDirection()	25
3.6.2.5 getPosition()	25
3.6.2.6 getType()	26
3.6.2.7 setColor()	26
3.6.2.8 setDiffuseMultiplier()	26
3.6.2.9 setDirection()	26
3.6.2.10 setPosition()	27
3.7 Raytracer::DLLoader Class Reference	27
3.7.1 Constructor & Destructor Documentation	27

3.7.1.1 DLoader()	27
3.7.2 Member Function Documentation	28
3.7.2.1 getInstance()	28
3.8 Raytracer::Factory Class Reference	28
3.9 FlatColor Class Reference	29
3.10 Light::ILight Class Reference	29
3.10.1 Member Function Documentation	29
3.10.1.1 computeColor()	29
3.10.1.2 getColor()	30
3.10.1.3 getType()	30
3.11 Material::IMaterial Class Reference	30
3.11.1 Member Function Documentation	30
3.11.1.1 getType()	31
3.12 Primitive::IPrimitive Class Reference	31
3.12.1 Member Function Documentation	31
3.12.1.1 getMaterial()	31
3.12.1.2 getNormal()	31
3.12.1.3 hitPoint()	32
3.13 Light::LightsContainer Class Reference	32
3.13.1 Member Function Documentation	33
3.13.1.1 add()	33
3.13.1.2 computeColor()	33
3.13.1.3 getLightsList()	33
3.14 Raytracer::Scene::ParserException Class Reference	34
3.15 Primitive::Plane Class Reference	34
3.15.1 Constructor & Destructor Documentation	35
3.15.1.1 Plane()	35
3.15.2 Member Function Documentation	35
3.15.2.1 getAxis()	35
3.15.2.2 getMaterial()	35
3.15.2.3 getNormal()	35
3.15.2.4 getPosition()	36
3.15.2.5 hitPoint()	36
3.15.2.6 setAxis()	36
3.15.2.7 setMaterial()	37
3.15.2.8 setPosition()	37
3.15.2.9 setRotation()	37
3.16 Light::Point Class Reference	38
3.16.1 Constructor & Destructor Documentation	38
3.16.1.1 Point()	38
3.16.2 Member Function Documentation	39
3.16.2.1 computeColor()	39

3.16.2.2	getColor()	39
3.16.2.3	getDiffuseMultiplier()	39
3.16.2.4	getPosition()	40
3.16.2.5	getType()	40
3.16.2.6	setColor()	40
3.16.2.7	setDiffuseMultiplier()	40
3.16.2.8	setPosition()	41
3.17	Primitive::PrimitivesContainer Class Reference	41
3.17.1	Member Function Documentation	41
3.17.1.1	add()	42
3.17.1.2	computeColor()	43
3.17.1.3	getColorPoint()	43
3.17.1.4	getPrimitivesList()	44
3.18	Raytracer::Ray Class Reference	44
3.18.1	Detailed Description	44
3.18.2	Constructor & Destructor Documentation	44
3.18.2.1	Ray()	44
3.18.3	Member Function Documentation	45
3.18.3.1	at()	45
3.18.3.2	direction()	45
3.18.3.3	origin()	45
3.19	Raytracer::Rectangle3D Class Reference	46
3.19.1	Constructor & Destructor Documentation	46
3.19.1.1	Rectangle3D()	46
3.19.2	Member Function Documentation	46
3.19.2.1	pointAt()	46
3.20	Raytracer::Renderer Class Reference	47
3.20.1	Constructor & Destructor Documentation	47
3.20.1.1	Renderer()	47
3.20.2	Member Function Documentation	47
3.20.2.1	writeColor()	48
3.21	Raytracer::Scene Class Reference	48
3.21.1	Constructor & Destructor Documentation	48
3.21.1.1	Scene()	48
3.21.2	Member Function Documentation	49
3.21.2.1	getCamera()	49
3.21.2.2	getLights()	49
3.21.2.3	getPrimitives()	49
3.22	Primitives::Shadow Class Reference	50
3.22.1	Constructor & Destructor Documentation	50
3.22.1.1	Shadow()	50
3.22.2	Member Function Documentation	50

3.22.2.1 isShadow()	50
3.23 Primitive::Sphere Class Reference	51
3.23.1 Constructor & Destructor Documentation	51
3.23.1.1 Sphere()	51
3.23.2 Member Function Documentation	52
3.23.2.1 getMaterial()	52
3.23.2.2 getNormal()	52
3.23.2.3 getOrigin()	53
3.23.2.4 getRadius()	53
3.23.2.5 hitPoint()	53
3.23.2.6 setMaterial()	53
3.23.2.7 setOrigin()	54
3.23.2.8 setRadius()	54
3.23.2.9 setRotation()	54
3.24 Math::Vector3D Class Reference	55
3.24.1 Detailed Description	55
3.24.2 Constructor & Destructor Documentation	56
3.24.2.1 Vector3D()	56
3.24.3 Member Function Documentation	56
3.24.3.1 dot()	56
3.24.3.2 length()	56
3.24.3.3 length_squared()	57
3.24.3.4 rotateX()	57
3.24.3.5 rotateY()	57
3.24.3.6 rotateZ()	57
3.24.3.7 translate()	58
3.24.3.8 x()	58
3.24.3.9 y()	58
3.24.3.10 z()	58
<b>Index</b>	<b>59</b>





# Chapter 1

## Hierarchical Index

### 1.1 Class Hierarchy

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

Raytracer::Camera . . . . .	8
Color . . . . .	11
Raytracer::DLLoader . . . . .	27
std::exception	
Raytracer::Scene::ParserException . . . . .	34
Raytracer::Factory . . . . .	28
Light::ILight . . . . .	29
Light::Ambient . . . . .	5
Light::Directional . . . . .	23
Light::Point . . . . .	38
Material::IMaterial . . . . .	30
FlatColor . . . . .	29
Primitive::IPrimitive . . . . .	31
Primitive::Cone . . . . .	14
Primitive::Cylinder . . . . .	19
Primitive::Plane . . . . .	34
Primitive::Sphere . . . . .	51
Light::LightsContainer . . . . .	32
Primitive::PrimitivesContainer . . . . .	41
Raytracer::Ray . . . . .	44
Raytracer::Rectangle3D . . . . .	46
Raytracer::Renderer . . . . .	47
Raytracer::Scene . . . . .	48
Primitives::Shadow . . . . .	50
Math::Vector3D . . . . .	55



## 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
Color	11
Primitive::Cone	14
Primitive::Cylinder	19
Light::Directional	23
Raytracer::DLLoader	27
Raytracer::Factory	28
FlatColor	29
Light::ILight	29
Material::IMaterial	30
Primitive::IPrimitive	31
Light::LightsContainer	32
Raytracer::Scene::ParserException	34
Primitive::Plane	34
Light::Point	38
Primitive::PrimitivesContainer	41
Raytracer::Ray	
Ray class, (point and direction vectors)	44
Raytracer::Rectangle3D	46
Raytracer::Rendered	47
Raytracer::Scene	48
Primitives::Shadow	50
Primitive::Sphere	51
Math::Vector3D	
Vector 3D class (x, y, z)	55



## 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](#) ()  
*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 <code>Camera</code> 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 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

### 3.2.2.2 setFov()

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

Set the Fov object.

#### Parameters

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

### 3.2.2.3 setOrigin()

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

Set the Origin object.

#### Parameters

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

### 3.2.2.4 setResolution()

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

Set the Resolution object.

## Parameters

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

**3.2.2.5 setRotation()**

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

Set the Rotation object.

## Parameters

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

**3.2.2.6 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 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 ( $0 \leq rgb \leq 255$ ).*

### 3.3.1 Constructor & Destructor Documentation

#### 3.3.1.1 `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

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

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

Construct a new `Color` object form a Point3D.

##### Parameters

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

## 3.3.2 Member Function Documentation

### 3.3.2.1 getB()

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

Get the blue value.

#### Returns

double - blue value

### 3.3.2.2 getG()

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

Get the green value.

#### Returns

double - green value

### 3.3.2.3 getR()

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

Get the red value.

#### Returns

double - red value

### 3.3.2.4 isWrongColor()

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

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

#### Returns

true

false

### 3.3.2.5 setB()

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

Set the blue value.

**Parameters**

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

**3.3.2.6 setG()**

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

Set the green value.

**Parameters**

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

**3.3.2.7 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

## 3.4 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 override  
*Get the Normal of the object.*

### 3.4.1 Constructor & Destructor Documentation

#### 3.4.1.1 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>

## 3.4.2 Member Function Documentation

### 3.4.2.1 `getAngle()`

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

Get the Angle object.

#### Returns

Angle of [Cone](#)

### 3.4.2.2 `getAxis()`

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

Get the Axis object.

#### Returns

Axis of [Cone](#)

### 3.4.2.3 `getMaterial()`

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

Get the Material object.

#### Returns

Material of [Cone](#)

Implements [Primitive::!Primitive](#).

### 3.4.2.4 `getNormal()`

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

Get the Normal of the object.



## Parameters

<i>hitPoint</i>	to have the normal
-----------------	--------------------

## Returns

[Math::Vector3D](#)Implements [Primitive::IPrimitive](#).

### 3.4.2.5 getOrigin()

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

Get the Origin object.

## Returns

Origin of [Cone](#)

### 3.4.2.6 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](#).

### 3.4.2.7 setAngle()

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

Set the Angle.

**Parameters**

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

**3.4.2.8 setAxis()**

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

Set the Axis.

**Parameters**

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

**3.4.2.9 setMaterial()**

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

Set the Material.

**Parameters**

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

**3.4.2.10 setOrigin()**

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

Set the Origin object.

**Parameters**

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

### 3.4.2.11 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

## 3.5 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 override  
*Get the Normal of the object.*

### 3.5.1 Constructor & Destructor Documentation

#### 3.5.1.1 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

### 3.5.2 Member Function Documentation

#### 3.5.2.1 getMaterial()

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

Get the Material object.

##### Returns

Material of cylinder

Implements [Primitive::!Primitive](#).

#### 3.5.2.2 getNormal()

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

Get the Normal of the object.

##### Parameters

<i>hitPoint</i>	to have the normal
-----------------	--------------------

## Returns

[Math::Vector3D](#)

Implements [Primitive::IPrimitive](#).

### 3.5.2.3 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](#).

### 3.5.2.4 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
-------------	------------------------------------

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

```
void Primitive::Cylinder::setAxis (
    double axis )
```

Set axis of cylinder.

## Parameters

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

**3.5.2.6 setMaterial()**

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

Set the Material object.

## Parameters

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

**3.5.2.7 setOrigin()**

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

Set origin of cylinder.

## Parameters

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

**3.5.2.8 setRadius()**

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

Set radius of cylinder.

## Parameters

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

## 3.5.2.9 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

## 3.6 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*

### 3.6.1 Constructor & Destructor Documentation

#### 3.6.1.1 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

### 3.6.2 Member Function Documentation

#### 3.6.2.1 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](#).



### 3.6.2.2 getColor()

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

Get the [Color](#) object.

Returns

[Color](#)

Implements [Light::ILight](#).

### 3.6.2.3 getDiffuseMultiplier()

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

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

Returns

double

### 3.6.2.4 getDirection()

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

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

Returns

[Math::Vector3D](#) direction

### 3.6.2.5 getPosition()

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

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

Returns

[Math::Point3D](#) position

### 3.6.2.6 getType()

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

Get type of Light.

#### Returns

The type of the light

Implements [Light::Light](#).

### 3.6.2.7 setColor()

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

Set the [Color](#) object.

#### Parameters

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

### 3.6.2.8 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
--------------------------	---

### 3.6.2.9 setDirection()

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

Set the Direction object.

## Parameters

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

**3.6.2.10 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

**3.7 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**

**3.7.1 Constructor & Destructor Documentation****3.7.1.1 DLLoader()**

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

Construct a new [DLLoader](#) object.

## Parameters

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

### 3.7.2 Member Function Documentation

#### 3.7.2.1 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

## 3.8 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

## 3.9 FlatColor Class Reference

Inheritance diagram for FlatColor:

## 3.10 Light::ILight Class Reference

Inheritance diagram for Light::ILight:

### Public Member Functions

- virtual `~ILight()`=default  
*Destroy the `ILight` object.*
- virtual `Light::LightType getType()` const =0  
*Get type of Light.*
- virtual `Color getColor()` 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.*

### 3.10.1 Member Function Documentation

#### 3.10.1.1 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`.

### 3.10.1.2 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](#).

### 3.10.1.3 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

## 3.11 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.*

### 3.11.1 Member Function Documentation

### 3.11.1.1 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

## 3.12 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 =0  
*Get the Normal of the object.*
- virtual std::shared\_ptr< [Material::IMaterial](#) > [getMaterial](#) () const =0  
*Get the Material object.*

### 3.12.1 Member Function Documentation

#### 3.12.1.1 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::Sphere](#), [Primitive::Plane](#), [Primitive::Cylinder](#), and [Primitive::Cone](#).

#### 3.12.1.2 getNormal()

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

Get the Normal of the object.

## Parameters

<i>hitPoint</i>	to compute the normal
-----------------	-----------------------

## Returns

[Math::Vector3D](#)

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

### 3.12.1.3 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>	<a href="#">Vector3D</a>
------------	--------------------------

## Returns

[Math::Point3D](#)

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

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

- `include/Primitives/IPrimitive.hpp`

## 3.13 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.*



### 3.13.1 Member Function Documentation

#### 3.13.1.1 add()

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

Add a Light to the container.

##### Parameters

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

#### 3.13.1.2 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

#### 3.13.1.3 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

### 3.14 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

### 3.15 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 override  
*Get the normal of the object.*

## 3.15.1 Constructor & Destructor Documentation

### 3.15.1.1 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

## 3.15.2 Member Function Documentation

### 3.15.2.1 getAxis()

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

Get the Axis object.

#### Returns

Axis The axis of [Plane](#) Object

### 3.15.2.2 getMaterial()

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

Get the Material object.

#### Returns

Material of plane

Implements [Primitive::!Primitive](#).

### 3.15.2.3 getNormal()

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

Get the normal of the object.

## Parameters

<i>hitPoint</i>	to have the normal
-----------------	--------------------

## Returns

[Math::Vector3D](#)

Implements [Primitive::IPrimitive](#).

### 3.15.2.4 getPosition()

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

Get the Position object plane.

## Returns

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

### 3.15.2.5 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](#).

### 3.15.2.6 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
-------------	---

**3.15.2.7 setMaterial()**

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

Set the Material.

## Parameters

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

**3.15.2.8 setPosition()**

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

Set the Position object.

## Parameters

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

**3.15.2.9 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

## 3.16 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.*

### 3.16.1 Constructor & Destructor Documentation

#### 3.16.1.1 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

## 3.16.2 Member Function Documentation

### 3.16.2.1 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](#).

### 3.16.2.2 getColor()

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

Get the [Color](#) object.

#### Returns

[Color](#)

Implements [Light::ILight](#).

### 3.16.2.3 getDiffuseMultiplier()

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

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

#### Returns

double

### 3.16.2.4 getPosition()

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

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

#### Returns

Math::Point3D position

### 3.16.2.5 getType()

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

Get type of Light.

#### Returns

The type of the light

Implements [Light::ILight](#).

### 3.16.2.6 setColor()

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

Set the [Color](#) object.

#### Parameters

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

### 3.16.2.7 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
--------------------------	---

**3.16.2.8 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

**3.17 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  
*Compute the color pixel of a primitive's hitpoint.*

**3.17.1 Member Function Documentation**

### 3.17.1.1 add()

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

Add a Primitive to the container.

## Parameters

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

### 3.17.1.2 computeColor()

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

## Returns

Math::Point3D

### 3.17.1.3 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>	Math::Vector3D
<i>lights</i>	list of lights

## Returns

Color

### 3.17.1.4 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

## 3.18 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](#)*

### 3.18.1 Detailed Description

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

### 3.18.2 Constructor & Destructor Documentation

#### 3.18.2.1 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

### 3.18.3 Member Function Documentation

#### 3.18.3.1 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

#### 3.18.3.2 direction()

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

return the direction of the ray

## Returns

const Math::Vector3D&

#### 3.18.3.3 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

## 3.19 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.*

### 3.19.1 Constructor & Destructor Documentation

#### 3.19.1.1 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

### 3.19.2 Member Function Documentation

#### 3.19.2.1 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

## 3.20 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.*

### 3.20.1 Constructor & Destructor Documentation

#### 3.20.1.1 [Renderer\(\)](#)

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

Construct a new [Renderer](#) object.

#### Parameters

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

### 3.20.2 Member Function Documentation

### 3.20.2.1 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

## 3.21 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.*

### 3.21.1 Constructor & Destructor Documentation

#### 3.21.1.1 Scene()

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

Construct a new [Scene](#) object.



## Parameters

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

## 3.21.2 Member Function Documentation

### 3.21.2.1 getCamera()

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

Get the [Camera](#) object.

## Returns

[Camera](#) Object

### 3.21.2.2 getLights()

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

Get the Lights object.

## Returns

[Light::LightsContainer](#)

### 3.21.2.3 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

## 3.22 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.*

### 3.22.1 Constructor & Destructor Documentation

#### 3.22.1.1 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
-------------------	--------------------

### 3.22.2 Member Function Documentation

#### 3.22.2.1 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

## 3.23 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 override  
*Get the Normal of the object.*

### 3.23.1 Constructor & Destructor Documentation

#### 3.23.1.1 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>

### 3.23.2 Member Function Documentation

#### 3.23.2.1 `getMaterial()`

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

Get the Material object.

## Returns

Material of sphere

Implements [Primitive::IPrimitive](#).

#### 3.23.2.2 `getNormal()`

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

Get the Normal of the object.

## Parameters

<i>hitPoint</i>	to compute the normal
-----------------	-----------------------

## Returns

[Math::Vector3D](#)

Implements [Primitive::IPrimitive](#).

### 3.23.2.3 getOrigin()

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

Get the Origin object.

#### Returns

Origin of sphere

### 3.23.2.4 getRadius()

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

Get the Origin object.

#### Returns

Radius of sphere

### 3.23.2.5 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](#).

### 3.23.2.6 setMaterial()

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

Set the Material.

## Parameters

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

**3.23.2.7 setOrigin()**

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

Set the Origin object.

## Parameters

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

**3.23.2.8 setRadius()**

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

Set the Radius.

## Parameters

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

**3.23.2.9 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

## 3.24 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](#) [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)

### 3.24.1 Detailed Description

Vector 3D class (x, y, z)

### 3.24.2 Constructor & Destructor Documentation

#### 3.24.2.1 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

### 3.24.3 Member Function Documentation

#### 3.24.3.1 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

#### 3.24.3.2 length()

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

return the lenght of the vector

##### Returns

double



### 3.24.3.3 length\_squared()

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

return the square lenght of the vector

#### Returns

double

### 3.24.3.4 rotateX()

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

Rotate a vector with an angle on the axis X.

#### Parameters

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

### 3.24.3.5 rotateY()

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

Rotate a vector with an angle on the axis Y.

#### Parameters

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

### 3.24.3.6 rotateZ()

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

Rotate a vector with an angle on the axis Z.

#### Parameters

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

### 3.24.3.7 translate()

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

translate a vector with an other

#### Parameters

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

### 3.24.3.8 x()

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

return x coordinate vector

#### Returns

double

### 3.24.3.9 y()

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

return y coordinate vector

#### Returns

double

### 3.24.3.10 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, 33
  - Primitive::PrimitivesContainer, 41
- Ambient
  - Light::Ambient, 5
- at
  - Raytracer::Ray, 45
- Camera
  - Raytracer::Camera, 9
- Color, 11
  - Color, 12
  - getB, 13
  - getG, 13
  - getR, 13
  - isWrongColor, 13
  - setB, 13
  - setG, 14
  - setR, 14
- computeColor
  - Light::Ambient, 6
  - Light::Directional, 24
  - Light::ILight, 29
  - Light::LightsContainer, 33
  - Light::Point, 39
  - Primitive::PrimitivesContainer, 43
- Cone
  - Primitive::Cone, 15
- Cylinder
  - Primitive::Cylinder, 20
- direction
  - Raytracer::Ray, 45
- Directional
  - Light::Directional, 24
- DLLoader
  - Raytracer::DLLoader, 27
- dot
  - Math::Vector3D, 56
- FlatColor, 29
- getAngle
  - Primitive::Cone, 16
- getAxis
  - Primitive::Cone, 16
  - Primitive::Plane, 35
- getB
  - Color, 13
- getCamera
  - Raytracer::Scene, 49
- getColor
  - Light::Ambient, 6
  - Light::Directional, 24
  - Light::ILight, 29
  - Light::Point, 39
- getColorPoint
  - Primitive::PrimitivesContainer, 43
- getDiffuseMultiplier
  - Light::Ambient, 7
  - Light::Directional, 25
  - Light::Point, 39
- getDirection
  - Light::Directional, 25
- getG
  - Color, 13
- getInstance
  - Raytracer::DLLoader, 28
- getLights
  - Raytracer::Scene, 49
- getLightsList
  - Light::LightsContainer, 33
- getMaterial
  - Primitive::Cone, 16
  - Primitive::Cylinder, 20
  - Primitive::IPrimitive, 31
  - Primitive::Plane, 35
  - Primitive::Sphere, 52
- getMultiplier
  - Light::Ambient, 7
- getNormal
  - Primitive::Cone, 16
  - Primitive::Cylinder, 20
  - Primitive::IPrimitive, 31
  - Primitive::Plane, 35
  - Primitive::Sphere, 52
- getOrigin
  - Primitive::Cone, 17
  - Primitive::Sphere, 52
- getPosition
  - Light::Directional, 25
  - Light::Point, 39
  - Primitive::Plane, 36
- getPrimitives
  - Raytracer::Scene, 49
- getPrimitivesList
  - Primitive::PrimitivesContainer, 43
- getR
  - Color, 13

- getRadius
  - Primitive::Sphere, 53
- getType
  - Light::Ambient, 7
  - Light::Directional, 25
  - Light::ILight, 30
  - Light::Point, 40
  - Material::IMaterial, 30
- hitPoint
  - Primitive::Cone, 17
  - Primitive::Cylinder, 21
  - Primitive::IPrimitive, 32
  - Primitive::Plane, 36
  - Primitive::Sphere, 53
- isShadow
  - Primitives::Shadow, 50
- isWrongColor
  - Color, 13
- length
  - Math::Vector3D, 56
- length\_squared
  - Math::Vector3D, 56
- Light::Ambient, 5
  - Ambient, 5
  - computeColor, 6
  - getColor, 6
  - getDiffuseMultiplier, 7
  - getMultiplier, 7
  - getType, 7
  - setDiffuseMultiplier, 7
  - setMultiplier, 8
- Light::Directional, 23
  - computeColor, 24
  - Directional, 24
  - getColor, 24
  - getDiffuseMultiplier, 25
  - getDirection, 25
  - getPosition, 25
  - getType, 25
  - setColor, 26
  - setDiffuseMultiplier, 26
  - setDirection, 26
  - setPosition, 27
- Light::ILight, 29
  - computeColor, 29
  - getColor, 29
  - getType, 30
- Light::LightsContainer, 32
  - add, 33
  - computeColor, 33
  - getLightsList, 33
- Light::Point, 38
  - computeColor, 39
  - getColor, 39
  - getDiffuseMultiplier, 39
  - getPosition, 39
  - getType, 40
  - Point, 38
  - setColor, 40
  - setDiffuseMultiplier, 40
  - setPosition, 41
- Material::IMaterial, 30
  - getType, 30
- Math::Vector3D, 55
  - dot, 56
  - length, 56
  - length\_squared, 56
  - rotateX, 57
  - rotateY, 57
  - rotateZ, 57
  - translate, 58
  - Vector3D, 56
  - x, 58
  - y, 58
  - z, 58
- origin
  - Raytracer::Ray, 45
- Plane
  - Primitive::Plane, 35
- Point
  - Light::Point, 38
- pointAt
  - Raytracer::Rectangle3D, 46
- Primitive::Cone, 14
  - Cone, 15
  - getAngle, 16
  - getAxis, 16
  - getMaterial, 16
  - getNormal, 16
  - getOrigin, 17
  - hitPoint, 17
  - setAngle, 17
  - setAxis, 18
  - setMaterial, 18
  - setOrigin, 18
  - setRotation, 18
- Primitive::Cylinder, 19
  - Cylinder, 20
  - getMaterial, 20
  - getNormal, 20
  - hitPoint, 21
  - setAxis, 21
  - setMaterial, 22
  - setOrigin, 22
  - setRadius, 22
  - setRotation, 22
- Primitive::IPrimitive, 31
  - getMaterial, 31
  - getNormal, 31
  - hitPoint, 32
- Primitive::Plane, 34
  - getAxis, 35

- getMaterial, 35
- getNormal, 35
- getPosition, 36
- hitPoint, 36
- Plane, 35
- setAxis, 36
- setMaterial, 37
- setPosition, 37
- setRotation, 37
- Primitive::PrimitivesContainer, 41
  - add, 41
  - computeColor, 43
  - getColorPoint, 43
  - getPrimitivesList, 43
- Primitive::Sphere, 51
  - getMaterial, 52
  - getNormal, 52
  - getOrigin, 52
  - getRadius, 53
  - hitPoint, 53
  - setMaterial, 53
  - setOrigin, 54
  - setRadius, 54
  - setRotation, 54
  - Sphere, 51
- Primitives::Shadow, 50
  - isShadow, 50
  - Shadow, 50
- Ray
  - Raytracer::Ray, 44
- ray
  - Raytracer::Camera, 9
- Raytracer::Camera, 8
  - Camera, 9
  - ray, 9
  - setFov, 10
  - setOrigin, 10
  - setResolution, 10
  - setRotation, 11
  - setScreen, 11
- Raytracer::DLLoader, 27
  - DLLoader, 27
  - getInstance, 28
- Raytracer::Factory, 28
- Raytracer::Ray, 44
  - at, 45
  - direction, 45
  - origin, 45
  - Ray, 44
- Raytracer::Rectangle3D, 46
  - pointAt, 46
  - Rectangle3D, 46
- Raytracer::Renderer, 47
  - Renderer, 47
  - writeColor, 47
- Raytracer::Scene, 48
  - getCamera, 49
  - getLights, 49
  - getPrimitives, 49
  - Scene, 48
- Raytracer::Scene::ParseException, 34
- Rectangle3D
  - Raytracer::Rectangle3D, 46
- Renderer
  - Raytracer::Renderer, 47
- rotateX
  - Math::Vector3D, 57
- rotateY
  - Math::Vector3D, 57
- rotateZ
  - Math::Vector3D, 57
- Scene
  - Raytracer::Scene, 48
- setAngle
  - Primitive::Cone, 17
- setAxis
  - Primitive::Cone, 18
  - Primitive::Cylinder, 21
  - Primitive::Plane, 36
- setB
  - Color, 13
- setColor
  - Light::Directional, 26
  - Light::Point, 40
- setDiffuseMultiplier
  - Light::Ambient, 7
  - Light::Directional, 26
  - Light::Point, 40
- setDirection
  - Light::Directional, 26
- setFov
  - Raytracer::Camera, 10
- setG
  - Color, 14
- setMaterial
  - Primitive::Cone, 18
  - Primitive::Cylinder, 22
  - Primitive::Plane, 37
  - Primitive::Sphere, 53
- setMultiplier
  - Light::Ambient, 8
- setOrigin
  - Primitive::Cone, 18
  - Primitive::Cylinder, 22
  - Primitive::Sphere, 54
  - Raytracer::Camera, 10
- setPosition
  - Light::Directional, 27
  - Light::Point, 41
  - Primitive::Plane, 37
- setR
  - Color, 14
- setRadius
  - Primitive::Cylinder, 22
  - Primitive::Sphere, 54
- setResolution

- Raytracer::Camera, [10](#)
- setRotation
  - Primitive::Cone, [18](#)
  - Primitive::Cylinder, [22](#)
  - Primitive::Plane, [37](#)
  - Primitive::Sphere, [54](#)
  - Raytracer::Camera, [11](#)
- setScreen
  - Raytracer::Camera, [11](#)
- Shadow
  - Primitives::Shadow, [50](#)
- Sphere
  - Primitive::Sphere, [51](#)
- translate
  - Math::Vector3D, [58](#)
- Vector3D
  - Math::Vector3D, [56](#)
- writeColor
  - Raytracer::Renderer, [47](#)
- x
  - Math::Vector3D, [58](#)
- y
  - Math::Vector3D, [58](#)
- z
  - Math::Vector3D, [58](#)