Raytracer

Generated by Doxygen 1.9.1

Chapter 1

Hierarchical Index

1.1 Class Hierarchy

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

Chapter 2

Class Index

2.1 Class List

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

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 ambiant 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

multiplier	Multipler of ambient light
diffuseMultiplier	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 ambiant light

Parameters

primitiveNormal	normal to the hitpoint
hitPoint	hitpoint
color	color
shadow	Primitive::Shadow class to handle shadows

Returns

Color color

Implements Light::ILight.

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

diffuseMultiplier	New diffuse multiplier to set

3.1.2.7 setMultiplier()

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

Set the Multiplier object.

Parameters

multiplier	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.
- int getAntialiasing (void) const

  Get the antialiasing object.
- 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.
- void setAntialiasing (int antialiasing)

  Set the antialiasing 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

other	other Camera 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

origin	camera's origin point
screen	of camera

3.2.2 Member Function Documentation

3.2.2.1 getAntialiasing()

```
int Raytracer::Camera::getAntialiasing (
             void  ) const
```

Get the antialiasing object.

Returns

int Value of antialiasing

3.2.2.2 operator=()

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

Construct a new Camera object by another one.

Parameters

other	Camera object to duplicate

Returns

Camera& The new Camera object

3.2.2.3 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

u	location u in rectangle
v	location v in rectangle

3.2.2.4 setAntialiasing()

```
void Raytracer::Camera::setAntialiasing (
             int antialiasing )
```

Set the antialiasing object.

Parameters

antialiasing	value of antialiasing

3.2.2.5 setFov()

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

Set the Fov object.

Parameters

fov	Fov to set

3.2.2.6 setOrigin()

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

Set the Origin object.

Parameters

origin	Origin to set

3.2.2.7 setResolution()

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

Set the Resolution object.

Parameters

width	of image
height	of image

3.2.2.8 setRotation()

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

Set the Rotation object.

Parameters

rotation	Rotation Vector to set

3.2.2.9 setScreen()

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

Set the Screen object.

Parameters

screen	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.
- Raytracer::CameraBuilder & setAntialiasing (int antialiasing)

    Set the antialiasing 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 setAntialiasing()

```
Raytracer::CameraBuilder & Raytracer::CameraBuilder::setAntialiasing (
              int antialiasing )
```

Set the antialiasing object.

Parameters

antialiasing	Value of antialiasing

Returns

CameraBuilder& object to chain the setter

3.3.1.3 setFov()

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

Set the Fov object.

Parameters

fov	Fov to set

Returns

CameraBuilder& object to chain the setter

3.3.1.4 setOrigin()

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

Set the Origin object.

Parameters

origin	Origin to set

Returns

CameraBuilder& object to chain the setter

3.3.1.5 setResolution()

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

Set the Resolution object.

Parameters

width	of image
height	of image

Returns

CameraBuilder& object to chain the setter

3.3.1.6 setRotation()

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

Set the Rotation object.

Parameters

rotation	Rotation Vector to set

Returns

CameraBuilder& object to chain the setter

3.3.1.7 setScreen()

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

Set the Screen object.

Parameters

screen	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 <= rgb <= 255).
- Color & **operator+=** (const Color &other)
- Color & **operator/=** (float value)

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

r	red value
g	green value
b	blue value

3.4.1.2 Color() [2/2]

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

Construct a new Color object form a Point3D.

Parameters

color	

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 <= rgb <= 255).

Returns

true

false

3.4.2.5 setB()

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

Set the blue value.

Parameters

b	blue value

3.4.2.6 setG()

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

Set the green value.

Parameters

g	green value

3.4.2.7 setR()

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

Set the red value.

Parameters

r	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) 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

origin	center of the Cone
radius	of the Cone
axis	of the Cone
material	Material of Cone

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

hitPoint	to have the normal
ray	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 )  [override], [virtual]
```

return the hit point of the Cone.

Parameters

ray	vector3D

Returns

Point3D

Implements Primitive::IPrimitive.

3.5.2.8 setAngle()

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

Set the Angle.

Parameters

angle	New angle to set

3.5.2.9 setAxis()

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

Set the Axis.

Parameters

axis	New axis to set

3.5.2.10 setMaterial()

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

Set the Material.

Parameters

material	New material to set

3.5.2.11 setOrigin()

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

Set the Origin object.

Parameters

origin	New origin to set

3.5.2.12 setRotation()

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

Set the Rotation object.

Parameters

rotation	- 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)

  Indentify 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

minX	minX position
minY	minY position
minZ	minZ position
maxX	maxX position
maxY	maxY position
maxZ	maxZ position
primitives	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

collider	collider of the cube.
primitives	primitives to check.
nbRecursions	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

ray	ray to check.

Returns

PrimitivesContainer

3.6.2.10 identifyPrimitives()

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

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

Parameters

primitives	primitives to check.

3.6.2.11 setCollider()

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

Set the collider of the cube.

Parameters

collider	collider to set.

3.6.2.12 setMaxX()

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

Set the max X value.

Parameters

maxX	

3.6.2.13 setMaxY()

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

Set the max Y value.

Parameters

maxY	

3.6.2.14 setMaxZ()

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

Set the max Z value.

Parameters

maxZ	

3.6.2.15 setMinX()

```
void Cube::setMinX (
            double minX )
```

Set the min X value.

Parameters

minX	

3.6.2.16 setMinY()

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

Set the min Y value.

Parameters

minY	

3.6.2.17 setMinZ()

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

Set the min Z value.

Parameters

minZ	

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) 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

origin	center of the cylinder
radius	of the cylinder
axis	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

hitPoint	to have the normal
ray	of the camera

Returns

Math::Vector3D

Implements Primitive::IPrimitive.

3.8.2.4 hitPoint()

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

Return the hit point of the cylinder.

Parameters

ray	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

axis	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

axis	double

3.8.2.7 setMaterial()

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

Set the Material object.

Parameters

material	Material of cylinder

3.8.2.8 setOrigin()

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

Set origin of cylinder.

Parameters

hitPoint	Point3D

3.8.2.9 setRadius()

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

Set radius of cylinder.

Parameters

radius	double

3.8.2.10 setRotation()

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

Set the Rotation object.

Parameters

rotation	- 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

position	Position of Directionnal Light
direction	Direction of Directionnal Light
diffuseMultiplier	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

primitiveNormal	normal to the hitpoint
hitPoint	hitpoint
color	color
shadow	Primitive::Shadow class to handle shadows

Returns

Math::Point3D color

Implements Light::ILight.

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

rgb	color

3.9.2.8 setDiffuseMultiplier()

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

Set the Diffuse Multiplier object.

Parameters

diffuseMultiplier	New Diffuse Multiplier of Directional Light

3.9.2.9 setDirection()

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

Set the Direction object.

Parameters

direction	New Direction of Directional Light

3.9.2.10 setPosition()

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

Set the Position object.

Parameters

position	New position of Directional 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

libraryPath	

3.10.2 Member Function Documentation

3.10.2.1 getInstance()

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

Get the Instance object.

Parameters

functionName	

Returns

T

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

- include/Parser/DLLoader.hpp
- src/Core/Parser/DLLoader.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

primitiveNormal	normal to the hitpoint
hitPoint	hitpoint
color	color
shadow	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)=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::Mesh, 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::Mesh, 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

hitPoint	to compute the normal
ray	of the camera

Returns

Math::Vector3D

Implemented in Primitive::Triangle, Primitive::Sphere, Primitive::RectangularCuboid, Primitive::Plane, Primitive::Cylinder, Primitive::Cone, and Primitive::Mesh.

3.15.1.4 hitPoint()

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

compute the hit point of a primitive with a ray

Parameters

ray	Vector3D

Returns

Math::Point3D

Implemented in Primitive::Triangle, Primitive::Sphere, Primitive::RectangularCuboid, Primitive::Plane, Primitive::Mesh, Primitive::Cylinder, and Primitive::Cone.

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

- include/Primitives/IPrimitive.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

Light	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

primitiveNormal	normal to the hitpoint
hitPoint	hitpoint
color	color
shadow	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 Primitive::Mesh Class Reference

Inheritance diagram for Primitive::Mesh:

Collaboration diagram for Primitive::Mesh:

Public Member Functions

- Mesh ()

  Construct a new mesh object.
- ∼Mesh ()=default

  Destroy the mesh object.
- Math::Point3D hitPoint (const Raytracer::Ray &ray) override

  Compute the hit point of a Mesh with a ray.
- Math::Vector3D getNormal (const Math::Vector3D &hitPoint, const Raytracer::Ray &ray) const

  Get the normal of the object.
- void setMaterial (std::shared_ptr< Material::IMaterial > material)

  Set the material.
- std::shared_ptr< Material::IMaterial > getMaterial () const

  Get the material object.
- Optimisation::cubeCollider getColliderBox () const override

  Get the collider box object.
- void add (std::shared_ptr< Primitive::IPrimitive > primitive)

  Add a primitive to the mesh.

3.17.1 Member Function Documentation

3.17.1.1 add()

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

Add a primitive to the mesh.

Parameters

primitive	to add

3.17.1.2 getColliderBox()

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

Get the collider box object.

Returns

Octree::cubeCollider

Implements Primitive::IPrimitive.

3.17.1.3 getMaterial()

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

Get the material object.

Returns

std::shared_ptr<Material::IMaterial>

Implements Primitive::IPrimitive.

3.17.1.4 getNormal()

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

Get the normal of the object.

Parameters

hitPoint	to compute the normal
ray	of the camera

Returns

Math::Vector3D

Implements Primitive::IPrimitive.

3.17.1.5 hitPoint()

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

Compute the hit point of a Mesh with a ray.

Parameters

ray	Vector3D

Returns

Math::Point3D

Implements Primitive::IPrimitive.

3.17.1.6 setMaterial()

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

Set the material.

Parameters

material	New material to set

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

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

3.18 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.

3.18.1 Constructor & Destructor Documentation

3.18.1.1 Octree()

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

Construct a new Octree object with a list of primitives.

Parameters

primitives	primitives to study.
cubeCollider	collider of the cube.

3.18.2 Member Function Documentation

3.18.2.1 getPrimitivesHits()

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

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

Parameters

ray	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

3.19 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

3.20 Raytracer::ParserObj Class Reference

Classes

- class ParserObjException

  Class exception for the ParserObj.

Public Member Functions

- ParserObj (const std::string &filePath, std::shared_ptr< Primitive::Mesh > mesh)

  Construct a new parser obj object.
- ∼ParserObj ()=default

  Destroy the parser obj object.
- bool isVertex (std::string line)

  Check if a string start by 'v'.
- bool isFace (std::string line)

  Check if a string start by 'f'.
- void readVertex (std::string line)

  Read a vertex line.
- std::tuple< double, double, double, bool > parseVertex (std::string line)

  Parse a vertex line.
- void readFace (std::string line)

  Read a face line.
- std::tuple< int, int, int, bool > parseFace (std::string line)

  Parse a face line.
- void createMesh (void)

  Create a Mesh object with the parsing done.

3.20.1 Constructor & Destructor Documentation

3.20.1.1 ParserObj()

```
Raytracer::ParserObj::ParserObj (
             const std::string & filePath,
             std::shared_ptr< Primitive::Mesh > mesh )
```

Construct a new parser obj object.

Parameters

filePath	path to the OBJ file
mesh	to create with the file

3.20.2 Member Function Documentation

3.20.2.1 isFace()

```
bool Raytracer::ParserObj::isFace (
            std::string line )
```

Check if a string start by 'f'.

Parameters

line	to check

Returns

true

false

3.20.2.2 isVertex()

```
bool Raytracer::ParserObj::isVertex (
            std::string line )
```

Check if a string start by 'v'.

Parameters

line	to check

Returns

true

false

3.20.2.3 parseFace()

```
std::tuple< int, int, int, bool > Raytracer::ParserObj::parseFace (
            std::string line )
```

Parse a face line.

Parameters

line	to parse

Returns

std::tuple<int, int, int, bool> line parsed

3.20.2.4 parseVertex()

```
std::tuple< double, double, double, bool > Raytracer::ParserObj::parseVertex (
            std::string line )
```

Parse a vertex line.

Parameters

line	to parse

Returns

std::tuple<double, double, double, bool> line parsed

3.20.2.5 readFace()

```
void Raytracer::ParserObj::readFace (
            std::string line )
```

Read a face line.

Parameters

line	to read

3.20.2.6 readVertex()

```
void Raytracer::ParserObj::readVertex (
            std::string line )
```

Read a vertex line.

Parameters

line	to read

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

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

3.21 Raytracer::ParserObj::ParserObjException Class Reference

Class exception for the ParserObj.

```
#include <ParserObj.hpp>
```

Inheritance diagram for Raytracer::ParserObj::ParserObjException:

Collaboration diagram for Raytracer::ParserObj::ParserObjException:

Public Member Functions

- ParserObjException (const std::string &msg)
  Exception to throw when there is an error in parsing.
- virtual const char ∗ what () const noexcept override
  Return the error message.

3.21.1 Detailed Description

Class exception for the ParserObj.

3.21.2 Constructor & Destructor Documentation

3.21.2.1 ParserObjException()

```
Raytracer::ParserObj::ParserObjException::ParserObjException (
            const std::string & msg )  [inline], [explicit]
```

Exception to throw when there is an error in parsing.

Parameters

msg	to throw

3.21.3 Member Function Documentation

3.21.3.1 what()

```
virtual const char* Raytracer::ParserObj::ParserObjException::what ( ) const  [inline], [override],
[virtual], [noexcept]
```

Return the error message.

Returns

const char∗ message

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

- include/Parser/ParserObj.hpp

3.22 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) 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.

3.22.1 Constructor & Destructor Documentation

3.22.1.1 Plane()

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

Construct a new Plane object.

Parameters

axis	Axis of the plane
position	offset on axis

3.22.2 Member Function Documentation

3.22.2.1 getAxis()

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

Get the Axis object.

Returns

Axis The axis of Plane Object

3.22.2.2 getColliderBox()

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

Get the collider box object.

Returns

Octree::cubeCollider

Implements Primitive::IPrimitive.

3.22.2.3 getMaterial()

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

Get the Material object.

Returns

Material of plane

Implements Primitive::IPrimitive.

3.22.2.4 getNormal()

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

Get the normal of the object.

Parameters

hitPoint	to have the normal
ray	of the camera

Returns

Math::Vector3D

Implements Primitive::IPrimitive.

3.22.2.5 getPosition()

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

Get the Position object plane.

Returns

Math::Point3D Position of Plane Object

3.22.2.6 hitPoint()

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

Return the hit point of the plane.

Parameters

ray	ray to check vector3D

Returns

Point3D

Implements Primitive::IPrimitive.

3.22.2.7 setAxis()

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

Set the Axis object.

Parameters

axis	The axis of Plane Object to set

3.22.2.8 setMaterial()

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

Set the Material.

Parameters

material	New material to set

3.22.2.9 setPosition()

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

Set the Position object.

Parameters

position	Position to set

3.22.2.10 setRotation()

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

Set the Rotation object.

Parameters

rotation	- Rotation value

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

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

3.23 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.

3.23.1 Constructor & Destructor Documentation

3.23.1.1 Point()

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

Construct a new Point object.

Parameters

position	Position of Point Light
diffuseMultiplier	Diffuse multiplier of Point Light

3.23.2 Member Function Documentation

3.23.2.1 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

primitiveNormal	normal to the hitpoint
hitPoint	hitpoint
color	color
shadow	Primitive::Shadow class to handle shadows

Returns

Math::Point3D color

Implements Light::ILight.

3.23.2.2 getColor()

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

Get the Color object.

Returns

Color

Implements Light::ILight.

3.23.2.3 getDiffuseMultiplier()

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

Get the Diffuse Multiplier number of Point light.

Returns

double

3.23.2.4 getPosition()

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

Get the Position number of Point light.

Returns

Math::Point3D position

3.23.2.5 getType()

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

Get type of Light.

Returns

The type of the light

Implements Light::ILight.

3.23.2.6 setColor()

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

Set the Color object.

Parameters

rgb	color

3.23.2.7 setDiffuseMultiplier()

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

Set the Diffuse Multiplier object.

Parameters

diffuseMultiplier	New Diffuse Multiplier of Point Light

3.23.2.8 setPosition()

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

Set the Position object.

Parameters

position	New position of Point Light

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

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

3.24 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.

3.24.1 Member Function Documentation

3.24.1.1 add()

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

Add a Primitive to the container.

Parameters

primitive	to add

3.24.1.2 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

primitive	primitive to compute
hitPoint	to check
lights	list of lights
ray	ray of the camera

Returns

Math::Point3D

3.24.1.3 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

ray	Math::Vector3D
lights	list of lights

Returns

Color

3.24.1.4 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

3.25 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

3.25.1 Detailed Description

Ray class, (point and direction vectors)

3.25.2 Constructor & Destructor Documentation

3.25.2.1 Ray()

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

Construct a new Ray object.

Parameters

origin	point vector
direction	vector direction

3.25.3 Member Function Documentation

3.25.3.1 at()

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

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

Parameters

t	multiplication factor

Returns

Math::Point3D

3.25.3.2 direction()

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

return the direction of the ray

Returns

const Math::Vector3D&

3.25.3.3 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

3.26 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.

3.26.1 Constructor & Destructor Documentation

3.26.1.1 Rectangle3D()

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

Construct a new Rectangle 3D object.

Parameters

origin	bottom-left corner of the rectangle
bottom_side	vector from the bottom-left corner of the rectangle
left_side	vector from the bottom-left corner of the rectangle

3.26.2 Member Function Documentation

3.26.2.1 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

u	location u in rectangle
v	location v in rectangle

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

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

3.27 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) 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.

3.27.1 Constructor & Destructor Documentation

3.27.1.1 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

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

3.27.2 Member Function Documentation

3.27.2.1 getColliderBox()

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

Get the collider box object.

Returns

Octree::cubeCollider

Implements Primitive::IPrimitive.

3.27.2.2 getMaterial()

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

Get the Material object.

Returns

Material of rectangular cuboid

Implements Primitive::IPrimitive.

3.27.2.3 getMaxX()

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

Get the max x value.

Returns

double | max x value.

3.27.2.4 getMaxY()

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

Get the max y value.

Returns

double | max y value.

3.27.2.5 getMaxZ()

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

Get the max z value.

Returns

double | max z value.

3.27.2.6 getMinX()

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

Get the min x value.

Returns

double | min x value.

3.27.2.7 getMinY()

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

Get the min y value.

Returns

double | min y value.

3.27.2.8 getMinZ()

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

Get the min z value.

Returns

double | min z value.

3.27.2.9 getNormal()

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

Get the Normal of the object.

Parameters

hitPoint	to have the normal
ray	of the camera

Returns

Math::Vector3D

Implements Primitive::IPrimitive.

3.27.2.10 hitPoint()

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

Return the hit point of the rectangular cuboid.

Parameters

ray	vector3D

Returns

Point3D

Implements Primitive::IPrimitive.

3.27.2.11 setMaterial()

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

Set the Material object.

Parameters

material	Material of rectangular cuboid

3.27.2.12 setMaxX()

```
void Primitive::RectangularCuboid::setMaxX (
            double maxX )
```

Set the max x value.

Parameters

maxX	double max x value.

3.27.2.13 setMaxY()

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

Set the max y value.

Parameters

maxY	double max y value.

3.27.2.14 setMaxZ()

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

Set the max z value.

Parameters

maxZ	double max z value.

3.27.2.15 setMinX()

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

Set the min x value.

Parameters

minX	double min x value.

3.27.2.16 setMinY()

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

Set the min y value.

Parameters

minY	double min y value.

3.27.2.17 setMinZ()

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

Set the min z value.

Parameters

minZ	double min z value.

3.27.2.18 setRotation()

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

Set the Rotation of the object.

Parameters

rotation	- Rotation value

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

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

3.28 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.

3.28.1 Constructor & Destructor Documentation

3.28.1.1 Renderer()

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

Construct a new Renderer object.

Parameters

scene	to render

3.28.2 Member Function Documentation

3.28.2.1 writeColor()

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

Write a rgb color in a stream.

Parameters

o	stream to write in
color	color to write

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

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

3.29 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.

3.29.1 Constructor & Destructor Documentation

3.29.1.1 Scene()

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

Construct a new Scene object.

Parameters

filePath	File to parse

3.29.2 Member Function Documentation

3.29.2.1 getCamera()

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

Get the Camera object.

Returns

Camera Object

3.29.2.2 getLights()

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

Get the Lights object.

Returns

Light::LightsContainer

3.29.2.3 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

3.30 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.

3.30.1 Constructor & Destructor Documentation

3.30.1.1 Shadow()

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

Construct a new Shadow object.

Parameters

primitives	list of primitives

3.30.2 Member Function Documentation

3.30.2.1 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

3.31 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) 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.

3.31.1 Constructor & Destructor Documentation

3.31.1.1 Sphere()

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

Construct a new Sphere object.

Parameters

origin	center of the sphere
radius	of the sphere
material	Material of Sphere

3.31.2 Member Function Documentation

3.31.2.1 getColliderBox()

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

Get the collider box object.

Returns

Octree::cubeCollider

Implements Primitive::IPrimitive.

3.31.2.2 getMaterial()

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

Get the Material object.

Returns

Material of sphere

Implements Primitive::IPrimitive.

3.31.2.3 getNormal()

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

Get the Normal of the object.

Parameters

hitPoint	to compute the normal
ray	of the camera

Returns

Math::Vector3D

Implements Primitive::IPrimitive.

3.31.2.4 getOrigin()

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

Get the Origin object.

Returns

Origin of sphere

3.31.2.5 getRadius()

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

Get the Origin object.

Returns

Radius of sphere

3.31.2.6 hitPoint()

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

Return the hit point of the sphere.

Parameters

ray	vector3D

Returns

Point3D

Implements Primitive::IPrimitive.

3.31.2.7 setMaterial()

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

Set the Material.

Parameters

material	New material to set

3.31.2.8 setOrigin()

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

Set the Origin object.

Parameters

origin	New origin to set

3.31.2.9 setRadius()

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

Set the Radius.

Parameters

radius	New radius to set

3.31.2.10 setRotation()

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

Set the Rotation object.

Parameters

rotation	- Rotation value

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

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

3.32 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 (const Math::Point3D &vertex1, const Math::Point3D &vertex2, const Math::Point3D &vertex3)

    Construct a new triangle object.
- ∼Triangle ()=default

    Destroy the triangle object.
- Math::Point3D hitPoint (const Raytracer::Ray &ray) 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.

3.32.1 Constructor & Destructor Documentation

3.32.1.1 Triangle() [1/2]

```
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

vertex1	first edge of the triangle
vertex2	second edge of the triangle
vertex3	third edge of the triangle
material	of the triangle

3.32.1.2 Triangle() [2/2]

```
Primitive::Triangle::Triangle (
            const Math::Point3D & vertex1,
            const Math::Point3D & vertex2,
            const Math::Point3D & vertex3 )
```

Construct a new triangle object.

Parameters

vertex1	first edge of the triangle
vertex2	second edge of the triangle
vertex3	third edge of the triangle

3.32.2 Member Function Documentation

3.32.2.1 getColliderBox()

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

Get the collider box object.

Returns

Octree::cubeCollider

Implements Primitive::IPrimitive.

3.32.2.2 getMaterial()

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

Get the material object.

Returns

Material of triangle

Implements Primitive::IPrimitive.

3.32.2.3 getNormal()

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

Get the normal of the object.

Parameters

hitPoint	to compute the normal
ray	of the camera

Returns

Math::Vector3D

Implements Primitive::IPrimitive.

3.32.2.4 getTriangleInverseNormal()

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

Get the triangle inverse normal object.

Returns

Math::Vector3D

3.32.2.5 getTriangleNormal()

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

Get the triangle normal object.

Returns

Math::Vector3D

3.32.2.6 getVertex1()

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

Get the vertex1 object.

Returns

Math::Point3D

3.32.2.7 getVertex2()

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

Get the vertex2 object.

Returns

Math::Point3D

3.32.2.8 getVertex3()

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

Get the vertex3 object.

Returns

Math::Point3D

3.32.2.9 hitPoint()

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

Return the hit point of the triangle.

Parameters

ray	vector3D

Returns

Point3D

Implements Primitive::IPrimitive.

3.32.2.10 setMaterial()

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

Set the material.

Parameters

material	New material to set

3.32.2.11 setRotation()

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

Set the rotation object.

Parameters

rotation	- Rotation value

3.32.2.12 setTriangleInverseNormal()

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

Set the triangle inverse normal object.

Parameters

inverseNormal	new value to set

3.32.2.13 setTriangleNormal()

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

Set the triangle normal object.

Parameters

normal	new value to set

3.32.2.14 setVertex1()

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

Set the vertex1 object.

Parameters

vertex1	first edge of the triangle to set

3.32.2.15 setVertex2()

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

Set the vertex2 object.

Parameters

vertex2	second edge of the triangle to set

3.32.2.16 setVertex3()

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

Set the vertex3 object.

Parameters

vertex3	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

3.33 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)

3.33.1 Detailed Description

Vector 3D class (x, y, z)

3.33.2 Constructor & Destructor Documentation

3.33.2.1 Vector3D()

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

Construct a new Vector 3D object.

Parameters

x	position vector
y	position vector
z	position vector

3.33.3 Member Function Documentation

3.33.3.1 cross()

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

Return the cross product with the vector.

Parameters

ptr	within do the cross product

Returns

Vector3D cross product

3.33.3.2 dot()

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

return the dot product with an other Vector

Parameters

ptr	vector to dot with

Returns

double result of dot product

3.33.3.3 length()

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

return the lenght of the vector

Returns

double

3.33.3.4 length_squared()

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

return the square lenght of the vector

Returns

double

3.33.3.5 rotateX()

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

Rotate a vector with an angle on the axis X.

Parameters

degrees	- Rotation value

3.33.3.6 rotateY()

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

Rotate a vector with an angle on the axis Y.

Parameters

degrees	- Rotation value

3.33.3.7 rotateZ()

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

Rotate a vector with an angle on the axis Z.

Parameters

degrees	- Rotation value

3.33.3.8 translate()

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

translate a vector with an other

Parameters

ptr	vector of translation

3.33.3.9 x()

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

return x coordinate vector

Returns

double

3.33.3.10 y()

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

return y coordinate vector

Returns

double

3.33.3.11 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