

RayTracer

Generated by Doxygen 1.13.2



<b>1 Namespace Index</b>	<b>1</b>
1.1 Namespace List	1
<b>2 Hierarchical Index</b>	<b>3</b>
2.1 Class Hierarchy	3
<b>3 Class Index</b>	<b>5</b>
3.1 Class List	5
<b>4 File Index</b>	<b>7</b>
4.1 File List	7
<b>5 Namespace Documentation</b>	<b>9</b>
5.1 Math Namespace Reference	9
5.2 RayTracer Namespace Reference	9
5.2.1 Function Documentation	9
5.2.1.1 createCameraPlugin()	9
5.2.1.2 createConePrimitive()	10
5.2.1.3 createCylinderPrimitive()	10
<b>6 Class Documentation</b>	<b>11</b>
6.1 AmbientLight Class Reference	11
6.1.1 Constructor & Destructor Documentation	12
6.1.1.1 AmbientLight()	12
6.1.1.2 ~AmbientLight()	12
6.1.2 Member Function Documentation	12
6.1.2.1 applyLight()	12
6.1.2.2 getDirection()	12
6.1.2.3 getIntensity()	12
6.1.2.4 getLightType()	12
6.1.2.5 getType()	12
6.1.2.6 setDirection()	13
6.1.2.7 setIntensity()	13
6.2 BasicRenderer Class Reference	13
6.2.1 Constructor & Destructor Documentation	14
6.2.1.1 BasicRenderer()	14
6.2.1.2 ~BasicRenderer()	14
6.2.2 Member Function Documentation	14
6.2.2.1 getRendererType()	14
6.2.2.2 getType()	14
6.2.2.3 renderScene()	14
6.2.2.4 setCamera()	14
6.2.2.5 setLights()	15
6.2.2.6 setPrimitives()	15

6.2.2.7 setRendererType()	15
6.3 RayTracer::Camera Class Reference	15
6.3.1 Constructor & Destructor Documentation	16
6.3.1.1 Camera() [1/2]	16
6.3.1.2 Camera() [2/2]	16
6.3.1.3 ~Camera()	16
6.3.2 Member Function Documentation	17
6.3.2.1 getFieldOfView()	17
6.3.2.2 getHeight()	17
6.3.2.3 getLookAt()	17
6.3.2.4 getPosition()	17
6.3.2.5 getResolution()	17
6.3.2.6 getType()	17
6.3.2.7 getWidth()	17
6.3.2.8 rayAt()	18
6.3.2.9 rotate()	18
6.3.2.10 setFieldOfView()	18
6.3.2.11 setLookAt()	18
6.3.2.12 setPosition()	18
6.3.2.13 setResolution()	18
6.3.2.14 translate()	19
6.3.2.15 updateScreen()	19
6.4 Math::Color Class Reference	19
6.4.1 Constructor & Destructor Documentation	19
6.4.1.1 Color()	19
6.4.1.2 ~Color()	19
6.4.2 Member Function Documentation	20
6.4.2.1 getB()	20
6.4.2.2 getG()	20
6.4.2.3 getR()	20
6.4.2.4 operator+()	20
6.4.2.5 operator+=()	20
6.5 RayTracer::Cone Class Reference	20
6.5.1 Constructor & Destructor Documentation	21
6.5.1.1 Cone()	21
6.5.1.2 ~Cone()	21
6.5.2 Member Function Documentation	21
6.5.2.1 getCenter()	21
6.5.2.2 getColor()	22
6.5.2.3 getHeight()	22
6.5.2.4 getNormal()	22
6.5.2.5 getPosition()	22

6.5.2.6 <a href="#">getPrimitiveType()</a>	22
6.5.2.7 <a href="#">getRadius()</a>	22
6.5.2.8 <a href="#">getType()</a>	22
6.5.2.9 <a href="#">hits()</a>	22
6.5.2.10 <a href="#">intersect()</a>	23
6.5.2.11 <a href="#">rotate()</a>	23
6.5.2.12 <a href="#">setColor()</a>	23
6.5.2.13 <a href="#">setHeight()</a>	23
6.5.2.14 <a href="#">setNormal()</a>	23
6.5.2.15 <a href="#">setPosition()</a>	23
6.5.2.16 <a href="#">setRadius()</a>	23
6.5.2.17 <a href="#">translate()</a>	24
6.6 Core Class Reference	24
6.6.1 Constructor & Destructor Documentation	24
6.6.1.1 <a href="#">Core()</a>	24
6.6.2 Member Function Documentation	24
6.6.2.1 <a href="#">addPlugin()</a>	24
6.6.2.2 <a href="#">convertVectorPluginToVectorLight()</a>	24
6.6.2.3 <a href="#">convertVectorPluginToVectorPrimitive()</a>	25
6.6.2.4 <a href="#">getPlugins()</a>	25
6.6.2.5 <a href="#">isFileValid()</a>	25
6.7 RayTracer::Cube Class Reference	25
6.7.1 Constructor & Destructor Documentation	26
6.7.1.1 <a href="#">Cube()</a>	26
6.7.1.2 <a href="#">~Cube()</a>	26
6.7.2 Member Function Documentation	26
6.7.2.1 <a href="#">getCenter()</a>	26
6.7.2.2 <a href="#">getColor()</a>	26
6.7.2.3 <a href="#">getHeight()</a>	26
6.7.2.4 <a href="#">getNormal()</a>	26
6.7.2.5 <a href="#">getPosition()</a>	27
6.7.2.6 <a href="#">getPrimitiveType()</a>	27
6.7.2.7 <a href="#">getRadius()</a>	27
6.7.2.8 <a href="#">getSize()</a>	27
6.7.2.9 <a href="#">getType()</a>	27
6.7.2.10 <a href="#">hits()</a>	27
6.7.2.11 <a href="#">intersect()</a>	27
6.7.2.12 <a href="#">rotate()</a>	28
6.7.2.13 <a href="#">setColor()</a>	28
6.7.2.14 <a href="#">setHeight()</a>	28
6.7.2.15 <a href="#">setNormal()</a>	28
6.7.2.16 <a href="#">setPosition()</a>	28

6.7.2.17 setRadius()	28
6.7.2.18 setSize()	28
6.7.2.19 translate()	29
6.8 RayTracer::Cylinder Class Reference	29
6.8.1 Constructor & Destructor Documentation	30
6.8.1.1 Cylinder()	30
6.8.1.2 ~Cylinder()	30
6.8.2 Member Function Documentation	30
6.8.2.1 getCenter()	30
6.8.2.2 getColor()	30
6.8.2.3 getHeight()	30
6.8.2.4 getNormal()	30
6.8.2.5 getPosition()	30
6.8.2.6 getPrimitiveType()	31
6.8.2.7 getRadius()	31
6.8.2.8 hits()	31
6.8.2.9 intersect()	31
6.8.2.10 rotate()	31
6.8.2.11 setColor()	31
6.8.2.12 setHeight()	31
6.8.2.13 setNormal()	32
6.8.2.14 setPosition()	32
6.8.2.15 setRadius()	32
6.8.2.16 translate()	32
6.9 DirectionnalLights Class Reference	32
6.9.1 Constructor & Destructor Documentation	33
6.9.1.1 DirectionnalLights()	33
6.9.1.2 ~DirectionnalLights()	33
6.9.2 Member Function Documentation	33
6.9.2.1 applyLight()	33
6.9.2.2 getDirection()	34
6.9.2.3 getIntensity()	34
6.9.2.4 getLightType()	34
6.9.2.5 getType()	34
6.9.2.6 setDirection()	34
6.9.2.7 setIntensity()	34
6.10 Exception Class Reference	35
6.10.1 Constructor & Destructor Documentation	35
6.10.1.1 Exception()	35
6.11 RayTracer::HitRecord Struct Reference	35
6.11.1 Member Data Documentation	35
6.11.1.1 color	35

6.11.1.2 normal	36
6.11.1.3 point	36
6.11.1.4 t	36
6.12 RayTracer::ICamera Class Reference	36
6.12.1 Constructor & Destructor Documentation	37
6.12.1.1 ~ICamera()	37
6.12.2 Member Function Documentation	37
6.12.2.1 getFieldOfView()	37
6.12.2.2 getHeight()	37
6.12.2.3 getLookAt()	37
6.12.2.4 getPosition()	37
6.12.2.5 getResolution()	37
6.12.2.6 getWidth()	38
6.12.2.7 rayAt()	38
6.12.2.8 rotate()	38
6.12.2.9 setFieldOfView()	38
6.12.2.10 setLookAt()	38
6.12.2.11 setPosition()	38
6.12.2.12 setResolution()	38
6.12.2.13 translate()	39
6.12.2.14 updateScreen()	39
6.13 ILight Class Reference	39
6.13.1 Constructor & Destructor Documentation	40
6.13.1.1 ILight()	40
6.13.1.2 ~ILight()	40
6.13.2 Member Function Documentation	40
6.13.2.1 applyLight()	40
6.13.2.2 getDirection()	40
6.13.2.3 getIntensity()	40
6.13.2.4 getLightType()	40
6.13.2.5 setDirection()	40
6.13.2.6 setIntensity()	41
6.14 IPlugin Class Reference	41
6.14.1 Constructor & Destructor Documentation	41
6.14.1.1 IPlugin()	41
6.14.1.2 ~IPlugin()	41
6.14.2 Member Function Documentation	42
6.14.2.1 getType()	42
6.15 IPrimitive Class Reference	42
6.15.1 Constructor & Destructor Documentation	43
6.15.1.1 IPrimitive()	43
6.15.1.2 ~IPrimitive()	43

6.15.2 Member Function Documentation	43
6.15.2.1 getCenter()	43
6.15.2.2 getColor()	43
6.15.2.3 getHeight()	43
6.15.2.4 getNormal()	43
6.15.2.5 getPosition()	43
6.15.2.6 getPrimitiveType()	44
6.15.2.7 getRadius()	44
6.15.2.8 getType()	44
6.15.2.9 hits()	44
6.15.2.10 intersect()	44
6.15.2.11 rotate()	44
6.15.2.12 setColor()	44
6.15.2.13 setHeight()	45
6.15.2.14 setNormal()	45
6.15.2.15 setPosition()	45
6.15.2.16 setRadius()	45
6.15.2.17 translate()	45
6.16 IRenderer Class Reference	45
6.16.1 Constructor & Destructor Documentation	46
6.16.1.1 IRenderer()	46
6.16.1.2 ~IRenderer()	46
6.16.2 Member Function Documentation	46
6.16.2.1 getRendererType()	46
6.16.2.2 renderScene()	46
6.16.2.3 setCamera()	46
6.16.2.4 setLights()	47
6.16.2.5 setPrimitives()	47
6.16.2.6 setRendererType()	47
6.17 Math::Matrix4x4 Class Reference	47
6.17.1 Constructor & Destructor Documentation	47
6.17.1.1 Matrix4x4()	47
6.17.1.2 ~Matrix4x4()	48
6.17.2 Member Function Documentation	48
6.17.2.1 applyToPoint()	48
6.17.2.2 applyToVector()	48
6.17.2.3 identity()	48
6.17.2.4 rotationMatrix()	48
6.18 ParsingException Class Reference	48
6.18.1 Constructor & Destructor Documentation	49
6.18.1.1 ParsingException()	49
6.19 RayTracer::Plane Class Reference	49



6.19.1 Constructor & Destructor Documentation	50
6.19.1.1 Plane()	50
6.19.1.2 ~Plane()	50
6.19.2 Member Function Documentation	50
6.19.2.1 getCenter()	50
6.19.2.2 getColor()	50
6.19.2.3 getHeight()	50
6.19.2.4 getNormal()	50
6.19.2.5 getPosition()	51
6.19.2.6 getPrimitiveType()	51
6.19.2.7 getRadius()	51
6.19.2.8 getType()	51
6.19.2.9 hits()	51
6.19.2.10 intersect()	51
6.19.2.11 rotate()	51
6.19.2.12 setColor()	52
6.19.2.13 setHeight()	52
6.19.2.14 setNormal()	52
6.19.2.15 setPosition()	52
6.19.2.16 setRadius()	52
6.19.2.17 translate()	52
6.20 PluginLoader Class Reference	53
6.20.1 Constructor & Destructor Documentation	53
6.20.1.1 PluginLoader()	53
6.20.1.2 ~PluginLoader()	54
6.20.2 Member Function Documentation	54
6.20.2.1 chooseHandleByPath()	54
6.20.2.2 createPrimitive()	54
6.20.2.3 getInstance()	54
6.20.2.4 loadCameraPlugin()	54
6.20.2.5 loadlightsPlugin()	54
6.20.2.6 loadRenderPlugin()	54
6.20.2.7 loadSharedLibrary()	54
6.20.2.8 loadSymbol()	55
6.20.2.9 storeHandle()	55
6.20.2.10 unloadCamera()	55
6.20.2.11 unloadLights()	55
6.20.2.12 unloadPrimitives()	55
6.20.2.13 unloadRender()	55
6.20.3 Member Data Documentation	55
6.20.3.1 camera	55
6.20.3.2 coneHandle	55

6.20.3.3 cubeHandle . . . . .	56
6.20.3.4 cylinderHandle . . . . .	56
6.20.3.5 lights . . . . .	56
6.20.3.6 planeHandle . . . . .	56
6.20.3.7 primitive . . . . .	56
6.20.3.8 render . . . . .	56
6.20.3.9 sphereHandle . . . . .	56
6.21 Math::Point3D Class Reference . . . . .	56
6.21.1 Constructor & Destructor Documentation . . . . .	57
6.21.1.1 Point3D() . . . . .	57
6.21.1.2 ~Point3D() . . . . .	57
6.21.2 Member Function Documentation . . . . .	57
6.21.2.1 getX() . . . . .	57
6.21.2.2 getY() . . . . .	57
6.21.2.3 getZ() . . . . .	57
6.21.2.4 operator+() . . . . .	57
6.21.2.5 operator+=() . . . . .	57
6.22 RayTracer::Ray Class Reference . . . . .	57
6.22.1 Constructor & Destructor Documentation . . . . .	58
6.22.1.1 Ray() . . . . .	58
6.22.2 Member Function Documentation . . . . .	58
6.22.2.1 getDirection() . . . . .	58
6.22.2.2 getOrigin() . . . . .	58
6.23 RayTracerException Class Reference . . . . .	58
6.23.1 Constructor & Destructor Documentation . . . . .	59
6.23.1.1 RayTracerException() . . . . .	59
6.24 Math::Rectangle3D Class Reference . . . . .	59
6.24.1 Constructor & Destructor Documentation . . . . .	59
6.24.1.1 Rectangle3D() . . . . .	59
6.24.2 Member Function Documentation . . . . .	59
6.24.2.1 pointAt() . . . . .	59
6.24.3 Member Data Documentation . . . . .	60
6.24.3.1 bottom_side . . . . .	60
6.24.3.2 left_side . . . . .	60
6.24.3.3 origin . . . . .	60
6.25 SceneLoader Class Reference . . . . .	60
6.25.1 Constructor & Destructor Documentation . . . . .	60
6.25.1.1 SceneLoader() . . . . .	60
6.25.1.2 ~SceneLoader() . . . . .	61
6.25.2 Member Function Documentation . . . . .	61
6.25.2.1 castPlugin() . . . . .	61
6.25.2.2 checkCfgError() . . . . .	61

6.25.2.3 fillCone()	61
6.25.2.4 fillCube()	61
6.25.2.5 fillCylinder()	61
6.25.2.6 fillPlane()	61
6.25.2.7 fillSphere()	61
6.25.2.8 loadCamera()	62
6.25.2.9 loadLights()	62
6.25.2.10 loadPrimitives()	62
6.25.2.11 loadRender()	62
6.26 RayTracer::Sphere Class Reference	62
6.26.1 Constructor & Destructor Documentation	63
6.26.1.1 Sphere()	63
6.26.1.2 ~Sphere()	63
6.26.2 Member Function Documentation	63
6.26.2.1 getCenter()	63
6.26.2.2 getColor()	64
6.26.2.3 getHeight()	64
6.26.2.4 getNormal()	64
6.26.2.5 getPosition()	64
6.26.2.6 getPrimitiveType()	64
6.26.2.7 getRadius()	64
6.26.2.8 getType()	64
6.26.2.9 hits()	64
6.26.2.10 intersect()	65
6.26.2.11 rotate()	65
6.26.2.12 setColor()	65
6.26.2.13 setHeight()	65
6.26.2.14 setNormal()	65
6.26.2.15 setPosition()	65
6.26.2.16 setRadius()	65
6.26.2.17 translate()	66
6.27 Math::Vector3D Class Reference	66
6.27.1 Constructor & Destructor Documentation	66
6.27.1.1 Vector3D()	66
6.27.1.2 ~Vector3D()	66
6.27.2 Member Function Documentation	66
6.27.2.1 dot()	66
6.27.2.2 getX()	67
6.27.2.3 getY()	67
6.27.2.4 getZ()	67
6.27.2.5 operator+()	67
6.27.2.6 operator-()	67

<b>7 File Documentation</b>	<b>69</b>
7.1 src/core/Core.cpp File Reference	69
7.1.1 Function Documentation	69
7.1.1.1 core_run()	69
7.2 src/core/Core.hpp File Reference	69
7.3 Core.hpp	70
7.4 src/exceptions/Exceptions.hpp File Reference	70
7.5 Exceptions.hpp	71
7.6 src/loaders/PluginLoader.cpp File Reference	71
7.7 src/loaders/PluginLoader.hpp File Reference	71
7.8 PluginLoader.hpp	72
7.9 src/loaders/SceneLoader.cpp File Reference	72
7.10 src/loaders/SceneLoader.hpp File Reference	72
7.11 SceneLoader.hpp	73
7.12 src/main.cpp File Reference	74
7.12.1 Function Documentation	74
7.12.1.1 core_run()	74
7.12.1.2 display_usage()	74
7.12.1.3 main()	74
7.13 src/plugins/camera/Camera.cpp File Reference	74
7.14 src/plugins/camera/Camera.hpp File Reference	75
7.15 Camera.hpp	75
7.16 src/plugins/camera/ICamera.hpp File Reference	76
7.17 ICamera.hpp	76
7.18 src/plugins/IPlugin.hpp File Reference	77
7.18.1 Enumeration Type Documentation	77
7.18.1.1 PluginType	77
7.19 IPlugin.hpp	77
7.20 src/plugins/lights/AmbientLight.cpp File Reference	78
7.20.1 Function Documentation	78
7.20.1.1 createAmbientLight()	78
7.21 src/plugins/lights/AmbientLight.hpp File Reference	78
7.22 AmbientLight.hpp	78
7.23 src/plugins/lights/DirectionnalLights.cpp File Reference	79
7.23.1 Function Documentation	79
7.23.1.1 createDirectionnalLights()	79
7.24 src/plugins/lights/DirectionnalLights.hpp File Reference	79
7.25 DirectionnalLights.hpp	79
7.26 src/plugins/lights/ILight.hpp File Reference	80
7.26.1 Enumeration Type Documentation	80
7.26.1.1 LightType	80
7.27 ILight.hpp	81

7.28 src/plugins/math/Color.cpp File Reference . . . . .	81
7.29 src/plugins/math/Color.hpp File Reference . . . . .	81
7.30 Color.hpp . . . . .	81
7.31 src/plugins/math/Matrix.cpp File Reference . . . . .	82
7.32 src/plugins/math/Matrix.hpp File Reference . . . . .	82
7.33 Matrix.hpp . . . . .	82
7.34 src/plugins/math/Point.cpp File Reference . . . . .	83
7.35 src/plugins/math/Point.hpp File Reference . . . . .	83
7.36 Point.hpp . . . . .	83
7.37 src/plugins/math/Rectangle.cpp File Reference . . . . .	84
7.38 src/plugins/math/Rectangle.hpp File Reference . . . . .	84
7.39 Rectangle.hpp . . . . .	84
7.40 src/plugins/math/Vector.cpp File Reference . . . . .	84
7.41 src/plugins/math/Vector.hpp File Reference . . . . .	85
7.42 Vector.hpp . . . . .	85
7.43 src/plugins/primitives/Cone.cpp File Reference . . . . .	85
7.44 src/plugins/primitives/Cone.hpp File Reference . . . . .	86
7.45 Cone.hpp . . . . .	86
7.46 src/plugins/primitives/Cube.cpp File Reference . . . . .	87
7.46.1 Function Documentation . . . . .	87
7.46.1.1 createCubePrimitive() . . . . .	87
7.47 src/plugins/primitives/Cube.hpp File Reference . . . . .	87
7.48 Cube.hpp . . . . .	88
7.49 src/plugins/primitives/Cylinder.cpp File Reference . . . . .	89
7.50 src/plugins/primitives/Cylinder.hpp File Reference . . . . .	89
7.51 Cylinder.hpp . . . . .	90
7.52 src/plugins/primitives/HitRecord.hpp File Reference . . . . .	90
7.53 HitRecord.hpp . . . . .	91
7.54 src/plugins/primitives/IPrimitive.hpp File Reference . . . . .	91
7.54.1 Enumeration Type Documentation . . . . .	91
7.54.1.1 PrimitiveType . . . . .	91
7.55 IPrimitive.hpp . . . . .	92
7.56 src/plugins/primitives/Plane.cpp File Reference . . . . .	92
7.56.1 Function Documentation . . . . .	93
7.56.1.1 createPlanePrimitive() . . . . .	93
7.57 src/plugins/primitives/Plane.hpp File Reference . . . . .	93
7.58 Plane.hpp . . . . .	93
7.59 src/plugins/primitives/Ray.cpp File Reference . . . . .	94
7.60 src/plugins/primitives/Ray.hpp File Reference . . . . .	94
7.61 Ray.hpp . . . . .	95
7.62 src/plugins/primitives/Sphere.cpp File Reference . . . . .	95
7.62.1 Function Documentation . . . . .	95

---

7.62.1.1 createSpherePrimitive()	95
7.63 src/plugins/primitives/Sphere.hpp File Reference	96
7.64 Sphere.hpp	96
7.65 src/plugins/render/BasicRenderer.cpp File Reference	97
7.65.1 Function Documentation	97
7.65.1.1 createRenderPlugin()	97
7.66 src/plugins/render/BasicRenderer.hpp File Reference	97
7.67 BasicRenderer.hpp	98
7.68 src/plugins/render/IRenderer.hpp File Reference	98
7.68.1 Enumeration Type Documentation	99
7.68.1.1 RendererType	99
7.69 IRenderer.hpp	99
<b>Index</b>	<b>101</b>

# Chapter 1

## Namespace Index

### 1.1 Namespace List

Here is a list of all namespaces with brief descriptions:

<a href="#">Math</a>	.....	9
<a href="#">RayTracer</a>	.....	9





## Chapter 2

# Hierarchical Index

### 2.1 Class Hierarchy

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

Math::Color . . . . .	19
Core . . . . .	24
RayTracer::HitRecord . . . . .	35
IPlugin . . . . .	41
ILight . . . . .	39
AmbientLight . . . . .	11
DirectionnalLights . . . . .	32
IPrimitive . . . . .	42
RayTracer::Cone . . . . .	20
RayTracer::Cube . . . . .	25
RayTracer::Cylinder . . . . .	29
RayTracer::Plane . . . . .	49
RayTracer::Sphere . . . . .	62
IRenderer . . . . .	45
BasicRenderer . . . . .	13
RayTracer::ICamera . . . . .	36
RayTracer::Camera . . . . .	15
Math::Matrix4x4 . . . . .	47
PluginLoader . . . . .	53
Math::Point3D . . . . .	56
RayTracer::Ray . . . . .	57
Math::Rectangle3D . . . . .	59
std::runtime_error . . . . .	
Exception . . . . .	35
ParsingException . . . . .	48
RayTracerException . . . . .	58
SceneLoader . . . . .	60
Math::Vector3D . . . . .	66



## Chapter 3

# Class Index

### 3.1 Class List

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

AmbientLight	11
BasicRenderer	13
RayTracer::Camera	15
Math::Color	19
RayTracer::Cone	20
Core	24
RayTracer::Cube	25
RayTracer::Cylinder	29
DirectionnalLights	32
Exception	35
RayTracer::HitRecord	35
RayTracer::ICamera	36
ILight	39
IPlugin	41
IPrimitive	42
IRenderer	45
Math::Matrix4x4	47
ParsingException	48
RayTracer::Plane	49
PluginLoader	53
Math::Point3D	56
RayTracer::Ray	57
RayTracerException	58
Math::Rectangle3D	59
SceneLoader	60
RayTracer::Sphere	62
Math::Vector3D	66



## Chapter 4

# File Index

### 4.1 File List

Here is a list of all files with brief descriptions:

src/main.cpp	74
src/core/Core.cpp	69
src/core/Core.hpp	69
src/exceptions/Exceptions.hpp	70
src/loaders/PluginLoader.cpp	71
src/loaders/PluginLoader.hpp	71
src/loaders/SceneLoader.cpp	72
src/loaders/SceneLoader.hpp	72
src/plugins/IPlugin.hpp	77
src/plugins/camera/Camera.cpp	74
src/plugins/camera/Camera.hpp	75
src/plugins/camera/ICamera.hpp	76
src/plugins/lights/AmbientLight.cpp	78
src/plugins/lights/AmbientLight.hpp	78
src/plugins/lights/DirectionnalLights.cpp	79
src/plugins/lights/DirectionnalLights.hpp	79
src/plugins/lights/ILight.hpp	80
src/plugins/math/Color.cpp	81
src/plugins/math/Color.hpp	81
src/plugins/math/Matrix.cpp	82
src/plugins/math/Matrix.hpp	82
src/plugins/math/Point.cpp	83
src/plugins/math/Point.hpp	83
src/plugins/math/Rectangle.cpp	84
src/plugins/math/Rectangle.hpp	84
src/plugins/math/Vector.cpp	84
src/plugins/math/Vector.hpp	85
src/plugins/primitives/Cone.cpp	85
src/plugins/primitives/Cone.hpp	86
src/plugins/primitives/Cube.cpp	87
src/plugins/primitives/Cube.hpp	87
src/plugins/primitives/Cylinder.cpp	89
src/plugins/primitives/Cylinder.hpp	89
src/plugins/primitives/HitRecord.hpp	90
src/plugins/primitives/IPrimitive.hpp	91

src/plugins/primitives/ <a href="#">Plane.cpp</a> . . . . .	92
src/plugins/primitives/ <a href="#">Plane.hpp</a> . . . . .	93
src/plugins/primitives/ <a href="#">Ray.cpp</a> . . . . .	94
src/plugins/primitives/ <a href="#">Ray.hpp</a> . . . . .	94
src/plugins/primitives/ <a href="#">Sphere.cpp</a> . . . . .	95
src/plugins/primitives/ <a href="#">Sphere.hpp</a> . . . . .	96
src/plugins/render/ <a href="#">BasicRenderer.cpp</a> . . . . .	97
src/plugins/render/ <a href="#">BasicRenderer.hpp</a> . . . . .	97
src/plugins/render/ <a href="#">IRenderer.hpp</a> . . . . .	98

## Chapter 5

# Namespace Documentation

### 5.1 Math Namespace Reference

#### Classes

- class [Color](#)
- class [Matrix4x4](#)
- class [Point3D](#)
- class [Rectangle3D](#)
- class [Vector3D](#)

### 5.2 RayTracer Namespace Reference

#### Classes

- class [Camera](#)
- class [Cone](#)
- class [Cube](#)
- class [Cylinder](#)
- struct [HitRecord](#)
- class [ICamera](#)
- class [Plane](#)
- class [Ray](#)
- class [Sphere](#)

#### Functions

- [IPlugin](#) \* [createCameraPlugin](#) ()
- [IPlugin](#) \* [createConePrimitive](#) ()
- [IPlugin](#) \* [createCylinderPrimitive](#) ()

#### 5.2.1 Function Documentation

##### 5.2.1.1 createCameraPlugin()

[IPlugin](#) \* [RayTracer::createCameraPlugin](#) ()

#### 5.2.1.2 createConePrimitive()

```
IPlugin * RayTracer::createConePrimitive ()
```

#### 5.2.1.3 createCylinderPrimitive()

```
IPlugin * RayTracer::createCylinderPrimitive ()
```



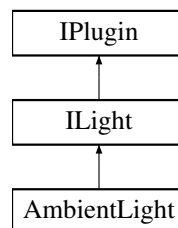
# Chapter 6

## Class Documentation

### 6.1 AmbientLight Class Reference

```
#include <AmbientLight.hpp>
```

Inheritance diagram for AmbientLight:



#### Public Member Functions

- [AmbientLight](#) (float intensity=1.0f)
- [~AmbientLight](#) ()=default
- [PluginType getType](#) () const override
- [LightType getLightType](#) () const override
- void [setIntensity](#) (float intensity) override
- float [getIntensity](#) () const override
- void [applyLight](#) (float &r, float &g, float &b) const override
- void [setDirection](#) (float, float, float) override
- [Math::Vector3D getDirection](#) () const override

#### Public Member Functions inherited from [ILight](#)

- [ILight](#) ()=default
- virtual [~ILight](#) ()=default

#### Public Member Functions inherited from [IPlugin](#)

- [IPlugin](#) ()=default
- virtual [~IPlugin](#) ()=default

## 6.1.1 Constructor & Destructor Documentation

### 6.1.1.1 AmbientLight()

```
AmbientLight::AmbientLight (
    float intensity = 1.0f)
```

### 6.1.1.2 ~AmbientLight()

```
AmbientLight::~AmbientLight () [default]
```

## 6.1.2 Member Function Documentation

### 6.1.2.1 applyLight()

```
void AmbientLight::applyLight (
    float & r,
    float & g,
    float & b) const [override], [virtual]
```

Implements [ILight](#).

### 6.1.2.2 getDirection()

```
Math::Vector3D AmbientLight::getDirection () const [inline], [override], [virtual]
```

Implements [ILight](#).

### 6.1.2.3 getIntensity()

```
float AmbientLight::getIntensity () const [override], [virtual]
```

Implements [ILight](#).

### 6.1.2.4 getLightType()

```
LightType AmbientLight::getLightType () const [inline], [override], [virtual]
```

Implements [ILight](#).

### 6.1.2.5 getType()

```
PluginType AmbientLight::getType () const [inline], [override], [virtual]
```

Implements [IPlugin](#).

### 6.1.2.6 setDirection()

```
void AmbientLight::setDirection (
    float ,
    float ,
    float ) [inline], [override], [virtual]
```

Implements [ILight](#).

### 6.1.2.7 setIntensity()

```
void AmbientLight::setIntensity (
    float intensity) [override], [virtual]
```

Implements [ILight](#).

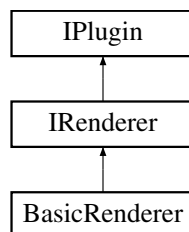
The documentation for this class was generated from the following files:

- src/plugins/lights/[AmbientLight.hpp](#)
- src/plugins/lights/[AmbientLight.cpp](#)

## 6.2 BasicRenderer Class Reference

```
#include <BasicRenderer.hpp>
```

Inheritance diagram for BasicRenderer:



### Public Member Functions

- [BasicRenderer](#) ()
- [~BasicRenderer](#) ()=default
- [PluginType](#) [getType](#) () const override
- void [setRendererType](#) ([RendererType](#) type) override
- [RendererType](#) [getRendererType](#) () const override
- void [renderScene](#) () override
- void [setCamera](#) ([RayTracer::ICamera](#) \*camera) override
- void [setLights](#) (std::vector< [ILight](#) \* > &lights) override
- void [setPrimitives](#) (std::vector< [IPrimitive](#) \* > &primitives) override

## Public Member Functions inherited from [IRenderer](#)

- [IRenderer](#) ()=default
- virtual [~IRenderer](#) ()=default

## Public Member Functions inherited from [IPlugin](#)

- [IPlugin](#) ()=default
- virtual [~IPlugin](#) ()=default

## 6.2.1 Constructor & Destructor Documentation

### 6.2.1.1 BasicRenderer()

```
BasicRenderer::BasicRenderer ()
```

### 6.2.1.2 ~BasicRenderer()

```
BasicRenderer::~~BasicRenderer () [default]
```

## 6.2.2 Member Function Documentation

### 6.2.2.1 getRendererType()

```
RendererType BasicRenderer::getRendererType () const [inline], [override], [virtual]
```

Implements [IRenderer](#).

### 6.2.2.2 getType()

```
PluginType BasicRenderer::getType () const [inline], [override], [virtual]
```

Implements [IPlugin](#).

### 6.2.2.3 renderScene()

```
void BasicRenderer::renderScene () [override], [virtual]
```

Implements [IRenderer](#).

### 6.2.2.4 setCamera()

```
void BasicRenderer::setCamera (
    RayTracer::ICamera * camera) [override], [virtual]
```

Implements [IRenderer](#).

#### 6.2.2.5 setLights()

```
void BasicRenderer::setLights (
    std::vector< ILight * > & lights) [override], [virtual]
```

Implements [IRenderer](#).

#### 6.2.2.6 setPrimitives()

```
void BasicRenderer::setPrimitives (
    std::vector< IPrimitive * > & primitives) [override], [virtual]
```

Implements [IRenderer](#).

#### 6.2.2.7 setRendererType()

```
void BasicRenderer::setRendererType (
    RendererType type) [inline], [override], [virtual]
```

Implements [IRenderer](#).

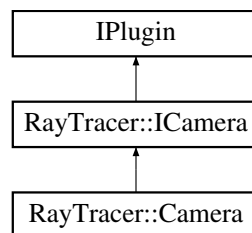
The documentation for this class was generated from the following files:

- src/plugins/render/[BasicRenderer.hpp](#)
- src/plugins/render/[BasicRenderer.cpp](#)

## 6.3 RayTracer::Camera Class Reference

```
#include <Camera.hpp>
```

Inheritance diagram for RayTracer::Camera:



## Public Member Functions

- [Camera](#) ()
- [Camera](#) (const [Math::Point3D](#) &origin, const [Math::Rectangle3D](#) &screen, float fov)
- [~Camera](#) ()=default
- [PluginType](#) [getType](#) () const override
- void [rotate](#) (float angle, const [Math::Vector3D](#) &axis) override
- void [translate](#) (const [Math::Vector3D](#) &translation) override
- void [setPosition](#) ([Math::Point3D](#) position) override
- [Math::Point3D](#) [getPosition](#) () const override
- void [setResolution](#) (int width, int height) override
- void [getResolution](#) (int &width, int &height) const override
- void [setFieldOfView](#) (float fov) override
- float [getFieldOfView](#) () const override
- int [getWidth](#) () const override
- int [getHeight](#) () const override
- void [updateScreen](#) () override
- [Ray](#) [rayAt](#) (double u, double v) const override
- void [setLookAt](#) (const [Math::Point3D](#) &lookAt) override
- [Math::Point3D](#) [getLookAt](#) () const override

## Public Member Functions inherited from [RayTracer::ICamera](#)

- virtual [~ICamera](#) ()=default

## Public Member Functions inherited from [IPlugin](#)

- [IPlugin](#) ()=default
- virtual [~IPlugin](#) ()=default

## 6.3.1 Constructor & Destructor Documentation

### 6.3.1.1 [Camera\(\)](#) [1/2]

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

### 6.3.1.2 [Camera\(\)](#) [2/2]

```
RayTracer::Camera::Camera (
    const Math::Point3D & origin,
    const Math::Rectangle3D & screen,
    float fov)
```

### 6.3.1.3 [~Camera\(\)](#)

```
RayTracer::Camera::~~Camera () [default]
```

## 6.3.2 Member Function Documentation

### 6.3.2.1 getFieldOfView()

```
float RayTracer::Camera::getFieldOfView () const [override], [virtual]
```

Implements [RayTracer::ICamera](#).

### 6.3.2.2 getHeight()

```
int RayTracer::Camera::getHeight () const [inline], [override], [virtual]
```

Implements [RayTracer::ICamera](#).

### 6.3.2.3 getLookAt()

```
Math::Point3D RayTracer::Camera::getLookAt () const [inline], [override], [virtual]
```

Implements [RayTracer::ICamera](#).

### 6.3.2.4 getPosition()

```
Math::Point3D RayTracer::Camera::getPosition () const [override], [virtual]
```

Implements [RayTracer::ICamera](#).

### 6.3.2.5 getResolution()

```
void RayTracer::Camera::getResolution (
    int & width,
    int & height) const [override], [virtual]
```

Implements [RayTracer::ICamera](#).

### 6.3.2.6 getType()

```
PluginType RayTracer::Camera::getType () const [inline], [override], [virtual]
```

Implements [IPlugin](#).

### 6.3.2.7 getWidth()

```
int RayTracer::Camera::getWidth () const [inline], [override], [virtual]
```

Implements [RayTracer::ICamera](#).

#### 6.3.2.8 rayAt()

```
Ray RayTracer::Camera::rayAt (
    double u,
    double v) const [override], [virtual]
```

Implements [RayTracer::ICamera](#).

#### 6.3.2.9 rotate()

```
void RayTracer::Camera::rotate (
    float angle,
    const Math::Vector3D & axis) [override], [virtual]
```

Implements [RayTracer::ICamera](#).

#### 6.3.2.10 setFieldOfView()

```
void RayTracer::Camera::setFieldOfView (
    float fov) [override], [virtual]
```

Implements [RayTracer::ICamera](#).

#### 6.3.2.11 setLookAt()

```
void RayTracer::Camera::setLookAt (
    const Math::Point3D & lookAt) [override], [virtual]
```

Implements [RayTracer::ICamera](#).

#### 6.3.2.12 setPosition()

```
void RayTracer::Camera::setPosition (
    Math::Point3D position) [inline], [override], [virtual]
```

Implements [RayTracer::ICamera](#).

#### 6.3.2.13 setResolution()

```
void RayTracer::Camera::setResolution (
    int width,
    int height) [override], [virtual]
```

Implements [RayTracer::ICamera](#).



#### 6.3.2.14 translate()

```
void RayTracer::Camera::translate (
    const Math::Vector3D & translation) [override], [virtual]
```

Implements [RayTracer::ICamera](#).

#### 6.3.2.15 updateScreen()

```
void RayTracer::Camera::updateScreen () [override], [virtual]
```

Implements [RayTracer::ICamera](#).

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

- src/plugins/camera/[Camera.hpp](#)
- src/plugins/camera/[Camera.cpp](#)

## 6.4 Math::Color Class Reference

```
#include <Color.hpp>
```

### Public Member Functions

- [Color](#) (float r, float g, float b)
- [~Color](#) ()=default
- float [getR](#) () const
- float [getG](#) () const
- float [getB](#) () const
- [Color operator+](#) (const [Color](#) &other) const
- [Color operator+=](#) (const [Color](#) &other)

### 6.4.1 Constructor & Destructor Documentation

#### 6.4.1.1 Color()

```
Math::Color::Color (
    float r,
    float g,
    float b) [inline]
```

#### 6.4.1.2 ~Color()

```
Math::Color::~~Color () [default]
```

## 6.4.2 Member Function Documentation

### 6.4.2.1 getB()

```
float Math::Color::getB () const [inline]
```

### 6.4.2.2 getG()

```
float Math::Color::getG () const [inline]
```

### 6.4.2.3 getR()

```
float Math::Color::getR () const [inline]
```

### 6.4.2.4 operator+()

```
Color Math::Color::operator+ (  
    const Color & other) const [inline]
```

### 6.4.2.5 operator+=()

```
Color Math::Color::operator+= (  
    const Color & other) [inline]
```

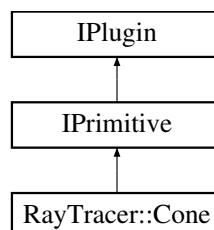
The documentation for this class was generated from the following file:

- [src/plugins/math/Color.hpp](#)

## 6.5 RayTracer::Cone Class Reference

```
#include <Cone.hpp>
```

Inheritance diagram for RayTracer::Cone:



**Public Member Functions**

- [Cone](#) ()
- [~Cone](#) ()=default
- bool [hits](#) (const [Ray](#) &ray) const override
- bool [intersect](#) (const [Ray](#) &ray, [HitRecord](#) &rec) const override
- void [translate](#) (const [Math::Vector3D](#) &translation) override
- void [rotate](#) (float angle, const [Math::Vector3D](#) &axis) override
- const [Math::Point3D](#) & [getCenter](#) () const override
- float [getRadius](#) () const override
- const [Math::Color](#) & [getColor](#) () const override
- [PluginType](#) [getType](#) () const override
- [PrimitiveType](#) [getPrimitiveType](#) () const override
- [Math::Point3D](#) [getPosition](#) () const override
- void [setPosition](#) (const [Math::Point3D](#) &position) override
- void [setRadius](#) (float newRadius) override
- void [setNormal](#) (const [Math::Vector3D](#) &newNormal) override
- [Math::Vector3D](#) [getNormal](#) () const override
- void [setColor](#) ([Math::Color](#) newColor) override
- float [getHeight](#) () const override
- void [setHeight](#) (float newHeight) override

**Public Member Functions inherited from [IPrimitive](#)**

- [IPrimitive](#) ()=default
- virtual [~IPrimitive](#) ()=default
- [PluginType](#) [getType](#) () const override

**Public Member Functions inherited from [IPlugin](#)**

- [IPlugin](#) ()=default
- virtual [~IPlugin](#) ()=default

**6.5.1 Constructor & Destructor Documentation****6.5.1.1 [Cone](#)()**

```
RayTracer::Cone::Cone ()
```

**6.5.1.2 [~Cone](#)()**

```
RayTracer::Cone::~Cone () [default]
```

**6.5.2 Member Function Documentation****6.5.2.1 [getCenter](#)()**

```
const Math::Point3D & RayTracer::Cone::getCenter () const [inline], [override], [virtual]
```

Implements [IPrimitive](#).

#### 6.5.2.2 getColor()

```
const Math::Color & RayTracer::Cone::getColor () const [inline], [override], [virtual]
```

Implements [IPrimitive](#).

#### 6.5.2.3 getHeight()

```
float RayTracer::Cone::getHeight () const [inline], [override], [virtual]
```

Implements [IPrimitive](#).

#### 6.5.2.4 getNormal()

```
Math::Vector3D RayTracer::Cone::getNormal () const [inline], [override], [virtual]
```

Implements [IPrimitive](#).

#### 6.5.2.5 getPosition()

```
Math::Point3D RayTracer::Cone::getPosition () const [inline], [override], [virtual]
```

Implements [IPrimitive](#).

#### 6.5.2.6 getPrimitiveType()

```
PrimitiveType RayTracer::Cone::getPrimitiveType () const [inline], [override], [virtual]
```

Implements [IPrimitive](#).

#### 6.5.2.7 getRadius()

```
float RayTracer::Cone::getRadius () const [inline], [override], [virtual]
```

Implements [IPrimitive](#).

#### 6.5.2.8 getType()

```
PluginType RayTracer::Cone::getType () const [inline], [override], [virtual]
```

Implements [IPlugin](#).

#### 6.5.2.9 hits()

```
bool RayTracer::Cone::hits (  
    const Ray & ray) const [override], [virtual]
```

Implements [IPrimitive](#).

#### 6.5.2.10 intersect()

```
bool RayTracer::Cone::intersect (
    const Ray & ray,
    HitRecord & rec) const [override], [virtual]
```

Implements [IPrimitive](#).

#### 6.5.2.11 rotate()

```
void RayTracer::Cone::rotate (
    float angle,
    const Math::Vector3D & axis) [override], [virtual]
```

Implements [IPrimitive](#).

#### 6.5.2.12 setColor()

```
void RayTracer::Cone::setColor (
    Math::Color newColor) [inline], [override], [virtual]
```

Implements [IPrimitive](#).

#### 6.5.2.13 setHeight()

```
void RayTracer::Cone::setHeight (
    float newHeight) [inline], [override], [virtual]
```

Implements [IPrimitive](#).

#### 6.5.2.14 setNormal()

```
void RayTracer::Cone::setNormal (
    const Math::Vector3D & newNormal) [inline], [override], [virtual]
```

Implements [IPrimitive](#).

#### 6.5.2.15 setPosition()

```
void RayTracer::Cone::setPosition (
    const Math::Point3D & position) [inline], [override], [virtual]
```

Implements [IPrimitive](#).

#### 6.5.2.16 setRadius()

```
void RayTracer::Cone::setRadius (
    float newRadius) [inline], [override], [virtual]
```

Implements [IPrimitive](#).

### 6.5.2.17 translate()

```
void RayTracer::Cone::translate (
    const Math::Vector3D & translation)    [override], [virtual]
```

Implements [IPrimitive](#).

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

- [src/plugins/primitives/Cone.hpp](#)
- [src/plugins/primitives/Cone.cpp](#)

## 6.6 Core Class Reference

```
#include <Core.hpp>
```

### Public Member Functions

- [Core](#) ()=default
- bool [isFileValid](#) (const std::string &filename)
- std::vector< [IPlugin](#) \* > [getPlugins](#) () const
- void [addPlugin](#) ([IPlugin](#) \*plugin)
- void [convertVectorPluginToVectorLight](#) (std::vector< [IPlugin](#) \* > &plugins, std::vector< [ILight](#) \* > &lights)
- void [convertVectorPluginToVectorPrimitive](#) (std::vector< [IPlugin](#) \* > &plugins, std::vector< [IPrimitive](#) \* > &primitives)

### 6.6.1 Constructor & Destructor Documentation

#### 6.6.1.1 Core()

```
Core::Core ()    [default]
```

### 6.6.2 Member Function Documentation

#### 6.6.2.1 addPlugin()

```
void Core::addPlugin (
    IPlugin * plugin)    [inline]
```

#### 6.6.2.2 convertVectorPluginToVectorLight()

```
void Core::convertVectorPluginToVectorLight (
    std::vector< IPlugin * > & plugins,
    std::vector< ILight * > & lights)
```

### 6.6.2.3 convertVectorPluginToVectorPrimitive()

```
void Core::convertVectorPluginToVectorPrimitive (
    std::vector< IPlugin * > & plugins,
    std::vector< IPrimitive * > & primitives)
```

### 6.6.2.4 getPlugins()

```
std::vector< IPlugin * > Core::getPlugins () const [inline]
```

### 6.6.2.5 isFileValid()

```
bool Core::isFileValid (
    const std::string & filename)
```

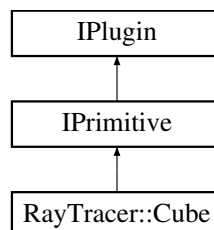
The documentation for this class was generated from the following files:

- [src/core/Core.hpp](#)
- [src/core/Core.cpp](#)

## 6.7 RayTracer::Cube Class Reference

```
#include <Cube.hpp>
```

Inheritance diagram for RayTracer::Cube:



### Public Member Functions

- [Cube](#) ()
- [~Cube](#) ()=default
- bool [hits](#) (const [Ray](#) &ray) const override
- bool [intersect](#) (const [Ray](#) &ray, [HitRecord](#) &rec) const override
- void [translate](#) (const [Math::Vector3D](#) &translation) override
- void [rotate](#) (float angle, const [Math::Vector3D](#) &axis) override
- const [Math::Point3D](#) & [getCenter](#) () const override
- float [getRadius](#) () const override
- const [Math::Color](#) & [getColor](#) () const override
- [PluginType](#) [getType](#) () const override
- [PrimitiveType](#) [getPrimitiveType](#) () const override
- [Math::Point3D](#) [getPosition](#) () const override
- void [setPosition](#) (const [Math::Point3D](#) &position) override
- void [setRadius](#) (float newSize) override
- void [setNormal](#) (const [Math::Vector3D](#) &newNormal) override
- [Math::Vector3D](#) [getNormal](#) () const override
- void [setColor](#) ([Math::Color](#) newColor) override
- void [setSize](#) (float newSize)
- float [getSize](#) () const
- void [setHeight](#) (float newHeight) override
- float [getHeight](#) () const override

## Public Member Functions inherited from [IPrimitive](#)

- [IPrimitive](#) ()=default
- virtual [~IPrimitive](#) ()=default
- [PluginType](#) [getType](#) () const override

## Public Member Functions inherited from [IPlugin](#)

- [IPlugin](#) ()=default
- virtual [~IPlugin](#) ()=default

## 6.7.1 Constructor & Destructor Documentation

### 6.7.1.1 [Cube](#)()

```
RayTracer::Cube::Cube ()
```

### 6.7.1.2 [~Cube](#)()

```
RayTracer::Cube::~Cube () [default]
```

## 6.7.2 Member Function Documentation

### 6.7.2.1 [getCenter](#)()

```
const Math::Point3D & RayTracer::Cube::getCenter () const [inline], [override], [virtual]
```

Implements [IPrimitive](#).

### 6.7.2.2 [getColor](#)()

```
const Math::Color & RayTracer::Cube::getColor () const [inline], [override], [virtual]
```

Implements [IPrimitive](#).

### 6.7.2.3 [getHeight](#)()

```
float RayTracer::Cube::getHeight () const [inline], [override], [virtual]
```

Implements [IPrimitive](#).

### 6.7.2.4 [getNormal](#)()

```
Math::Vector3D RayTracer::Cube::getNormal () const [inline], [override], [virtual]
```

Implements [IPrimitive](#).



#### 6.7.2.5 getPosition()

```
Math::Point3D RayTracer::Cube::getPosition () const [inline], [override], [virtual]
```

Implements [IPrimitive](#).

#### 6.7.2.6 getPrimitiveType()

```
PrimitiveType RayTracer::Cube::getPrimitiveType () const [inline], [override], [virtual]
```

Implements [IPrimitive](#).

#### 6.7.2.7 getRadius()

```
float RayTracer::Cube::getRadius () const [inline], [override], [virtual]
```

Implements [IPrimitive](#).

#### 6.7.2.8 getSize()

```
float RayTracer::Cube::getSize () const [inline]
```

#### 6.7.2.9 getType()

```
PluginType RayTracer::Cube::getType () const [inline], [override], [virtual]
```

Implements [IPlugin](#).

#### 6.7.2.10 hits()

```
bool RayTracer::Cube::hits (  
    const Ray & ray) const [override], [virtual]
```

Implements [IPrimitive](#).

#### 6.7.2.11 intersect()

```
bool RayTracer::Cube::intersect (  
    const Ray & ray,  
    RayTracer::HitRecord & rec) const [override], [virtual]
```

Implements [IPrimitive](#).

#### 6.7.2.12 rotate()

```
void RayTracer::Cube::rotate (
    float angle,
    const Math::Vector3D & axis) [override], [virtual]
```

Implements [IPrimitive](#).

#### 6.7.2.13 setColor()

```
void RayTracer::Cube::setColor (
    Math::Color newColor) [inline], [override], [virtual]
```

Implements [IPrimitive](#).

#### 6.7.2.14 setHeight()

```
void RayTracer::Cube::setHeight (
    float newHeight) [inline], [override], [virtual]
```

Implements [IPrimitive](#).

#### 6.7.2.15 setNormal()

```
void RayTracer::Cube::setNormal (
    const Math::Vector3D & newNormal) [inline], [override], [virtual]
```

Implements [IPrimitive](#).

#### 6.7.2.16 setPosition()

```
void RayTracer::Cube::setPosition (
    const Math::Point3D & position) [inline], [override], [virtual]
```

Implements [IPrimitive](#).

#### 6.7.2.17 setRadius()

```
void RayTracer::Cube::setRadius (
    float newSize) [inline], [override], [virtual]
```

Implements [IPrimitive](#).

#### 6.7.2.18 setSize()

```
void RayTracer::Cube::setSize (
    float newSize) [inline]
```

### 6.7.2.19 translate()

```
void RayTracer::Cube::translate (
    const Math::Vector3D & translation) [override], [virtual]
```

Implements [IPrimitive](#).

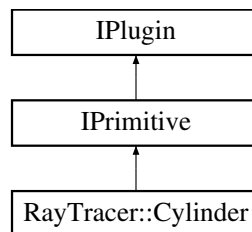
The documentation for this class was generated from the following files:

- [src/plugins/primitives/Cube.hpp](#)
- [src/plugins/primitives/Cube.cpp](#)

## 6.8 RayTracer::Cylinder Class Reference

```
#include <Cylinder.hpp>
```

Inheritance diagram for RayTracer::Cylinder:



### Public Member Functions

- [Cylinder](#) ()
- [~Cylinder](#) ()=default
- bool [hits](#) (const [Ray](#) &ray) const override
- bool [intersect](#) (const [Ray](#) &ray, [HitRecord](#) &rec) const override
- void [translate](#) (const [Math::Vector3D](#) &translation) override
- void [rotate](#) (float angle, const [Math::Vector3D](#) &axis) override
- const [Math::Point3D](#) & [getCenter](#) () const override
- float [getRadius](#) () const override
- const [Math::Color](#) & [getColor](#) () const override
- [PrimitiveType](#) [getPrimitiveType](#) () const override
- [Math::Point3D](#) [getPosition](#) () const override
- void [setPosition](#) (const [Math::Point3D](#) &position) override
- void [setRadius](#) (float newRadius) override
- void [setNormal](#) (const [Math::Vector3D](#) &newNormal) override
- [Math::Vector3D](#) [getNormal](#) () const override
- void [setColor](#) ([Math::Color](#) newColor) override
- float [getHeight](#) () const override
- void [setHeight](#) (float newHeight) override

### Public Member Functions inherited from [IPrimitive](#)

- [IPrimitive](#) ()=default
- virtual [~IPrimitive](#) ()=default
- [PluginType](#) [getType](#) () const override

## Public Member Functions inherited from [IPlugin](#)

- [IPlugin](#) ()=default
- virtual [~IPlugin](#) ()=default

## 6.8.1 Constructor & Destructor Documentation

### 6.8.1.1 Cylinder()

```
RayTracer::Cylinder::Cylinder ()
```

### 6.8.1.2 ~Cylinder()

```
RayTracer::Cylinder::~~Cylinder () [default]
```

## 6.8.2 Member Function Documentation

### 6.8.2.1 getCenter()

```
const Math::Point3D & RayTracer::Cylinder::getCenter () const [inline], [override], [virtual]
```

Implements [IPrimitive](#).

### 6.8.2.2 getColor()

```
const Math::Color & RayTracer::Cylinder::getColor () const [inline], [override], [virtual]
```

Implements [IPrimitive](#).

### 6.8.2.3 getHeight()

```
float RayTracer::Cylinder::getHeight () const [inline], [override], [virtual]
```

Implements [IPrimitive](#).

### 6.8.2.4 getNormal()

```
Math::Vector3D RayTracer::Cylinder::getNormal () const [inline], [override], [virtual]
```

Implements [IPrimitive](#).

### 6.8.2.5 getPosition()

```
Math::Point3D RayTracer::Cylinder::getPosition () const [inline], [override], [virtual]
```

Implements [IPrimitive](#).

### 6.8.2.6 getPrimitiveType()

```
PrimitiveType RayTracer::Cylinder::getPrimitiveType () const [inline], [override], [virtual]
```

Implements [IPrimitive](#).

### 6.8.2.7 getRadius()

```
float RayTracer::Cylinder::getRadius () const [inline], [override], [virtual]
```

Implements [IPrimitive](#).

### 6.8.2.8 hits()

```
bool RayTracer::Cylinder::hits (  
    const Ray & ray) const [override], [virtual]
```

Implements [IPrimitive](#).

### 6.8.2.9 intersect()

```
bool RayTracer::Cylinder::intersect (  
    const Ray & ray,  
    HitRecord & rec) const [override], [virtual]
```

Implements [IPrimitive](#).

### 6.8.2.10 rotate()

```
void RayTracer::Cylinder::rotate (  
    float angle,  
    const Math::Vector3D & axis) [override], [virtual]
```

Implements [IPrimitive](#).

### 6.8.2.11 setColor()

```
void RayTracer::Cylinder::setColor (  
    Math::Color newColor) [inline], [override], [virtual]
```

Implements [IPrimitive](#).

### 6.8.2.12 setHeight()

```
void RayTracer::Cylinder::setHeight (  
    float newHeight) [inline], [override], [virtual]
```

Implements [IPrimitive](#).

### 6.8.2.13 setNormal()

```
void RayTracer::Cylinder::setNormal (
    const Math::Vector3D & newNormal) [inline], [override], [virtual]
```

Implements [IPrimitive](#).

### 6.8.2.14 setPosition()

```
void RayTracer::Cylinder::setPosition (
    const Math::Point3D & position) [inline], [override], [virtual]
```

Implements [IPrimitive](#).

### 6.8.2.15 setRadius()

```
void RayTracer::Cylinder::setRadius (
    float newRadius) [inline], [override], [virtual]
```

Implements [IPrimitive](#).

### 6.8.2.16 translate()

```
void RayTracer::Cylinder::translate (
    const Math::Vector3D & translation) [override], [virtual]
```

Implements [IPrimitive](#).

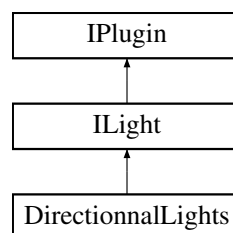
The documentation for this class was generated from the following files:

- [src/plugins/primitives/Cylinder.hpp](#)
- [src/plugins/primitives/Cylinder.cpp](#)

## 6.9 DirectionnalLights Class Reference

```
#include <DirectionnalLights.hpp>
```

Inheritance diagram for DirectionnalLights:



## Public Member Functions

- [DirectionnalLights](#) (float intensity=1.0f, [Math::Vector3D](#) direction=[Math::Vector3D](#)(0, 0, -1))
- [~DirectionnalLights](#) ()=default
- [PluginType](#) [getType](#) () const override
- [LightType](#) [getLightType](#) () const override
- void [setIntensity](#) (float intensity) override
- float [getIntensity](#) () const override
- void [setDirection](#) (float x, float y, float z) override
- [Math::Vector3D](#) [getDirection](#) () const override
- void [applyLight](#) (float &r, float &g, float &b) const override

## Public Member Functions inherited from [ILight](#)

- [ILight](#) ()=default
- virtual [~ILight](#) ()=default

## Public Member Functions inherited from [IPlugin](#)

- [IPlugin](#) ()=default
- virtual [~IPlugin](#) ()=default

## 6.9.1 Constructor & Destructor Documentation

### 6.9.1.1 [DirectionnalLights](#)()

```
DirectionnalLights::DirectionnalLights (
    float intensity = 1.0f,
    Math::Vector3D direction = Math::Vector3D(0, 0, -1))
```

### 6.9.1.2 [~DirectionnalLights](#)()

```
DirectionnalLights::~~DirectionnalLights () [default]
```

## 6.9.2 Member Function Documentation

### 6.9.2.1 [applyLight](#)()

```
void DirectionnalLights::applyLight (
    float & r,
    float & g,
    float & b) const [override], [virtual]
```

Implements [ILight](#).

### 6.9.2.2 `getDirection()`

```
Math::Vector3D DirectionnalLights::getDirection () const [override], [virtual]
```

Implements [ILight](#).

### 6.9.2.3 `getIntensity()`

```
float DirectionnalLights::getIntensity () const [override], [virtual]
```

Implements [ILight](#).

### 6.9.2.4 `getLightType()`

```
LightType DirectionnalLights::getLightType () const [inline], [override], [virtual]
```

Implements [ILight](#).

### 6.9.2.5 `getType()`

```
PluginType DirectionnalLights::getType () const [inline], [override], [virtual]
```

Implements [IPlugin](#).

### 6.9.2.6 `setDirection()`

```
void DirectionnalLights::setDirection (  
    float x,  
    float y,  
    float z) [override], [virtual]
```

Implements [ILight](#).

### 6.9.2.7 `setIntensity()`

```
void DirectionnalLights::setIntensity (  
    float intensity) [override], [virtual]
```

Implements [ILight](#).

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

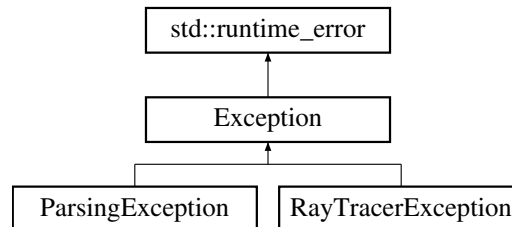
- `src/plugins/lights/DirectionnalLights.hpp`
- `src/plugins/lights/DirectionnalLights.cpp`



## 6.10 Exception Class Reference

```
#include <Exceptions.hpp>
```

Inheritance diagram for Exception:



### Public Member Functions

- [Exception](#) (const std::string &message)

### 6.10.1 Constructor & Destructor Documentation

#### 6.10.1.1 Exception()

```
Exception::Exception (
    const std::string & message) [inline], [explicit]
```

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

- src/exceptions/[Exceptions.hpp](#)

## 6.11 RayTracer::HitRecord Struct Reference

```
#include <HitRecord.hpp>
```

### Public Attributes

- [Math::Point3D](#) point = [Math::Point3D](#)(0,0,0)
- [Math::Vector3D](#) normal = [Math::Vector3D](#)(0,0,0)
- float t = 0
- [Math::Color](#) color = [Math::Color](#)(0,0,0)

### 6.11.1 Member Data Documentation

#### 6.11.1.1 color

```
Math::Color RayTracer::HitRecord::color = Math::Color(0,0,0)
```

### 6.11.1.2 normal

```
Math::Vector3D RayTracer::HitRecord::normal = Math::Vector3D(0,0,0)
```

### 6.11.1.3 point

```
Math::Point3D RayTracer::HitRecord::point = Math::Point3D(0,0,0)
```

### 6.11.1.4 t

```
float RayTracer::HitRecord::t = 0
```

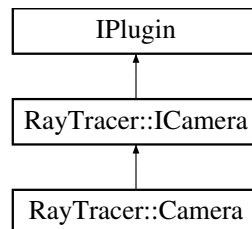
The documentation for this struct was generated from the following file:

- [src/plugins/primitives/HitRecord.hpp](#)

## 6.12 RayTracer::ICamera Class Reference

```
#include <ICamera.hpp>
```

Inheritance diagram for RayTracer::ICamera:



### Public Member Functions

- virtual [~ICamera](#) ()=default
- virtual void [rotate](#) (float angle, const [Math::Vector3D](#) &axis)=0
- virtual void [translate](#) (const [Math::Vector3D](#) &translation)=0
- virtual void [setPosition](#) ([Math::Point3D](#) position)=0
- virtual [Math::Point3D](#) [getPosition](#) () const =0
- virtual void [setResolution](#) (int width, int height)=0
- virtual void [getResolution](#) (int &width, int &height) const =0
- virtual void [setFieldOfView](#) (float fov)=0
- virtual float [getFieldOfView](#) () const =0
- virtual int [getWidth](#) () const =0
- virtual int [getHeight](#) () const =0
- virtual void [updateScreen](#) ()=0
- virtual [Ray](#) [rayAt](#) (double u, double v) const =0
- virtual void [setLookAt](#) (const [Math::Point3D](#) &lookAt)=0
- virtual [Math::Point3D](#) [getLookAt](#) () const =0

## Public Member Functions inherited from [IPlugin](#)

- [IPlugin](#) ()=default
- virtual [~IPlugin](#) ()=default
- virtual [PluginType](#) [getType](#) () const =0

## 6.12.1 Constructor & Destructor Documentation

### 6.12.1.1 [~ICamera\(\)](#)

```
virtual RayTracer::ICamera::~~ICamera () [virtual], [default]
```

## 6.12.2 Member Function Documentation

### 6.12.2.1 [getFieldOfView\(\)](#)

```
virtual float RayTracer::ICamera::getFieldOfView () const [pure virtual]
```

Implemented in [RayTracer::Camera](#).

### 6.12.2.2 [getHeight\(\)](#)

```
virtual int RayTracer::ICamera::getHeight () const [pure virtual]
```

Implemented in [RayTracer::Camera](#).

### 6.12.2.3 [getLookAt\(\)](#)

```
virtual Math::Point3D RayTracer::ICamera::getLookAt () const [pure virtual]
```

Implemented in [RayTracer::Camera](#).

### 6.12.2.4 [getPosition\(\)](#)

```
virtual Math::Point3D RayTracer::ICamera::getPosition () const [pure virtual]
```

Implemented in [RayTracer::Camera](#).

### 6.12.2.5 [getResolution\(\)](#)

```
virtual void RayTracer::ICamera::getResolution (  
    int & width,  
    int & height) const [pure virtual]
```

Implemented in [RayTracer::Camera](#).

#### 6.12.2.6 getWidth()

```
virtual int RayTracer::ICamera::getWidth () const [pure virtual]
```

Implemented in [RayTracer::Camera](#).

#### 6.12.2.7 rayAt()

```
virtual Ray RayTracer::ICamera::rayAt (  
    double u,  
    double v) const [pure virtual]
```

Implemented in [RayTracer::Camera](#).

#### 6.12.2.8 rotate()

```
virtual void RayTracer::ICamera::rotate (  
    float angle,  
    const Math::Vector3D & axis) [pure virtual]
```

Implemented in [RayTracer::Camera](#).

#### 6.12.2.9 setFieldOfView()

```
virtual void RayTracer::ICamera::setFieldOfView (  
    float fov) [pure virtual]
```

Implemented in [RayTracer::Camera](#).

#### 6.12.2.10 setLookAt()

```
virtual void RayTracer::ICamera::setLookAt (  
    const Math::Point3D & lookAt) [pure virtual]
```

Implemented in [RayTracer::Camera](#).

#### 6.12.2.11 setPosition()

```
virtual void RayTracer::ICamera::setPosition (  
    Math::Point3D position) [pure virtual]
```

Implemented in [RayTracer::Camera](#).

#### 6.12.2.12 setResolution()

```
virtual void RayTracer::ICamera::setResolution (  
    int width,  
    int height) [pure virtual]
```

Implemented in [RayTracer::Camera](#).

## 6.12.2.13 translate()

```
virtual void RayTracer::ICamera::translate (
    const Math::Vector3D & translation) [pure virtual]
```

Implemented in [RayTracer::Camera](#).

## 6.12.2.14 updateScreen()

```
virtual void RayTracer::ICamera::updateScreen () [pure virtual]
```

Implemented in [RayTracer::Camera](#).

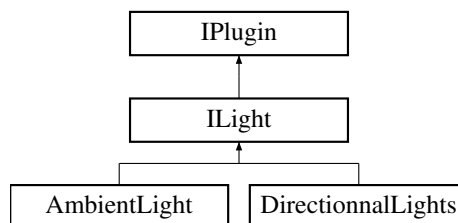
The documentation for this class was generated from the following file:

- [src/plugins/camera/ICamera.hpp](#)

## 6.13 ILight Class Reference

```
#include <ILight.hpp>
```

Inheritance diagram for ILight:



## Public Member Functions

- [ILight](#) ()=default
- virtual [~ILight](#) ()=default
- virtual [LightType](#) [getLightType](#) () const =0
- virtual void [setIntensity](#) (float intensity)=0
- virtual float [getIntensity](#) () const =0
- virtual void [applyLight](#) (float &r, float &g, float &b) const =0
- virtual void [setDirection](#) (float x, float y, float z)=0
- virtual [Math::Vector3D](#) [getDirection](#) () const =0

Public Member Functions inherited from [IPlugin](#)

- [IPlugin](#) ()=default
- virtual [~IPlugin](#) ()=default
- virtual [PluginType](#) [getType](#) () const =0

## 6.13.1 Constructor & Destructor Documentation

### 6.13.1.1 ILight()

```
ILight::ILight () [default]
```

### 6.13.1.2 ~ILight()

```
virtual ILight::~~ILight () [virtual], [default]
```

## 6.13.2 Member Function Documentation

### 6.13.2.1 applyLight()

```
virtual void ILight::applyLight (  
    float & r,  
    float & g,  
    float & b) const [pure virtual]
```

Implemented in [AmbientLight](#), and [DirectionnalLights](#).

### 6.13.2.2 getDirection()

```
virtual Math::Vector3D ILight::getDirection () const [pure virtual]
```

Implemented in [AmbientLight](#), and [DirectionnalLights](#).

### 6.13.2.3 getIntensity()

```
virtual float ILight::getIntensity () const [pure virtual]
```

Implemented in [AmbientLight](#), and [DirectionnalLights](#).

### 6.13.2.4 getLightType()

```
virtual LightType ILight::getLightType () const [pure virtual]
```

Implemented in [AmbientLight](#), and [DirectionnalLights](#).

### 6.13.2.5 setDirection()

```
virtual void ILight::setDirection (  
    float x,  
    float y,  
    float z) [pure virtual]
```

Implemented in [AmbientLight](#), and [DirectionnalLights](#).

### 6.13.2.6 setIntensity()

```
virtual void ILight::setIntensity (
    float intensity) [pure virtual]
```

Implemented in [AmbientLight](#), and [DirectionnalLights](#).

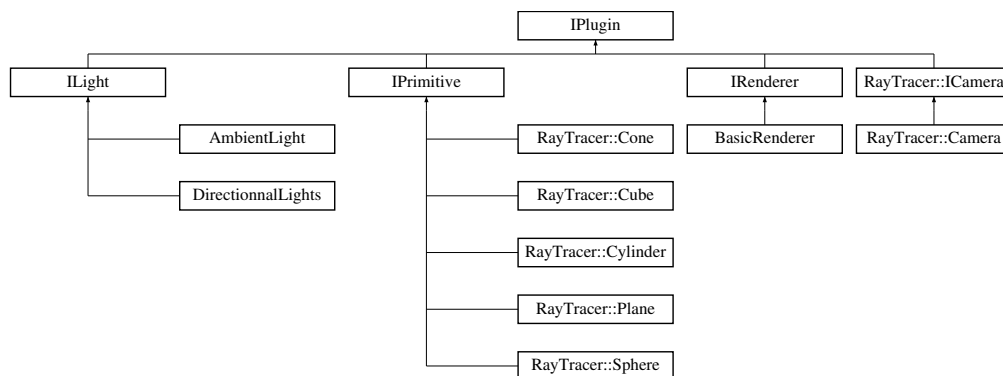
The documentation for this class was generated from the following file:

- [src/plugins/lights/ILight.hpp](#)

## 6.14 IPlugin Class Reference

```
#include <IPlugin.hpp>
```

Inheritance diagram for IPlugin:



### Public Member Functions

- [IPlugin\(\)](#)=default
- virtual [~IPlugin\(\)](#)=default
- virtual [PluginType getType\(\)](#) const =0

### 6.14.1 Constructor & Destructor Documentation

#### 6.14.1.1 IPlugin()

```
IPlugin::IPlugin () [default]
```

#### 6.14.1.2 ~IPlugin()

```
virtual IPlugin::~~IPlugin () [virtual], [default]
```

## 6.14.2 Member Function Documentation

### 6.14.2.1 getType()

```
virtual PluginType IPlugin::getType () const [pure virtual]
```

Implemented in [AmbientLight](#), [BasicRenderer](#), [DirectionnalLights](#), [IPrimitive](#), [RayTracer::Camera](#), [RayTracer::Cone](#), [RayTracer::Cube](#), [RayTracer::Plane](#), and [RayTracer::Sphere](#).

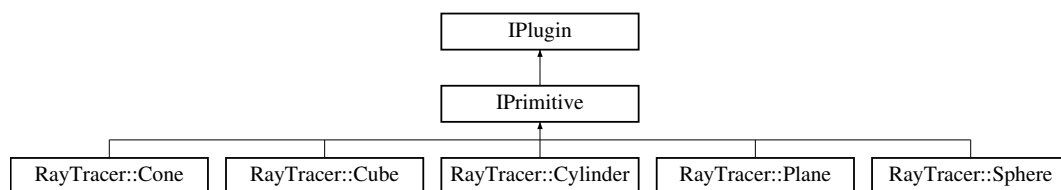
The documentation for this class was generated from the following file:

- [src/plugins/IPlugin.hpp](#)

## 6.15 IPrimitive Class Reference

```
#include <IPrimitive.hpp>
```

Inheritance diagram for IPrimitive:



### Public Member Functions

- [IPrimitive](#) ()=default
- virtual [~IPrimitive](#) ()=default
- virtual bool [hits](#) (const [RayTracer::Ray](#) &ray) const =0
- virtual void [translate](#) (const [Math::Vector3D](#) &translation)=0
- virtual void [rotate](#) (float angle, const [Math::Vector3D](#) &axis)=0
- virtual bool [intersect](#) (const [RayTracer::Ray](#) &ray, [RayTracer::HitRecord](#) &rec) const =0
- virtual const [Math::Point3D](#) & [getCenter](#) () const =0
- virtual float [getRadius](#) () const =0
- virtual const [Math::Color](#) & [getColor](#) () const =0
- virtual [Math::Point3D](#) [getPosition](#) () const =0
- virtual [PrimitiveType](#) [getPrimitiveType](#) () const =0
- virtual [Math::Vector3D](#) [getNormal](#) () const =0
- virtual float [getHeight](#) () const =0
- virtual void [setPosition](#) (const [Math::Point3D](#) &position)=0
- virtual void [setRadius](#) (float radius)=0
- virtual void [setColor](#) ([Math::Color](#) color)=0
- virtual void [setNormal](#) (const [Math::Vector3D](#) &normal)=0
- virtual void [setHeight](#) (float height)=0
- [PluginType](#) [getType](#) () const override



## Public Member Functions inherited from [IPPlugin](#)

- [IPPlugin](#) ()=default
- virtual [~IPPlugin](#) ()=default

### 6.15.1 Constructor & Destructor Documentation

#### 6.15.1.1 IPrimitive()

```
IPrimitive::IPrimitive () [default]
```

#### 6.15.1.2 ~IPrimitive()

```
virtual IPrimitive::~~IPrimitive () [virtual], [default]
```

### 6.15.2 Member Function Documentation

#### 6.15.2.1 getCenter()

```
virtual const Math::Point3D & IPrimitive::getCenter () const [pure virtual]
```

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

#### 6.15.2.2 getColor()

```
virtual const Math::Color & IPrimitive::getColor () const [pure virtual]
```

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

#### 6.15.2.3 getHeight()

```
virtual float IPrimitive::getHeight () const [pure virtual]
```

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

#### 6.15.2.4 getNormal()

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

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

#### 6.15.2.5 getPosition()

```
virtual Math::Point3D IPrimitive::getPosition () const [pure virtual]
```

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

#### 6.15.2.6 getPrimitiveType()

```
virtual PrimitiveType IPrimitive::getPrimitiveType () const [pure virtual]
```

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

#### 6.15.2.7 getRadius()

```
virtual float IPrimitive::getRadius () const [pure virtual]
```

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

#### 6.15.2.8 getType()

```
PluginType IPrimitive::getType () const [inline], [override], [virtual]
```

Implements [IPlugin](#).

Reimplemented in [RayTracer::Plane](#), and [RayTracer::Sphere](#).

#### 6.15.2.9 hits()

```
virtual bool IPrimitive::hits (  
    const RayTracer::Ray & ray) const [pure virtual]
```

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

#### 6.15.2.10 intersect()

```
virtual bool IPrimitive::intersect (  
    const RayTracer::Ray & ray,  
    RayTracer::HitRecord & rec) const [pure virtual]
```

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

#### 6.15.2.11 rotate()

```
virtual void IPrimitive::rotate (  
    float angle,  
    const Math::Vector3D & axis) [pure virtual]
```

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

#### 6.15.2.12 setColor()

```
virtual void IPrimitive::setColor (  
    Math::Color color) [pure virtual]
```

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

**6.15.2.13 setHeight()**

```
virtual void IPrimitive::setHeight (
    float height) [pure virtual]
```

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

**6.15.2.14 setNormal()**

```
virtual void IPrimitive::setNormal (
    const Math::Vector3D & normal) [pure virtual]
```

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

**6.15.2.15 setPosition()**

```
virtual void IPrimitive::setPosition (
    const Math::Point3D & position) [pure virtual]
```

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

**6.15.2.16 setRadius()**

```
virtual void IPrimitive::setRadius (
    float radius) [pure virtual]
```

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

**6.15.2.17 translate()**

```
virtual void IPrimitive::translate (
    const Math::Vector3D & translation) [pure virtual]
```

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

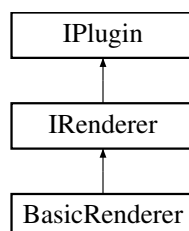
The documentation for this class was generated from the following file:

- [src/plugins/primitives/IPrimitive.hpp](#)

**6.16 IRenderer Class Reference**

```
#include <IRenderer.hpp>
```

Inheritance diagram for IRenderer:



## Public Member Functions

- [IRenderer](#) ()=default
- virtual [~IRenderer](#) ()=default
- virtual void [setRendererType](#) ([RendererType](#) type)=0
- virtual [RendererType](#) [getRendererType](#) () const =0
- virtual void [renderScene](#) ()=0
- virtual void [setCamera](#) ([RayTracer::ICamera](#) \*camera)=0
- virtual void [setLights](#) (std::vector< [ILight](#) \* > &lights)=0
- virtual void [setPrimitives](#) (std::vector< [IPrimitive](#) \* > &primitives)=0

## Public Member Functions inherited from [IPlugin](#)

- [IPlugin](#) ()=default
- virtual [~IPlugin](#) ()=default
- virtual [PluginType](#) [getType](#) () const =0

## 6.16.1 Constructor & Destructor Documentation

### 6.16.1.1 [IRenderer](#)()

```
IRenderer::IRenderer () [default]
```

### 6.16.1.2 [~IRenderer](#)()

```
virtual IRenderer::~~IRenderer () [virtual], [default]
```

## 6.16.2 Member Function Documentation

### 6.16.2.1 [getRendererType](#)()

```
virtual RendererType IRenderer::getRendererType () const [pure virtual]
```

Implemented in [BasicRenderer](#).

### 6.16.2.2 [renderScene](#)()

```
virtual void IRenderer::renderScene () [pure virtual]
```

Implemented in [BasicRenderer](#).

### 6.16.2.3 [setCamera](#)()

```
virtual void IRenderer::setCamera (  
    RayTracer::ICamera * camera) [pure virtual]
```

Implemented in [BasicRenderer](#).

#### 6.16.2.4 setLights()

```
virtual void IRenderer::setLights (
    std::vector< ILight * > & lights) [pure virtual]
```

Implemented in [BasicRenderer](#).

#### 6.16.2.5 setPrimitives()

```
virtual void IRenderer::setPrimitives (
    std::vector< IPrimitive * > & primitives) [pure virtual]
```

Implemented in [BasicRenderer](#).

#### 6.16.2.6 setRendererType()

```
virtual void IRenderer::setRendererType (
    RendererType type) [pure virtual]
```

Implemented in [BasicRenderer](#).

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

- [src/plugins/render/IRenderer.hpp](#)

## 6.17 Math::Matrix4x4 Class Reference

```
#include <Matrix.hpp>
```

### Public Member Functions

- [Matrix4x4](#) ()
- [~Matrix4x4](#) ()=default
- void [identity](#) ()
- void [rotationMatrix](#) (float angle, const [Vector3D](#) &axis)
- [Point3D](#) [applyToPoint](#) (const [Point3D](#) &point) const
- [Vector3D](#) [applyToVector](#) (const [Vector3D](#) &vector) const

### 6.17.1 Constructor & Destructor Documentation

#### 6.17.1.1 Matrix4x4()

```
Math::Matrix4x4::Matrix4x4 ()
```

### 6.17.1.2 ~Matrix4x4()

```
Math::Matrix4x4::~~Matrix4x4 () [default]
```

## 6.17.2 Member Function Documentation

### 6.17.2.1 applyToPoint()

```
Math::Point3D Math::Matrix4x4::applyToPoint (  
    const Point3D & point) const
```

### 6.17.2.2 applyToVector()

```
Math::Vector3D Math::Matrix4x4::applyToVector (  
    const Vector3D & vector) const
```

### 6.17.2.3 identity()

```
void Math::Matrix4x4::identity ()
```

### 6.17.2.4 rotationMatrix()

```
void Math::Matrix4x4::rotationMatrix (  
    float angle,  
    const Vector3D & axis)
```

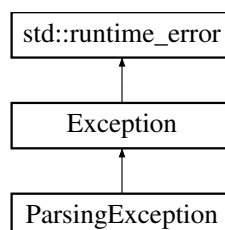
The documentation for this class was generated from the following files:

- [src/plugins/math/Matrix.hpp](#)
- [src/plugins/math/Matrix.cpp](#)

## 6.18 ParsingException Class Reference

```
#include <Exceptions.hpp>
```

Inheritance diagram for ParsingException:



**Public Member Functions**

- [ParsingException](#) (const std::string &message)

**Public Member Functions inherited from [Exception](#)**

- [Exception](#) (const std::string &message)

**6.18.1 Constructor & Destructor Documentation****6.18.1.1 ParsingException()**

```
ParsingException::ParsingException (
    const std::string & message) [inline], [explicit]
```

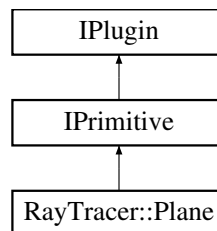
The documentation for this class was generated from the following file:

- src/exceptions/[Exceptions.hpp](#)

**6.19 RayTracer::Plane Class Reference**

```
#include <Plane.hpp>
```

Inheritance diagram for RayTracer::Plane:

**Public Member Functions**

- [Plane](#) ()
- [~Plane](#) ()=default
- bool [hits](#) (const [RayTracer::Ray](#) &ray) const override
- void [translate](#) (const [Math::Vector3D](#) &translation) override
- void [rotate](#) (float angle, const [Math::Vector3D](#) &axis) override
- virtual bool [intersect](#) (const [RayTracer::Ray](#) &ray, [RayTracer::HitRecord](#) &rec) const override
- const [Math::Point3D](#) & [getCenter](#) () const override
- float [getRadius](#) () const override
- const [Math::Color](#) & [getColor](#) () const override
- [PluginType](#) [getType](#) () const override
- void [setPosition](#) (const [Math::Point3D](#) &newPoint) override
- [Math::Point3D](#) [getPosition](#) () const override
- void [setRadius](#) (float newRadius) override
- void [setNormal](#) (const [Math::Vector3D](#) &newNormal) override
- [Math::Vector3D](#) [getNormal](#) () const override
- void [setColor](#) ([Math::Color](#) newColor) override
- [PrimitiveType](#) [getPrimitiveType](#) () const override
- void [setHeight](#) (float newHeight) override
- float [getHeight](#) () const override

## Public Member Functions inherited from [IPrimitive](#)

- [IPrimitive](#) ()=default
- virtual [~IPrimitive](#) ()=default

## Public Member Functions inherited from [IPlugin](#)

- [IPlugin](#) ()=default
- virtual [~IPlugin](#) ()=default

## 6.19.1 Constructor & Destructor Documentation

### 6.19.1.1 Plane()

```
RayTracer::Plane::Plane ()
```

### 6.19.1.2 ~Plane()

```
RayTracer::Plane::~~Plane () [default]
```

## 6.19.2 Member Function Documentation

### 6.19.2.1 getCenter()

```
const Math::Point3D & RayTracer::Plane::getCenter () const [inline], [override], [virtual]
```

Implements [IPrimitive](#).

### 6.19.2.2 getColor()

```
const Math::Color & RayTracer::Plane::getColor () const [inline], [override], [virtual]
```

Implements [IPrimitive](#).

### 6.19.2.3 getHeight()

```
float RayTracer::Plane::getHeight () const [inline], [override], [virtual]
```

Implements [IPrimitive](#).

### 6.19.2.4 getNormal()

```
Math::Vector3D RayTracer::Plane::getNormal () const [inline], [override], [virtual]
```

Implements [IPrimitive](#).



### 6.19.2.5 getPosition()

```
Math::Point3D RayTracer::Plane::getPosition () const [inline], [override], [virtual]
```

Implements [IPrimitive](#).

### 6.19.2.6 getPrimitiveType()

```
PrimitiveType RayTracer::Plane::getPrimitiveType () const [inline], [override], [virtual]
```

Implements [IPrimitive](#).

### 6.19.2.7 getRadius()

```
float RayTracer::Plane::getRadius () const [inline], [override], [virtual]
```

Implements [IPrimitive](#).

### 6.19.2.8 getType()

```
PluginType RayTracer::Plane::getType () const [inline], [override], [virtual]
```

Reimplemented from [IPrimitive](#).

### 6.19.2.9 hits()

```
bool RayTracer::Plane::hits (  
    const RayTracer::Ray & ray) const [override], [virtual]
```

Implements [IPrimitive](#).

### 6.19.2.10 intersect()

```
bool RayTracer::Plane::intersect (  
    const RayTracer::Ray & ray,  
    RayTracer::HitRecord & rec) const [override], [virtual]
```

Implements [IPrimitive](#).

### 6.19.2.11 rotate()

```
void RayTracer::Plane::rotate (  
    float angle,  
    const Math::Vector3D & axis) [override], [virtual]
```

Implements [IPrimitive](#).

#### 6.19.2.12 setColor()

```
void RayTracer::Plane::setColor (
    Math::Color newColor) [inline], [override], [virtual]
```

Implements [IPrimitive](#).

#### 6.19.2.13 setHeight()

```
void RayTracer::Plane::setHeight (
    float newHeight) [inline], [override], [virtual]
```

Implements [IPrimitive](#).

#### 6.19.2.14 setNormal()

```
void RayTracer::Plane::setNormal (
    const Math::Vector3D & newNormal) [inline], [override], [virtual]
```

Implements [IPrimitive](#).

#### 6.19.2.15 setPosition()

```
void RayTracer::Plane::setPosition (
    const Math::Point3D & newPoint) [inline], [override], [virtual]
```

Implements [IPrimitive](#).

#### 6.19.2.16 setRadius()

```
void RayTracer::Plane::setRadius (
    float newRadius) [inline], [override], [virtual]
```

Implements [IPrimitive](#).

#### 6.19.2.17 translate()

```
void RayTracer::Plane::translate (
    const Math::Vector3D & translation) [override], [virtual]
```

Implements [IPrimitive](#).

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

- [src/plugins/primitives/Plane.hpp](#)
- [src/plugins/primitives/Plane.cpp](#)

## 6.20 PluginLoader Class Reference

```
#include <PluginLoader.hpp>
```

### Public Member Functions

- [PluginLoader](#) ()=default
- [~PluginLoader](#) ()=default
- void [storeHandle](#) (void \*handle, const std::string &path)
- [IPlugin \\*](#) [loadCameraPlugin](#) (const std::string &path)
- std::vector< [IPlugin \\*](#) > [loadlightsPlugin](#) (const std::string &path)
- [IPlugin \\*](#) [loadRenderPlugin](#) (const std::string &path)
- void \* [loadSharedLibrary](#) (const std::string &path, void \*&handle)
- void \* [loadSymbol](#) (void \*handle, const std::string &symbolName)
- [IPlugin \\*](#) [createPrimitive](#) (const std::string &libPath, const std::string &symbolName)
- void [unloadCamera](#) ([IPlugin \\*](#)camera)
- void [unloadRender](#) ([IPlugin \\*](#)plugin)
- void [unloadLights](#) (std::vector< [IPlugin \\*](#) > lights)
- void [unloadPrimitives](#) (std::vector< [IPlugin \\*](#) > primitive)

### Static Public Member Functions

- static void \*& [chooseHandleByPath](#) (const std::string &path)
- static void \*& [getInstance](#) (const std::string &path)

### Public Attributes

- [IPlugin \\*](#) camera
- [IPlugin \\*](#) render
- std::vector< [IPlugin \\*](#) > lights
- std::vector< [IPlugin \\*](#) > primitive

### Static Public Attributes

- static void \* [planeHandle](#) = nullptr
- static void \* [sphereHandle](#) = nullptr
- static void \* [cylinderHandle](#) = nullptr
- static void \* [coneHandle](#) = nullptr
- static void \* [cubeHandle](#) = nullptr

## 6.20.1 Constructor & Destructor Documentation

### 6.20.1.1 PluginLoader()

```
PluginLoader::PluginLoader () [default]
```

### 6.20.1.2 ~PluginLoader()

```
PluginLoader::~~PluginLoader () [default]
```

## 6.20.2 Member Function Documentation

### 6.20.2.1 chooseHandleByPath()

```
static void *& PluginLoader::chooseHandleByPath (  
    const std::string & path) [static]
```

### 6.20.2.2 createPrimitive()

```
IPlugin * PluginLoader::createPrimitive (  
    const std::string & libPath,  
    const std::string & symbolName)
```

### 6.20.2.3 getInstance()

```
void *& PluginLoader::getInstance (  
    const std::string & path) [static]
```

### 6.20.2.4 loadCameraPlugin()

```
IPlugin * PluginLoader::loadCameraPlugin (  
    const std::string & path)
```

### 6.20.2.5 loadlightsPlugin()

```
std::vector< IPlugin * > PluginLoader::loadlightsPlugin (  
    const std::string & path)
```

### 6.20.2.6 loadRenderPlugin()

```
IPlugin * PluginLoader::loadRenderPlugin (  
    const std::string & path)
```

### 6.20.2.7 loadSharedLibrary()

```
void * PluginLoader::loadSharedLibrary (  
    const std::string & path,  
    void *& handle)
```

### 6.20.2.8 loadSymbol()

```
void * PluginLoader::loadSymbol (
    void * handle,
    const std::string & symbolName)
```

### 6.20.2.9 storeHandle()

```
void PluginLoader::storeHandle (
    void * handle,
    const std::string & path)
```

### 6.20.2.10 unloadCamera()

```
void PluginLoader::unloadCamera (
    IPlugin * camera)
```

### 6.20.2.11 unloadLights()

```
void PluginLoader::unloadLights (
    std::vector< IPlugin * > lights)
```

### 6.20.2.12 unloadPrimitives()

```
void PluginLoader::unloadPrimitives (
    std::vector< IPlugin * > primitive)
```

### 6.20.2.13 unloadRender()

```
void PluginLoader::unloadRender (
    IPlugin * plugin)
```

## 6.20.3 Member Data Documentation

### 6.20.3.1 camera

```
IPlugin* PluginLoader::camera
```

### 6.20.3.2 coneHandle

```
void * PluginLoader::coneHandle = nullptr [static]
```

### 6.20.3.3 cubeHandle

```
void * PluginLoader::cubeHandle = nullptr [static]
```

### 6.20.3.4 cylinderHandle

```
void * PluginLoader::cylinderHandle = nullptr [static]
```

### 6.20.3.5 lights

```
std::vector<IPlugin *> PluginLoader::lights
```

### 6.20.3.6 planeHandle

```
void * PluginLoader::planeHandle = nullptr [static]
```

### 6.20.3.7 primitive

```
std::vector<IPlugin *> PluginLoader::primitive
```

### 6.20.3.8 render

```
IPlugin* PluginLoader::render
```

### 6.20.3.9 sphereHandle

```
void * PluginLoader::sphereHandle = nullptr [static]
```

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

- src/loaders/[PluginLoader.hpp](#)
- src/loaders/[PluginLoader.cpp](#)

## 6.21 Math::Point3D Class Reference

```
#include <Point.hpp>
```

### Public Member Functions

- [Point3D](#) (float x, float y, float z)
- [~Point3D](#) ()=default
- float [getX](#) () const
- float [getY](#) () const
- float [getZ](#) () const
- [Point3D operator+](#) (const [Point3D](#) &other) const
- [Point3D operator+=](#) (const [Point3D](#) &other)

## 6.21.1 Constructor & Destructor Documentation

### 6.21.1.1 Point3D()

```
Math::Point3D::Point3D (
    float x,
    float y,
    float z) [inline]
```

### 6.21.1.2 ~Point3D()

```
Math::Point3D::~~Point3D () [default]
```

## 6.21.2 Member Function Documentation

### 6.21.2.1 getX()

```
float Math::Point3D::getX () const [inline]
```

### 6.21.2.2 getY()

```
float Math::Point3D::getY () const [inline]
```

### 6.21.2.3 getZ()

```
float Math::Point3D::getZ () const [inline]
```

### 6.21.2.4 operator+()

```
Point3D Math::Point3D::operator+ (
    const Point3D & other) const [inline]
```

### 6.21.2.5 operator+=()

```
Point3D Math::Point3D::operator+= (
    const Point3D & other) [inline]
```

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

- src/plugins/math/[Point.hpp](#)

## 6.22 RayTracer::Ray Class Reference

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

## Public Member Functions

- [Ray](#) (const [Math::Point3D](#) &origin, const [Math::Vector3D](#) &direction)
- const [Math::Point3D](#) & [getOrigin](#) () const
- const [Math::Vector3D](#) & [getDirection](#) () const

## 6.22.1 Constructor & Destructor Documentation

### 6.22.1.1 Ray()

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

## 6.22.2 Member Function Documentation

### 6.22.2.1 getDirection()

```
const Math::Vector3D & RayTracer::Ray::getDirection () const [inline]
```

### 6.22.2.2 getOrigin()

```
const Math::Point3D & RayTracer::Ray::getOrigin () const [inline]
```

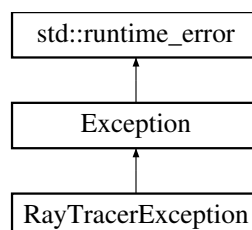
The documentation for this class was generated from the following files:

- src/plugins/primitives/[Ray.hpp](#)
- src/plugins/primitives/[Ray.cpp](#)

## 6.23 RayTracerException Class Reference

```
#include <Exceptions.hpp>
```

Inheritance diagram for RayTracerException:



## Public Member Functions

- [RayTracerException](#) (const std::string &message)



## Public Member Functions inherited from [Exception](#)

- [Exception](#) (const std::string &message)

### 6.23.1 Constructor & Destructor Documentation

#### 6.23.1.1 RayTracerException()

```
RayTracerException::RayTracerException (
    const std::string & message) [inline], [explicit]
```

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

- src/exceptions/[Exceptions.hpp](#)

## 6.24 Math::Rectangle3D Class Reference

```
#include <Rectangle.hpp>
```

### Public Member Functions

- [Rectangle3D](#) (const [Math::Point3D](#) &origin, const [Math::Vector3D](#) &bottom\_side, const [Math::Vector3D](#) &left\_side)
- [Math::Point3D pointAt](#) (double u, double v) const

### Public Attributes

- [Math::Point3D origin](#)
- [Math::Vector3D bottom\\_side](#)
- [Math::Vector3D left\\_side](#)

### 6.24.1 Constructor & Destructor Documentation

#### 6.24.1.1 Rectangle3D()

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

### 6.24.2 Member Function Documentation

#### 6.24.2.1 pointAt()

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

### 6.24.3 Member Data Documentation

#### 6.24.3.1 bottom\_side

`Math::Vector3D Math::Rectangle3D::bottom_side`

#### 6.24.3.2 left\_side

`Math::Vector3D Math::Rectangle3D::left_side`

#### 6.24.3.3 origin

`Math::Point3D Math::Rectangle3D::origin`

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

- `src/plugins/math/Rectangle.hpp`
- `src/plugins/math/Rectangle.cpp`

## 6.25 SceneLoader Class Reference

```
#include <SceneLoader.hpp>
```

### Public Member Functions

- `SceneLoader()`=default
- `~SceneLoader()`=default
- `bool checkCfgError (const std::string &filename)`
- `void loadCamera (IPlugin *camera)`
- `void loadLights (std::vector< IPlugin * > lights)`
- `void loadRender (IPlugin *plugin)`
- `std::vector< IPlugin * > loadPrimitives (PluginLoader &pluginLoader)`
- `void fillSphere (IPlugin *sphere, Setting &sphereSetting)`
- `void fillPlane (IPlugin *plane, Setting &planeSetting)`
- `void fillCylinder (IPlugin *cylinder, Setting &cylinderSetting)`
- `void fillCone (IPlugin *cone, Setting &coneSetting)`
- `void fillCube (IPlugin *cube, Setting &cubeSetting)`
- `template<typename T>`  
`T * castPlugin (IPlugin *plugin)`

### 6.25.1 Constructor & Destructor Documentation

#### 6.25.1.1 SceneLoader()

`SceneLoader::SceneLoader () [default]`

### 6.25.1.2 ~SceneLoader()

```
SceneLoader::~~SceneLoader () [default]
```

## 6.25.2 Member Function Documentation

### 6.25.2.1 castPlugin()

```
template<typename T>  
T * SceneLoader::castPlugin (  
    IPlugin * plugin) [inline]
```

### 6.25.2.2 checkCfgError()

```
bool SceneLoader::checkCfgError (  
    const std::string & filename)
```

### 6.25.2.3 fillCone()

```
void SceneLoader::fillCone (  
    IPlugin * cone,  
    Setting & coneSetting)
```

### 6.25.2.4 fillCube()

```
void SceneLoader::fillCube (  
    IPlugin * cube,  
    Setting & cubeSetting)
```

### 6.25.2.5 fillCylinder()

```
void SceneLoader::fillCylinder (  
    IPlugin * cylinder,  
    Setting & cylinderSetting)
```

### 6.25.2.6 fillPlane()

```
void SceneLoader::fillPlane (  
    IPlugin * plane,  
    Setting & planeSetting)
```

### 6.25.2.7 fillSphere()

```
void SceneLoader::fillSphere (  
    IPlugin * sphere,  
    Setting & sphereSetting)
```

### 6.25.2.8 loadCamera()

```
void SceneLoader::loadCamera (  
    IPlugin * camera)
```

### 6.25.2.9 loadLights()

```
void SceneLoader::loadLights (  
    std::vector< IPlugin * > lights)
```

### 6.25.2.10 loadPrimitives()

```
std::vector< IPlugin * > SceneLoader::loadPrimitives (  
    PluginLoader & pluginLoader)
```

### 6.25.2.11 loadRender()

```
void SceneLoader::loadRender (  
    IPlugin * plugin)
```

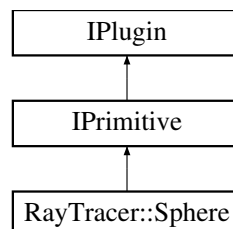
The documentation for this class was generated from the following files:

- src/loaders/[SceneLoader.hpp](#)
- src/loaders/[SceneLoader.cpp](#)

## 6.26 RayTracer::Sphere Class Reference

```
#include <Sphere.hpp>
```

Inheritance diagram for RayTracer::Sphere:



## Public Member Functions

- [Sphere](#) ()
- [~Sphere](#) ()=default
- [PluginType](#) [getType](#) () const override
- [PrimitiveType](#) [getPrimitiveType](#) () const override
- bool [hits](#) (const [Ray](#) &ray) const override
- bool [intersect](#) (const [Ray](#) &ray, [RayTracer::HitRecord](#) &rec) const override
- void [translate](#) (const [Math::Vector3D](#) &translation) override
- void [rotate](#) (float angle, const [Math::Vector3D](#) &axis) override
- const [Math::Point3D](#) & [getCenter](#) () const override
- float [getRadius](#) () const override
- const [Math::Color](#) & [getColor](#) () const override
- void [setPosition](#) (const [Math::Point3D](#) &newCenter) override
- [Math::Point3D](#) [getPosition](#) () const override
- void [setRadius](#) (float newRadius) override
- void [setNormal](#) (const [Math::Vector3D](#) &newNormal) override
- [Math::Vector3D](#) [getNormal](#) () const override
- void [setColor](#) ([Math::Color](#) newColor) override
- float [getHeight](#) () const override
- void [setHeight](#) (float newHeight) override

## Public Member Functions inherited from [IPrimitive](#)

- [IPrimitive](#) ()=default
- virtual [~IPrimitive](#) ()=default

## Public Member Functions inherited from [IPlugin](#)

- [IPlugin](#) ()=default
- virtual [~IPlugin](#) ()=default

## 6.26.1 Constructor & Destructor Documentation

### 6.26.1.1 Sphere()

```
RayTracer::Sphere::Sphere ()
```

### 6.26.1.2 ~Sphere()

```
RayTracer::Sphere::~Sphere () [default]
```

## 6.26.2 Member Function Documentation

### 6.26.2.1 getCenter()

```
const Math::Point3D & RayTracer::Sphere::getCenter () const [inline], [override], [virtual]
```

Implements [IPrimitive](#).

### 6.26.2.2 getColor()

```
const Math::Color & RayTracer::Sphere::getColor () const [inline], [override], [virtual]
```

Implements [IPrimitive](#).

### 6.26.2.3 getHeight()

```
float RayTracer::Sphere::getHeight () const [inline], [override], [virtual]
```

Implements [IPrimitive](#).

### 6.26.2.4 getNormal()

```
Math::Vector3D RayTracer::Sphere::getNormal () const [inline], [override], [virtual]
```

Implements [IPrimitive](#).

### 6.26.2.5 getPosition()

```
Math::Point3D RayTracer::Sphere::getPosition () const [inline], [override], [virtual]
```

Implements [IPrimitive](#).

### 6.26.2.6 getPrimitiveType()

```
PrimitiveType RayTracer::Sphere::getPrimitiveType () const [inline], [override], [virtual]
```

Implements [IPrimitive](#).

### 6.26.2.7 getRadius()

```
float RayTracer::Sphere::getRadius () const [inline], [override], [virtual]
```

Implements [IPrimitive](#).

### 6.26.2.8 getType()

```
PluginType RayTracer::Sphere::getType () const [inline], [override], [virtual]
```

Reimplemented from [IPrimitive](#).

### 6.26.2.9 hits()

```
bool RayTracer::Sphere::hits (  
    const Ray & ray) const [override], [virtual]
```

Implements [IPrimitive](#).

#### 6.26.2.10 intersect()

```
bool RayTracer::Sphere::intersect (
    const Ray & ray,
    RayTracer::HitRecord & rec) const [override], [virtual]
```

Implements [IPrimitive](#).

#### 6.26.2.11 rotate()

```
void RayTracer::Sphere::rotate (
    float angle,
    const Math::Vector3D & axis) [override], [virtual]
```

Implements [IPrimitive](#).

#### 6.26.2.12 setColor()

```
void RayTracer::Sphere::setColor (
    Math::Color newColor) [inline], [override], [virtual]
```

Implements [IPrimitive](#).

#### 6.26.2.13 setHeight()

```
void RayTracer::Sphere::setHeight (
    float newHeight) [inline], [override], [virtual]
```

Implements [IPrimitive](#).

#### 6.26.2.14 setNormal()

```
void RayTracer::Sphere::setNormal (
    const Math::Vector3D & newNormal) [inline], [override], [virtual]
```

Implements [IPrimitive](#).

#### 6.26.2.15 setPosition()

```
void RayTracer::Sphere::setPosition (
    const Math::Point3D & newCenter) [inline], [override], [virtual]
```

Implements [IPrimitive](#).

#### 6.26.2.16 setRadius()

```
void RayTracer::Sphere::setRadius (
    float newRadius) [inline], [override], [virtual]
```

Implements [IPrimitive](#).

### 6.26.2.17 translate()

```
void RayTracer::Sphere::translate (
    const Math::Vector3D & translation) [override], [virtual]
```

Implements [IPrimitive](#).

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

- [src/plugins/primitives/Sphere.hpp](#)
- [src/plugins/primitives/Sphere.cpp](#)

## 6.27 Math::Vector3D Class Reference

```
#include <Vector.hpp>
```

### Public Member Functions

- [Vector3D](#) (float x, float y, float z)
- [~Vector3D](#) ()=default
- float [getX](#) () const
- float [getY](#) () const
- float [getZ](#) () const
- [Vector3D operator+](#) (const [Vector3D](#) &other) const
- [Vector3D operator-](#) (const [Vector3D](#) &other) const
- double [dot](#) (const [Vector3D](#) &other) const

### 6.27.1 Constructor & Destructor Documentation

#### 6.27.1.1 Vector3D()

```
Math::Vector3D::Vector3D (
    float x,
    float y,
    float z) [inline]
```

#### 6.27.1.2 ~Vector3D()

```
Math::Vector3D::~Vector3D () [default]
```

### 6.27.2 Member Function Documentation

#### 6.27.2.1 dot()

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



### 6.27.2.2 getX()

```
float Math::Vector3D::getX () const [inline]
```

### 6.27.2.3 getY()

```
float Math::Vector3D::getY () const [inline]
```

### 6.27.2.4 getZ()

```
float Math::Vector3D::getZ () const [inline]
```

### 6.27.2.5 operator+()

```
Vector3D Math::Vector3D::operator+ (  
    const Vector3D & other) const [inline]
```

### 6.27.2.6 operator-()

```
Vector3D Math::Vector3D::operator- (  
    const Vector3D & other) const [inline]
```

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

- [src/plugins/math/Vector.hpp](#)
- [src/plugins/math/Vector.cpp](#)



# Chapter 7

## File Documentation

### 7.1 src/core/Core.cpp File Reference

```
#include "Core.hpp"
```

#### Functions

- void [core\\_run](#) (char \*filename)

#### 7.1.1 Function Documentation

##### 7.1.1.1 core\_run()

```
void core_run (  
    char * filename)
```

### 7.2 src/core/Core.hpp File Reference

```
#include "PluginLoader.hpp"  
#include "SceneLoader.hpp"  
#include "IPrimitive.hpp"  
#include "Exceptions.hpp"  
#include "IRenderer.hpp"  
#include "ICamera.hpp"  
#include "IPlugin.hpp"  
#include "ILight.hpp"  
#include <SFML/Graphics.hpp>  
#include <SFML/Window.hpp>  
#include <iostream>  
#include <fstream>  
#include <cstring>  
#include <dlfcn.h>  
#include <string>  
#include <vector>
```

## Classes

- class [Core](#)

## 7.3 Core.hpp

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

```

00001  /*
00002  ** EPITECH PROJECT, 2025
00003  ** RayTracer
00004  ** File description:
00005  ** Core
00006  */
00007
00008  #pragma once
00009
00010  #include "PluginLoader.hpp"
00011  #include "SceneLoader.hpp"
00012  #include "IPrimitive.hpp"
00013  #include "Exceptions.hpp"
00014  #include "IRenderer.hpp"
00015  #include "ICamera.hpp"
00016  #include "IPlugin.hpp"
00017  #include "ILight.hpp"
00018
00019  #include <SFML/Graphics.hpp>
00020  #include <SFML/Window.hpp>
00021  #include <iostream>
00022  #include <fstream>
00023  #include <cstring>
00024  #include <dlfcn.h>
00025  #include <string>
00026  #include <vector>
00027
00028  class Core {
00029  public:
00030      Core() = default;
00031
00032      bool isValid(const std::string &filename);
00033
00034      std::vector<IPlugin*> getPlugins() const {
00035          return plugins;
00036      }
00037
00038      void addPlugin(IPlugin *plugin) {
00039          plugins.push_back(plugin);
00040      }
00041
00042      void convertVectorPluginToVectorLight(std::vector<IPlugin*> &plugins, std::vector<ILight*>
&lights);
00043      void convertVectorPluginToVectorPrimitive(std::vector<IPlugin*> &plugins,
std::vector<IPrimitive*> &primitives);
00044  private:
00045      std::vector<IPlugin*> plugins;
00046  };
00047  
```

## 7.4 src/exceptions/Exceptions.hpp File Reference

```

#include <stdexcept>
#include <string>

```

## Classes

- class [Exception](#)
- class [ParsingException](#)
- class [RayTracerException](#)

## 7.5 Exceptions.hpp

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

```
00001 /*
00002 ** EPITECH PROJECT, 2025
00003 ** B-OOP-400-MPL-4-1-raytracer-vincent.bichat
00004 ** File description:
00005 ** exceptions
00006 */
00007
00008 #pragma once
00009
00010 #include <stdexcept>
00011 #include <string>
00012
00013 // Base exception class
00014 class Exception : public std::runtime_error {
00015 public:
00016     explicit Exception(const std::string& message)
00017         : std::runtime_error(message) {}
00018 };
00019
00020 // Parsing error
00021 class ParsingException : public Exception {
00022 public:
00023     explicit ParsingException(const std::string& message)
00024         : Exception("Parsing Error: " + message) {}
00025 };
00026
00027 // RayTracer error
00028 class RayTracerException : public Exception {
00029 public:
00030     explicit RayTracerException(const std::string& message)
00031         : Exception("RayTracer Error: " + message) {}
00032 };
```

## 7.6 src/loaders/PluginLoader.cpp File Reference

```
#include "PluginLoader.hpp"
```

## 7.7 src/loaders/PluginLoader.hpp File Reference

```
#include "IPlugin.hpp"
#include <unordered_map>
#include <iostream>
#include <dlfcn.h>
#include <vector>
#include <string>
```

### Classes

- class [PluginLoader](#)

## 7.8 PluginLoader.hpp

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

```

00001  /*
00002  ** EPITECH PROJECT, 2025
00003  ** RayTracer
00004  ** File description:
00005  ** PluginLoader
00006  */
00007
00008  #pragma once
00009
00010  #include "IPlugin.hpp"
00011
00012  #include <unordered_map>
00013  #include <iostream>
00014  #include <dlfcn.h>
00015  #include <vector>
00016  #include <string>
00017
00018  class PluginLoader {
00019  public:
00020      PluginLoader() = default;
00021      ~PluginLoader() = default;
00022
00023      // Load the plugin from the given path
00024      void storeHandle(void *handle, const std::string &path);
00025      IPlugin *loadCameraPlugin(const std::string &path);
00026      std::vector<IPlugin *> loadlightsPlugin(const std::string &path);
00027      IPlugin *loadRenderPlugin(const std::string &path);
00028
00029      void *loadSharedLibrary(const std::string &path, void *&handle);
00030      void *loadSymbol(void *handle, const std::string &symbolName);
00031      IPlugin *createPrimitive(const std::string &libPath, const std::string &symbolName);
00032      static void *&chooseHandleByPath(const std::string &path);
00033
00034      // Unload the plugin
00035      void unloadCamera(IPlugin *camera);
00036      void unloadRender(IPlugin *plugin);
00037      void unloadLights(std::vector<IPlugin *> lights);
00038      void unloadPrimitives(std::vector<IPlugin *> primitive);
00039
00040      // Store the loaded plugins
00041      IPlugin *camera;
00042      IPlugin *render;
00043      std::vector<IPlugin *> lights;
00044      std::vector<IPlugin *> primitive;
00045      static void *planeHandle;
00046      static void *sphereHandle;
00047      static void *cylinderHandle;
00048      static void *coneHandle;
00049      static void *cubeHandle;
00050
00051      // Store the loaded plugin handles
00052      static void *&getInstance(const std::string &path);
00053
00054  private:
00055      static std::unordered_map<std::string, void **> _factory;
00056      // Handle for the loaded plugin
00057      // This is a pointer to the shared library
00058      void *cameraHandle = nullptr;
00059      void *lightHandle = nullptr;
00060      void *renderHandle = nullptr;
00061  };

```

## 7.9 src/loaders/SceneLoader.cpp File Reference

```
#include "SceneLoader.hpp"
```

## 7.10 src/loaders/SceneLoader.hpp File Reference

```
#include "PluginLoader.hpp"
#include "SceneLoader.hpp"
```

```
#include "Exceptions.hpp"
#include "IRenderer.hpp"
#include "ICamera.hpp"
#include "IPlugin.hpp"
#include "ILight.hpp"
#include <libconfig.h++>
#include <iostream>
#include <vector>
```

## Classes

- class [SceneLoader](#)

## 7.11 SceneLoader.hpp

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

```
00001 /*
00002 ** EPITECH PROJECT, 2025
00003 ** RayTracer
00004 ** File description:
00005 ** SceneLoader
00006 */
00007
00008 #pragma once
00009
00010 #include "PluginLoader.hpp"
00011 #include "SceneLoader.hpp"
00012 #include "Exceptions.hpp"
00013 #include "IRenderer.hpp"
00014 #include "ICamera.hpp"
00015 #include "IPlugin.hpp"
00016 #include "ILight.hpp"
00017
00018 #include <libconfig.h++>
00019
00020 #include <iostream>
00021 #include <vector>
00022
00023 using namespace libconfig;
00024
00025 class SceneLoader {
00026     public:
00027         SceneLoader() = default;
00028         ~SceneLoader() = default;
00029
00030         bool checkCfgError(const std::string &filename);
00031         void loadCamera(IPlugin *camera);
00032         void loadLights(std::vector<IPlugin *> lights);
00033         void loadRender(IPlugin *plugin);
00034         std::vector<IPlugin *> loadPrimitives(PluginLoader &pluginLoader);
00035
00036         void fillSphere(IPlugin *sphere, Setting &sphereSetting);
00037         void fillPlane(IPlugin *plane, Setting &planeSetting);
00038         void fillCylinder(IPlugin *cylinder, Setting &cylinderSetting);
00039         void fillCone(IPlugin *cone, Setting &coneSetting);
00040         void fillCube(IPlugin *cube, Setting &cubeSetting);
00041
00042         template <typename T>
00043         T *castPlugin(IPlugin *plugin) {
00044             return dynamic_cast<T *>(plugin);
00045         }
00046
00047     private:
00048         Config cfg;
00049 };
```

## 7.12 src/main.cpp File Reference

```
#include "Exceptions.hpp"
#include <iostream>
```

### Functions

- void [core\\_run](#) (char \*filename)
- void [display\\_usage](#) ()
- int [main](#) (int argc, char \*\*argv)

### 7.12.1 Function Documentation

#### 7.12.1.1 core\_run()

```
void core_run (
    char * filename)
```

#### 7.12.1.2 display\_usage()

```
void display_usage ()
```

#### 7.12.1.3 main()

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

## 7.13 src/plugins/camera/Camera.cpp File Reference

```
#include "Camera.hpp"
```

### Namespaces

- namespace [RayTracer](#)

### Functions

- [IPlugin \\* RayTracer::createCameraPlugin](#) ()



## 7.14 src/plugins/camera/Camera.hpp File Reference

```
#include "Rectangle.hpp"
#include "ICamera.hpp"
#include "Point.hpp"
#include "Ray.hpp"
#include <cmath>
```

### Classes

- class [RayTracer::Camera](#)

### Namespaces

- namespace [RayTracer](#)

## 7.15 Camera.hpp

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

```
00001 /*
00002 ** EPITECH PROJECT, 2025
00003 ** RayTracer
00004 ** File description:
00005 ** Camera
00006 */
00007
00008 #pragma once
00009
00010 #include "Rectangle.hpp"
00011 #include "ICamera.hpp"
00012 #include "Point.hpp"
00013 #include "Ray.hpp"
00014
00015 #include <cmath>
00016
00017 namespace RayTracer {
00018     class Camera : public ICamera {
00019     public:
00020         Camera();
00021         Camera(const Math::Point3D& origin, const Math::Rectangle3D& screen, float fov);
00022         ~Camera() = default;
00023
00024         // Get the type of the plugin
00025         PluginType getType() const override {
00026             return PluginType::CAMERA;
00027         }
00028
00029         // Rotate the camera around the axis
00030         void rotate(float angle, const Math::Vector3D &axis) override;
00031         // Translate the camera
00032         void translate(const Math::Vector3D &translation) override;
00033
00034         // Setters and Getters for camera position
00035         void setPosition(Math::Point3D position) override { origin = position; }
00036         Math::Point3D getPosition() const override;
00037
00038         // Setters and Getters for camera resolution
00039         void setResolution(int width, int height) override;
00040         void getResolution(int &width, int &height) const override;
00041
00042         // Setters and Getters for camera field of view
00043         void setFieldOfView(float fov) override;
00044         float getFieldOfView() const override;
00045
00046         int getWidth() const override { return width; }
00047         int getHeight() const override { return height; }
00048     }
```

```

00049         // Update the screen rectangle based on the field of view
00050         void updateScreen() override;
00051
00052         Ray rayAt(double u, double v) const override;
00053
00054         void setLookAt(const Math::Point3D &lookAt) override;
00055         Math::Point3D getLookAt() const override { return lookAt_; }
00056
00057     private:
00058         Math::Point3D origin;
00059         Math::Point3D lookAt_;
00060         Math::Rectangle3D screen;
00061         float fieldOfView;
00062         int width;
00063         int height;
00064     };
00065 }

```

## 7.16 src/plugins/camera/ICamera.hpp File Reference

```

#include "IPlugin.hpp"
#include "Point.hpp"
#include "Ray.hpp"

```

### Classes

- class [RayTracer::ICamera](#)

### Namespaces

- namespace [RayTracer](#)

## 7.17 ICamera.hpp

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

```

00001 /*
00002 ** EPITECH PROJECT, 2025
00003 ** RayTracer
00004 ** File description:
00005 ** ICamera
00006 */
00007
00008 #pragma once
00009
00010 #include "IPlugin.hpp"
00011 #include "Point.hpp"
00012 #include "Ray.hpp"
00013
00014 namespace RayTracer {
00015     class ICamera : public IPlugin {
00016     public:
00017         virtual ~ICamera() = default;
00018
00019         //rotate the camera around the axis
00020         virtual void rotate(float angle, const Math::Vector3D &axis) = 0;
00021         virtual void translate(const Math::Vector3D &translation) = 0;
00022
00023         virtual void setPosition(Math::Point3D position) = 0;
00024         virtual Math::Point3D getPosition() const = 0;
00025
00026         virtual void setResolution(int width, int height) = 0;
00027         virtual void getResolution(int &width, int &height) const = 0;
00028
00029         virtual void setFieldOfView(float fov) = 0;
00030         virtual float getFieldOfView() const = 0;

```

```

00031
00032     virtual int getWidth() const = 0;
00033     virtual int getHeight() const = 0;
00034
00035     virtual void updateScreen() = 0;
00036
00037     virtual Ray rayAt(double u, double v) const = 0;
00038
00039     virtual void setLookAt(const Math::Point3D &lookAt) = 0;
00040     virtual Math::Point3D getLookAt() const = 0;
00041 };
00042 }

```

## 7.18 src/plugins/IPlugin.hpp File Reference

### Classes

- class [IPlugin](#)

### Enumerations

- enum class [PluginType](#) { [CAMERA](#) , [LIGHT](#) , [PRIMITIVE](#) , [RENDERER](#) }

### 7.18.1 Enumeration Type Documentation

#### 7.18.1.1 PluginType

```
enum class PluginType [strong]
```

#### Enumerator

CAMERA	
LIGHT	
PRIMITIVE	
RENDERER	

## 7.19 IPlugin.hpp

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

```

00001 /*
00002  ** EPITECH PROJECT, 2025
00003  ** RayTracer
00004  ** File description:
00005  ** IPlugin
00006  */
00007
00008 #pragma once
00009
00010 enum class PluginType {
00011     CAMERA,
00012     LIGHT,
00013     PRIMITIVE,
00014     RENDERER,
00015 };
00016
00017 class IPlugin {
00018 public:
00019     IPlugin() = default;
00020     virtual ~IPlugin() = default;
00021
00022     virtual PluginType getType() const = 0;
00023 };

```

## 7.20 src/plugins/lights/AmbientLight.cpp File Reference

```
#include "AmbientLight.hpp"
```

### Functions

- [IPlugin \\* createAmbientLight \(\)](#)

### 7.20.1 Function Documentation

#### 7.20.1.1 createAmbientLight()

```
IPlugin * createAmbientLight ()
```

## 7.21 src/plugins/lights/AmbientLight.hpp File Reference

```
#include "ILight.hpp"
```

### Classes

- class [AmbientLight](#)

## 7.22 AmbientLight.hpp

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

```
00001 /*
00002 ** EPITECH PROJECT, 2025
00003 ** RayTracer
00004 ** File description:
00005 ** AmbientLight
00006 */
00007
00008 #pragma once
00009
00010 #include "ILight.hpp"
00011
00012 class AmbientLight : public ILight {
00013     public:
00014         AmbientLight(float intensity = 1.0f);
00015         ~AmbientLight() = default;
00016
00017         // Get the type of the plugin
00018         PluginType getType() const override {
00019             return PluginType::LIGHT;
00020         }
00021
00022         // Get the light type
00023         LightType getLightType() const override {
00024             return LightType::AMBIENT;
00025         }
00026
00027         // Setters and Getters for light intensity
00028         void setIntensity(float intensity) override;
00029         float getIntensity() const override;
00030
00031         // function to Apply the light to a color
00032         void applyLight(float &r, float &g, float &b) const override;
00033
00034         void setDirection(float, float, float) override {} // No direction for ambient light
00035         Math::Vector3D getDirection() const override {return Math::Vector3D(0, 0, 0);} // No direction
00036     private:
00037         float intensity;
00038 };
```

## 7.23 src/plugins/lights/DirectionnalLights.cpp File Reference

```
#include "DirectionnalLights.hpp"
#include "Rectangle.hpp"
#include "Vector.hpp"
#include "Point.hpp"
```

### Functions

- [IPlugin \\* createDirectionnalLights \(\)](#)

### 7.23.1 Function Documentation

#### 7.23.1.1 createDirectionnalLights()

```
IPlugin * createDirectionnalLights ()
```

## 7.24 src/plugins/lights/DirectionnalLights.hpp File Reference

```
#include "ILight.hpp"
```

### Classes

- class [DirectionnalLights](#)

## 7.25 DirectionnalLights.hpp

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

```
00001 /*
00002 ** EPITECH PROJECT, 2025
00003 ** RayTracer
00004 ** File description:
00005 ** DirectionnalLights
00006 */
00007
00008 #pragma once
00009
00010 #include "ILight.hpp"
00011
00012 class DirectionnalLights : public ILight {
00013 public:
00014     DirectionnalLights(float intensity = 1.0f, Math::Vector3D direction = Math::Vector3D(0, 0,
00015 -1));
00016     ~DirectionnalLights() = default;
00017
00018     // Get the type of the plugin
00019     PluginType getType() const override {
00020         return PluginType::LIGHT;
00021     }
00022
00023     // Get the light type
00024     LightType getLightType() const override {
00025         return LightType::DIRECTIONAL;
00026     }
00027 }
```

```

00026
00027     // Setters and Getters for light intensity
00028     void setIntensity(float intensity) override;
00029     float getIntensity() const override;
00030
00031     // Setters and Getters for light direction
00032     void setDirection(float x, float y, float z) override;
00033     Math::Vector3D getDirection() const override;
00034
00035     // function to Apply the light to a color
00036     void applyLight(float &r, float &g, float &b) const override;
00037     private:
00038     float intensity;
00039     Math::Point3D position;
00040     Math::Vector3D direction;
00041 };

```

## 7.26 src/plugins/lights/ILight.hpp File Reference

```

#include "IPlugin.hpp"
#include "Vector.hpp"
#include "Point.hpp"

```

### Classes

- class [ILight](#)

### Enumerations

- enum class [LightType](#) { [AMBIENT](#) , [DIRECTIONAL](#) }

## 7.26.1 Enumeration Type Documentation

### 7.26.1.1 LightType

```
enum class LightType [strong]
```

#### Enumerator

AMBIENT	
DIRECTIONAL	

## 7.27 ILight.hpp

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

```

00001 /*
00002 ** EPITECH PROJECT, 2025
00003 ** RayTracer
00004 ** File description:
00005 ** ILight
00006 */
00007
00008 #pragma once
00009
00010 #include "IPlugin.hpp"
00011 #include "Vector.hpp"
00012 #include "Point.hpp"
00013
00014 enum class LightType {
00015     AMBIENT,
00016     DIRECTIONAL
00017 };
00018
00019 class ILight : public IPlugin {
00020     public:
00021         ILight() = default;
00022         virtual ~ILight() = default;
00023
00024         virtual LightType getLightType() const = 0;
00025
00026         virtual void setIntensity(float intensity) = 0;
00027         virtual float getIntensity() const = 0;
00028         virtual void applyLight(float &r, float &g, float &b) const = 0;
00029
00030         virtual void setDirection(float x, float y, float z) = 0;
00031         virtual Math::Vector3D getDirection() const = 0;
00032 };

```

## 7.28 src/plugins/math/Color.cpp File Reference

```
#include "Color.hpp"
```

## 7.29 src/plugins/math/Color.hpp File Reference

### Classes

- class [Math::Color](#)

### Namespaces

- namespace [Math](#)

## 7.30 Color.hpp

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

```

00001 /*
00002 ** EPITECH PROJECT, 2025
00003 ** RayTracer
00004 ** File description:
00005 ** Point
00006 */
00007

```

```

00008 #pragma once
00009
00010 namespace Math {
00011     class Color {
00012     public:
00013         Color(float r, float g, float b) : r(r), g(g), b(b) {}
00014         ~Color() = default;
00015
00016         float getR() const { return r; }
00017         float getG() const { return g; }
00018         float getB() const { return b; }
00019
00020         Color operator+(const Color &other) const { return Color(r + other.r, g + other.g, b +
00021             other.b); }
00022         Color operator+=(const Color &other) { r += other.r; g += other.g; b += other.b; return
00023             *this; }
00024     private:
00025         float r;
00026         float g;
00027         float b;
00028     };
00029 }

```

## 7.31 src/plugins/math/Matrix.cpp File Reference

```
#include "Matrix.hpp"
```

## 7.32 src/plugins/math/Matrix.hpp File Reference

```

#include "Vector.hpp"
#include "Point.hpp"
#include <cmath>

```

### Classes

- class [Math::Matrix4x4](#)

### Namespaces

- namespace [Math](#)

## 7.33 Matrix.hpp

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

```

00001 /*
00002 ** EPITECH PROJECT, 2025
00003 ** RayTracer
00004 ** File description:
00005 ** Matrix
00006 */
00007
00008 #pragma once
00009
00010 #include "Vector.hpp"
00011 #include "Point.hpp"
00012
00013 #include <cmath>

```



```

00014
00015 namespace Math {
00016     class Matrix4x4 {
00017     public:
00018         Matrix4x4();
00019         ~Matrix4x4() = default;
00020
00021         // Create identity matrix
00022         void identity();
00023
00024         // Create rotation matrix around an axis
00025         void rotationMatrix(float angle, const Vector3D &axis);
00026
00027         // Apply matrix to a point
00028         Point3D applyToPoint(const Point3D &point) const;
00029
00030         // Apply matrix to a vector
00031         Vector3D applyToVector(const Vector3D &vector) const;
00032
00033     private:
00034         float data[4][4];
00035     };
00036 }

```

## 7.34 src/plugins/math/Point.cpp File Reference

```
#include "Point.hpp"
```

## 7.35 src/plugins/math/Point.hpp File Reference

### Classes

- class [Math::Point3D](#)

### Namespaces

- namespace [Math](#)

## 7.36 Point.hpp

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

```

00001 /*
00002 ** EPITECH PROJECT, 2025
00003 ** RayTracer
00004 ** File description:
00005 ** Point
00006 */
00007
00008 #pragma once
00009
00010 namespace Math {
00011     class Point3D {
00012     public:
00013         Point3D(float x, float y, float z) : x(x), y(y), z(z) {}
00014         ~Point3D() = default;
00015
00016         float getX() const { return x; }
00017         float getY() const { return y; }
00018         float getZ() const { return z; }
00019
00020         Point3D operator+(const Point3D &other) const { return Point3D(x + other.x, y + other.y, z
+ other.z); }
00021         Point3D operator+=(const Point3D &other) { x += other.x; y += other.y; z += other.z;
return *this; }
00022
00023     private:
00024         float x;
00025         float y;
00026         float z;
00027     };
00028 }

```

## 7.37 src/plugins/math/Rectangle.cpp File Reference

```
#include "Rectangle.hpp"
```

### Namespaces

- namespace [Math](#)

## 7.38 src/plugins/math/Rectangle.hpp File Reference

```
#include "Vector.hpp"  
#include "Point.hpp"
```

### Classes

- class [Math::Rectangle3D](#)

### Namespaces

- namespace [Math](#)

## 7.39 Rectangle.hpp

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

```
00001 /*  
00002 ** EPITECH PROJECT, 2025  
00003 ** RayTracer  
00004 ** File description:  
00005 ** Rectangle  
00006 */  
00007  
00008 #pragma once  
00009  
00010 #include "Vector.hpp"  
00011 #include "Point.hpp"  
00012  
00013 namespace Math {  
00014     class Rectangle3D {  
00015     public:  
00016         Rectangle3D(const Math::Point3D& origin, const Math::Vector3D& bottom_side, const  
00017                     Math::Vector3D& left_side);  
00018         Math::Point3D origin; // Bottom-left corner of the rectangle  
00019         Math::Vector3D bottom_side;  
00020         Math::Vector3D left_side;  
00021  
00022         Math::Point3D pointAt(double u, double v) const;  
00023     };  
00024 }
```

## 7.40 src/plugins/math/Vector.cpp File Reference

```
#include "Vector.hpp"
```

**Namespaces**

- namespace [Math](#)

**7.41 src/plugins/math/Vector.hpp File Reference****Classes**

- class [Math::Vector3D](#)

**Namespaces**

- namespace [Math](#)

**7.42 Vector.hpp**

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

```

00001 /*
00002 ** EPITECH PROJECT, 2025
00003 ** RayTracer
00004 ** File description:
00005 ** Vector
00006 */
00007
00008 #pragma once
00009
00010 namespace Math {
00011     class Vector3D {
00012     public:
00013         Vector3D(float x, float y, float z) : x(x), y(y), z(z) {}
00014         ~Vector3D() = default;
00015
00016         float getX() const { return x; }
00017         float getY() const { return y; }
00018         float getZ() const { return z; }
00019
00020         Vector3D operator+(const Vector3D &other) const { return Vector3D(x + other.x, y +
00021 other.y, z + other.z); }
00022         Vector3D operator-(const Vector3D &other) const { return Vector3D(x - other.x, y -
00023 other.y, z - other.z); }
00024
00025         double dot(const Vector3D& other) const;
00026     private:
00027         float x;
00028         float y;
00029         float z;
00030     };
00031 }
```

**7.43 src/plugins/primitives/Cone.cpp File Reference**

```
#include "Cone.hpp"
```

**Namespaces**

- namespace [RayTracer](#)

## Functions

- [IPlugin \\* RayTracer::createConePrimitive \(\)](#)

## 7.44 src/plugins/primitives/Cone.hpp File Reference

```
#include "IPrimitive.hpp"
#include "Vector.hpp"
#include "Point.hpp"
#include "Color.hpp"
#include "Ray.hpp"
#include <iostream>
#include <cmath>
```

## Classes

- class [RayTracer::Cone](#)

## Namespaces

- namespace [RayTracer](#)

## 7.45 Cone.hpp

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

```
00001 /*
00002 ** EPITECH PROJECT, 2025
00003 ** RayTracer
00004 ** File description:
00005 ** Cone
00006 */
00007
00008 #pragma once
00009
00010 #include "IPrimitive.hpp"
00011 #include "Vector.hpp"
00012 #include "Point.hpp"
00013 #include "Color.hpp"
00014 #include "Ray.hpp"
00015
00016 #include <iostream>
00017 #include <cmath>
00018
00019 namespace RayTracer {
00020     class Cone : public IPrimitive {
00021     public:
00022         // Constructeur : base est le sommet du cône, axis est la direction de l'axe,
00023         // radius est le rayon de la base, height est la hauteur
00024         Cone();
00025         ~Cone() = default;
00026
00027         // Interface IPrimitive
00028         bool hits(const Ray &ray) const override;
00029         bool intersect(const Ray &ray, HitRecord &rec) const override;
00030         void translate(const Math::Vector3D &translation) override;
00031         void rotate(float angle, const Math::Vector3D &axis) override;
00032
00033         // Getters
00034         const Math::Point3D &getCenter() const override { return vertex; }
00035         float getRadius() const override { return radius; }
00036         const Math::Color &getColor() const override { return color; }
00037     }
```

```

00038         // Get the type of the plugin
00039         PluginType getType() const override {
00040             return PluginType::PRIMITIVE;
00041         }
00042
00043         // Get the primitive type
00044         PrimitiveType getPrimitiveType() const override{
00045             return PrimitiveType::CONE;
00046         }
00047
00048         Math::Point3D getPosition() const override { return vertex; }
00049         void setPosition(const Math::Point3D &position) override { vertex = position; }
00050         void setRadius(float newRadius) override { radius = newRadius; }
00051         void setNormal(const Math::Vector3D &newNormal) override { axis = newNormal; }
00052         Math::Vector3D getNormal() const override { return axis; }
00053         void setColor(Math::Color newColor) override { color = newColor; }
00054
00055         float getHeight() const override { return height; }
00056         void setHeight(float newHeight) override { height = newHeight; }
00057
00058     private:
00059         Math::Point3D vertex;      // Sommet du cône
00060         Math::Vector3D axis;       // Direction de l'axe
00061         float radius;             // Rayon de la base
00062         float height;             // Hauteur du cône
00063         Math::Color color;        // Couleur du cône
00064     };
00065 }

```

## 7.46 src/plugins/primitives/Cube.cpp File Reference

```
#include "Cube.hpp"
```

### Functions

- [IPlugin \\* createCubePrimitive \(\)](#)

### 7.46.1 Function Documentation

#### 7.46.1.1 createCubePrimitive()

```
IPlugin * createCubePrimitive ()
```

## 7.47 src/plugins/primitives/Cube.hpp File Reference

```

#include "IPrimitive.hpp"
#include "Vector.hpp"
#include "Point.hpp"
#include "Ray.hpp"
#include <algorithm>
#include <cmath>

```

### Classes

- class [RayTracer::Cube](#)

## Namespaces

- namespace [RayTracer](#)

## 7.48 Cube.hpp

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

```

00001  /*
00002  ** EPITECH PROJECT, 2025
00003  ** RayTracer
00004  ** File description:
00005  ** Cube
00006  */
00007
00008  #pragma once
00009
00010  #include "IPrimitive.hpp"
00011  #include "Vector.hpp"
00012  #include "Point.hpp"
00013  #include "Ray.hpp"
00014
00015  #include <algorithm>
00016  #include <cmath>
00017
00018  namespace RayTracer {
00019      class Cube : public IPrimitive {
00020      public:
00021          Cube();
00022          ~Cube() = default;
00023
00024          bool hits(const Ray &ray) const override;
00025          bool intersect(const Ray &ray, HitRecord &rec) const override;
00026          void translate(const Math::Vector3D &translation) override;
00027          void rotate(float angle, const Math::Vector3D &axis) override;
00028
00029          // Getters
00030          const Math::Point3D &getCenter() const override {
00031              return center;
00032          }
00033          float getRadius() const override {
00034              return size / 2.0f; // Demi-taille du cube
00035          }
00036          const Math::Color &getColor() const override {
00037              return color;
00038          }
00039
00040          // Get the type of the plugin
00041          PluginType getType() const override {
00042              return PluginType::PRIMITIVE;
00043          }
00044
00045          // Get the primitive type
00046          PrimitiveType getPrimitiveType() const override {
00047              return PrimitiveType::CUBE;
00048          }
00049
00050          Math::Point3D getPosition() const override {
00051              return center;
00052          }
00053          void setPosition(const Math::Point3D &position) override {
00054              center = position;
00055          }
00056          void setRadius(float newSize) override {
00057              size = newSize * 2.0f; // Le rayon est la moitié de la taille
00058          }
00059          void setNormal(const Math::Vector3D &newNormal) override {
00060              // Pas applicable directement pour un cube
00061              (void)newNormal;
00062          }
00063          Math::Vector3D getNormal() const override {
00064              // Un cube n'a pas une seule normale
00065              return Math::Vector3D(0, 0, 0);
00066          }
00067          void setColor(Math::Color newColor) override {
00068              color = newColor;
00069          }
00070
00071          // Spécifique au cube
00072          void setSize(float newSize) {
00073              size = newSize;

```

```

00074     }
00075     float getSize() const {
00076         return size;
00077     }
00078
00079     // Set height
00080     void setHeight(float newHeight) override {
00081         // Pas applicable directement pour un cube
00082         (void)newHeight;
00083     }
00084     float getHeight() const override {
00085         // Pas applicable directement pour un cube
00086         return size;
00087     }
00088
00089     private:
00090     Math::Point3D center; // Centre du cube
00091     float size;          // Taille du cube (longueur d'une arête)
00092     Math::Color color;    // Couleur du cube
00093
00094     // Directions des axes du cube (pour supporter la rotation)
00095     Math::Vector3D xAxis;
00096     Math::Vector3D yAxis;
00097     Math::Vector3D zAxis;
00098 };
00099 }

```

## 7.49 src/plugins/primitives/Cylinder.cpp File Reference

```
#include "Cylinder.hpp"
```

### Namespaces

- namespace [RayTracer](#)

### Functions

- [IPlugin \\* RayTracer::createCylinderPrimitive \(\)](#)

## 7.50 src/plugins/primitives/Cylinder.hpp File Reference

```

#include "IPrimitive.hpp"
#include "Vector.hpp"
#include "Color.hpp"
#include "Point.hpp"
#include "Ray.hpp"
#include <iostream>
#include <cmath>

```

### Classes

- class [RayTracer::Cylinder](#)

### Namespaces

- namespace [RayTracer](#)

## 7.51 Cylinder.hpp

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

```

00001 /*
00002 ** EPITECH PROJECT, 2025
00003 ** RayTracer [WSL: Ubuntu-24.04]
00004 ** File description:
00005 ** Cylinder
00006 */
00007
00008 #pragma once
00009
00010 #include "IPrimitive.hpp"
00011 #include "Vector.hpp"
00012 #include "Color.hpp"
00013 #include "Point.hpp"
00014 #include "Ray.hpp"
00015
00016 #include <iostream>
00017 #include <cmath>
00018
00019 namespace RayTracer {
00020     class Cylinder : public IPrimitive {
00021     public:
00022         Cylinder();
00023         ~Cylinder() = default;
00024
00025         bool hits(const Ray &ray) const override;
00026         bool intersect(const Ray &ray, HitRecord &rec) const override;
00027         void translate(const Math::Vector3D &translation) override;
00028         void rotate(float angle, const Math::Vector3D &axis) override;
00029
00030         const Math::Point3D &getCenter() const override { return base; }
00031         float getRadius() const override { return radius; }
00032         const Math::Color &getColor() const override { return color; }
00033
00034         PrimitiveType getPrimitiveType() const override { return PrimitiveType::CYLINDER; }
00035         Math::Point3D getPosition() const override { return base; }
00036         void setPosition(const Math::Point3D &position) override { base = position; }
00037         void setRadius(float newRadius) override { radius = newRadius; }
00038         void setNormal(const Math::Vector3D &newNormal) override { axis = newNormal; }
00039         Math::Vector3D getNormal() const override { return axis; }
00040         void setColor(Math::Color newColor) override { color = newColor; }
00041
00042         float getHeight() const override { return height; }
00043         void setHeight(float newHeight) override { height = newHeight; }
00044
00045     private:
00046         Math::Point3D base;
00047         Math::Vector3D axis;
00048         float radius;
00049         float height;
00050         Math::Color color;
00051     };
00052 }

```

## 7.52 src/plugins/primitives/HitRecord.hpp File Reference

```

#include "Vector.hpp"
#include "Point.hpp"
#include "Color.hpp"

```

### Classes

- struct [RayTracer::HitRecord](#)

### Namespaces

- namespace [RayTracer](#)



## 7.53 HitRecord.hpp

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

```

00001  /*
00002  ** EPITECH PROJECT, 2025
00003  ** RayTracer [WSL: Ubuntu-24.04]
00004  ** File description:
00005  ** HitRecord
00006  */
00007
00008  #pragma once
00009
00010  #include "Vector.hpp"
00011  #include "Point.hpp"
00012  #include "Color.hpp"
00013
00014  namespace RayTracer {
00015      struct HitRecord {
00016          Math::Point3D point = Math::Point3D(0,0,0);
00017          Math::Vector3D normal = Math::Vector3D(0,0,0);
00018          float t = 0;
00019          Math::Color color = Math::Color(0,0,0);
00020      };
00021  }
```

## 7.54 src/plugins/primitives/IPrimitive.hpp File Reference

```

#include "HitRecord.hpp"
#include "IPlugin.hpp"
#include "Vector.hpp"
#include "Color.hpp"
#include "Point.hpp"
#include "Ray.hpp"
#include <vector>
```

### Classes

- class [IPrimitive](#)

### Enumerations

- enum class [PrimitiveType](#) {  
[SPHERE](#) , [PLANE](#) , [CUBE](#) , [CYLINDER](#) ,  
[CONE](#) , [DONUT](#) }

### 7.54.1 Enumeration Type Documentation

#### 7.54.1.1 PrimitiveType

```
enum class PrimitiveType [strong]
```

#### Enumerator

SPHERE	
PLANE	
CUBE	
CYLINDER	
CONE	
DONUT	

## 7.55 IPrimitive.hpp

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

```

00001  /*
00002  ** EPITECH PROJECT, 2025
00003  ** RayTracer
00004  ** File description:
00005  ** IPrimitive
00006  */
00007
00008  #pragma once
00009
00010  #include "HitRecord.hpp"
00011  #include "IPlugin.hpp"
00012  #include "Vector.hpp"
00013  #include "Color.hpp"
00014  #include "Point.hpp"
00015  #include "Ray.hpp"
00016
00017  #include <vector>
00018
00019  enum class PrimitiveType {
00020      SPHERE,
00021      PLANE,
00022      CUBE,
00023      CYLINDER,
00024      CONE,
00025      DONUT,
00026  };
00027
00028  class IPrimitive : public IPlugin {
00029  public:
00030      IPrimitive() = default;
00031      virtual ~IPrimitive() = default;
00032
00033      bool virtual hits(const RayTracer::Ray &ray) const = 0;
00034      void virtual translate(const Math::Vector3D &translation) = 0;
00035      void virtual rotate(float angle, const Math::Vector3D &axis) = 0;
00036      virtual bool intersect(const RayTracer::Ray &ray, RayTracer::HitRecord &rec) const = 0;
00037
00038      // Getters
00039      virtual const Math::Point3D &getCenter() const = 0;
00040      virtual float getRadius() const = 0;
00041      virtual const Math::Color &getColor() const = 0;
00042      virtual Math::Point3D getPosition() const = 0;
00043      virtual PrimitiveType getPrimitiveType() const = 0;
00044      virtual Math::Vector3D getNormal() const = 0;
00045      virtual float getHeight() const = 0;
00046
00047
00048
00049      // Setters
00050      virtual void setPosition(const Math::Point3D &position) = 0;
00051      virtual void setRadius(float radius) = 0;
00052      virtual void setColor(Math::Color color) = 0;
00053      virtual void setNormal(const Math::Vector3D &normal) = 0;
00054      virtual void setHeight(float height) = 0;
00055
00056      // Get the type of the plugin
00057      PluginType getType() const override {
00058          return PluginType::PRIMITIVE;
00059      }
00060  };

```

## 7.56 src/plugins/primitives/Plane.cpp File Reference

```
#include "Plane.hpp"
```

### Functions

- [IPlugin \\* createPlanePrimitive \(\)](#)

## 7.56.1 Function Documentation

### 7.56.1.1 createPlanePrimitive()

```
IPlugin * createPlanePrimitive ()
```

## 7.57 src/plugins/primitives/Plane.hpp File Reference

```
#include "IPrimitive.hpp"
#include "Vector.hpp"
#include "Point.hpp"
#include "Ray.hpp"
#include <cmath>
```

### Classes

- class [RayTracer::Plane](#)

### Namespaces

- namespace [RayTracer](#)

## 7.58 Plane.hpp

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

```
00001 /*
00002 ** EPITECH PROJECT, 2025
00003 ** RayTracer
00004 ** File description:
00005 ** Plane
00006 */
00007
00008 #pragma once
00009
00010 #include "IPrimitive.hpp"
00011 #include "Vector.hpp"
00012 #include "Point.hpp"
00013 #include "Ray.hpp"
00014
00015 #include <cmath>
00016
00017 namespace RayTracer {
00018     class Plane : public IPrimitive {
00019     public:
00020         Plane();
00021         ~Plane() = default;
00022
00023         bool hits(const RayTracer::Ray &ray) const override;
00024         void translate(const Math::Vector3D &translation) override;
00025         void rotate(float angle, const Math::Vector3D &axis) override;
00026
00027         virtual bool intersect(const RayTracer::Ray &ray, RayTracer::HitRecord &rec) const
00028         override;
00029
00029         // Getters
00030         const Math::Point3D &getCenter() const override {
00031             return point;
00032         }
00033         float getRadius() const override {
00034             return 0; // Planes do not have a radius
00035         }
00036     };
00037 }
```

```

00035         }
00036         const Math::Color &getColor() const override {
00037             return color;
00038         }
00039
00040         // Get the type of the plugin
00041         PluginType getType() const override {
00042             return PluginType::PRIMITIVE;
00043         }
00044
00045         void setPosition(const Math::Point3D &newPoint) override {
00046             point = newPoint;
00047         }
00048         Math::Point3D getPosition() const override {
00049             return point;
00050         }
00051         void setRadius(float newRadius) override {
00052             // Not applicable for Plane
00053             (void)newRadius;
00054         }
00055         void setNormal(const Math::Vector3D &newNormal) override {
00056             normal = newNormal;
00057         }
00058
00059         Math::Vector3D getNormal() const override {
00060             return normal;
00061         }
00062
00063         void setColor(Math::Color newColor) override {
00064             color = newColor;
00065         }
00066
00067         // Get the primitive type
00068         PrimitiveType getPrimitiveType() const override{
00069             return PrimitiveType::PLANE;
00070         }
00071
00072         // Set the height (not applicable for Plane)
00073         void setHeight(float newHeight) override {
00074             // Not applicable for Plane
00075             (void)newHeight;
00076         }
00077         float getHeight() const override {
00078             return 0; // Planes do not have a height
00079         }
00080
00081     private:
00082         Math::Point3D point;
00083         Math::Vector3D normal;
00084         Math::Color color; // Color of the plane
00085     };
00086 }
00087

```

## 7.59 src/plugins/primitives/Ray.cpp File Reference

```
#include "Ray.hpp"
```

## 7.60 src/plugins/primitives/Ray.hpp File Reference

```
#include "Vector.hpp"
#include "Point.hpp"
```

### Classes

- class [RayTracer::Ray](#)

## Namespaces

- namespace [RayTracer](#)

## 7.61 Ray.hpp

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

```
00001 /*
00002 ** EPITECH PROJECT, 2025
00003 ** RayTracer
00004 ** File description:
00005 ** Ray
00006 */
00007
00008 #pragma once
00009
00010 #include "Vector.hpp"
00011 #include "Point.hpp"
00012
00013 namespace RayTracer {
00014     class Ray {
00015     public:
00016         Ray(const Math::Point3D &origin, const Math::Vector3D &direction);
00017
00018         //Getters
00019
00020         const Math::Point3D &getOrigin() const {
00021             return origin;
00022         }
00023         const Math::Vector3D &getDirection() const {
00024             return direction;
00025         }
00026
00027     private:
00028         Math::Point3D origin;
00029         Math::Vector3D direction;
00030     };
00031 }
```

## 7.62 src/plugins/primitives/Sphere.cpp File Reference

```
#include "Sphere.hpp"
```

### Functions

- [IPlugin \\* createSpherePrimitive \(\)](#)

### 7.62.1 Function Documentation

#### 7.62.1.1 createSpherePrimitive()

```
IPlugin * createSpherePrimitive ()
```

## 7.63 src/plugins/primitives/Sphere.hpp File Reference

```
#include "IPrimitive.hpp"
#include "Matrix.hpp"
#include "Point.hpp"
#include "Ray.hpp"
#include <cmath>
```

### Classes

- class [RayTracer::Sphere](#)

### Namespaces

- namespace [RayTracer](#)

## 7.64 Sphere.hpp

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

```
00001 /*
00002 ** EPITECH PROJECT, 2025
00003 ** RayTracer
00004 ** File description:
00005 ** Sphere
00006 */
00007
00008 #pragma once
00009
00010 #include "IPrimitive.hpp"
00011 #include "Matrix.hpp"
00012 #include "Point.hpp"
00013 #include "Ray.hpp"
00014
00015 #include <cmath>
00016
00017 namespace RayTracer {
00018     class Sphere : public IPrimitive {
00019     public:
00020         Sphere();
00021         ~Sphere() = default;
00022
00023         // Get the type of the plugin
00024         PluginType getType() const override {
00025             return PluginType::PRIMITIVE;
00026         }
00027
00028         // Get the primitive type
00029         PrimitiveType getPrimitiveType() const override {
00030             return PrimitiveType::SPHERE;
00031         }
00032         // hits function
00033         bool hits(const Ray &ray) const override;
00034         bool intersect(const Ray &ray, RayTracer::HitRecord &rec) const override;
00035         void translate(const Math::Vector3D &translation) override;
00036         void rotate(float angle, const Math::Vector3D &axis) override;
00037
00038         // Getters
00039         const Math::Point3D &getCenter() const override {
00040             return center;
00041         }
00042         float getRadius() const override {
00043             return radius;
00044         }
00045
00046         // Get color
00047         const Math::Color &getColor() const override {
00048             return color;
```

```

00049         }
00050
00051         void setPosition(const Math::Point3D &newCenter) override {
00052             center = newCenter;
00053         }
00054         Math::Point3D getPosition() const override {
00055             return center;
00056         }
00057         void setRadius(float newRadius) override {
00058             radius = newRadius;
00059         }
00060         void setNormal(const Math::Vector3D &newNormal) override {
00061             (void)newNormal; // Not applicable for Sphere
00062         }
00063         Math::Vector3D getNormal() const override {
00064             return Math::Vector3D(0, 0, 0); // Not applicable for Sphere
00065         }
00066         void setColor(Math::Color newColor) override {
00067             color = newColor;
00068         }
00069
00070         float getHeight() const override {
00071             return 0; // Not applicable for Sphere
00072         }
00073         void setHeight(float newHeight) override {
00074             (void)newHeight; // Not applicable for Sphere
00075         }
00076
00077     private:
00078         Math::Point3D center;
00079         float radius;
00080         Math::Color color; // Color of the sphere
00081     };
00082 }

```

## 7.65 src/plugins/render/BasicRenderer.cpp File Reference

```
#include "BasicRenderer.hpp"
```

### Functions

- [IPlugin \\* createRenderPlugin \(\)](#)

### 7.65.1 Function Documentation

#### 7.65.1.1 createRenderPlugin()

```
IPlugin * createRenderPlugin ()
```

## 7.66 src/plugins/render/BasicRenderer.hpp File Reference

```

#include "IRenderer.hpp"
#include "ICamera.hpp"
#include "Sphere.hpp"
#include "Ray.hpp"
#include <iostream>
#include <fstream>
#include <cmath>

```

## Classes

- class [BasicRenderer](#)

## 7.67 BasicRenderer.hpp

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

```

00001  /*
00002  ** EPITECH PROJECT, 2025
00003  ** RayTracer
00004  ** File description:
00005  ** BasicRenderer
00006  */
00007
00008  #include "IRenderer.hpp"
00009  #include "ICamera.hpp"
00010  #include "Sphere.hpp"
00011  #include "Ray.hpp"
00012
00013  #include <iostream>
00014  #include <fstream>
00015  #include <cmath>
00016
00017  #pragma once
00018
00019  class BasicRenderer : public IRenderer {
00020  public:
00021      BasicRenderer();
00022      ~BasicRenderer() = default;
00023
00024      // Get the type of the plugin
00025      PluginType getType() const override {
00026          return PluginType::RENDERER;
00027      }
00028
00029      // Set the renderer type
00030      void setRendererType(RendererType type) override {
00031          this->type = type;
00032      }
00033
00034      // Get the renderer type
00035      RendererType getRendererType() const override {
00036          return this->type;
00037      }
00038
00039      // Function to render the scene
00040      void renderScene() override;
00041
00042      // Function to set the camera
00043      void setCamera(RayTracer::ICamera *camera) override;
00044
00045      // Function to set the lights
00046      void setLights(std::vector<ILight *> &lights) override;
00047
00048      // Function to set the primitives
00049      void setPrimitives(std::vector<IPrimitive *> &primitives) override;
00050
00051  private:
00052      RendererType type;
00053      RayTracer::ICamera *camera;
00054      std::vector<ILight *> lights;
00055      std::vector<IPrimitive *> primitives;
00056  };

```

## 7.68 src/plugins/render/IRenderer.hpp File Reference

```

#include "IPrimitive.hpp"
#include "IPlugin.hpp"
#include "ICamera.hpp"
#include "ILight.hpp"
#include <vector>

```



**Classes**

- class [IRenderer](#)

**Enumerations**

- enum class [RendererType](#) { [BASIC](#) , [SFML](#) , [FILE](#) }

**7.68.1 Enumeration Type Documentation****7.68.1.1 RendererType**

```
enum class RendererType [strong]
```

**Enumerator**

BASIC	
SFML	
FILE	

**7.69 IRenderer.hpp**

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

```
00001 /*
00002 ** EPITECH PROJECT, 2025
00003 ** RayTracer
00004 ** File description:
00005 ** IRenderer
00006 */
00007
00008 #pragma once
00009
00010 #include "IPrimitive.hpp"
00011 #include "IPlugin.hpp"
00012 #include "ICamera.hpp"
00013 #include "ILight.hpp"
00014
00015 #include <vector>
00016
00017 enum class RendererType {
00018     BASIC,
00019     SFML,
00020     FILE,
00021 };
00022
00023 class IRenderer : public IPlugin {
00024     public:
00025         IRenderer() = default;
00026         virtual ~IRenderer() = default;
00027
00028         // Set RendererType
00029         virtual void setRendererType(RendererType type) = 0;
00030
00031         // Get the type of the renderer
00032         virtual RendererType getRendererType() const = 0;
00033
00034         // Function to render the scene
00035         virtual void renderScene() = 0;
00036
00037         // Function to set the camera
00038         virtual void setCamera(RayTracer::ICamera *camera) = 0;
00039
00040         // Function to set the lights
00041         virtual void setLights(std::vector<ILight *> &lights) = 0;
00042
00043         // Function to set the primitives
00044         virtual void setPrimitives(std::vector<IPrimitive *> &primitives) = 0;
00045 };
```



# Index

- ~AmbientLight
  - AmbientLight, [12](#)
- ~BasicRenderer
  - BasicRenderer, [14](#)
- ~Camera
  - RayTracer::Camera, [16](#)
- ~Color
  - Math::Color, [19](#)
- ~Cone
  - RayTracer::Cone, [21](#)
- ~Cube
  - RayTracer::Cube, [26](#)
- ~Cylinder
  - RayTracer::Cylinder, [30](#)
- ~DirectionnalLights
  - DirectionnalLights, [33](#)
- ~ICamera
  - RayTracer::ICamera, [37](#)
- ~ILight
  - ILight, [40](#)
- ~IPlugin
  - IPlugin, [41](#)
- ~IPrimitive
  - IPrimitive, [43](#)
- ~IRenderer
  - IRenderer, [46](#)
- ~Matrix4x4
  - Math::Matrix4x4, [47](#)
- ~Plane
  - RayTracer::Plane, [50](#)
- ~PluginLoader
  - PluginLoader, [53](#)
- ~Point3D
  - Math::Point3D, [57](#)
- ~SceneLoader
  - SceneLoader, [60](#)
- ~Sphere
  - RayTracer::Sphere, [63](#)
- ~Vector3D
  - Math::Vector3D, [66](#)
- addPlugin
  - Core, [24](#)
- AMBIENT
  - ILight.hpp, [80](#)
- AmbientLight, [11](#)
  - ~AmbientLight, [12](#)
  - AmbientLight, [12](#)
  - applyLight, [12](#)
  - getDirection, [12](#)
  - getIntensity, [12](#)
  - getLightType, [12](#)
  - getType, [12](#)
  - setDirection, [12](#)
  - setIntensity, [13](#)
- AmbientLight.cpp
  - createAmbientLight, [78](#)
- applyLight
  - AmbientLight, [12](#)
  - DirectionnalLights, [33](#)
  - ILight, [40](#)
- applyToPoint
  - Math::Matrix4x4, [48](#)
- applyToVector
  - Math::Matrix4x4, [48](#)
- BASIC
  - IRenderer.hpp, [99](#)
- BasicRenderer, [13](#)
  - ~BasicRenderer, [14](#)
  - BasicRenderer, [14](#)
  - getRendererType, [14](#)
  - getType, [14](#)
  - renderScene, [14](#)
  - setCamera, [14](#)
  - setLights, [14](#)
  - setPrimitives, [15](#)
  - setRendererType, [15](#)
- BasicRenderer.cpp
  - createRenderPlugin, [97](#)
- bottom\_side
  - Math::Rectangle3D, [60](#)
- CAMERA
  - IPlugin.hpp, [77](#)
- Camera
  - RayTracer::Camera, [16](#)
- camera
  - PluginLoader, [55](#)
- castPlugin
  - SceneLoader, [61](#)
- checkCfgError
  - SceneLoader, [61](#)
- chooseHandleByPath
  - PluginLoader, [54](#)
- Color
  - Math::Color, [19](#)
- color
  - RayTracer::HitRecord, [35](#)
- CONE

- IPrimitive.hpp, 91
- Cone
  - RayTracer::Cone, 21
- coneHandle
  - PluginLoader, 55
- convertVectorPluginToVectorLight
  - Core, 24
- convertVectorPluginToVectorPrimitive
  - Core, 24
- Core, 24
  - addPlugin, 24
  - convertVectorPluginToVectorLight, 24
  - convertVectorPluginToVectorPrimitive, 24
  - Core, 24
  - getPlugins, 25
  - isFileValid, 25
- Core.cpp
  - core\_run, 69
- core\_run
  - Core.cpp, 69
  - main.cpp, 74
- createAmbientLight
  - AmbientLight.cpp, 78
- createCameraPlugin
  - RayTracer, 9
- createConePrimitive
  - RayTracer, 9
- createCubePrimitive
  - Cube.cpp, 87
- createCylinderPrimitive
  - RayTracer, 10
- createDirectionnalLights
  - DirectionnalLights.cpp, 79
- createPlanePrimitive
  - Plane.cpp, 93
- createPrimitive
  - PluginLoader, 54
- createRenderPlugin
  - BasicRenderer.cpp, 97
- createSpherePrimitive
  - Sphere.cpp, 95
- CUBE
  - IPrimitive.hpp, 91
- Cube
  - RayTracer::Cube, 26
- Cube.cpp
  - createCubePrimitive, 87
- cubeHandle
  - PluginLoader, 55
- CYLINDER
  - IPrimitive.hpp, 91
- Cylinder
  - RayTracer::Cylinder, 30
- cylinderHandle
  - PluginLoader, 56
- DIRECTIONAL
  - ILight.hpp, 80
- DirectionnalLights, 32
  - ~DirectionnalLights, 33
  - applyLight, 33
  - DirectionnalLights, 33
  - getDirection, 33
  - getIntensity, 34
  - getLightType, 34
  - getType, 34
  - setDirection, 34
  - setIntensity, 34
- DirectionnalLights.cpp
  - createDirectionnalLights, 79
- display\_usage
  - main.cpp, 74
- DONUT
  - IPrimitive.hpp, 91
- dot
  - Math::Vector3D, 66
- Exception, 35
  - Exception, 35
- FILE
  - IRenderer.hpp, 99
- fillCone
  - SceneLoader, 61
- fillCube
  - SceneLoader, 61
- fillCylinder
  - SceneLoader, 61
- fillPlane
  - SceneLoader, 61
- fillSphere
  - SceneLoader, 61
- getB
  - Math::Color, 20
- getCenter
  - IPrimitive, 43
  - RayTracer::Cone, 21
  - RayTracer::Cube, 26
  - RayTracer::Cylinder, 30
  - RayTracer::Plane, 50
  - RayTracer::Sphere, 63
- getColor
  - IPrimitive, 43
  - RayTracer::Cone, 21
  - RayTracer::Cube, 26
  - RayTracer::Cylinder, 30
  - RayTracer::Plane, 50
  - RayTracer::Sphere, 63
- getDirection
  - AmbientLight, 12
  - DirectionnalLights, 33
  - ILight, 40
  - RayTracer::Ray, 58
- getFieldOfView
  - RayTracer::Camera, 17
  - RayTracer::ICamera, 37
- getG

- Math::Color, 20
- getHeight
  - IPrimitive, 43
  - RayTracer::Camera, 17
  - RayTracer::Cone, 22
  - RayTracer::Cube, 26
  - RayTracer::Cylinder, 30
  - RayTracer::ICamera, 37
  - RayTracer::Plane, 50
  - RayTracer::Sphere, 64
- getInstance
  - PluginLoader, 54
- getIntensity
  - AmbientLight, 12
  - DirectionnalLights, 34
  - ILight, 40
- getLightType
  - AmbientLight, 12
  - DirectionnalLights, 34
  - ILight, 40
- getLookAt
  - RayTracer::Camera, 17
  - RayTracer::ICamera, 37
- getNormal
  - IPrimitive, 43
  - RayTracer::Cone, 22
  - RayTracer::Cube, 26
  - RayTracer::Cylinder, 30
  - RayTracer::Plane, 50
  - RayTracer::Sphere, 64
- getOrigin
  - RayTracer::Ray, 58
- getPlugins
  - Core, 25
- getPosition
  - IPrimitive, 43
  - RayTracer::Camera, 17
  - RayTracer::Cone, 22
  - RayTracer::Cube, 26
  - RayTracer::Cylinder, 30
  - RayTracer::ICamera, 37
  - RayTracer::Plane, 50
  - RayTracer::Sphere, 64
- getPrimitiveType
  - IPrimitive, 43
  - RayTracer::Cone, 22
  - RayTracer::Cube, 27
  - RayTracer::Cylinder, 30
  - RayTracer::Plane, 51
  - RayTracer::Sphere, 64
- getR
  - Math::Color, 20
- getRadius
  - IPrimitive, 44
  - RayTracer::Cone, 22
  - RayTracer::Cube, 27
  - RayTracer::Cylinder, 31
  - RayTracer::Plane, 51
  - RayTracer::Sphere, 64
- getRendererType
  - BasicRenderer, 14
  - IRenderer, 46
- getResolution
  - RayTracer::Camera, 17
  - RayTracer::ICamera, 37
- getSize
  - RayTracer::Cube, 27
- getType
  - AmbientLight, 12
  - BasicRenderer, 14
  - DirectionnalLights, 34
  - IPlugin, 42
  - IPrimitive, 44
  - RayTracer::Camera, 17
  - RayTracer::Cone, 22
  - RayTracer::Cube, 27
  - RayTracer::Plane, 51
  - RayTracer::Sphere, 64
- getWidth
  - RayTracer::Camera, 17
  - RayTracer::ICamera, 37
- getX
  - Math::Point3D, 57
  - Math::Vector3D, 66
- getY
  - Math::Point3D, 57
  - Math::Vector3D, 67
- getZ
  - Math::Point3D, 57
  - Math::Vector3D, 67
- hits
  - IPrimitive, 44
  - RayTracer::Cone, 22
  - RayTracer::Cube, 27
  - RayTracer::Cylinder, 31
  - RayTracer::Plane, 51
  - RayTracer::Sphere, 64
- identity
  - Math::Matrix4x4, 48
- ILight, 39
  - ~ILight, 40
  - applyLight, 40
  - getDirection, 40
  - getIntensity, 40
  - getLightType, 40
  - ILight, 40
  - setDirection, 40
  - setIntensity, 40
- ILight.hpp
  - AMBIENT, 80
  - DIRECTIONAL, 80
  - LightType, 80
- intersect
  - IPrimitive, 44
  - RayTracer::Cone, 22

- RayTracer::Cube, 27
- RayTracer::Cylinder, 31
- RayTracer::Plane, 51
- RayTracer::Sphere, 64
- IPlugin, 41
  - ~IPlugin, 41
  - getType, 42
  - IPlugin, 41
- IPlugin.hpp
  - CAMERA, 77
  - LIGHT, 77
  - PluginType, 77
  - PRIMITIVE, 77
  - RENDERER, 77
- IPrimitive, 42
  - ~IPrimitive, 43
  - getCenter, 43
  - getColor, 43
  - getHeight, 43
  - getNormal, 43
  - getPosition, 43
  - getPrimitiveType, 43
  - getRadius, 44
  - getType, 44
  - hits, 44
  - intersect, 44
  - IPrimitive, 43
  - rotate, 44
  - setColor, 44
  - setHeight, 44
  - setNormal, 45
  - setPosition, 45
  - setRadius, 45
  - translate, 45
- IPrimitive.hpp
  - CONE, 91
  - CUBE, 91
  - CYLINDER, 91
  - DONUT, 91
  - PLANE, 91
  - PrimitiveType, 91
  - SPHERE, 91
- IRenderer, 45
  - ~IRenderer, 46
  - getRendererType, 46
  - IRenderer, 46
  - renderScene, 46
  - setCamera, 46
  - setLights, 46
  - setPrimitives, 47
  - setRendererType, 47
- IRenderer.hpp
  - BASIC, 99
  - FILE, 99
  - RendererType, 99
  - SFML, 99
- isFileValid
  - Core, 25
- left\_side
  - Math::Rectangle3D, 60
- LIGHT
  - IPlugin.hpp, 77
- lights
  - PluginLoader, 56
- LightType
  - ILight.hpp, 80
- loadCamera
  - SceneLoader, 61
- loadCameraPlugin
  - PluginLoader, 54
- loadLights
  - SceneLoader, 62
- loadlightsPlugin
  - PluginLoader, 54
- loadPrimitives
  - SceneLoader, 62
- loadRender
  - SceneLoader, 62
- loadRenderPlugin
  - PluginLoader, 54
- loadSharedLibrary
  - PluginLoader, 54
- loadSymbol
  - PluginLoader, 54
- main
  - main.cpp, 74
- main.cpp
  - core\_run, 74
  - display\_usage, 74
  - main, 74
- Math, 9
- Math::Color, 19
  - ~Color, 19
  - Color, 19
  - getB, 20
  - getG, 20
  - getR, 20
  - operator+, 20
  - operator+=", 20
- Math::Matrix4x4, 47
  - ~Matrix4x4, 47
  - applyToPoint, 48
  - applyToVector, 48
  - identity, 48
  - Matrix4x4, 47
  - rotationMatrix, 48
- Math::Point3D, 56
  - ~Point3D, 57
  - getX, 57
  - getY, 57
  - getZ, 57
  - operator+, 57
  - operator+=", 57
  - Point3D, 57
- Math::Rectangle3D, 59
  - bottom\_side, 60

- left\_side, 60
- origin, 60
- pointAt, 59
- Rectangle3D, 59
- Math::Vector3D, 66
  - ~Vector3D, 66
  - dot, 66
  - getX, 66
  - getY, 67
  - getZ, 67
  - operator+, 67
  - operator-, 67
  - Vector3D, 66
- Matrix4x4
  - Math::Matrix4x4, 47
- normal
  - RayTracer::HitRecord, 35
- operator+
  - Math::Color, 20
  - Math::Point3D, 57
  - Math::Vector3D, 67
- operator+=
  - Math::Color, 20
  - Math::Point3D, 57
- operator-
  - Math::Vector3D, 67
- origin
  - Math::Rectangle3D, 60
- ParsingException, 48
  - ParsingException, 49
- PLANE
  - IPrimitive.hpp, 91
- Plane
  - RayTracer::Plane, 50
- Plane.cpp
  - createPlanePrimitive, 93
- planeHandle
  - PluginLoader, 56
- PluginLoader, 53
  - ~PluginLoader, 53
  - camera, 55
  - chooseHandleByPath, 54
  - coneHandle, 55
  - createPrimitive, 54
  - cubeHandle, 55
  - cylinderHandle, 56
  - getInstance, 54
  - lights, 56
  - loadCameraPlugin, 54
  - loadlightsPlugin, 54
  - loadRenderPlugin, 54
  - loadSharedLibrary, 54
  - loadSymbol, 54
  - planeHandle, 56
  - PluginLoader, 53
  - primitive, 56
  - render, 56
  - sphereHandle, 56
  - storeHandle, 55
  - unloadCamera, 55
  - unloadLights, 55
  - unloadPrimitives, 55
  - unloadRender, 55
- PluginType
  - IPlugin.hpp, 77
- point
  - RayTracer::HitRecord, 36
- Point3D
  - Math::Point3D, 57
- pointAt
  - Math::Rectangle3D, 59
- PRIMITIVE
  - IPlugin.hpp, 77
- primitive
  - PluginLoader, 56
- PrimitiveType
  - IPrimitive.hpp, 91
- Ray
  - RayTracer::Ray, 58
- rayAt
  - RayTracer::Camera, 17
  - RayTracer::ICamera, 38
- RayTracer, 9
  - createCameraPlugin, 9
  - createConePrimitive, 9
  - createCylinderPrimitive, 10
- RayTracer::Camera, 15
  - ~Camera, 16
  - Camera, 16
  - getFieldOfView, 17
  - getHeight, 17
  - getLookAt, 17
  - getPosition, 17
  - getResolution, 17
  - getType, 17
  - getWidth, 17
  - rayAt, 17
  - rotate, 18
  - setFieldOfView, 18
  - setLookAt, 18
  - setPosition, 18
  - setResolution, 18
  - translate, 18
  - updateScreen, 19
- RayTracer::Cone, 20
  - ~Cone, 21
  - Cone, 21
  - getCenter, 21
  - getColor, 21
  - getHeight, 22
  - getNormal, 22
  - getPosition, 22
  - getPrimitiveType, 22
  - getRadius, 22

- getType, 22
  - hits, 22
  - intersect, 22
  - rotate, 23
  - setColor, 23
  - setHeight, 23
  - setNormal, 23
  - setPosition, 23
  - setRadius, 23
  - translate, 23
- RayTracer::Cube, 25
  - ~Cube, 26
  - Cube, 26
  - getCenter, 26
  - getColor, 26
  - getHeight, 26
  - getNormal, 26
  - getPosition, 26
  - getPrimitiveType, 27
  - getRadius, 27
  - getSize, 27
  - getType, 27
  - hits, 27
  - intersect, 27
  - rotate, 27
  - setColor, 28
  - setHeight, 28
  - setNormal, 28
  - setPosition, 28
  - setRadius, 28
  - setSize, 28
  - translate, 28
- RayTracer::Cylinder, 29
  - ~Cylinder, 30
  - Cylinder, 30
  - getCenter, 30
  - getColor, 30
  - getHeight, 30
  - getNormal, 30
  - getPosition, 30
  - getPrimitiveType, 30
  - getRadius, 31
  - hits, 31
  - intersect, 31
  - rotate, 31
  - setColor, 31
  - setHeight, 31
  - setNormal, 31
  - setPosition, 32
  - setRadius, 32
  - translate, 32
- RayTracer::HitRecord, 35
  - color, 35
  - normal, 35
  - point, 36
  - t, 36
- RayTracer::ICamera, 36
  - ~ICamera, 37
  - getFieldOfView, 37
  - getHeight, 37
  - getLookAt, 37
  - getPosition, 37
  - getResolution, 37
  - getWidth, 37
  - rayAt, 38
  - rotate, 38
  - setFieldOfView, 38
  - setLookAt, 38
  - setPosition, 38
  - setResolution, 38
  - translate, 38
  - updateScreen, 39
- RayTracer::Plane, 49
  - ~Plane, 50
  - getCenter, 50
  - getColor, 50
  - getHeight, 50
  - getNormal, 50
  - getPosition, 50
  - getPrimitiveType, 51
  - getRadius, 51
  - getType, 51
  - hits, 51
  - intersect, 51
  - Plane, 50
  - rotate, 51
  - setColor, 51
  - setHeight, 52
  - setNormal, 52
  - setPosition, 52
  - setRadius, 52
  - translate, 52
- RayTracer::Ray, 57
  - getDirection, 58
  - getOrigin, 58
  - Ray, 58
- RayTracer::Sphere, 62
  - ~Sphere, 63
  - getCenter, 63
  - getColor, 63
  - getHeight, 64
  - getNormal, 64
  - getPosition, 64
  - getPrimitiveType, 64
  - getRadius, 64
  - getType, 64
  - hits, 64
  - intersect, 64
  - rotate, 65
  - setColor, 65
  - setHeight, 65
  - setNormal, 65
  - setPosition, 65
  - setRadius, 65
  - Sphere, 63
  - translate, 65



- RayTracerException, 58
  - RayTracerException, 59
- Rectangle3D
  - Math::Rectangle3D, 59
- render
  - PluginLoader, 56
- RENDERER
  - IPlugin.hpp, 77
- RendererType
  - IRenderer.hpp, 99
- renderScene
  - BasicRenderer, 14
  - IRenderer, 46
- rotate
  - IPrimitive, 44
  - RayTracer::Camera, 18
  - RayTracer::Cone, 23
  - RayTracer::Cube, 27
  - RayTracer::Cylinder, 31
  - RayTracer::ICamera, 38
  - RayTracer::Plane, 51
  - RayTracer::Sphere, 65
- rotationMatrix
  - Math::Matrix4x4, 48
- SceneLoader, 60
  - ~SceneLoader, 60
  - castPlugin, 61
  - checkCfgError, 61
  - fillCone, 61
  - fillCube, 61
  - fillCylinder, 61
  - fillPlane, 61
  - fillSphere, 61
  - loadCamera, 61
  - loadLights, 62
  - loadPrimitives, 62
  - loadRender, 62
  - SceneLoader, 60
- setCamera
  - BasicRenderer, 14
  - IRenderer, 46
- setColor
  - IPrimitive, 44
  - RayTracer::Cone, 23
  - RayTracer::Cube, 28
  - RayTracer::Cylinder, 31
  - RayTracer::Plane, 51
  - RayTracer::Sphere, 65
- setDirection
  - AmbientLight, 12
  - DirectionnalLights, 34
  - ILight, 40
- setFieldOfView
  - RayTracer::Camera, 18
  - RayTracer::ICamera, 38
- setHeight
  - IPrimitive, 44
  - RayTracer::Cone, 23
  - RayTracer::Cube, 28
  - RayTracer::Cylinder, 31
  - RayTracer::Plane, 52
  - RayTracer::Sphere, 65
- setIntensity
  - AmbientLight, 13
  - DirectionnalLights, 34
  - ILight, 40
- setLights
  - BasicRenderer, 14
  - IRenderer, 46
- setLookAt
  - RayTracer::Camera, 18
  - RayTracer::ICamera, 38
- setNormal
  - IPrimitive, 45
  - RayTracer::Cone, 23
  - RayTracer::Cube, 28
  - RayTracer::Cylinder, 31
  - RayTracer::Plane, 52
  - RayTracer::Sphere, 65
- setPosition
  - IPrimitive, 45
  - RayTracer::Camera, 18
  - RayTracer::Cone, 23
  - RayTracer::Cube, 28
  - RayTracer::Cylinder, 32
  - RayTracer::ICamera, 38
  - RayTracer::Plane, 52
  - RayTracer::Sphere, 65
- setPrimitives
  - BasicRenderer, 15
  - IRenderer, 47
- setRadius
  - IPrimitive, 45
  - RayTracer::Cone, 23
  - RayTracer::Cube, 28
  - RayTracer::Cylinder, 32
  - RayTracer::Plane, 52
  - RayTracer::Sphere, 65
- setRendererType
  - BasicRenderer, 15
  - IRenderer, 47
- setResolution
  - RayTracer::Camera, 18
  - RayTracer::ICamera, 38
- setSize
  - RayTracer::Cube, 28
- SFML
  - IRenderer.hpp, 99
- SPHERE
  - IPrimitive.hpp, 91
- Sphere
  - RayTracer::Sphere, 63
- Sphere.cpp
  - createSpherePrimitive, 95
- sphereHandle
  - PluginLoader, 56

- src/core/Core.cpp, 69
- src/core/Core.hpp, 69, 70
- src/exceptions/Exceptions.hpp, 70, 71
- src/loaders/PluginLoader.cpp, 71
- src/loaders/PluginLoader.hpp, 71, 72
- src/loaders/SceneLoader.cpp, 72
- src/loaders/SceneLoader.hpp, 72, 73
- src/main.cpp, 74
- src/plugins/camera/Camera.cpp, 74
- src/plugins/camera/Camera.hpp, 75
- src/plugins/camera/ICamera.hpp, 76
- src/plugins/IPlugin.hpp, 77
- src/plugins/lights/AmbientLight.cpp, 78
- src/plugins/lights/AmbientLight.hpp, 78
- src/plugins/lights/DirectionnalLights.cpp, 79
- src/plugins/lights/DirectionnalLights.hpp, 79
- src/plugins/lights/ILight.hpp, 80, 81
- src/plugins/math/Color.cpp, 81
- src/plugins/math/Color.hpp, 81
- src/plugins/math/Matrix.cpp, 82
- src/plugins/math/Matrix.hpp, 82
- src/plugins/math/Point.cpp, 83
- src/plugins/math/Point.hpp, 83
- src/plugins/math/Rectangle.cpp, 84
- src/plugins/math/Rectangle.hpp, 84
- src/plugins/math/Vector.cpp, 84
- src/plugins/math/Vector.hpp, 85
- src/plugins/primitives/Cone.cpp, 85
- src/plugins/primitives/Cone.hpp, 86
- src/plugins/primitives/Cube.cpp, 87
- src/plugins/primitives/Cube.hpp, 87, 88
- src/plugins/primitives/Cylinder.cpp, 89
- src/plugins/primitives/Cylinder.hpp, 89, 90
- src/plugins/primitives/HitRecord.hpp, 90, 91
- src/plugins/primitives/IPrimitive.hpp, 91, 92
- src/plugins/primitives/Plane.cpp, 92
- src/plugins/primitives/Plane.hpp, 93
- src/plugins/primitives/Ray.cpp, 94
- src/plugins/primitives/Ray.hpp, 94, 95
- src/plugins/primitives/Sphere.cpp, 95
- src/plugins/primitives/Sphere.hpp, 96
- src/plugins/render/BasicRenderer.cpp, 97
- src/plugins/render/BasicRenderer.hpp, 97, 98
- src/plugins/render/IRenderer.hpp, 98, 99
- storeHandle
  - PluginLoader, 55
- t
  - RayTracer::HitRecord, 36
- translate
  - IPrimitive, 45
  - RayTracer::Camera, 18
  - RayTracer::Cone, 23
  - RayTracer::Cube, 28
  - RayTracer::Cylinder, 32
  - RayTracer::ICamera, 38
  - RayTracer::Plane, 52
  - RayTracer::Sphere, 65
- unloadCamera
  - PluginLoader, 55
- unloadLights
  - PluginLoader, 55
- unloadPrimitives
  - PluginLoader, 55
- unloadRender
  - PluginLoader, 55
- updateScreen
  - RayTracer::Camera, 19
  - RayTracer::ICamera, 39
- Vector3D
  - Math::Vector3D, 66