

## Arcade

Generated by Doxygen 1.9.1



<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 Data Structure Index</b>	<b>5</b>
3.1 Data Structures	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
5.1.1 Detailed Description	9
5.1.2 Function Documentation	9
5.1.2.1 operator<<() [1/2]	9
5.1.2.2 operator<<() [2/2]	10
5.2 RayTracer Namespace Reference	10
5.2.1 Detailed Description	11
<b>6 Data Structure Documentation</b>	<b>13</b>
6.1 RayTracer::ALight Class Reference	13
6.1.1 Constructor & Destructor Documentation	15
6.1.1.1 ~ALight()	15
6.1.2 Member Function Documentation	15
6.1.2.1 computeDiffuseLight()	15
6.1.2.2 computeSpecular()	15
6.1.2.3 getClosestIntersection()	15
6.1.2.4 isShadowIntersection()	16
6.1.3 Field Documentation	16
6.1.3.1 color_	16
6.1.3.2 direction_	16
6.1.3.3 intensity_	16
6.1.3.4 lightVector_	16
6.1.3.5 origin_	16
6.2 RayTracer::AmbientLight Class Reference	17
6.2.1 Constructor & Destructor Documentation	19
6.2.1.1 AmbientLight() [1/2]	19
6.2.1.2 AmbientLight() [2/2]	19
6.2.2 Member Function Documentation	19
6.2.2.1 checkJsonExistence()	19
6.2.2.2 checkJsonExistenceLight()	19
6.2.2.3 computeFast()	20

6.2.2.4 computeLight()	20
6.2.2.5 parseInfo()	20
6.3 RayTracer::APrimitives Class Reference	20
6.3.1 Constructor & Destructor Documentation	23
6.3.1.1 ~APrimitives()	23
6.3.2 Member Function Documentation	23
6.3.2.1 getMaterial()	23
6.3.2.2 setMaterial()	23
6.3.3 Field Documentation	23
6.3.3.1 material_	23
6.4 RayTracer::Camera Class Reference	24
6.4.1 Constructor & Destructor Documentation	25
6.4.1.1 Camera()	25
6.4.1.2 ~Camera()	25
6.4.2 Member Function Documentation	25
6.4.2.1 canvasToViewport()	25
6.4.2.2 getBackgroundColor()	26
6.4.2.3 setBackgroundColor()	26
6.4.3 Field Documentation	26
6.4.3.1 backgroundColor_	26
6.4.3.2 imageSize_	26
6.4.3.3 origin	27
6.4.3.4 reccursionDepth_	27
6.4.3.5 rotation_	27
6.4.3.6 screen	27
6.5 RayTracer::Color Class Reference	27
6.5.1 Constructor & Destructor Documentation	28
6.5.1.1 Color() [1/2]	28
6.5.1.2 ~Color()	28
6.5.1.3 Color() [2/2]	28
6.5.2 Member Function Documentation	29
6.5.2.1 getB()	29
6.5.2.2 getBIntensity()	29
6.5.2.3 getG()	29
6.5.2.4 getGIntensity()	29
6.5.2.5 getR()	30
6.5.2.6 getRIntensity()	30
6.5.2.7 operator+()	30
6.5.3 Field Documentation	30
6.5.3.1 b	30
6.5.3.2 g	30
6.5.3.3 r	31

6.6 RayTracer::Cone Class Reference	31
6.6.1 Constructor & Destructor Documentation	34
6.6.1.1 Cone() [1/2]	34
6.6.1.2 Cone() [2/2]	34
6.6.2 Member Function Documentation	34
6.6.2.1 checkJsonExistence()	34
6.6.2.2 checkJsonExistencePrimitive()	35
6.6.2.3 checkRangeValue()	35
6.6.2.4 getIntersection()	35
6.6.2.5 getNormalVector()	35
6.6.2.6 getRotationVector()	35
6.6.2.7 parseInfo()	35
6.6.3 Field Documentation	36
6.6.3.1 angle_	36
6.6.3.2 crossingPoint_	36
6.6.3.3 rotation_	36
6.7 RayTracer::Core Class Reference	36
6.7.1 Constructor & Destructor Documentation	38
6.7.1.1 Core()	38
6.7.1.2 ~Core()	38
6.7.2 Member Function Documentation	38
6.7.2.1 getEncoder()	39
6.7.2.2 getParser()	39
6.7.2.3 getRenderer()	39
6.7.2.4 handleCommand()	39
6.7.2.5 handleRealPreview()	39
6.7.2.6 loadEncoder()	39
6.7.2.7 loadParser()	40
6.7.2.8 loadRenderer()	40
6.7.2.9 loadScene()	40
6.7.2.10 renderImage()	40
6.7.2.11 resetPrimitives()	41
6.7.2.12 setEncoder()	41
6.7.2.13 setParser()	41
6.7.2.14 setRenderer()	41
6.7.2.15 startCli()	42
6.7.3 Field Documentation	42
6.7.3.1 encoder_	42
6.7.3.2 encoderLoader_	42
6.7.3.3 parser_	42
6.7.3.4 parserLoader_	42
6.7.3.5 renderer_	42

6.7.3.6 rendererLoader_ . . . . .	43
6.8 RayTracer::Cylinder Class Reference . . . . .	43
6.8.1 Constructor & Destructor Documentation . . . . .	46
6.8.1.1 Cylinder() [1/2] . . . . .	46
6.8.1.2 Cylinder() [2/2] . . . . .	46
6.8.2 Member Function Documentation . . . . .	46
6.8.2.1 checkJsonExistence() . . . . .	46
6.8.2.2 checkJsonExistencePrimitive() . . . . .	47
6.8.2.3 checkRangeValue() . . . . .	47
6.8.2.4 getIntersection() . . . . .	47
6.8.2.5 getNormalVector() . . . . .	47
6.8.2.6 getRotationVector() . . . . .	47
6.8.2.7 parseInfo() . . . . .	47
6.8.3 Field Documentation . . . . .	48
6.8.3.1 crossingPoint_ . . . . .	48
6.8.3.2 radius_ . . . . .	48
6.8.3.3 rotation_ . . . . .	48
6.9 RayTracer::DirectionalLight Class Reference . . . . .	48
6.9.1 Constructor & Destructor Documentation . . . . .	51
6.9.1.1 DirectionalLight() [1/2] . . . . .	51
6.9.1.2 DirectionalLight() [2/2] . . . . .	51
6.9.2 Member Function Documentation . . . . .	51
6.9.2.1 checkJsonExistence() . . . . .	51
6.9.2.2 checkJsonExistenceLight() . . . . .	51
6.9.2.3 computeFast() . . . . .	52
6.9.2.4 computeLight() . . . . .	52
6.9.2.5 parseInfo() . . . . .	52
6.10 DLLoader< T > Class Template Reference . . . . .	52
6.10.1 Constructor & Destructor Documentation . . . . .	53
6.10.1.1 DLLoader() . . . . .	54
6.10.1.2 ~DLLoader() . . . . .	54
6.10.2 Member Function Documentation . . . . .	54
6.10.2.1 closeLib() . . . . .	54
6.10.2.2 getInstance() . . . . .	54
6.10.3 Field Documentation . . . . .	54
6.10.3.1 actualLib_ . . . . .	54
6.11 RayTracer::Encoder Class Reference . . . . .	55
6.11.1 Detailed Description . . . . .	57
6.11.2 Constructor & Destructor Documentation . . . . .	57
6.11.2.1 Encoder() . . . . .	57
6.11.2.2 ~Encoder() . . . . .	57
6.11.3 Member Function Documentation . . . . .	57

6.11.3.1 canClose()	57
6.11.3.2 checkEvents()	57
6.11.3.3 encodeOutput()	57
6.11.3.4 openSupport()	58
6.11.3.5 writeHeaderToPpm()	58
6.11.3.6 writePixelDataToPpm()	58
6.11.4 Field Documentation	59
6.11.4.1 canClose_	59
6.11.4.2 result_	59
6.12 RayTracer::FastRenderer Class Reference	59
6.12.1 Constructor & Destructor Documentation	62
6.12.1.1 FastRenderer()	62
6.12.1.2 ~FastRenderer()	62
6.12.2 Member Function Documentation	62
6.12.2.1 canWrite()	63
6.12.2.2 computeLight()	63
6.12.2.3 computeScene()	63
6.12.2.4 getClosestIntersection()	63
6.12.2.5 getResult()	63
6.12.2.6 getScene()	64
6.12.2.7 initIterators()	64
6.12.2.8 isCommand()	64
6.12.2.9 isGenerationDone()	64
6.12.2.10 reflectRay()	65
6.12.2.11 traceRay()	65
6.12.3 Field Documentation	65
6.12.3.1 canWrite_	65
6.12.3.2 done_	65
6.12.3.3 index	65
6.12.3.4 result_	65
6.12.3.5 scene_	66
6.12.3.6 x	66
6.12.3.7 y	66
6.13 RayTracer::IEncoder Class Reference	66
6.13.1 Constructor & Destructor Documentation	68
6.13.1.1 ~IEncoder()	68
6.13.2 Member Function Documentation	68
6.13.2.1 canClose()	69
6.13.2.2 checkEvents()	69
6.13.2.3 encodeOutput()	69
6.13.2.4 openSupport()	69
6.14 RayTracer::ILight Class Reference	70

6.14.1 Constructor & Destructor Documentation	71
6.14.1.1 ~ILight()	71
6.14.2 Member Function Documentation	71
6.14.2.1 computeDiffuseLight()	72
6.14.2.2 computeFast()	72
6.14.2.3 computeLight()	72
6.14.2.4 computeSpecular()	72
6.14.2.5 getClosestIntersection()	73
6.14.2.6 isShadowIntersection()	73
6.14.2.7 parseInfo()	73
6.15 RayTracer::SfmlEncoder::Image Struct Reference	74
6.15.1 Field Documentation	74
6.15.1.1 sprite	74
6.15.1.2 texture	74
6.16 RayTracer::imageSize Struct Reference	75
6.16.1 Field Documentation	75
6.16.1.1 height	75
6.16.1.2 width	75
6.17 RayTracer::intersection Struct Reference	76
6.17.1 Detailed Description	76
6.17.2 Field Documentation	76
6.17.2.1 closestPrim	77
6.17.2.2 closestT	77
6.18 RayTracer::IParser Class Reference	77
6.18.1 Member Function Documentation	79
6.18.1.1 getEncoderName()	79
6.18.1.2 getFastRendererFileName()	80
6.18.1.3 getLights()	80
6.18.1.4 getOutputFileName()	80
6.18.1.5 getPrimitives()	80
6.18.1.6 getRendererName()	81
6.18.1.7 isFastRendererEnabled()	81
6.18.1.8 parse()	81
6.19 RayTracer::IPrimitives Class Reference	81
6.19.1 Constructor & Destructor Documentation	83
6.19.1.1 ~IPrimitives()	83
6.19.2 Member Function Documentation	83
6.19.2.1 getIntersection()	83
6.19.2.2 getMaterial()	83
6.19.2.3 getNormalVector()	83
6.19.2.4 getRotationVector()	83
6.19.2.5 parseInfo()	84



6.19.2.6 setMaterial()	84
6.20 RayTracer::IRenderer Class Reference	84
6.20.1 Member Function Documentation	85
6.20.1.1 canWrite()	85
6.20.1.2 computeScene()	86
6.20.1.3 getResult()	86
6.20.1.4 getScene()	86
6.20.1.5 initIterators()	86
6.20.1.6 isCommand()	87
6.20.1.7 isGenerationDone()	87
6.21 RayTracer::LiveRenderer Class Reference	87
6.21.1 Constructor & Destructor Documentation	90
6.21.1.1 LiveRenderer()	90
6.21.1.2 ~LiveRenderer()	90
6.21.2 Member Function Documentation	90
6.21.2.1 canWrite()	91
6.21.2.2 computeLight()	91
6.21.2.3 computeScene()	91
6.21.2.4 getClosestIntersection()	91
6.21.2.5 getResult()	91
6.21.2.6 getScene()	92
6.21.2.7 initIterators()	92
6.21.2.8 isCommand()	92
6.21.2.9 isGenerationDone()	92
6.21.2.10 reflectRay()	93
6.21.2.11 traceRay()	93
6.21.3 Field Documentation	93
6.21.3.1 canWrite_	93
6.21.3.2 done_	93
6.21.3.3 index	93
6.21.3.4 result_	93
6.21.3.5 scene_	94
6.21.3.6 x	94
6.21.3.7 y	94
6.22 RayTracer::Material Class Reference	94
6.22.1 Detailed Description	96
6.22.2 Constructor & Destructor Documentation	96
6.22.2.1 Material() [1/2]	96
6.22.2.2 ~Material()	96
6.22.2.3 Material() [2/2]	96
6.22.3 Member Function Documentation	97
6.22.3.1 getColor()	97

6.22.3.2 getReflective()	97
6.22.3.3 getSpecular()	97
6.22.3.4 getTransparency()	98
6.22.3.5 setColor()	98
6.22.3.6 setReflective()	98
6.22.3.7 setSpecular()	98
6.22.3.8 setTransparency()	99
6.22.4 Field Documentation	99
6.22.4.1 color_	99
6.22.4.2 reflective_	99
6.22.4.3 specular_	99
6.22.4.4 transparency_	99
6.23 RayTracer::Parser Class Reference	100
6.23.1 Detailed Description	103
6.23.2 Constructor & Destructor Documentation	103
6.23.2.1 Parser() [1/2]	103
6.23.2.2 Parser() [2/2]	103
6.23.2.3 ~Parser()	103
6.23.3 Member Function Documentation	103
6.23.3.1 checkJsonCamera()	103
6.23.3.2 checkJsonExistence()	104
6.23.3.3 checkJsonGlobal()	104
6.23.3.4 checkRangeValue()	104
6.23.3.5 getEncoderName()	105
6.23.3.6 getFastRendererFileName()	105
6.23.3.7 getLights()	105
6.23.3.8 getOutputFileName()	106
6.23.3.9 getParsedCamera()	106
6.23.3.10 getPrimitives()	106
6.23.3.11 getRendererName()	106
6.23.3.12 isFastRendererEnabled()	107
6.23.3.13 loadLight()	107
6.23.3.14 loadPrimitive()	107
6.23.3.15 parse()	108
6.23.3.16 parseCamera()	108
6.23.3.17 parseCorePlugins()	108
6.23.3.18 parseImportedGlobal()	108
6.23.3.19 parseImportedObj()	108
6.23.3.20 parseImportedScene()	109
6.23.3.21 parseLight()	109
6.23.3.22 parseObject()	109
6.23.3.23 parsePrimitive()	109

6.23.4 Field Documentation	109
6.23.4.1 cam_	110
6.23.4.2 encoderName_	110
6.23.4.3 fasterRenderEnabled_	110
6.23.4.4 fasterRenderName_	110
6.23.4.5 importedScene_	110
6.23.4.6 lightLoader_	110
6.23.4.7 lights_	110
6.23.4.8 outputFile_	110
6.23.4.9 primitiveLoader_	111
6.23.4.10 primitives_	111
6.23.4.11 rendererName_	111
6.23.4.12 scene_	111
6.24 ParsingError Class Reference	111
6.24.1 Constructor & Destructor Documentation	112
6.24.1.1 ParsingError()	112
6.24.1.2 ~ParsingError()	112
6.24.2 Member Function Documentation	113
6.24.2.1 what()	113
6.24.3 Field Documentation	113
6.24.3.1 msg_	113
6.25 RayTracer::Plan Class Reference	113
6.25.1 Constructor & Destructor Documentation	116
6.25.1.1 Plan() [1/4]	116
6.25.1.2 Plan() [2/4]	116
6.25.1.3 Plan() [3/4]	116
6.25.1.4 Plan() [4/4]	117
6.25.2 Member Function Documentation	117
6.25.2.1 checkJsonExistence()	117
6.25.2.2 checkJsonExistencePrimitive()	117
6.25.2.3 checkRangeValue()	117
6.25.2.4 getIntersection()	117
6.25.2.5 getNormalVector()	117
6.25.2.6 getRotationVector()	118
6.25.2.7 operator=() [1/2]	118
6.25.2.8 operator=() [2/2]	118
6.25.2.9 parseInfo()	118
6.25.3 Field Documentation	118
6.25.3.1 d	118
6.25.3.2 normalVector_	118
6.25.3.3 rotation_	119
6.26 Math::Point3D Class Reference	119

6.26.1 Detailed Description	120
6.26.2 Constructor & Destructor Documentation	120
6.26.2.1 Point3D() [1/4]	120
6.26.2.2 Point3D() [2/4]	120
6.26.2.3 Point3D() [3/4]	121
6.26.2.4 Point3D() [4/4]	121
6.26.2.5 ~Point3D()	121
6.26.3 Member Function Documentation	121
6.26.3.1 operator+()	121
6.26.3.2 operator=() [1/2]	122
6.26.3.3 operator=() [2/2]	122
6.26.3.4 rotateX()	122
6.26.3.5 rotateY()	123
6.26.3.6 rotateZ()	123
6.26.4 Field Documentation	123
6.26.4.1 x	123
6.26.4.2 y	123
6.26.4.3 z	124
6.27 RayTracer::PointLight Class Reference	124
6.27.1 Constructor & Destructor Documentation	127
6.27.1.1 PointLight() [1/2]	127
6.27.1.2 PointLight() [2/2]	127
6.27.1.3 ~PointLight()	127
6.27.2 Member Function Documentation	127
6.27.2.1 checkJsonExistence()	127
6.27.2.2 checkJsonExistenceLight()	128
6.27.2.3 computeFast()	128
6.27.2.4 computeLight()	128
6.27.2.5 parseInfo()	128
6.28 RayTracer::Ray Class Reference	129
6.28.1 Detailed Description	130
6.28.2 Constructor & Destructor Documentation	130
6.28.2.1 Ray() [1/4]	130
6.28.2.2 Ray() [2/4]	130
6.28.2.3 Ray() [3/4]	131
6.28.2.4 Ray() [4/4]	131
6.28.2.5 ~Ray()	131
6.28.3 Member Function Documentation	131
6.28.3.1 rotateRay()	131
6.28.4 Field Documentation	132
6.28.4.1 direction	132
6.28.4.2 origin	132

6.29 Math::Rectangle3D Class Reference	132
6.29.1 Detailed Description	134
6.29.2 Constructor & Destructor Documentation	134
6.29.2.1 Rectangle3D() [1/2]	134
6.29.2.2 Rectangle3D() [2/2]	134
6.29.2.3 ~Rectangle3D()	134
6.29.3 Field Documentation	134
6.29.3.1 bottom_side	135
6.29.3.2 left_side	135
6.29.3.3 origin	135
6.30 RayTracer::Renderer Class Reference	135
6.30.1 Constructor & Destructor Documentation	138
6.30.1.1 Renderer()	138
6.30.1.2 ~Renderer()	138
6.30.2 Member Function Documentation	138
6.30.2.1 canWrite()	139
6.30.2.2 computeLight()	139
6.30.2.3 computeScene()	139
6.30.2.4 getClosestIntersection()	139
6.30.2.5 getResult()	139
6.30.2.6 getScene()	140
6.30.2.7 initIterators()	140
6.30.2.8 isCommand()	140
6.30.2.9 isGenerationDone()	140
6.30.2.10 reflectRay()	141
6.30.2.11 traceRay()	141
6.30.3 Field Documentation	141
6.30.3.1 canWrite_	141
6.30.3.2 done_	141
6.30.3.3 index	141
6.30.3.4 result_	141
6.30.3.5 scene_	142
6.30.3.6 x	142
6.30.3.7 y	142
6.31 RayTracer::Scene Class Reference	142
6.31.1 Detailed Description	144
6.31.2 Constructor & Destructor Documentation	144
6.31.2.1 Scene()	144
6.31.2.2 ~Scene()	144
6.31.3 Member Function Documentation	144
6.31.3.1 addLight()	144
6.31.3.2 addPrimitive()	145

6.31.3.3	getCamera()	145
6.31.3.4	getLights()	145
6.31.3.5	getPrimitives()	146
6.31.3.6	setCamera()	146
6.31.4	Field Documentation	146
6.31.4.1	cam_	146
6.31.4.2	lights_	146
6.31.4.3	primitives_	146
6.32	RayTracer::SfmlEncoder Class Reference	147
6.32.1	Constructor & Destructor Documentation	149
6.32.1.1	SfmlEncoder()	149
6.32.1.2	~SfmlEncoder()	149
6.32.2	Member Function Documentation	150
6.32.2.1	canClose()	150
6.32.2.2	checkEvents()	150
6.32.2.3	clearWindow()	150
6.32.2.4	closeWindow()	150
6.32.2.5	createWindow()	150
6.32.2.6	displaySfml()	150
6.32.2.7	encodeOutput()	151
6.32.2.8	loadImage()	151
6.32.2.9	openSupport()	151
6.32.2.10	renderWindow()	152
6.32.2.11	setFrameRateLimit()	152
6.32.3	Field Documentation	152
6.32.3.1	canClose_	152
6.32.3.2	frameRateLimit_	152
6.32.3.3	height_	152
6.32.3.4	magicNumber_	152
6.32.3.5	maxValue_	153
6.32.3.6	result_	153
6.32.3.7	width_	153
6.32.3.8	window_	153
6.33	RayTracer::Sphere Class Reference	153
6.33.1	Constructor & Destructor Documentation	156
6.33.1.1	Sphere() [1/2]	156
6.33.1.2	Sphere() [2/2]	156
6.33.1.3	~Sphere()	156
6.33.2	Member Function Documentation	156
6.33.2.1	checkJsonExistence()	157
6.33.2.2	checkJsonExistencePrimitive()	157
6.33.2.3	checkRangeValue()	157

6.33.2.4 getIntersection()	157
6.33.2.5 getNormalVector()	157
6.33.2.6 getRotationVector()	157
6.33.2.7 parseInfo()	158
6.33.3 Field Documentation	158
6.33.3.1 center_	158
6.33.3.2 radius_	158
6.34 Math::Vector3D Class Reference	158
6.34.1 Constructor & Destructor Documentation	160
6.34.1.1 Vector3D() [1/4]	160
6.34.1.2 Vector3D() [2/4]	161
6.34.1.3 Vector3D() [3/4]	161
6.34.1.4 Vector3D() [4/4]	161
6.34.1.5 ~Vector3D()	161
6.34.2 Member Function Documentation	161
6.34.2.1 dot()	161
6.34.2.2 length()	162
6.34.2.3 operator*() [1/2]	162
6.34.2.4 operator*() [2/2]	162
6.34.2.5 operator*==() [1/2]	162
6.34.2.6 operator*==() [2/2]	163
6.34.2.7 operator+()	163
6.34.2.8 operator+=()	163
6.34.2.9 operator-()	165
6.34.2.10 operator-=()	165
6.34.2.11 operator/() [1/2]	165
6.34.2.12 operator/() [2/2]	167
6.34.2.13 operator/=( ) [1/2]	167
6.34.2.14 operator/=( ) [2/2]	167
6.34.2.15 operator=( ) [1/2]	169
6.34.2.16 operator=( ) [2/2]	169
6.34.2.17 rotate()	169
6.34.2.18 rotateX()	169
6.34.2.19 rotateY()	170
6.34.2.20 rotateZ()	170
6.34.3 Field Documentation	170
6.34.3.1 x	170
6.34.3.2 y	170
6.34.3.3 z	171
<b>7 File Documentation</b>	<b>173</b>
7.1 include/Camera.hpp File Reference	173

7.1.1 Macro Definition Documentation	174
7.1.1.1 PI_RAD	174
7.2 include/Core.hpp File Reference	174
7.3 include/DLLoader.hpp File Reference	175
7.4 include/encoder/IEncoder.hpp File Reference	176
7.5 include/Error.hpp File Reference	177
7.5.1 Macro Definition Documentation	177
7.5.1.1 EPITECH_ERROR	178
7.6 include/lights/ALight.hpp File Reference	178
7.7 include/lights/ILight.hpp File Reference	179
7.7.1 Typedef Documentation	180
7.7.1.1 json	180
7.8 include/material/Color.hpp File Reference	180
7.9 include/material/Material.hpp File Reference	181
7.10 include/maths/Point3D.hpp File Reference	181
7.11 include/maths/Ray.hpp File Reference	183
7.12 include/maths/Rectangle3D.hpp File Reference	184
7.13 include/maths/Vector3D.hpp File Reference	185
7.14 include/parser/IParser.hpp File Reference	186
7.15 include/parser/Parser.hpp File Reference	186
7.15.1 Typedef Documentation	187
7.15.1.1 json	188
7.16 include/primitives/APrimitives.hpp File Reference	188
7.16.1 Macro Definition Documentation	189
7.16.1.1 PI_RAD	189
7.17 include/primitives/IPrimitives.hpp File Reference	189
7.17.1 Typedef Documentation	190
7.17.1.1 json	190
7.18 include/RayTracer.hpp File Reference	190
7.19 include/renderer/IRenderer.hpp File Reference	191
7.20 include/Scene.hpp File Reference	192
7.21 include/Size.hpp File Reference	193
7.22 src/Core.cpp File Reference	193
7.23 src/lights/ALight.cpp File Reference	194
7.24 src/Main.cpp File Reference	195
7.24.1 Function Documentation	195
7.24.1.1 checkArgs()	195
7.24.1.2 main()	195
7.25 src/maths/Point3D.cpp File Reference	196
7.26 src/maths/Ray.cpp File Reference	196
7.27 src/maths/Rectangle3D.cpp File Reference	197
7.28 src/maths/Vector3D.cpp File Reference	198



7.29 src/parser/CheckArgs.cpp File Reference . . . . .	199
7.29.1 Function Documentation . . . . .	200
7.29.1.1 checkArgs() . . . . .	200
7.29.1.2 displayUsage() . . . . .	200
7.30 src/parser/CheckJsonExistence.cpp File Reference . . . . .	201
7.31 src/parser/Error.cpp File Reference . . . . .	201
7.32 src/parser/ParseImportedScene.cpp File Reference . . . . .	202
7.33 src/parser/ParseLight.cpp File Reference . . . . .	202
7.34 src/parser/ParsePrimitive.cpp File Reference . . . . .	203
7.35 src/parser/Parser.cpp File Reference . . . . .	203
7.36 src/plugins/encoder/ppmEncoder/Encoder.cpp File Reference . . . . .	204
7.37 src/plugins/encoder/sfmlEncoder/Encoder.cpp File Reference . . . . .	205
7.38 src/plugins/encoder/ppmEncoder/Encoder.hpp File Reference . . . . .	205
7.39 src/plugins/encoder/sfmlEncoder/Encoder.hpp File Reference . . . . .	206
7.39.1 Macro Definition Documentation . . . . .	207
7.39.1.1 BYTE_OF_RGBA_FORMAT . . . . .	207
7.39.1.2 FRAME_RATE_LIMIT . . . . .	207
7.39.1.3 HEADER_MAGIC_NUMBER . . . . .	208
7.39.1.4 MAX_VALUE . . . . .	208
7.40 src/plugins/encoder/ppmEncoder/EncoderEntry.cpp File Reference . . . . .	208
7.40.1 Function Documentation . . . . .	208
7.40.1.1 entryPoint() . . . . .	209
7.41 src/plugins/encoder/sfmlEncoder/EncoderEntry.cpp File Reference . . . . .	209
7.41.1 Function Documentation . . . . .	209
7.41.1.1 entryPoint() . . . . .	209
7.42 src/plugins/lights/ambient/AmbientEntry.cpp File Reference . . . . .	210
7.42.1 Function Documentation . . . . .	210
7.42.1.1 entryPoint() . . . . .	210
7.43 src/plugins/lights/ambient/AmbientLight.cpp File Reference . . . . .	211
7.44 src/plugins/lights/ambient/AmbientLight.hpp File Reference . . . . .	211
7.45 src/plugins/lights/directional/DirectionalEntry.cpp File Reference . . . . .	213
7.45.1 Function Documentation . . . . .	213
7.45.1.1 entryPoint() . . . . .	214
7.46 src/plugins/lights/directional/DirectionalLight.cpp File Reference . . . . .	214
7.47 src/plugins/lights/directional/DirectionalLight.hpp File Reference . . . . .	215
7.48 src/plugins/lights/point/PointEntry.cpp File Reference . . . . .	216
7.48.1 Function Documentation . . . . .	217
7.48.1.1 entryPoint() . . . . .	217
7.49 src/plugins/lights/point/PointLight.cpp File Reference . . . . .	217
7.50 src/plugins/lights/point/PointLight.hpp File Reference . . . . .	218
7.51 src/plugins/primitives/cone/Cone.cpp File Reference . . . . .	219
7.52 src/plugins/primitives/cone/Cone.hpp File Reference . . . . .	220

7.53 src/plugins/primitives/cone/ConeEntry.cpp File Reference	221
7.53.1 Function Documentation	221
7.53.1.1 entryPoint()	221
7.54 src/plugins/primitives/cylinder/Cylinder.cpp File Reference	222
7.55 src/plugins/primitives/cylinder/Cylinder.hpp File Reference	222
7.56 src/plugins/primitives/cylinder/CylinderEntry.cpp File Reference	224
7.56.1 Function Documentation	224
7.56.1.1 entryPoint()	224
7.57 src/plugins/primitives/plan/Plan.cpp File Reference	225
7.58 src/plugins/primitives/plan/Plan.hpp File Reference	225
7.58.1 Typedef Documentation	227
7.58.1.1 json	227
7.59 src/plugins/primitives/plan/PlanEntry.cpp File Reference	227
7.59.1 Function Documentation	228
7.59.1.1 entryPoint()	228
7.60 src/plugins/primitives/sphere/Sphere.cpp File Reference	228
7.61 src/plugins/primitives/sphere/Sphere.hpp File Reference	229
7.61.1 Typedef Documentation	230
7.61.1.1 json	230
7.62 src/plugins/primitives/sphere/SphereEntry.cpp File Reference	230
7.62.1 Function Documentation	231
7.62.1.1 entryPoint()	231
7.63 src/plugins/renderer/baseRenderer/Renderer.cpp File Reference	231
7.64 src/plugins/renderer/fastRenderer/Renderer.cpp File Reference	231
7.65 src/plugins/renderer/liveRenderer/Renderer.cpp File Reference	232
7.66 src/plugins/renderer/baseRenderer/Renderer.hpp File Reference	233
7.67 src/plugins/renderer/fastRenderer/Renderer.hpp File Reference	234
7.68 src/plugins/renderer/liveRenderer/Renderer.hpp File Reference	235
7.69 src/plugins/renderer/baseRenderer/RendererEntry.cpp File Reference	236
7.69.1 Function Documentation	236
7.69.1.1 entryPoint()	236
7.70 src/plugins/renderer/fastRenderer/RendererEntry.cpp File Reference	237
7.70.1 Function Documentation	237
7.70.1.1 entryPoint()	237
7.71 src/plugins/renderer/liveRenderer/RendererEntry.cpp File Reference	238
7.71.1 Function Documentation	238
7.71.1.1 entryPoint()	238
7.72 src/utills/Camera.cpp File Reference	239
7.73 src/utills/Color.cpp File Reference	239
7.74 src/utills/Material.cpp File Reference	240
7.75 src/utills/Primitives.cpp File Reference	240
7.76 src/utills/Scene.cpp File Reference	241





# Chapter 1

## Namespace Index

### 1.1 Namespace List

Here is a list of all namespaces with brief descriptions:

<a href="#">Math</a>	Namespace for the math functions . . . . .	<a href="#">9</a>
<a href="#">RayTracer</a>	Namespace for the raytracer . . . . .	<a href="#">10</a>



## Chapter 2

# Hierarchical Index

## 2.1 Class Hierarchy

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

RayTracer::Camera	24
RayTracer::Color	27
RayTracer::Core	36
DLLoader< T >	52
DLLoader< RayTracer::IEncoder >	52
DLLoader< RayTracer::ILight >	52
DLLoader< RayTracer::IParser >	52
DLLoader< RayTracer::IPrimitives >	52
DLLoader< RayTracer::IRenderer >	52
std::exception	
ParsingError	111
RayTracer::IEncoder	66
RayTracer::Encoder	55
RayTracer::SfmlEncoder	147
RayTracer::ILight	70
RayTracer::ALight	13
RayTracer::AmbientLight	17
RayTracer::DirectionalLight	48
RayTracer::PointLight	124
RayTracer::SfmlEncoder::Image	74
RayTracer::imageSize	75
RayTracer::intersection	76
RayTracer::IParser	77
RayTracer::Parser	100
RayTracer::IPrimitives	81
RayTracer::APrimitives	20
RayTracer::Cone	31
RayTracer::Cylinder	43
RayTracer::Plan	113
RayTracer::Sphere	153
RayTracer::IRenderer	84
RayTracer::FastRenderer	59
RayTracer::LiveRenderer	87
RayTracer::Renderer	135

RayTracer::Material . . . . .	94
Math::Point3D . . . . .	119
RayTracer::Ray . . . . .	129
Math::Rectangle3D . . . . .	132
RayTracer::Scene . . . . .	142
Math::Vector3D . . . . .	158



## Chapter 3

# Data Structure Index

### 3.1 Data Structures

Here are the data structures with brief descriptions:

RayTracer::ALight	13
RayTracer::AmbientLight	17
RayTracer::APrimitives	20
RayTracer::Camera	24
RayTracer::Color	27
RayTracer::Cone	31
RayTracer::Core	36
RayTracer::Cylinder	43
RayTracer::DirectionalLight	48
DLLoader< T >	52
RayTracer::Encoder	
Encoder class	55
RayTracer::FastRenderer	59
RayTracer::IEncoder	66
RayTracer::ILight	70
RayTracer::SfmlEncoder::Image	74
RayTracer::imageSize	75
RayTracer::intersection	
Struct for the intersection	76
RayTracer::IParser	77
RayTracer::IPrimitives	81
RayTracer::IRenderer	84
RayTracer::LiveRenderer	87
RayTracer::Material	
Material class representing the properties of a surface of an object	94
RayTracer::Parser	
Class for the parser	100
ParsingError	111
RayTracer::Plan	113
Math::Point3D	
Class for the 3D point	119
RayTracer::PointLight	124
RayTracer::Ray	
Class for the ray	129
Math::Rectangle3D	
Class for the 3D rectangle	132

<a href="#">RayTracer::Renderer</a>	135
<a href="#">RayTracer::Scene</a>	
Class for the scene	142
<a href="#">RayTracer::SfmlEncoder</a>	147
<a href="#">RayTracer::Sphere</a>	153
<a href="#">Math::Vector3D</a>	158

## Chapter 4

# File Index

### 4.1 File List

Here is a list of all files with brief descriptions:

include/Camera.hpp	173
include/Core.hpp	174
include/DLLoader.hpp	175
include/Error.hpp	177
include/RayTracer.hpp	190
include/Scene.hpp	192
include/Size.hpp	193
include/encoder/IEncoder.hpp	176
include/lights/ALight.hpp	178
include/lights/ILight.hpp	179
include/material/Color.hpp	180
include/material/Material.hpp	181
include/math/Point3D.hpp	181
include/math/Ray.hpp	183
include/math/Rectangle3D.hpp	184
include/math/Vector3D.hpp	185
include/parser/IParser.hpp	186
include/parser/Parser.hpp	186
include/primitives/APrimitives.hpp	188
include/primitives/IPrimitives.hpp	189
include/renderer/IRenderer.hpp	191
src/Core.cpp	193
src/Main.cpp	195
src/lights/ALight.cpp	194
src/math/Point3D.cpp	196
src/math/Ray.cpp	196
src/math/Rectangle3D.cpp	197
src/math/Vector3D.cpp	198
src/parser/CheckArgs.cpp	199
src/parser/CheckJsonExistence.cpp	201
src/parser/Error.cpp	201
src/parser/ParseImportedScene.cpp	202
src/parser/ParseLight.cpp	202
src/parser/ParsePrimitive.cpp	203
src/parser/Parser.cpp	203

src/plugins/encoder/ppmEncoder/Encoder.cpp	204
src/plugins/encoder/ppmEncoder/Encoder.hpp	205
src/plugins/encoder/ppmEncoder/EncoderEntry.cpp	208
src/plugins/encoder/sfmlEncoder/Encoder.cpp	205
src/plugins/encoder/sfmlEncoder/Encoder.hpp	206
src/plugins/encoder/sfmlEncoder/EncoderEntry.cpp	209
src/plugins/lights/ambient/AmbientEntry.cpp	210
src/plugins/lights/ambient/AmbientLight.cpp	211
src/plugins/lights/ambient/AmbientLight.hpp	211
src/plugins/lights/directional/DirectionalEntry.cpp	213
src/plugins/lights/directional/DirectionalLight.cpp	214
src/plugins/lights/directional/DirectionalLight.hpp	215
src/plugins/lights/point/PointEntry.cpp	216
src/plugins/lights/point/PointLight.cpp	217
src/plugins/lights/point/PointLight.hpp	218
src/plugins/primitives/cone/Cone.cpp	219
src/plugins/primitives/cone/Cone.hpp	220
src/plugins/primitives/cone/ConeEntry.cpp	221
src/plugins/primitives/cylinder/Cylinder.cpp	222
src/plugins/primitives/cylinder/Cylinder.hpp	222
src/plugins/primitives/cylinder/CylinderEntry.cpp	224
src/plugins/primitives/plan/Plan.cpp	225
src/plugins/primitives/plan/Plan.hpp	225
src/plugins/primitives/plan/PlanEntry.cpp	227
src/plugins/primitives/sphere/Sphere.cpp	228
src/plugins/primitives/sphere/Sphere.hpp	229
src/plugins/primitives/sphere/SphereEntry.cpp	230
src/plugins/renderer/baseRenderer/Renderer.cpp	231
src/plugins/renderer/baseRenderer/Renderer.hpp	233
src/plugins/renderer/baseRenderer/RendererEntry.cpp	236
src/plugins/renderer/fastRenderer/Renderer.cpp	231
src/plugins/renderer/fastRenderer/Renderer.hpp	234
src/plugins/renderer/fastRenderer/RendererEntry.cpp	237
src/plugins/renderer/liveRenderer/Renderer.cpp	232
src/plugins/renderer/liveRenderer/Renderer.hpp	235
src/plugins/renderer/liveRenderer/RendererEntry.cpp	238
src/utls/Camera.cpp	239
src/utls/Color.cpp	239
src/utls/Material.cpp	240
src/utls/Primitives.cpp	240
src/utls/Scene.cpp	241

## Chapter 5

# Namespace Documentation

### 5.1 Math Namespace Reference

Namespace for the math functions.

#### Data Structures

- class [Point3D](#)  
*Class for the 3D point.*
- class [Rectangle3D](#)  
*Class for the 3D rectangle.*
- class [Vector3D](#)

#### Functions

- `std::ostream & operator<< (std::ostream &os, const Point3D &vect)`  
*Outputs a human-readable representation of the given point to the given output stream.*
- `std::ostream & operator<< (std::ostream &os, const Vector3D &vect)`  
*Calculates the cross product of two vectors.*

#### 5.1.1 Detailed Description

Namespace for the math functions.

Namespace for the math library.

#### 5.1.2 Function Documentation

##### 5.1.2.1 `operator<<()` [1/2]

```
std::ostream & Math::operator<< (  
    std::ostream & os,  
    const Point3D & vect )
```

Outputs a human-readable representation of the given point to the given output stream.

**Parameters**

<i>os</i>	The output stream to write to.
<i>vect</i>	The point to output.

**Returns**

The output stream after writing the point.

**5.1.2.2 operator<<() [2/2]**

```
std::ostream & Math::operator<< (
    std::ostream & os,
    const Vector3D & vect )
```

Calculates the cross product of two vectors.

**Parameters**

<i>vector1</i>	The first vector.
<i>vector2</i>	The second vector.

**Returns**

The cross product of the two vectors as a new vector.

**5.2 RayTracer Namespace Reference**

Namespace for the raytracer.

**Data Structures**

- class [Camera](#)
- class [Core](#)
- class [IEncoder](#)
- class [ALight](#)
- class [ILight](#)
- class [Color](#)
- class [Material](#)
  - [Material](#) class representing the properties of a surface of an object.*
- class [Ray](#)
  - Class for the ray.*
- class [IParser](#)
- class [Parser](#)
  - Class for the parser.*

- class [APrimitives](#)
- class [IPrimitives](#)
- struct [intersection](#)  
*Struct for the intersection.*
- class [IRenderer](#)
- class [Scene](#)  
*Class for the scene.*
- struct [imageSize](#)
- class [Encoder](#)  
*Encoder class.*
- class [SfmlEncoder](#)
- class [AmbientLight](#)
- class [DirectionalLight](#)
- class [PointLight](#)
- class [Cone](#)
- class [Cylinder](#)
- class [Plan](#)
- class [Sphere](#)
- class [Renderer](#)
- class [FastRenderer](#)
- class [LiveRenderer](#)

### 5.2.1 Detailed Description

Namespace for the raytracer.





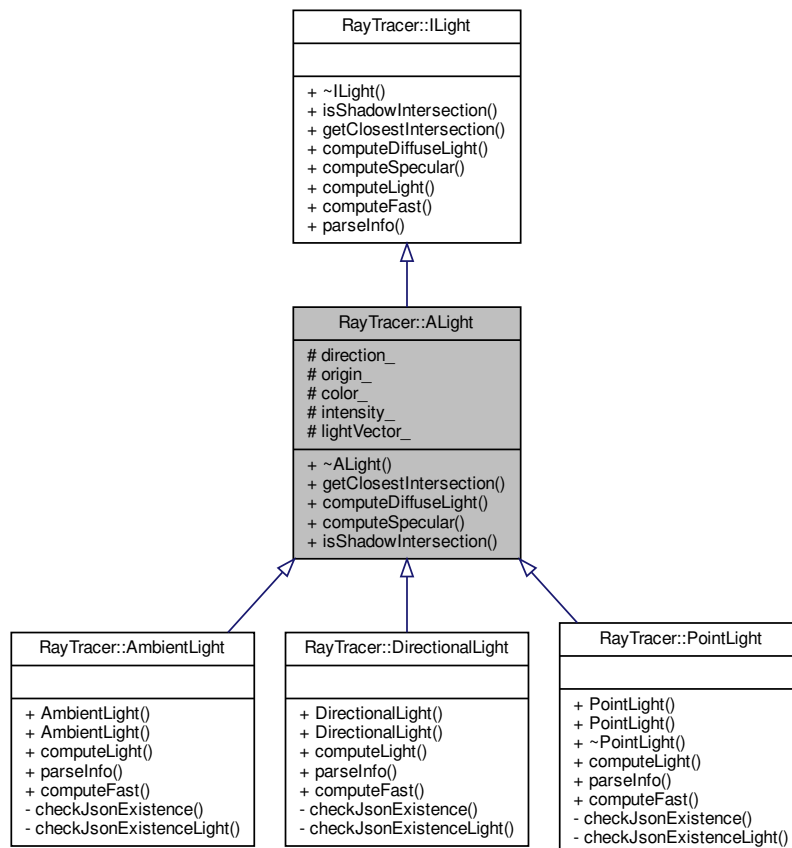
## Chapter 6

# Data Structure Documentation

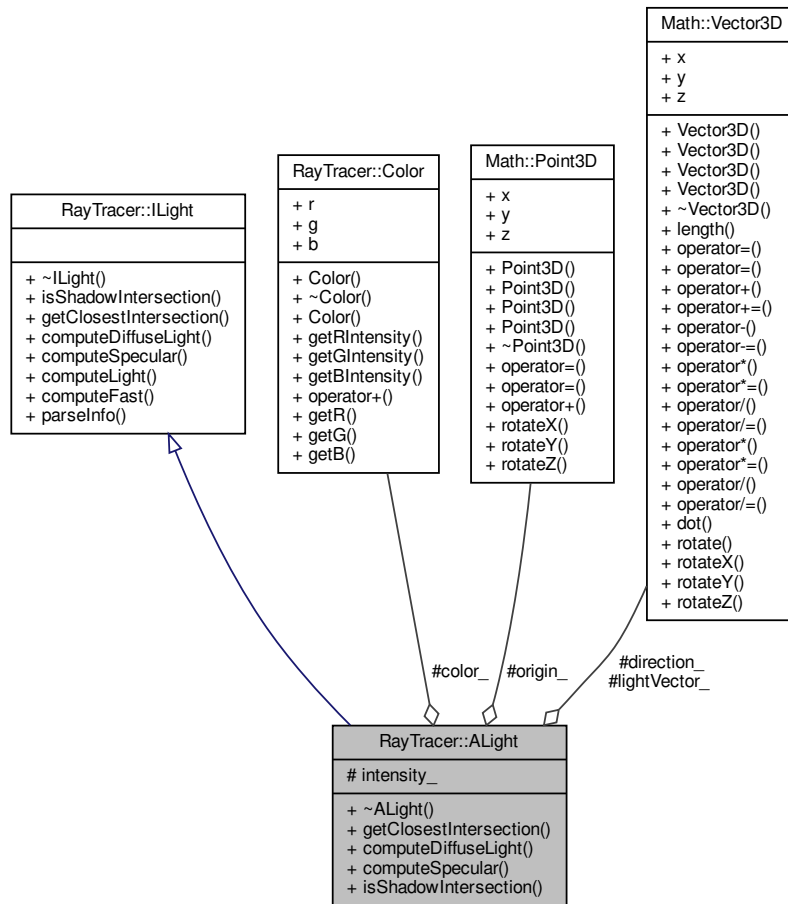
### 6.1 RayTracer::ALight Class Reference

```
#include <ALight.hpp>
```

Inheritance diagram for RayTracer::ALight:



Collaboration diagram for RayTracer::ALight:



## Public Member Functions

- `~ALight()` override=default
- `intersection getClosestIntersection` (`RayTracer::Ray` ray, double tMin, double tMax, `std::vector< std::unique_ptr< IPrimitives >> &primitives`) final
- `Color computeDiffuseLight` (`Math::Vector3D` normalVector) final
- `Color computeSpecular` (`Math::Vector3D` normalVector, int spec, `Math::Vector3D` rayDir) final
- `bool isShadowIntersection` (`Math::Point3D` intersectionPoint, `std::vector< std::unique_ptr< IPrimitives >> &primitives`) final

## Protected Attributes

- `Math::Vector3D direction_` {0, 0, 0}
- `Math::Point3D origin_` {0, 0, 0}
- `Color color_` {0, 0, 0}
- `double intensity_` {0.0}
- `Math::Vector3D lightVector_` {0, 0, 0}

## 6.1.1 Constructor & Destructor Documentation

### 6.1.1.1 ~ALight()

```
RayTracer::ALight::~~ALight ( ) [override], [default]
```

## 6.1.2 Member Function Documentation

### 6.1.2.1 computeDiffuseLight()

```
Color RayTracer::ALight::computeDiffuseLight (
    Math::Vector3D normalVector ) [final], [virtual]
```

Implements [RayTracer::ILight](#).

### 6.1.2.2 computeSpecular()

```
Color RayTracer::ALight::computeSpecular (
    Math::Vector3D normalVector,
    int spec,
    Math::Vector3D rayDir ) [final], [virtual]
```

Implements [RayTracer::ILight](#).

### 6.1.2.3 getClosestIntersection()

```
intersection RayTracer::ALight::getClosestIntersection (
    RayTracer::Ray ray,
    double tMin,
    double tMax,
    std::vector< std::unique_ptr< IPrimitives >> & primitives ) [final], [virtual]
```

Implements [RayTracer::ILight](#).

#### 6.1.2.4 isShadowIntersection()

```
bool RayTracer::ALight::isShadowIntersection (
    Math::Point3D intersectionPoint,
    std::vector< std::unique_ptr< IPrimitives >> & primitives ) [final], [virtual]
```

Implements [RayTracer::ILight](#).

### 6.1.3 Field Documentation

#### 6.1.3.1 color\_

```
Color RayTracer::ALight::color_ {0, 0, 0} [protected]
```

#### 6.1.3.2 direction\_

```
Math::Vector3D RayTracer::ALight::direction_ {0, 0, 0} [protected]
```

#### 6.1.3.3 intensity\_

```
double RayTracer::ALight::intensity_ {0.0} [protected]
```

#### 6.1.3.4 lightVector\_

```
Math::Vector3D RayTracer::ALight::lightVector_ {0, 0, 0} [protected]
```

#### 6.1.3.5 origin\_

```
Math::Point3D RayTracer::ALight::origin_ {0, 0, 0} [protected]
```

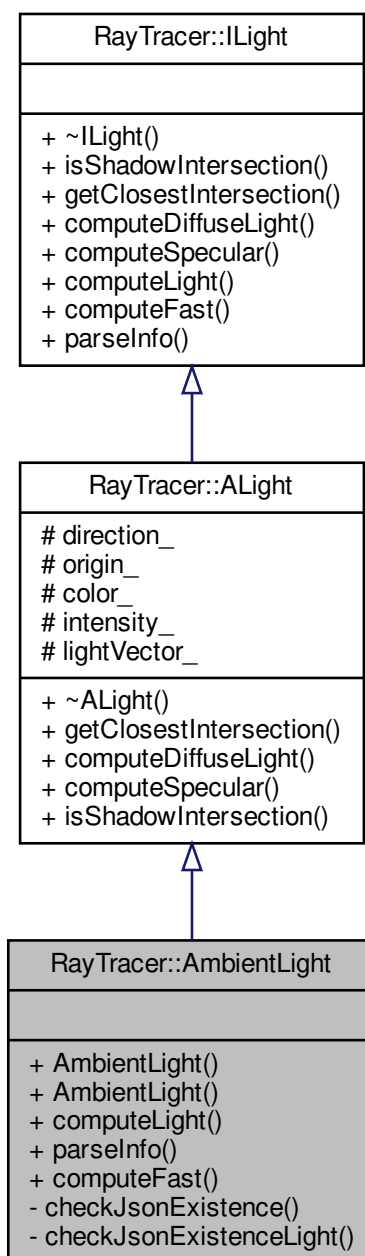
The documentation for this class was generated from the following files:

- [include/lights/ALight.hpp](#)
- [src/lights/ALight.cpp](#)

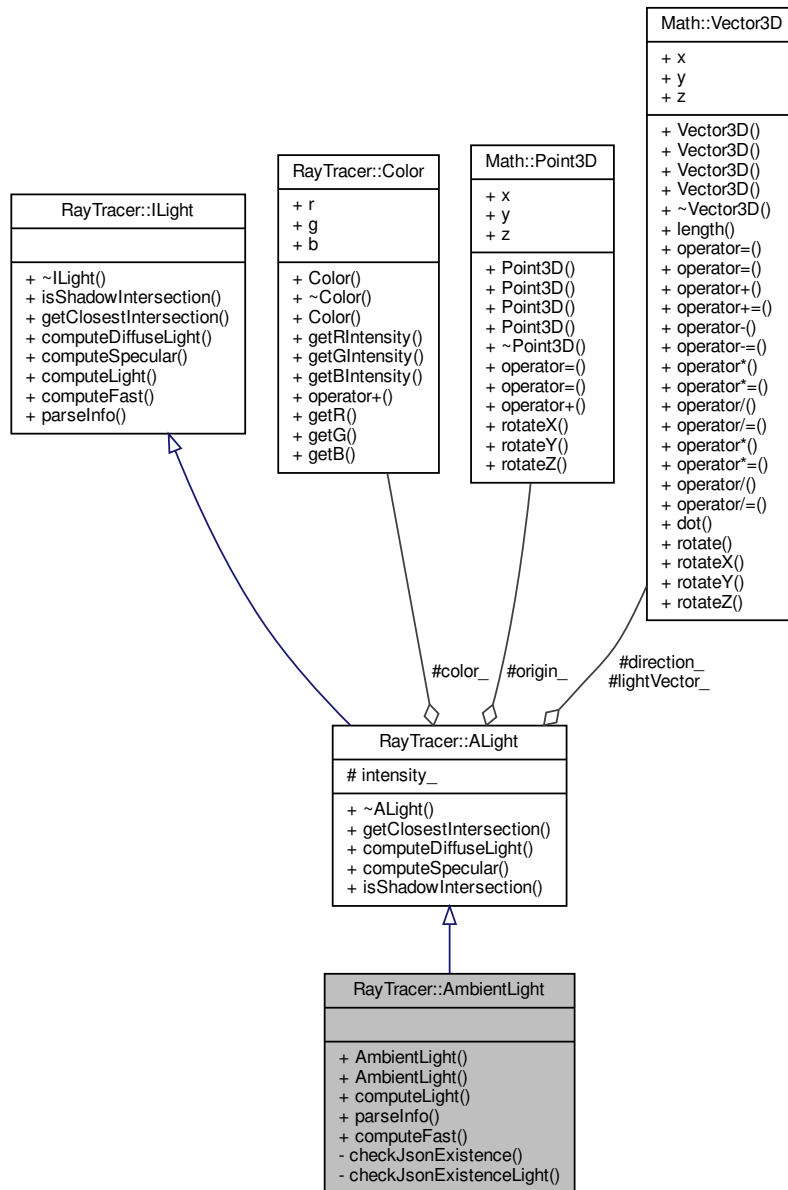
## 6.2 RayTracer::AmbientLight Class Reference

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

Inheritance diagram for RayTracer::AmbientLight:



Collaboration diagram for RayTracer::AmbientLight:



## Public Member Functions

- `AmbientLight()` = default
- `AmbientLight` (`Color` color, double intensity)
- `Color computeLight` (`Math::Vector3D` normalVector, int spec, `Math::Point3D` intersectionPoint, `Math::Vector3D` rayDir, `std::vector< std::unique_ptr< IPrimitives >> &primitives`) final
- void `parseInfo` (json object) final  
*Information parser to create the light object.*
- `Color computeFast` (`Math::Vector3D` normalVector, int spec, `Math::Point3D` intersectionPoint, `Math::Vector3D` rayDir, `std::vector< std::unique_ptr< IPrimitives >> &primitives`) final

## Private Member Functions

- void [checkJsonExistence](#) (const [json](#) &scene, const std::string &field\_name)
- void [checkJsonExistenceLight](#) (const [json](#) &scene)

## Additional Inherited Members

### 6.2.1 Constructor & Destructor Documentation

#### 6.2.1.1 AmbientLight() [1/2]

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

#### 6.2.1.2 AmbientLight() [2/2]

```
RayTracer::AmbientLight::AmbientLight (
    Color color,
    double intensity ) [explicit]
```

### 6.2.2 Member Function Documentation

#### 6.2.2.1 checkJsonExistence()

```
void RayTracer::AmbientLight::checkJsonExistence (
    const json & scene,
    const std::string & field_name ) [private]
```

#### 6.2.2.2 checkJsonExistenceLight()

```
void RayTracer::AmbientLight::checkJsonExistenceLight (
    const json & scene ) [private]
```

### 6.2.2.3 computeFast()

```
Color RayTracer::AmbientLight::computeFast (
    Math::Vector3D normalVector,
    int spec,
    Math::Point3D intersectionPoint,
    Math::Vector3D rayDir,
    std::vector< std::unique_ptr< IPrimitives >> & primitives ) [final], [virtual]
```

Implements [RayTracer::ILight](#).

### 6.2.2.4 computeLight()

```
Color RayTracer::AmbientLight::computeLight (
    Math::Vector3D normalVector,
    int spec,
    Math::Point3D intersectionPoint,
    Math::Vector3D rayDir,
    std::vector< std::unique_ptr< IPrimitives >> & primitives ) [final], [virtual]
```

Implements [RayTracer::ILight](#).

### 6.2.2.5 parseInfo()

```
void RayTracer::AmbientLight::parseInfo (
    json object ) [final], [virtual]
```

Information parser to create the light object.

#### Parameters

<i>object</i>	the json object containing light info
---------------	---------------------------------------

Implements [RayTracer::ILight](#).

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

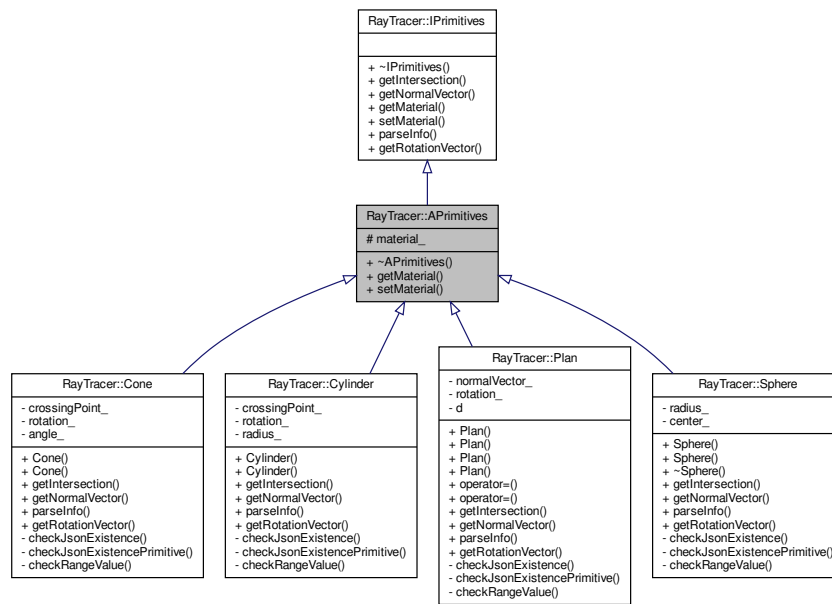
- [src/plugins/lights/ambient/AmbientLight.hpp](#)
- [src/plugins/lights/ambient/AmbientLight.cpp](#)

## 6.3 RayTracer::APrimitives Class Reference

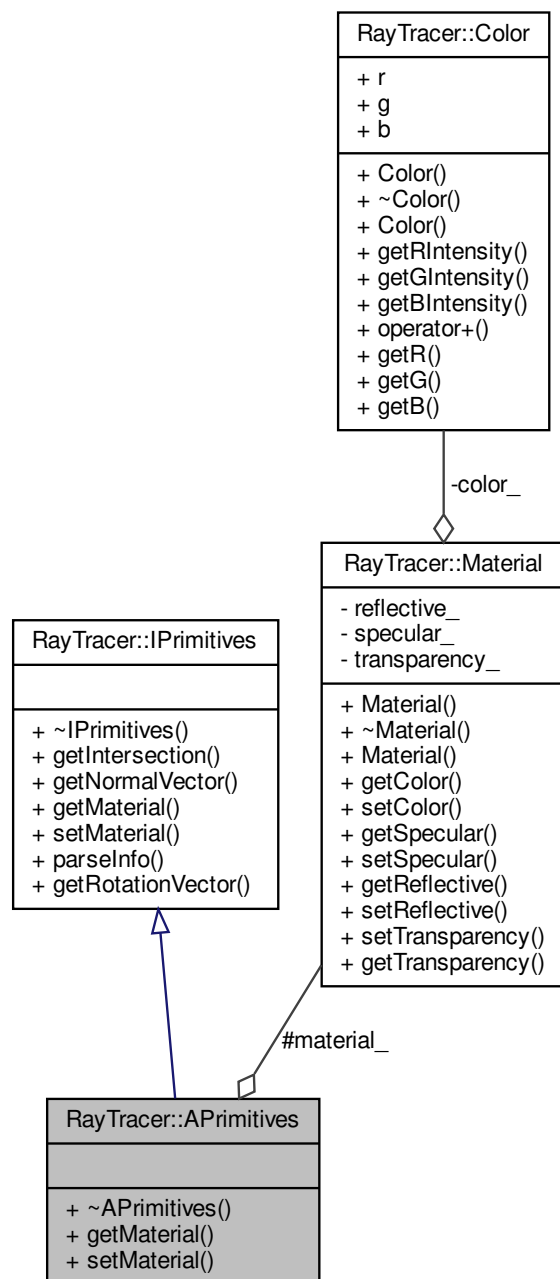
```
#include <APrimitives.hpp>
```



Inheritance diagram for RayTracer::APrimitives:



Collaboration diagram for RayTracer::APrimitives:



## Public Member Functions

- `~APrimitives ()` override=default
- `Material getMaterial ()` final
- `void setMaterial (Color color, double reflect, int specular, double transparency)` final

## Protected Attributes

- [Material material\\_](#)

## 6.3.1 Constructor & Destructor Documentation

### 6.3.1.1 ~APrimitives()

`RayTracer::APrimitives::~~APrimitives ( ) [override], [default]`

## 6.3.2 Member Function Documentation

### 6.3.2.1 getMaterial()

`RayTracer::Material RayTracer::APrimitives::getMaterial ( ) [final], [virtual]`

Implements [RayTracer::IPrimitives](#).

### 6.3.2.2 setMaterial()

```
void RayTracer::APrimitives::setMaterial (
    RayTracer::Color color,
    double reflect,
    int specular,
    double transparency ) [final], [virtual]
```

Implements [RayTracer::IPrimitives](#).

## 6.3.3 Field Documentation

### 6.3.3.1 material\_

`Material RayTracer::APrimitives::material_ [protected]`

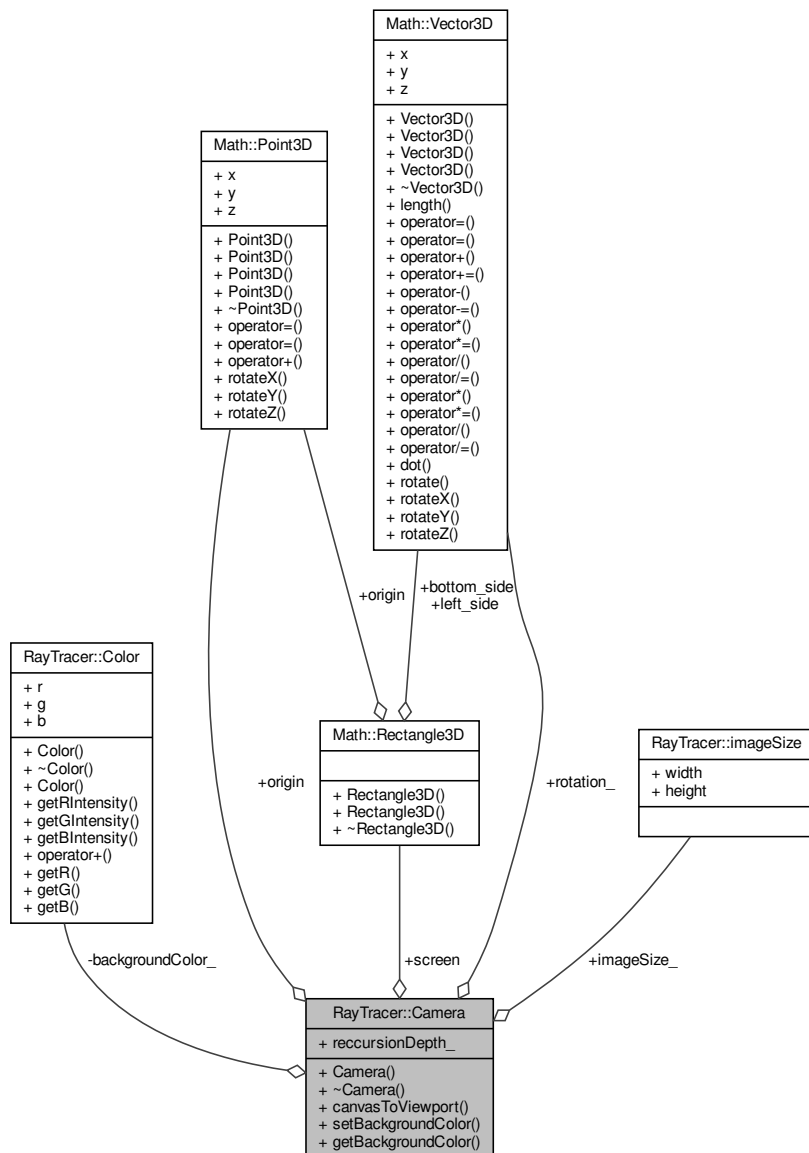
The documentation for this class was generated from the following files:

- include/primitives/[APrimitives.hpp](#)
- src/utis/[Primitives.cpp](#)

## 6.4 RayTracer::Camera Class Reference

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

Collaboration diagram for RayTracer::Camera:



### Public Member Functions

- `Camera ()`=default
- `~Camera ()`=default
- `Math::Vector3D canvasToViewport (double u, double v)`  
*Converts canvas coordinates to viewport coordinates.*
- `void setBackgroundColor (Color color)`  
*Sets the background color of the scene.*
- `Color getBackgroundColor () const`  
*Returns the background color of the scene.*

## Data Fields

- [Math::Point3D](#) `origin` { 0, 0, 0 }
- [Math::Rectangle3D](#) `screen`
- [Math::Vector3D](#) `rotation_`
- `imageSize` `imageSize_` { 400, 400 }
- `int` `reccursionDepth_` { 1 }

## Private Attributes

- [Color](#) `backgroundColor_`

## 6.4.1 Constructor & Destructor Documentation

### 6.4.1.1 Camera()

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

### 6.4.1.2 ~Camera()

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

## 6.4.2 Member Function Documentation

### 6.4.2.1 canvasToViewport()

```
Math::Vector3D RayTracer::Camera::canvasToViewport (
    double u,
    double v )
```

Converts canvas coordinates to viewport coordinates.

#### Parameters

<i>u</i>	The x coordinate of the canvas.
<i>v</i>	The y coordinate of the canvas.

**Returns**

A Vector3D representing the converted viewport coordinates.

**6.4.2.2 getBackgroundColor()**

```
Color RayTracer::Camera::getBackgroundColor ( ) const
```

Returns the background color of the scene.

**Returns**

The background color of the scene.

**6.4.2.3 setBackgroundColor()**

```
void RayTracer::Camera::setBackgroundColor (
    Color color )
```

Sets the background color of the scene.

**Parameters**

<i>color</i>	The new background color to set.
--------------	----------------------------------

**6.4.3 Field Documentation****6.4.3.1 backgroundColor\_**

```
Color RayTracer::Camera::backgroundColor_ [private]
```

**6.4.3.2 imageSize\_**

```
imageSize RayTracer::Camera::imageSize_ { 400, 400 }
```

### 6.4.3.3 origin

```
Math::Point3D RayTracer::Camera::origin { 0, 0, 0 }
```

### 6.4.3.4 reccursionDepth\_

```
int RayTracer::Camera::reccursionDepth_ { 1 }
```

### 6.4.3.5 rotation\_

```
Math::Vector3D RayTracer::Camera::rotation_
```

### 6.4.3.6 screen

```
Math::Rectangle3D RayTracer::Camera::screen
```

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

- include/[Camera.hpp](#)
- src/utis/[Camera.cpp](#)

## 6.5 RayTracer::Color Class Reference

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

Collaboration diagram for RayTracer::Color:

RayTracer::Color
+ r + g + b
+ Color() + ~Color() + Color() + getRIntensity() + getGIntensity() + getBIntensity() + operator+() + getR() + getG() + getB()

## Public Member Functions

- `Color()`=default
- `~Color()`=default
- `Color(double r, double g, double b)`  
*Constructs a `Color` object with the given RGB values.*
- `double getRIntensity() const`  
*Returns the intensity of the red component of the color.*
- `double getGIntensity() const`  
*Returns the intensity of the green component of the color.*
- `double getBIntensity() const`  
*Returns the intensity of the blue component of the color.*
- `Color operator+ (const Color &)`  
*Adds the given color to this color.*
- `double getR() const`
- `double getG() const`
- `double getB() const`

## Data Fields

- `double r`
- `double g`
- `double b`

## 6.5.1 Constructor & Destructor Documentation

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

```
RayTracer::Color::Color ( ) [default]
```

### 6.5.1.2 `~Color()`

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

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

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

Constructs a `Color` object with the given RGB values.



## Parameters

<i>r</i>	The red component of the color.
<i>g</i>	The green component of the color.
<i>b</i>	The blue component of the color.

## 6.5.2 Member Function Documentation

### 6.5.2.1 getB()

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

### 6.5.2.2 getBIntensity()

```
double RayTracer::Color::getBIntensity ( ) const
```

Returns the intensity of the blue component of the color.

## Returns

A double representing the intensity of the blue component.

### 6.5.2.3 getG()

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

### 6.5.2.4 getGIntensity()

```
double RayTracer::Color::getGIntensity ( ) const
```

Returns the intensity of the green component of the color.

## Returns

A double representing the intensity of the green component.

#### 6.5.2.5 getR()

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

#### 6.5.2.6 getRIntensity()

```
double RayTracer::Color::getRIntensity ( ) const
```

Returns the intensity of the red component of the color.

##### Returns

A double representing the intensity of the red component.

#### 6.5.2.7 operator+()

```
Color RayTracer::Color::operator+ (
    const Color & color )
```

Adds the given color to this color.

##### Parameters

<i>color</i>	The color to be added.
--------------	------------------------

##### Returns

A [Color](#) object representing the sum of the two colors.

### 6.5.3 Field Documentation

#### 6.5.3.1 b

```
double RayTracer::Color::b
```

#### 6.5.3.2 g

```
double RayTracer::Color::g
```

### 6.5.3.3 r

```
double RayTracer::Color::r
```

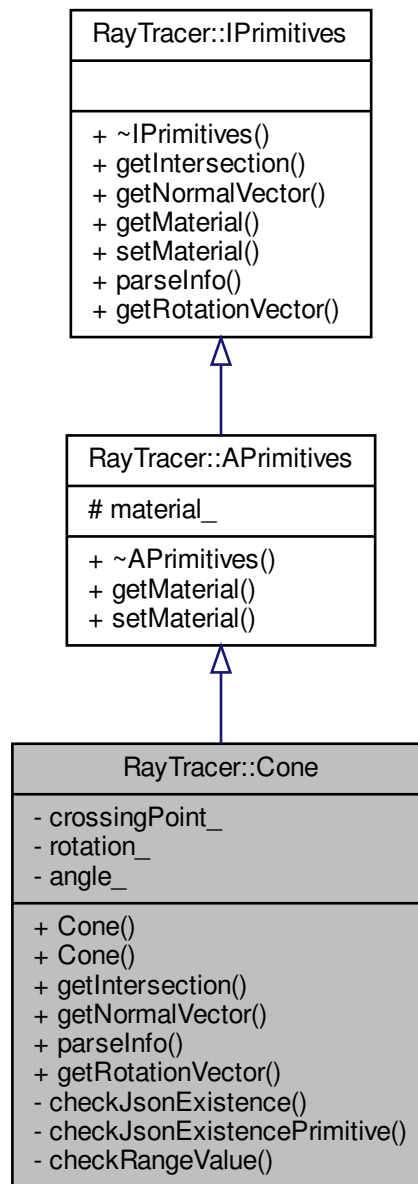
The documentation for this class was generated from the following files:

- [include/material/Color.hpp](#)
- [src/utls/Color.cpp](#)

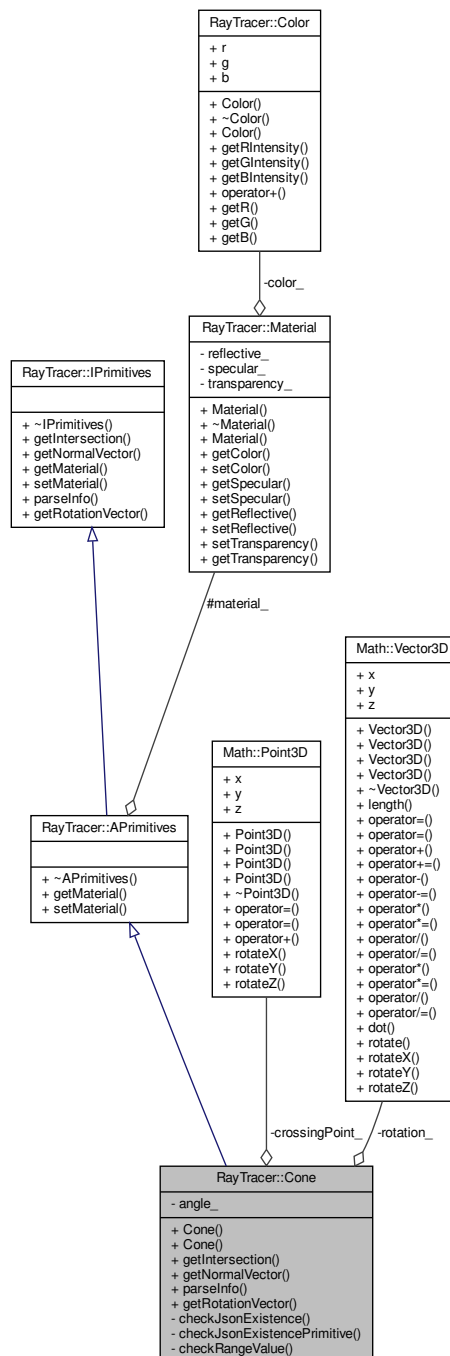
## 6.6 RayTracer::Cone Class Reference

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

Inheritance diagram for RayTracer::Cone:



Collaboration diagram for RayTracer::Cone:



## Public Member Functions

- `Cone (Math::Point3D crossingPoint, double angle)`
- `Cone ()=default`
- `std::vector< double > getIntersection (RayTracer::Ray) final`
- `Math::Vector3D getNormalVector (Math::Point3D point) final`
- `void parseInfo (json object) final`
- `Math::Vector3D getRotationVector ()`

## Private Member Functions

- void [checkJsonExistence](#) (const [json](#) &scene, const std::string &field\_name)
- void [checkJsonExistencePrimitive](#) (const [json](#) &scene)
- void [checkRangeValue](#) (const [json](#) &scene, const std::string &field\_name, const std::string &comparison\_↔ sign, double value)

## Private Attributes

- [Math::Point3D](#) [crossingPoint\\_](#)
- [Math::Vector3D](#) [rotation\\_](#)
- double [angle\\_](#)

## Additional Inherited Members

### 6.6.1 Constructor & Destructor Documentation

#### 6.6.1.1 Cone() [1/2]

```
RayTracer::Cone::Cone (
    Math::Point3D crossingPoint,
    double angle )
```

#### 6.6.1.2 Cone() [2/2]

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

### 6.6.2 Member Function Documentation

#### 6.6.2.1 checkJsonExistence()

```
void RayTracer::Cone::checkJsonExistence (
    const json & scene,
    const std::string & field_name ) [private]
```

### 6.6.2.2 checkJsonExistencePrimitive()

```
void RayTracer::Cone::checkJsonExistencePrimitive (
    const json & scene ) [private]
```

### 6.6.2.3 checkRangeValue()

```
void RayTracer::Cone::checkRangeValue (
    const json & scene,
    const std::string & field_name,
    const std::string & comparison_sign,
    double value ) [private]
```

### 6.6.2.4 getIntersection()

```
std::vector< double > RayTracer::Cone::getIntersection (
    RayTracer::Ray ray ) [final], [virtual]
```

Implements [RayTracer::IPrimitives](#).

### 6.6.2.5 getNormalVector()

```
Math::Vector3D RayTracer::Cone::getNormalVector (
    Math::Point3D point ) [final], [virtual]
```

Implements [RayTracer::IPrimitives](#).

### 6.6.2.6 getRotationVector()

```
Math::Vector3D RayTracer::Cone::getRotationVector ( ) [virtual]
```

Implements [RayTracer::IPrimitives](#).

### 6.6.2.7 parseInfo()

```
void RayTracer::Cone::parseInfo (
    json object ) [final], [virtual]
```

Implements [RayTracer::IPrimitives](#).

### 6.6.3 Field Documentation

#### 6.6.3.1 angle\_

```
double RayTracer::Cone::angle_ [private]
```

#### 6.6.3.2 crossingPoint\_

```
Math::Point3D RayTracer::Cone::crossingPoint_ [private]
```

#### 6.6.3.3 rotation\_

```
Math::Vector3D RayTracer::Cone::rotation_ [private]
```

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

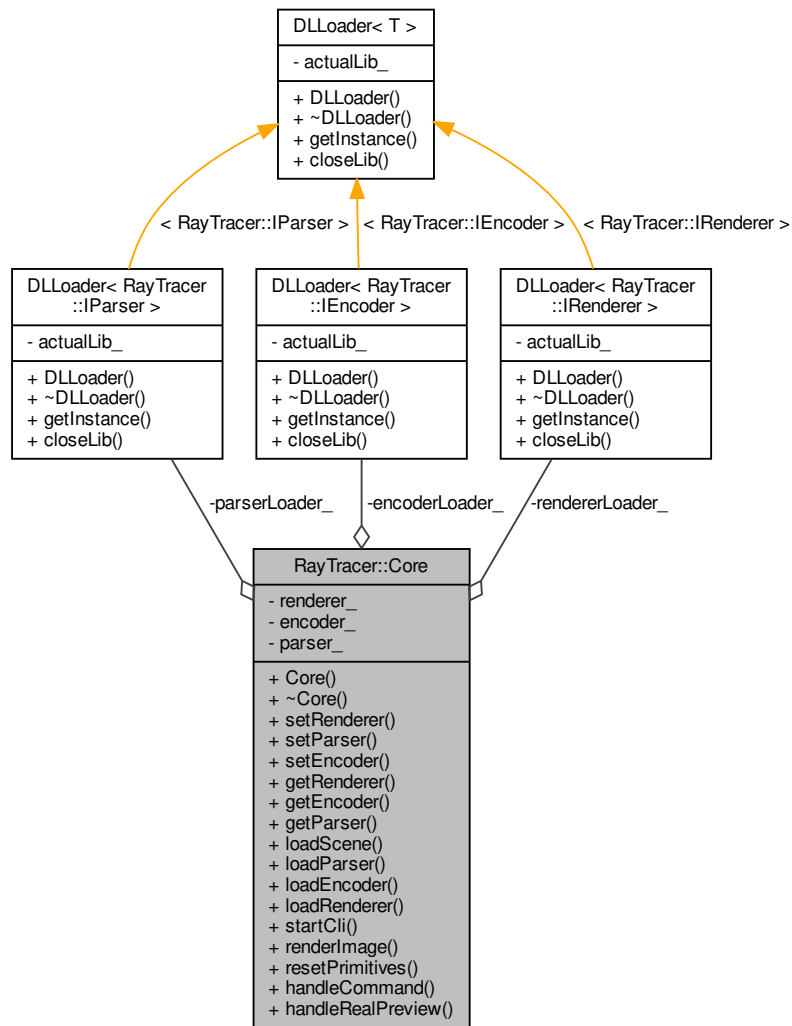
- [src/plugins/primitives/cone/Cone.hpp](#)
- [src/plugins/primitives/cone/Cone.cpp](#)

## 6.7 RayTracer::Core Class Reference

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



Collaboration diagram for RayTracer::Core:



## Public Member Functions

- `Core()`=default
- `~Core()`=default
- `void setRenderer (std::unique_ptr< IRenderer > renderer)`
- `void setParser (std::unique_ptr< IParser > parser)`
- `void setEncoder (std::unique_ptr< IEncoder > encoder)`
- `std::unique_ptr< IRenderer > & getRenderer ()`
- `std::unique_ptr< IEncoder > & getEncoder ()`
- `std::unique_ptr< IParser > & getParser ()`
- `void loadScene ()`
- `void loadParser (const std::string &name)`
- `void loadEncoder (const std::string &name)`
- `void loadRenderer (const std::string &name)`
- `void startCli (const char *config)`

*Starts the command-line interface with the specified configuration.*

- void [renderImage](#) ()

*Renders the final image of the scene.*

- void [resetPrimitives](#) ()

*Resets the primitives in the scene.*

## Static Public Member Functions

- static void [handleCommand](#) ()

*Handles a command received through the command-line interface.*

- static bool [handleRealPreview](#) ()

*Handles real-time preview of the scene.*

## Private Attributes

- [DLLoader](#)< [IEncoder](#) > [encoderLoader\\_](#)
- [DLLoader](#)< [IRenderer](#) > [rendererLoader\\_](#)
- [DLLoader](#)< [IParser](#) > [parserLoader\\_](#)
- std::unique\_ptr< [IRenderer](#) > [renderer\\_](#) { nullptr }
- std::unique\_ptr< [IEncoder](#) > [encoder\\_](#) { nullptr }
- std::unique\_ptr< [IParser](#) > [parser\\_](#) { nullptr }

## 6.7.1 Constructor & Destructor Documentation

### 6.7.1.1 Core()

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

### 6.7.1.2 ~Core()

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

## 6.7.2 Member Function Documentation

### 6.7.2.1 getEncoder()

```
std::unique_ptr< IEncoder > & RayTracer::Core::getEncoder ( )
```

Get the encoder of the scene.

#### Returns

A reference to a unique pointer to the [IEncoder](#) object.

### 6.7.2.2 getParser()

```
std::unique_ptr< IParser > & RayTracer::Core::getParser ( )
```

Get the parser of the scene.

#### Returns

A reference to a unique pointer to the [IParser](#) object.

### 6.7.2.3 getRenderer()

```
std::unique_ptr< IRenderer > & RayTracer::Core::getRenderer ( )
```

Get the renderer of the scene.

#### Returns

A reference to a unique pointer to the [IRenderer](#) object.

### 6.7.2.4 handleCommand()

```
void RayTracer::Core::handleCommand ( ) [static]
```

Handles a command received through the command-line interface.

### 6.7.2.5 handleRealPreview()

```
bool RayTracer::Core::handleRealPreview ( ) [static]
```

Handles real-time preview of the scene.

#### Returns

True if the real-time preview is successfully handled, False otherwise.

### 6.7.2.6 loadEncoder()

```
void RayTracer::Core::loadEncoder (
    const std::string & name )
```

Load an encoder by its name.

**Parameters**

<i>name</i>	A constant reference to the string name of the encoder.
-------------	---

**6.7.2.7 loadParser()**

```
void RayTracer::Core::loadParser (
    const std::string & name )
```

Load a parser by its name.

**Parameters**

<i>name</i>	A constant reference to the string name of the parser.
-------------	--

**6.7.2.8 loadRenderer()**

```
void RayTracer::Core::loadRenderer (
    const std::string & name )
```

Load a renderer by its name.

**Parameters**

<i>name</i>	A constant reference to the string name of the renderer.
-------------	--

**6.7.2.9 loadScene()**

```
void RayTracer::Core::loadScene ( )
```

Load the scene.

**Exceptions**

<i>throw</i>	an exception if the scene cannot be loaded
--------------	--

**6.7.2.10 renderImage()**

```
void RayTracer::Core::renderImage ( )
```

Renders the final image of the scene.

#### 6.7.2.11 resetPrimitives()

```
void RayTracer::Core::resetPrimitives ( )
```

Resets the primitives in the scene.

#### 6.7.2.12 setEncoder()

```
void RayTracer::Core::setEncoder (
    std::unique_ptr< IEncoder > encoder )
```

Set the encoder for the scene.

##### Parameters

<i>encoder</i>	A unique pointer to the <a href="#">IEncoder</a> object.
----------------	--

#### 6.7.2.13 setParser()

```
void RayTracer::Core::setParser (
    std::unique_ptr< IParser > parser )
```

Set the parser for the scene.

##### Parameters

<i>parser</i>	A unique pointer to the <a href="#">IParser</a> object.
---------------	---

#### 6.7.2.14 setRenderer()

```
void RayTracer::Core::setRenderer (
    std::unique_ptr< IRenderer > renderer )
```

Set the renderer for the scene.

##### Parameters

<i>renderer</i>	A unique pointer to the <a href="#">IRenderer</a> object.
-----------------	---

#### 6.7.2.15 startCli()

```
void RayTracer::Core::startCli (
    const char * config )
```

Starts the command-line interface with the specified configuration.

##### Parameters

<i>config</i>	The configuration file to use.
---------------	--------------------------------

### 6.7.3 Field Documentation

#### 6.7.3.1 encoder\_

```
std::unique_ptr<IEncoder> RayTracer::Core::encoder_ { nullptr } [private]
```

#### 6.7.3.2 encoderLoader\_

```
DLLoader<IEncoder> RayTracer::Core::encoderLoader_ [private]
```

#### 6.7.3.3 parser\_

```
std::unique_ptr<IParser> RayTracer::Core::parser_ { nullptr } [private]
```

#### 6.7.3.4 parserLoader\_

```
DLLoader<IParser> RayTracer::Core::parserLoader_ [private]
```

#### 6.7.3.5 renderer\_

```
std::unique_ptr<IRenderer> RayTracer::Core::renderer_ { nullptr } [private]
```

### 6.7.3.6 rendererLoader\_

```
DLLoader<IRenderer> RayTracer::Core::rendererLoader_ [private]
```

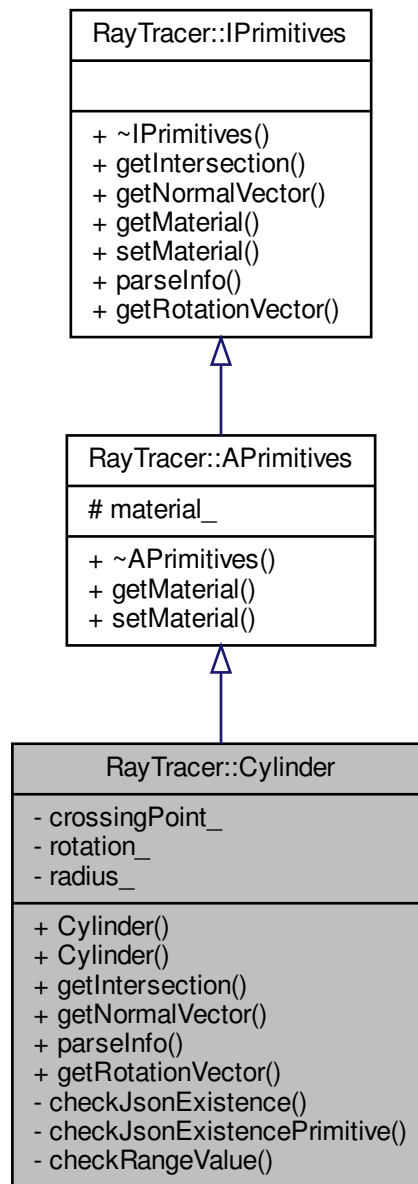
The documentation for this class was generated from the following files:

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

## 6.8 RayTracer::Cylinder Class Reference

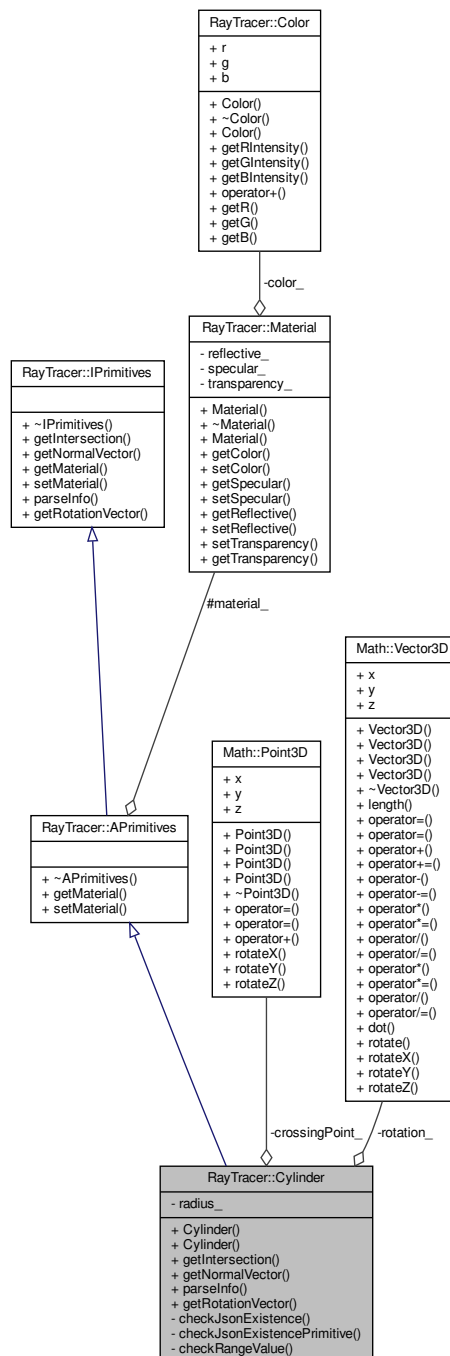
```
#include <Cylinder.hpp>
```

Inheritance diagram for RayTracer::Cylinder:





Collaboration diagram for RayTracer::Cylinder:



## Public Member Functions

- `Cylinder (Math::Point3D crossingPoint, double radius)`
- `Cylinder ()=default`
- `std::vector< double > getIntersection (RayTracer::Ray ray) final`
- `Math::Vector3D getNormalVector (Math::Point3D point) final`
- `void parseInfo (json object) final`
- `Math::Vector3D getRotationVector ()`

## Private Member Functions

- void [checkJsonExistence](#) (const [json](#) &scene, const std::string &field\_name)
- void [checkJsonExistencePrimitive](#) (const [json](#) &scene)
- void [checkRangeValue](#) (const [json](#) &scene, const std::string &field\_name, const std::string &comparison\_↔ sign, double value)

## Private Attributes

- [Math::Point3D](#) [crossingPoint\\_](#)
- [Math::Vector3D](#) [rotation\\_](#)
- double [radius\\_](#)

## Additional Inherited Members

### 6.8.1 Constructor & Destructor Documentation

#### 6.8.1.1 Cylinder() [1/2]

```
RayTracer::Cylinder::Cylinder (
    Math::Point3D crossingPoint,
    double radius )
```

#### 6.8.1.2 Cylinder() [2/2]

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

### 6.8.2 Member Function Documentation

#### 6.8.2.1 checkJsonExistence()

```
void RayTracer::Cylinder::checkJsonExistence (
    const json & scene,
    const std::string & field_name ) [private]
```

### 6.8.2.2 checkJsonExistencePrimitive()

```
void RayTracer::Cylinder::checkJsonExistencePrimitive (
    const json & scene ) [private]
```

### 6.8.2.3 checkRangeValue()

```
void RayTracer::Cylinder::checkRangeValue (
    const json & scene,
    const std::string & field_name,
    const std::string & comparison_sign,
    double value ) [private]
```

### 6.8.2.4 getIntersection()

```
std::vector< double > RayTracer::Cylinder::getIntersection (
    RayTracer::Ray ray ) [final], [virtual]
```

Implements [RayTracer::IPrimitives](#).

### 6.8.2.5 getNormalVector()

```
Math::Vector3D RayTracer::Cylinder::getNormalVector (
    Math::Point3D point ) [final], [virtual]
```

Implements [RayTracer::IPrimitives](#).

### 6.8.2.6 getRotationVector()

```
Math::Vector3D RayTracer::Cylinder::getRotationVector ( ) [virtual]
```

Implements [RayTracer::IPrimitives](#).

### 6.8.2.7 parseInfo()

```
void RayTracer::Cylinder::parseInfo (
    json object ) [final], [virtual]
```

Implements [RayTracer::IPrimitives](#).

### 6.8.3 Field Documentation

#### 6.8.3.1 crossingPoint\_

```
Math::Point3D RayTracer::Cylinder::crossingPoint_ [private]
```

#### 6.8.3.2 radius\_

```
double RayTracer::Cylinder::radius_ [private]
```

#### 6.8.3.3 rotation\_

```
Math::Vector3D RayTracer::Cylinder::rotation_ [private]
```

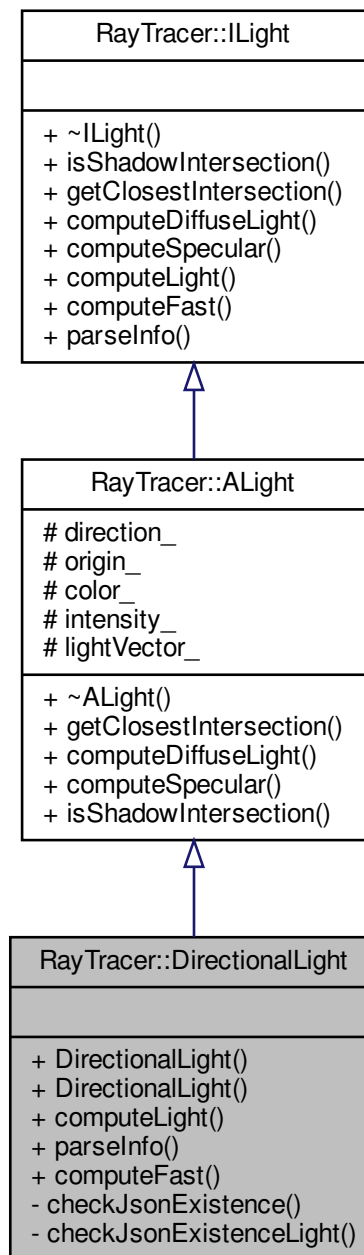
The documentation for this class was generated from the following files:

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

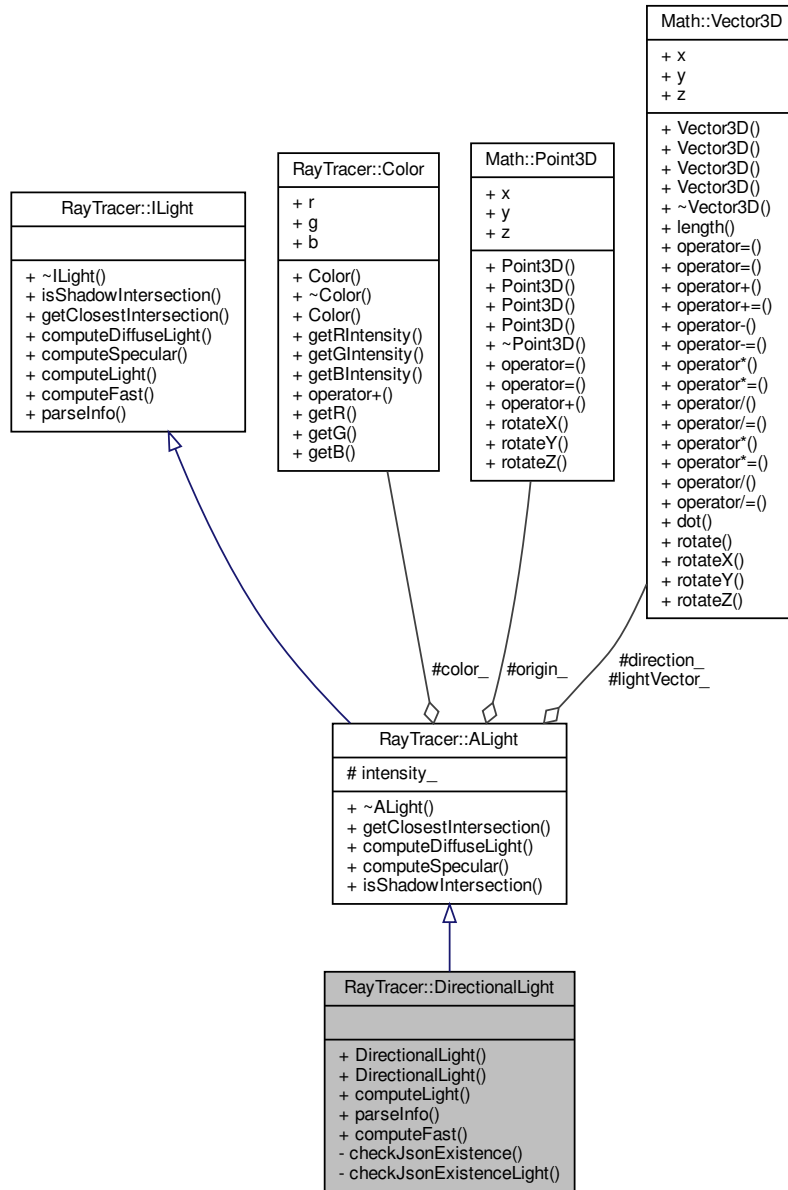
## 6.9 RayTracer::DirectionalLight Class Reference

```
#include <DirectionalLight.hpp>
```

Inheritance diagram for RayTracer::DirectionalLight:



Collaboration diagram for RayTracer::DirectionalLight:



## Public Member Functions

- [DirectionalLight](#) ()=default
- [DirectionalLight](#) ([Color](#) color, double intensity, [Math::Vector3D](#) direction)
- [Color](#) [computeLight](#) ([Math::Vector3D](#) normalVector, int spec, [Math::Point3D](#) intersectionPoint, [Math::Vector3D](#) rayDir, std::vector< std::unique\_ptr< [IPrimitives](#) >> &primitives) final
- void [parseInfo](#) (json object) final  
*Information parser to create the light object.*
- [Color](#) [computeFast](#) ([Math::Vector3D](#) normalVector, int spec, [Math::Point3D](#) intersectionPoint, [Math::Vector3D](#) rayDir, std::vector< std::unique\_ptr< [IPrimitives](#) >> &primitives) final

## Private Member Functions

- void [checkJsonExistence](#) (const [json](#) &scene, const std::string &field\_name)
- void [checkJsonExistenceLight](#) (const [json](#) &scene)

## Additional Inherited Members

### 6.9.1 Constructor & Destructor Documentation

#### 6.9.1.1 DirectionalLight() [1/2]

```
RayTracer::DirectionalLight::DirectionalLight ( ) [default]
```

#### 6.9.1.2 DirectionalLight() [2/2]

```
RayTracer::DirectionalLight::DirectionalLight (
    Color color,
    double intensity,
    Math::Vector3D direction )
```

### 6.9.2 Member Function Documentation

#### 6.9.2.1 checkJsonExistence()

```
void RayTracer::DirectionalLight::checkJsonExistence (
    const json & scene,
    const std::string & field_name ) [private]
```

#### 6.9.2.2 checkJsonExistenceLight()

```
void RayTracer::DirectionalLight::checkJsonExistenceLight (
    const json & scene ) [private]
```

### 6.9.2.3 computeFast()

```
Color RayTracer::DirectionalLight::computeFast (
    Math::Vector3D normalVector,
    int spec,
    Math::Point3D intersectionPoint,
    Math::Vector3D rayDir,
    std::vector< std::unique_ptr< IPrimitives >> & primitives ) [final], [virtual]
```

Implements [RayTracer::ILight](#).

### 6.9.2.4 computeLight()

```
Color RayTracer::DirectionalLight::computeLight (
    Math::Vector3D normalVector,
    int spec,
    Math::Point3D intersectionPoint,
    Math::Vector3D rayDir,
    std::vector< std::unique_ptr< IPrimitives >> & primitives ) [final], [virtual]
```

Implements [RayTracer::ILight](#).

### 6.9.2.5 parseInfo()

```
void RayTracer::DirectionalLight::parseInfo (
    json object ) [final], [virtual]
```

Information parser to create the light object.

#### Parameters

<i>object</i>	the json object containing light info
---------------	---------------------------------------

Implements [RayTracer::ILight](#).

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

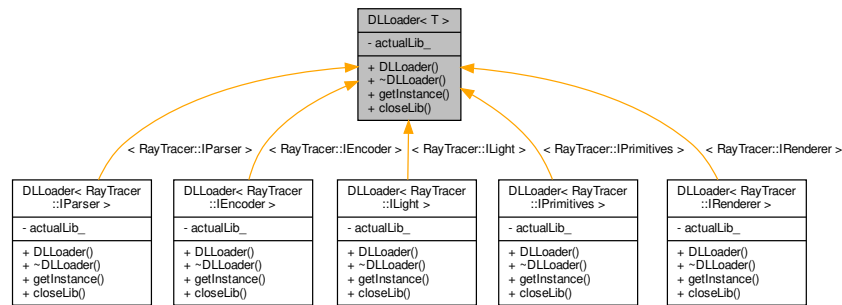
- [src/plugins/lights/directional/DirectionalLight.hpp](#)
- [src/plugins/lights/directional/DirectionalLight.cpp](#)

## 6.10 DLLoader< T > Class Template Reference

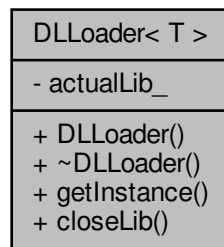
```
#include <DLLoader.hpp>
```



Inheritance diagram for DLLoader< T >:



Collaboration diagram for DLLoader< T >:



## Public Member Functions

- [DLLoader](#) ()=default
- [~DLLoader](#) ()=default
- `std::unique_ptr< T >` [getInstance](#) (const `std::string` &filename)
- void [closeLib](#) ()

## Private Attributes

- void \* [actualLib\\_](#) { nullptr }

### 6.10.1 Constructor & Destructor Documentation

#### 6.10.1.1 DLoader()

```
template<typename T >
DLoader< T >::DLoader ( ) [default]
```

#### 6.10.1.2 ~DLoader()

```
template<typename T >
DLoader< T >::~~DLoader ( ) [default]
```

### 6.10.2 Member Function Documentation

#### 6.10.2.1 closeLib()

```
template<typename T >
void DLoader< T >::closeLib ( ) [inline]
```

#### 6.10.2.2 getInstance()

```
template<typename T >
std::unique_ptr<T> DLoader< T >::getInstance (
    const std::string & filename ) [inline]
```

### 6.10.3 Field Documentation

#### 6.10.3.1 actualLib\_

```
template<typename T >
void* DLoader< T >::actualLib_ { nullptr } [private]
```

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

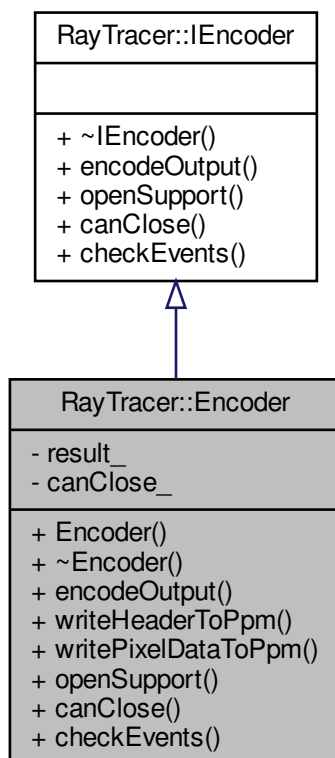
- include/DLoader.hpp

## 6.11 RayTracer::Encoder Class Reference

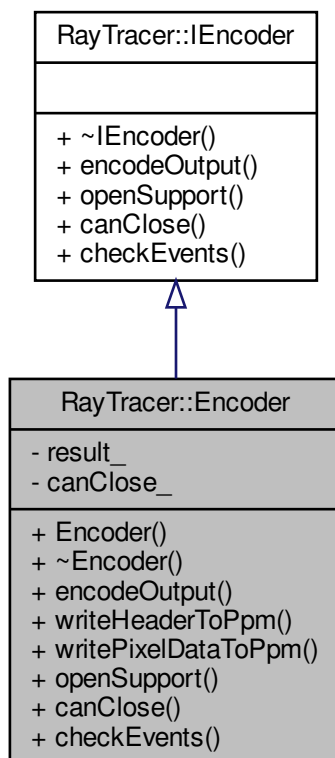
[Encoder](#) class.

```
#include <Encoder.hpp>
```

Inheritance diagram for RayTracer::Encoder:



Collaboration diagram for RayTracer::Encoder:



## Public Member Functions

- [Encoder](#) ()=default
- [~Encoder](#) ()=default
- void [encodeOutput](#) (std::vector< std::vector< [Color](#) >> [result\\_](#), const std::string &filename) override
- void [writeHeaderToPpm](#) (std::ofstream &ofs, long unsigned int width, long unsigned int height)
- void [writePixelDataToPpm](#) (std::ofstream &ofs, const std::vector< std::vector< [Color](#) >> &[result\\_](#))
- void [openSupport](#) (const std::string &name, [imageSize](#) size) override
  - Opens the necessary support for encoding.*
- bool [canClose](#) () override
  - Checks if the encoder can be closed.*
- void [checkEvents](#) () override
  - Checks for any pending events or actions.*

## Private Attributes

- std::vector< std::vector< [Color](#) > > [result\\_](#)
- bool [canClose\\_](#) { true }

### 6.11.1 Detailed Description

[Encoder](#) class.

### 6.11.2 Constructor & Destructor Documentation

#### 6.11.2.1 Encoder()

```
RayTracer::Encoder::Encoder ( ) [default]
```

#### 6.11.2.2 ~Encoder()

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

### 6.11.3 Member Function Documentation

#### 6.11.3.1 canClose()

```
bool RayTracer::Encoder::canClose ( ) [override], [virtual]
```

Checks if the encoder can be closed.

##### Returns

True if the encoder can be closed, False otherwise

Implements [RayTracer::IEncoder](#).

#### 6.11.3.2 checkEvents()

```
void RayTracer::Encoder::checkEvents ( ) [override], [virtual]
```

Checks for any pending events or actions.

Implements [RayTracer::IEncoder](#).

#### 6.11.3.3 encodeOutput()

```
void RayTracer::Encoder::encodeOutput (
    std::vector< std::vector< Color >> result_,
    const std::string & filename ) [override], [virtual]
```

## Parameters

<i>result</i> ↔	
—	
<i>filename</i>	

Implements [RayTracer::IEncoder](#).

#### 6.11.3.4 openSupport()

```
void RayTracer::Encoder::openSupport (
    const std::string & name,
    imageSize size ) [override], [virtual]
```

Opens the necessary support for encoding.

## Parameters

<i>name</i>	The name of the support to be opened
<i>size</i>	The size of the image to be encoded

Implements [RayTracer::IEncoder](#).

#### 6.11.3.5 writeHeaderToPpm()

```
void RayTracer::Encoder::writeHeaderToPpm (
    std::ofstream & ofs,
    long unsigned int width,
    long unsigned int height )
```

## Parameters

<i>ofs</i>	
<i>width</i>	
<i>height</i>	

#### 6.11.3.6 writePixelDataToPpm()

```
void RayTracer::Encoder::writePixelDataToPpm (
    std::ofstream & ofs,
    const std::vector< std::vector< Color >> & result_ )
```

## Parameters

<i>ofs</i>	
<i>result</i> $\leftrightarrow$	
—	

### 6.11.4 Field Documentation

#### 6.11.4.1 canClose\_

```
bool RayTracer::Encoder::canClose_ { true } [private]
```

#### 6.11.4.2 result\_

```
std::vector<std::vector<Color> > RayTracer::Encoder::result_ [private]
```

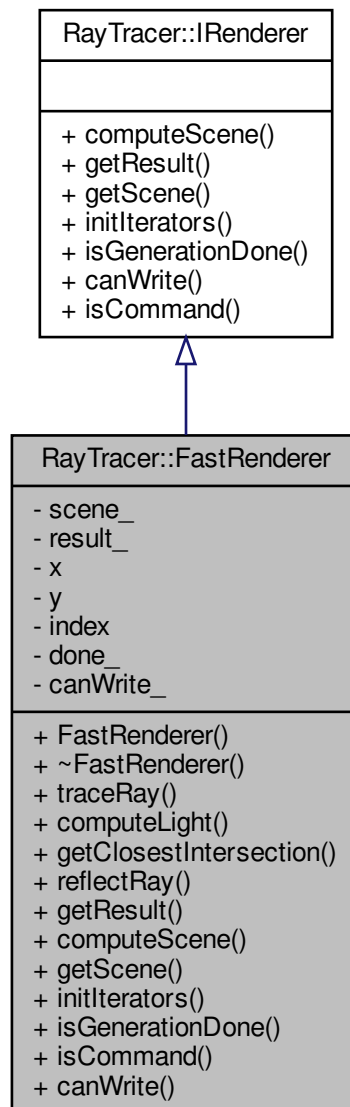
The documentation for this class was generated from the following files:

- src/plugins/encoder/ppmEncoder/[Encoder.hpp](#)
- src/plugins/encoder/ppmEncoder/[Encoder.cpp](#)

## 6.12 RayTracer::FastRenderer Class Reference

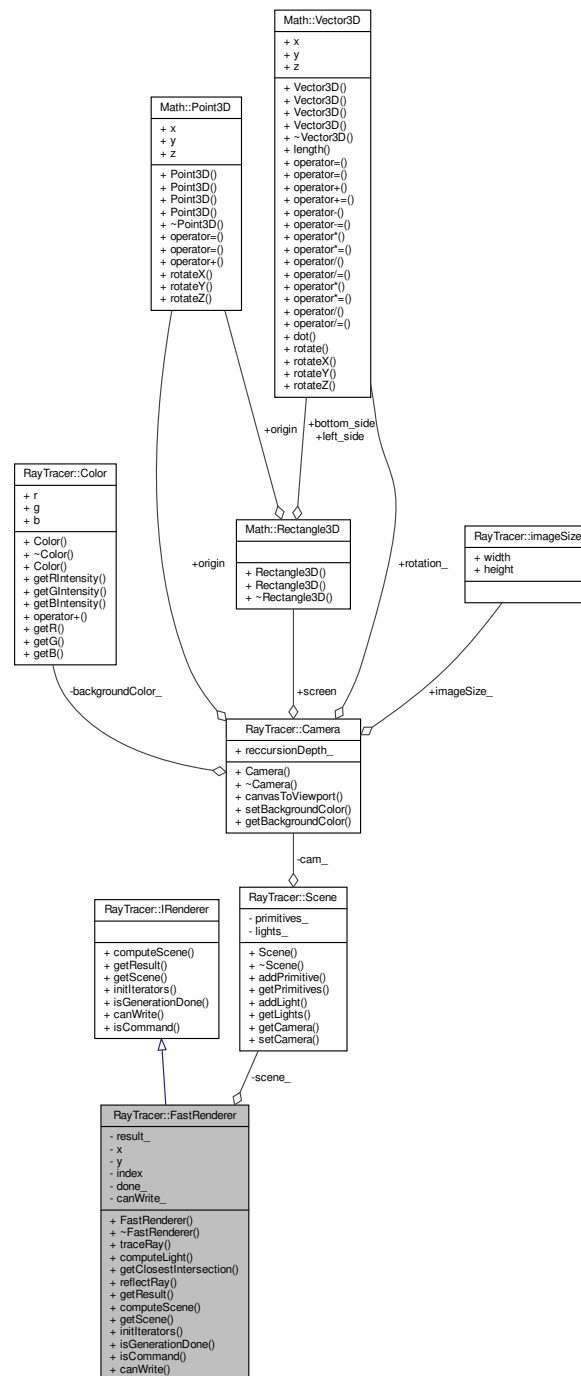
```
#include <Renderer.hpp>
```

Inheritance diagram for RayTracer::FastRenderer:





Collaboration diagram for RayTracer::FastRenderer:



## Public Member Functions

- [FastRenderer](#) ()=default
- [~FastRenderer](#) ()=default
- [Color traceRay](#) ([RayTracer::Ray](#) ray, double tMin, double tMax, int recursion\_depth)
- [Color computeLight](#) ([Math::Point3D](#) intersectionPoint, [Math::Vector3D](#) normalVector, [Math::Vector3D](#) rayDir, int spec)

- `intersection getClosestIntersection` (`RayTracer::Ray` ray, double tMin, double tMax)
- `Math::Vector3D reflectRay` (`Math::Vector3D` normalVector, `Math::Vector3D` ray)
- `std::vector< std::vector< Color > > getResult` () override  
*Get the result of the rendering as a 2D vector of colors.*
- void `computeScene` () override  
*Compute the scene and render the result.*
- `Scene & getScene` () override  
*Get the scene used for rendering.*
- void `initIterators` () override  
*Initializes the iterators for generation.*
- bool `isGenerationDone` () override  
*Checks if the generation process is done.*
- bool `isCommand` () override  
*Checks if the current action is a command.*
- bool `canWrite` () override  
*Checks if writing is allowed.*

## Private Attributes

- `Scene scene_` {}
- `std::vector< std::vector< Color > > result_`
- double `x`
- double `y`
- `std::size_t index` = 0
- bool `done_` { false }
- bool `canWrite_` { false }

## 6.12.1 Constructor & Destructor Documentation

### 6.12.1.1 FastRenderer()

```
RayTracer::FastRenderer::FastRenderer ( ) [default]
```

### 6.12.1.2 ~FastRenderer()

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

## 6.12.2 Member Function Documentation

### 6.12.2.1 canWrite()

```
bool RayTracer::FastRenderer::canWrite ( ) [override], [virtual]
```

Checks if writing is allowed.

#### Returns

True if writing is allowed, False otherwise

Implements [RayTracer::IRenderer](#).

### 6.12.2.2 computeLight()

```
Color RayTracer::FastRenderer::computeLight (
    Math::Point3D intersectionPoint,
    Math::Vector3D normalVector,
    Math::Vector3D rayDir,
    int spec )
```

### 6.12.2.3 computeScene()

```
void RayTracer::FastRenderer::computeScene ( ) [override], [virtual]
```

Compute the scene and render the result.

Implements [RayTracer::IRenderer](#).

### 6.12.2.4 getClosestIntersection()

```
intersection RayTracer::FastRenderer::getClosestIntersection (
    RayTracer::Ray ray,
    double tMin,
    double tMax )
```

### 6.12.2.5 getResult()

```
std::vector< std::vector< Color > > RayTracer::FastRenderer::getResult ( ) [override], [virtual]
```

Get the result of the rendering as a 2D vector of colors.

#### Returns

The result of the rendering as a 2D vector of colors.

Implements [RayTracer::IRenderer](#).

#### 6.12.2.6 `getScene()`

```
Scene & RayTracer::FastRenderer::getScene ( ) [override], [virtual]
```

Get the scene used for rendering.

##### Returns

The scene used for rendering.

Implements [RayTracer::IRenderer](#).

#### 6.12.2.7 `initIterators()`

```
void RayTracer::FastRenderer::initIterators ( ) [override], [virtual]
```

Initializes the iterators for generation.

Implements [RayTracer::IRenderer](#).

#### 6.12.2.8 `isCommand()`

```
bool RayTracer::FastRenderer::isCommand ( ) [override], [virtual]
```

Checks if the current action is a command.

##### Returns

True if the current action is a command, False otherwise

Implements [RayTracer::IRenderer](#).

#### 6.12.2.9 `isGenerationDone()`

```
bool RayTracer::FastRenderer::isGenerationDone ( ) [override], [virtual]
```

Checks if the generation process is done.

##### Returns

True if the generation is done, False otherwise

Implements [RayTracer::IRenderer](#).

#### 6.12.2.10 reflectRay()

```
Math::Vector3D RayTracer::FastRenderer::reflectRay (
    Math::Vector3D normalVector,
    Math::Vector3D ray )
```

#### 6.12.2.11 traceRay()

```
Color RayTracer::FastRenderer::traceRay (
    RayTracer::Ray ray,
    double tMin,
    double tMax,
    int recursion_depth )
```

### 6.12.3 Field Documentation

#### 6.12.3.1 canWrite\_

```
bool RayTracer::FastRenderer::canWrite_ { false } [private]
```

#### 6.12.3.2 done\_

```
bool RayTracer::FastRenderer::done_ { false } [private]
```

#### 6.12.3.3 index

```
std::size_t RayTracer::FastRenderer::index = 0 [private]
```

#### 6.12.3.4 result\_

```
std::vector<std::vector<Color> > RayTracer::FastRenderer::result_ [private]
```

#### 6.12.3.5 scene\_

```
Scene RayTracer::FastRenderer::scene_ {} [private]
```

#### 6.12.3.6 x

```
double RayTracer::FastRenderer::x [private]
```

#### 6.12.3.7 y

```
double RayTracer::FastRenderer::y [private]
```

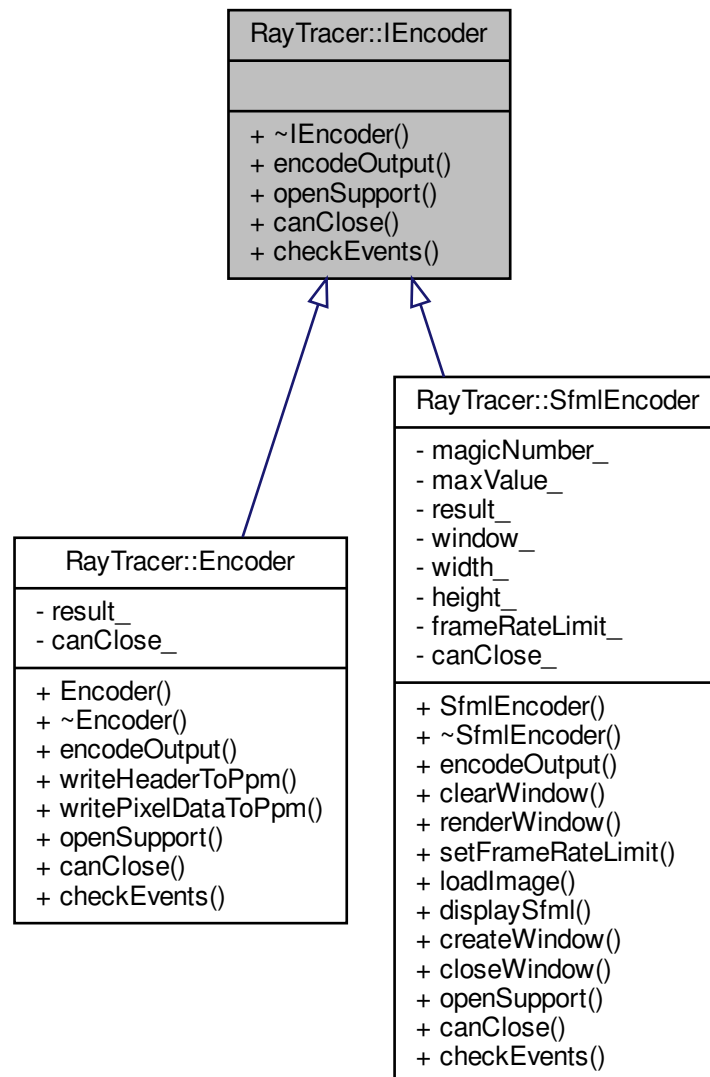
The documentation for this class was generated from the following files:

- src/plugins/renderer/fastRenderer/[Renderer.hpp](#)
- src/plugins/renderer/fastRenderer/[Renderer.cpp](#)

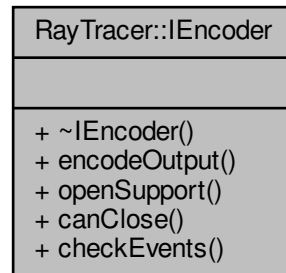
## 6.13 RayTracer::IEncoder Class Reference

```
#include <IEncoder.hpp>
```

Inheritance diagram for RayTracer::IEncoder:



Collaboration diagram for RayTracer::IEncoder:



## Public Member Functions

- virtual [~IEncoder](#) ()=default
- virtual void [encodeOutput](#) (std::vector< std::vector< [Color](#) >> result, const std::string &filename)=0  
*Encodes and saves the rendering result to an output file.*
- virtual void [openSupport](#) (const std::string &name, [imageSize](#) size)=0  
*Opens the necessary support for encoding.*
- virtual bool [canClose](#) ()=0  
*Checks if the encoder can be closed.*
- virtual void [checkEvents](#) ()=0  
*Checks for any pending events or actions.*

## 6.13.1 Constructor & Destructor Documentation

### 6.13.1.1 ~IEncoder()

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

## 6.13.2 Member Function Documentation



### 6.13.2.1 canClose()

```
virtual bool RayTracer::IEncoder::canClose ( ) [pure virtual]
```

Checks if the encoder can be closed.

#### Returns

True if the encoder can be closed, False otherwise

Implemented in [RayTracer::SfmlEncoder](#), and [RayTracer::Encoder](#).

### 6.13.2.2 checkEvents()

```
virtual void RayTracer::IEncoder::checkEvents ( ) [pure virtual]
```

Checks for any pending events or actions.

Implemented in [RayTracer::SfmlEncoder](#), and [RayTracer::Encoder](#).

### 6.13.2.3 encodeOutput()

```
virtual void RayTracer::IEncoder::encodeOutput (
    std::vector< std::vector< Color >> result,
    const std::string & filename ) [pure virtual]
```

Encodes and saves the rendering result to an output file.

#### Parameters

<i>result</i>	The rendering result as a 2D vector of colors
<i>filename</i>	The filename of the output file

Implemented in [RayTracer::SfmlEncoder](#), and [RayTracer::Encoder](#).

### 6.13.2.4 openSupport()

```
virtual void RayTracer::IEncoder::openSupport (
    const std::string & name,
    imageSize size ) [pure virtual]
```

Opens the necessary support for encoding.

## Parameters

<i>name</i>	The name of the support to be opened
<i>size</i>	The size of the image to be encoded

Implemented in [RayTracer::SfmlEncoder](#), and [RayTracer::Encoder](#).

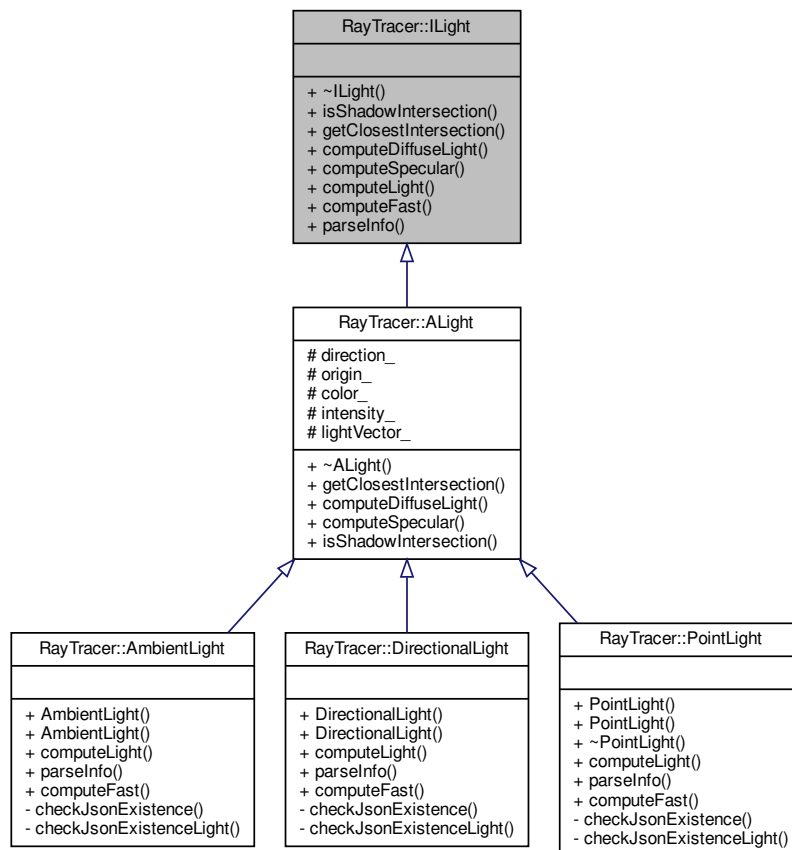
The documentation for this class was generated from the following file:

- [include/encoder/LEncoder.hpp](#)

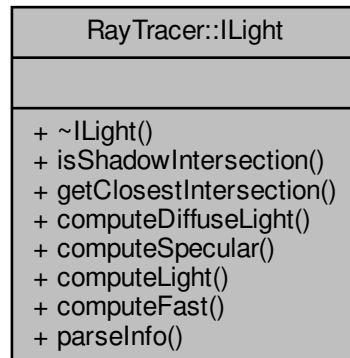
## 6.14 RayTracer::ILight Class Reference

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

Inheritance diagram for RayTracer::ILight:



Collaboration diagram for RayTracer::ILight:



## Public Member Functions

- virtual [~ILight\(\)](#)=default
- virtual bool [isShadowIntersection](#) ([Math::Point3D](#) intersectionPoint, std::vector< std::unique\_ptr< [IPrimitives](#) >> &primitives)=0
- virtual [intersection](#) [getClosestIntersection](#) ([RayTracer::Ray](#) ray, double tMin, double tMax, std::vector< std::unique\_ptr< [IPrimitives](#) >> &primitives)=0
- virtual [Color](#) [computeDiffuseLight](#) ([Math::Vector3D](#) normalVector)=0
- virtual [Color](#) [computeSpecular](#) ([Math::Vector3D](#) normalVector, int spec, [Math::Vector3D](#) rayDir)=0
- virtual [Color](#) [computeLight](#) ([Math::Vector3D](#) normalVector, int spec, [Math::Point3D](#) intersectionPoint, [Math::Vector3D](#) rayDir, std::vector< std::unique\_ptr< [IPrimitives](#) >> &primitives)=0
- virtual [Color](#) [computeFast](#) ([Math::Vector3D](#) normalVector, int spec, [Math::Point3D](#) intersectionPoint, [Math::Vector3D](#) rayDir, std::vector< std::unique\_ptr< [IPrimitives](#) >> &primitives)=0
- virtual void [parseInfo](#) ([json](#) object)=0

*Information parser to create the light object.*

### 6.14.1 Constructor & Destructor Documentation

#### 6.14.1.1 ~ILight()

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

### 6.14.2 Member Function Documentation

#### 6.14.2.1 computeDiffuseLight()

```
virtual Color RayTracer::ILight::computeDiffuseLight (
    Math::Vector3D normalVector ) [pure virtual]
```

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

#### 6.14.2.2 computeFast()

```
virtual Color RayTracer::ILight::computeFast (
    Math::Vector3D normalVector,
    int spec,
    Math::Point3D intersectionPoint,
    Math::Vector3D rayDir,
    std::vector< std::unique_ptr< IPrimitives >> & primitives ) [pure virtual]
```

Implemented in [RayTracer::PointLight](#), [RayTracer::DirectionalLight](#), and [RayTracer::AmbientLight](#).

#### 6.14.2.3 computeLight()

```
virtual Color RayTracer::ILight::computeLight (
    Math::Vector3D normalVector,
    int spec,
    Math::Point3D intersectionPoint,
    Math::Vector3D rayDir,
    std::vector< std::unique_ptr< IPrimitives >> & primitives ) [pure virtual]
```

Implemented in [RayTracer::PointLight](#), [RayTracer::DirectionalLight](#), and [RayTracer::AmbientLight](#).

#### 6.14.2.4 computeSpecular()

```
virtual Color RayTracer::ILight::computeSpecular (
    Math::Vector3D normalVector,
    int spec,
    Math::Vector3D rayDir ) [pure virtual]
```

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

#### 6.14.2.5 getClosestIntersection()

```
virtual intersection RayTracer::ILight::getClosestIntersection (
    RayTracer::Ray ray,
    double tMin,
    double tMax,
    std::vector< std::unique_ptr< IPrimitives >> & primitives ) [pure virtual]
```

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

#### 6.14.2.6 isShadowIntersection()

```
virtual bool RayTracer::ILight::isShadowIntersection (
    Math::Point3D intersectionPoint,
    std::vector< std::unique_ptr< IPrimitives >> & primitives ) [pure virtual]
```

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

#### 6.14.2.7 parseInfo()

```
virtual void RayTracer::ILight::parseInfo (
    json object ) [pure virtual]
```

Information parser to create the light object.

##### Parameters

<i>object</i>	the json object containing light info
---------------	---------------------------------------

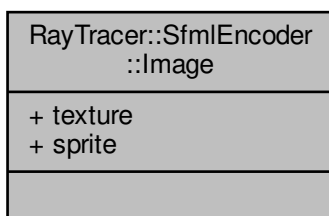
Implemented in [RayTracer::PointLight](#), [RayTracer::DirectionalLight](#), and [RayTracer::AmbientLight](#).

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

- [include/lights/ILight.hpp](#)

## 6.15 RayTracer::SfmlEncoder::Image Struct Reference

Collaboration diagram for RayTracer::SfmlEncoder::Image:



### Data Fields

- sf::Texture [texture](#)
- sf::Sprite [sprite](#)

### 6.15.1 Field Documentation

#### 6.15.1.1 [sprite](#)

```
sf::Sprite RayTracer::SfmlEncoder::Image::sprite
```

#### 6.15.1.2 [texture](#)

```
sf::Texture RayTracer::SfmlEncoder::Image::texture
```

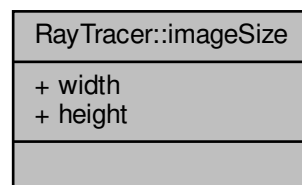
The documentation for this struct was generated from the following file:

- [src/plugins/encoder/sfmlEncoder/Encoder.hpp](#)

## 6.16 RayTracer::imageSize Struct Reference

```
#include <Size.hpp>
```

Collaboration diagram for RayTracer::imageSize:



### Data Fields

- double [width](#)
- double [height](#)

### 6.16.1 Field Documentation

#### 6.16.1.1 height

```
double RayTracer::imageSize::height
```

#### 6.16.1.2 width

```
double RayTracer::imageSize::width
```

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

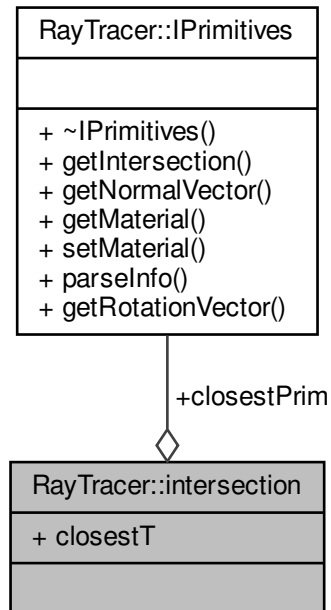
- [include/Size.hpp](#)

## 6.17 RayTracer::intersection Struct Reference

Struct for the intersection.

```
#include <RayTracer.hpp>
```

Collaboration diagram for RayTracer::intersection:



### Data Fields

- [IPrimitives](#) & [closestPrim](#)
- double [closestT](#)

#### 6.17.1 Detailed Description

Struct for the intersection.

#### 6.17.2 Field Documentation



### 6.17.2.1 closestPrim

```
IPrimitives& RayTracer::intersection::closestPrim
```

### 6.17.2.2 closestT

```
double RayTracer::intersection::closestT
```

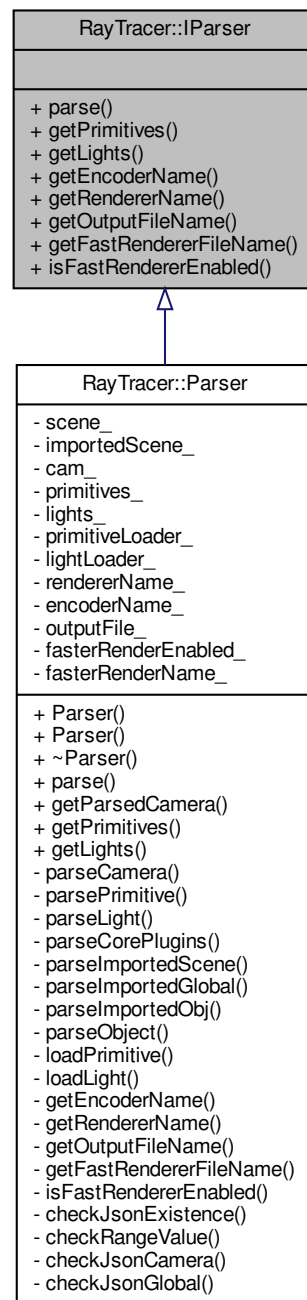
The documentation for this struct was generated from the following file:

- include/[RayTracer.hpp](#)

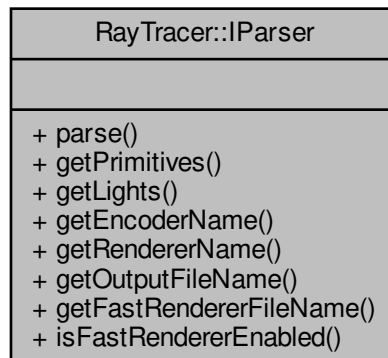
## 6.18 RayTracer::IParser Class Reference

```
#include <IParser.hpp>
```

Inheritance diagram for RayTracer::IParser:



Collaboration diagram for RayTracer::IParser:



## Public Member Functions

- virtual void [parse](#) ([RayTracer::Camera](#) &cam)=0  
*Parse the scene and fill the [Camera](#) object.*
- virtual std::vector< std::unique\_ptr< [RayTracer::IPrimitives](#) > > & [getPrimitives](#) ()=0  
*Get a vector of unique pointers to the scene primitives.*
- virtual std::vector< std::unique\_ptr< [RayTracer::ILight](#) > > & [getLights](#) ()=0  
*Get a vector of unique pointers to the scene lights.*
- virtual std::string [getEncoderName](#) () const =0  
*Get the name of the encoder.*
- virtual std::string [getRendererName](#) () const =0  
*Get the name of the renderer.*
- virtual std::string [getOutputFileName](#) () const =0  
*Get the name of the output file.*
- virtual std::string [getFastRendererFileName](#) () const =0
- virtual bool [isFastRendererEnabled](#) () const =0

## 6.18.1 Member Function Documentation

### 6.18.1.1 [getEncoderName\(\)](#)

```
virtual std::string RayTracer::IParser::getEncoderName ( ) const [pure virtual]
```

Get the name of the encoder.

#### Returns

std::string The name of the encoder

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

#### 6.18.1.2 getFastRendererFileName()

```
virtual std::string RayTracer::IParser::getFastRendererFileName ( ) const [pure virtual]
```

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

#### 6.18.1.3 getLights()

```
virtual std::vector<std::unique_ptr<RayTracer::ILight> >& RayTracer::IParser::getLights ( )  
[pure virtual]
```

Get a vector of unique pointers to the scene lights.

##### Returns

std::vector<std::unique\_ptr<RayTracer::ILight>>& Vector of unique pointers to the scene lights

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

#### 6.18.1.4 getOutputFileName()

```
virtual std::string RayTracer::IParser::getOutputFileName ( ) const [pure virtual]
```

Get the name of the output file.

##### Returns

std::string The name of the output file

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

#### 6.18.1.5 getPrimitives()

```
virtual std::vector<std::unique_ptr<RayTracer::IPrimitives> >& RayTracer::IParser::get↵  
Primitives ( ) [pure virtual]
```

Get a vector of unique pointers to the scene primitives.

##### Returns

std::vector<std::unique\_ptr<RayTracer::IPrimitives>>& Vector of unique pointers to the scene primitives

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

#### 6.18.1.6 getRendererName()

```
virtual std::string RayTracer::IParser::getRendererName ( ) const [pure virtual]
```

Get the name of the renderer.

##### Returns

std::string The name of the renderer

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

#### 6.18.1.7 isFastRendererEnabled()

```
virtual bool RayTracer::IParser::isFastRendererEnabled ( ) const [pure virtual]
```

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

#### 6.18.1.8 parse()

```
virtual void RayTracer::IParser::parse (
    RayTracer::Camera & cam ) [pure virtual]
```

Parse the scene and fill the [Camera](#) object.

##### Parameters

<i>cam</i>	The <a href="#">Camera</a> object to be filled
------------	--

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

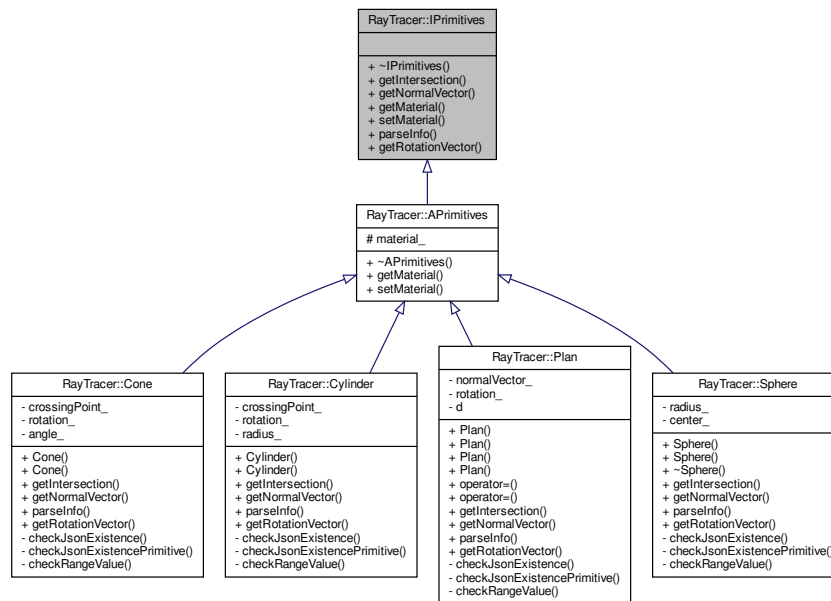
The documentation for this class was generated from the following file:

- [include/parser/IParser.hpp](#)

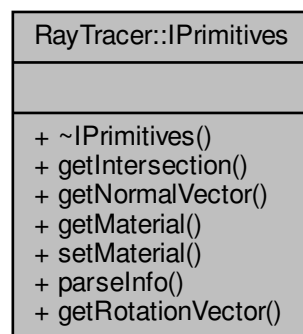
## 6.19 RayTracer::IPrimitives Class Reference

```
#include <IPrimitives.hpp>
```

Inheritance diagram for RayTracer::IPrimitives:



Collaboration diagram for RayTracer::IPrimitives:



## Public Member Functions

- virtual `~IPrimitives()` = default
- virtual `std::vector< double > getIntersection (RayTracer::Ray ray)=0`
- virtual `Math::Vector3D getNormalVector (Math::Point3D point)=0`
- virtual `Material getMaterial ()=0`
- virtual `void setMaterial (Color color, double reflect, int specular, double transparency)=0`
- virtual `void parseInfo (json object)=0`
- virtual `Math::Vector3D getRotationVector ()=0`

## 6.19.1 Constructor & Destructor Documentation

### 6.19.1.1 ~IPrimitives()

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

## 6.19.2 Member Function Documentation

### 6.19.2.1 getIntersection()

```
virtual std::vector<double> RayTracer::IPrimitives::getIntersection (
    RayTracer::Ray ray ) [pure virtual]
```

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

### 6.19.2.2 getMaterial()

```
virtual Material RayTracer::IPrimitives::getMaterial ( ) [pure virtual]
```

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

### 6.19.2.3 getNormalVector()

```
virtual Math::Vector3D RayTracer::IPrimitives::getNormalVector (
    Math::Point3D point ) [pure virtual]
```

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

### 6.19.2.4 getRotationVector()

```
virtual Math::Vector3D RayTracer::IPrimitives::getRotationVector ( ) [pure virtual]
```

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

### 6.19.2.5 parseInfo()

```
virtual void RayTracer::IPrimitives::parseInfo (
    json object ) [pure virtual]
```

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

### 6.19.2.6 setMaterial()

```
virtual void RayTracer::IPrimitives::setMaterial (
    Color color,
    double reflect,
    int specular,
    double transparency ) [pure virtual]
```

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

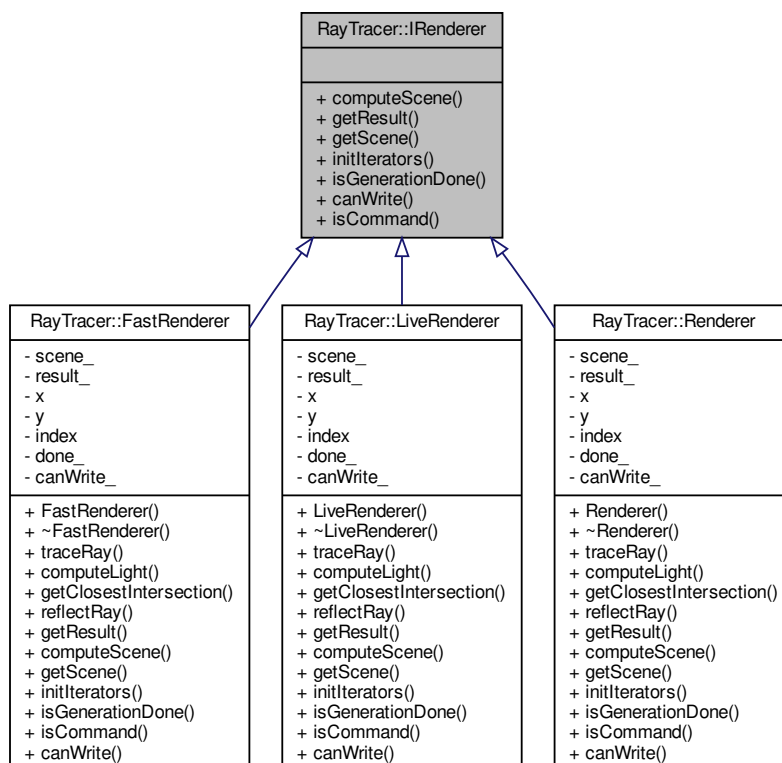
The documentation for this class was generated from the following file:

- [include/primitives/IPrimitives.hpp](#)

## 6.20 RayTracer::IRenderer Class Reference

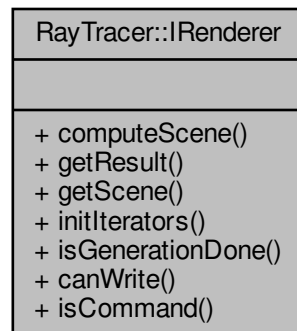
```
#include <IRenderer.hpp>
```

Inheritance diagram for RayTracer::IRenderer:





Collaboration diagram for RayTracer::IRenderer:



## Public Member Functions

- virtual void [computeScene](#) ()=0  
*Compute the scene and render the result.*
- virtual std::vector< std::vector< [Color](#) > > [getResult](#) ()=0  
*Get the result of the rendering as a 2D vector of colors.*
- virtual [Scene](#) & [getScene](#) ()=0  
*Get the scene used for rendering.*
- virtual void [initIterators](#) ()=0  
*Initializes the iterators for generation.*
- virtual bool [isGenerationDone](#) ()=0  
*Checks if the generation process is done.*
- virtual bool [canWrite](#) ()=0  
*Checks if writing is allowed.*
- virtual bool [isCommand](#) ()=0  
*Checks if the current action is a command.*

## 6.20.1 Member Function Documentation

### 6.20.1.1 [canWrite\(\)](#)

```
virtual bool RayTracer::IRenderer::canWrite ( ) [pure virtual]
```

Checks if writing is allowed.

#### Returns

True if writing is allowed, False otherwise

Implemented in [RayTracer::LiveRenderer](#), [RayTracer::FastRenderer](#), and [RayTracer::Renderer](#).

#### 6.20.1.2 computeScene()

```
virtual void RayTracer::IRenderer::computeScene ( ) [pure virtual]
```

Compute the scene and render the result.

Implemented in [RayTracer::LiveRenderer](#), [RayTracer::FastRenderer](#), and [RayTracer::Renderer](#).

#### 6.20.1.3 getResult()

```
virtual std::vector<std::vector<Color> > RayTracer::IRenderer::getResult ( ) [pure virtual]
```

Get the result of the rendering as a 2D vector of colors.

##### Returns

The result of the rendering as a 2D vector of colors.

Implemented in [RayTracer::LiveRenderer](#), [RayTracer::FastRenderer](#), and [RayTracer::Renderer](#).

#### 6.20.1.4 getScene()

```
virtual Scene& RayTracer::IRenderer::getScene ( ) [pure virtual]
```

Get the scene used for rendering.

##### Returns

The scene used for rendering.

Implemented in [RayTracer::LiveRenderer](#), [RayTracer::FastRenderer](#), and [RayTracer::Renderer](#).

#### 6.20.1.5 initIterators()

```
virtual void RayTracer::IRenderer::initIterators ( ) [pure virtual]
```

Initializes the iterators for generation.

Implemented in [RayTracer::LiveRenderer](#), [RayTracer::FastRenderer](#), and [RayTracer::Renderer](#).

### 6.20.1.6 isCommand()

```
virtual bool RayTracer::IRenderer::isCommand ( ) [pure virtual]
```

Checks if the current action is a command.

#### Returns

True if the current action is a command, False otherwise

Implemented in [RayTracer::LiveRenderer](#), [RayTracer::FastRenderer](#), and [RayTracer::Renderer](#).

### 6.20.1.7 isGenerationDone()

```
virtual bool RayTracer::IRenderer::isGenerationDone ( ) [pure virtual]
```

Checks if the generation process is done.

#### Returns

True if the generation is done, False otherwise

Implemented in [RayTracer::LiveRenderer](#), [RayTracer::FastRenderer](#), and [RayTracer::Renderer](#).

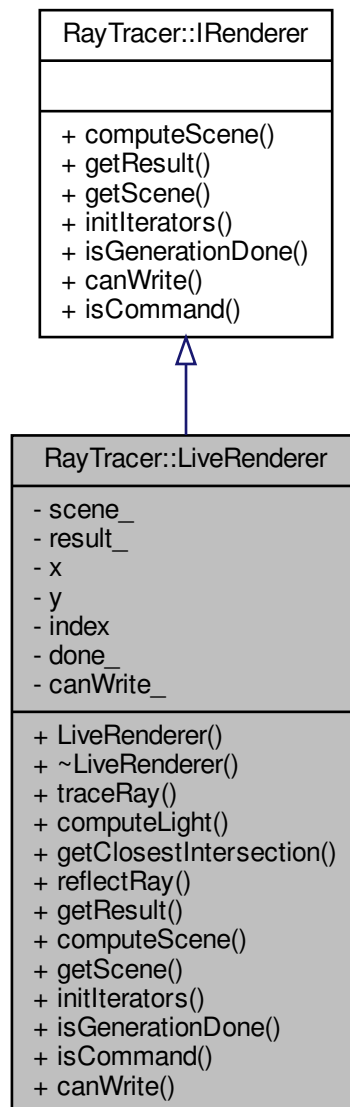
The documentation for this class was generated from the following file:

- [include/renderer/IRenderer.hpp](#)

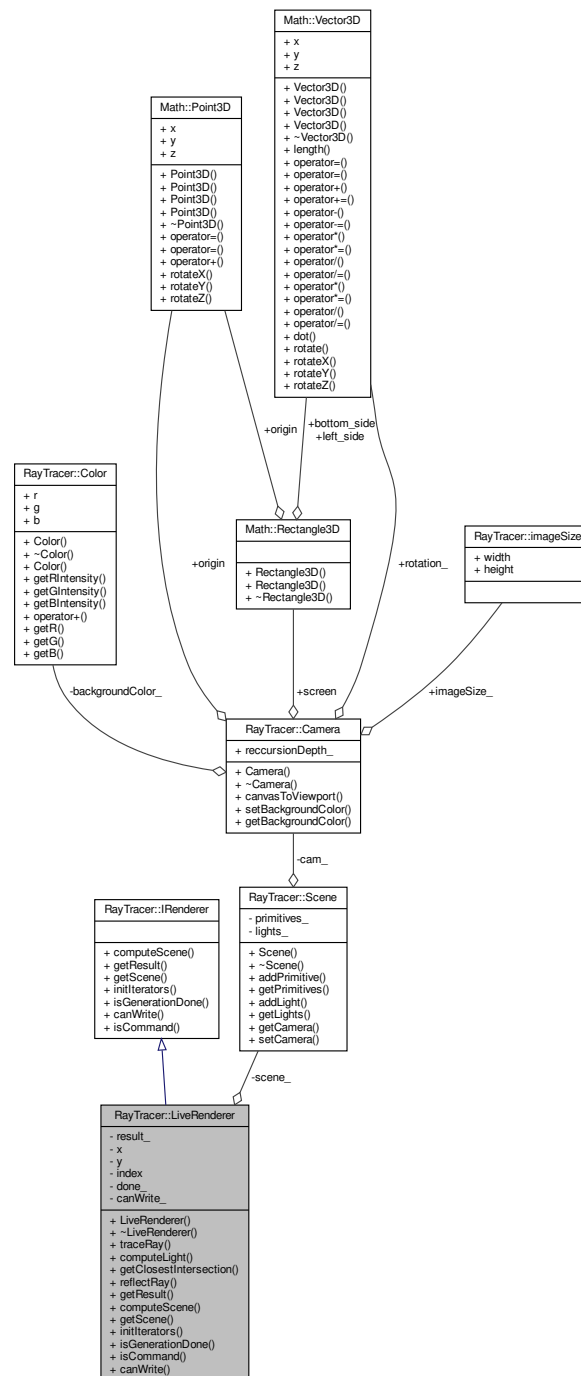
## 6.21 RayTracer::LiveRenderer Class Reference

```
#include <Renderer.hpp>
```

Inheritance diagram for RayTracer::LiveRenderer:



Collaboration diagram for RayTracer::LiveRenderer:



## Public Member Functions

- [LiveRenderer](#) ()=default
- [~LiveRenderer](#) ()=default
- [Color traceRay](#) ([RayTracer::Ray](#) ray, double tMin, double tMax, int recursion\_depth)
- [Color computeLight](#) ([Math::Point3D](#) intersectionPoint, [Math::Vector3D](#) normalVector, [Math::Vector3D](#) rayDir, int spec)

- `intersection getClosestIntersection` (`RayTracer::Ray` ray, double tMin, double tMax)
- `Math::Vector3D reflectRay` (`Math::Vector3D` normalVector, `Math::Vector3D` ray)
- `std::vector< std::vector< Color > > getResult` () override  
*Get the result of the rendering as a 2D vector of colors.*
- void `computeScene` () override  
*Compute the scene and render the result.*
- `Scene & getScene` () override  
*Get the scene used for rendering.*
- void `initIterators` () override  
*Initializes the iterators for generation.*
- bool `isGenerationDone` () override  
*Checks if the generation process is done.*
- bool `isCommand` () override  
*Checks if the current action is a command.*
- bool `canWrite` () override  
*Checks if writing is allowed.*

## Private Attributes

- `Scene scene_` {}
- `std::vector< std::vector< Color > > result_`
- double `x`
- double `y`
- `std::size_t index` = 0
- bool `done_` { false }
- bool `canWrite_` { false }

## 6.21.1 Constructor & Destructor Documentation

### 6.21.1.1 LiveRenderer()

```
RayTracer::LiveRenderer::LiveRenderer ( ) [default]
```

### 6.21.1.2 ~LiveRenderer()

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

## 6.21.2 Member Function Documentation

### 6.21.2.1 canWrite()

```
bool RayTracer::LiveRenderer::canWrite ( ) [override], [virtual]
```

Checks if writing is allowed.

#### Returns

True if writing is allowed, False otherwise

Implements [RayTracer::IRenderer](#).

### 6.21.2.2 computeLight()

```
Color RayTracer::LiveRenderer::computeLight (
    Math::Point3D intersectionPoint,
    Math::Vector3D normalVector,
    Math::Vector3D rayDir,
    int spec )
```

### 6.21.2.3 computeScene()

```
void RayTracer::LiveRenderer::computeScene ( ) [override], [virtual]
```

Compute the scene and render the result.

Implements [RayTracer::IRenderer](#).

### 6.21.2.4 getClosestIntersection()

```
intersection RayTracer::LiveRenderer::getClosestIntersection (
    RayTracer::Ray ray,
    double tMin,
    double tMax )
```

### 6.21.2.5 getResult()

```
std::vector< std::vector< Color > > RayTracer::LiveRenderer::getResult ( ) [override], [virtual]
```

Get the result of the rendering as a 2D vector of colors.

#### Returns

The result of the rendering as a 2D vector of colors.

Implements [RayTracer::IRenderer](#).

#### 6.21.2.6 `getScene()`

```
Scene & RayTracer::LiveRenderer::getScene ( ) [override], [virtual]
```

Get the scene used for rendering.

##### Returns

The scene used for rendering.

Implements [RayTracer::IRenderer](#).

#### 6.21.2.7 `initIterators()`

```
void RayTracer::LiveRenderer::initIterators ( ) [override], [virtual]
```

Initializes the iterators for generation.

Implements [RayTracer::IRenderer](#).

#### 6.21.2.8 `isCommand()`

```
bool RayTracer::LiveRenderer::isCommand ( ) [override], [virtual]
```

Checks if the current action is a command.

##### Returns

True if the current action is a command, False otherwise

Implements [RayTracer::IRenderer](#).

#### 6.21.2.9 `isGenerationDone()`

```
bool RayTracer::LiveRenderer::isGenerationDone ( ) [override], [virtual]
```

Checks if the generation process is done.

##### Returns

True if the generation is done, False otherwise

Implements [RayTracer::IRenderer](#).



### 6.21.2.10 reflectRay()

```
Math::Vector3D RayTracer::LiveRenderer::reflectRay (
    Math::Vector3D normalVector,
    Math::Vector3D ray )
```

### 6.21.2.11 traceRay()

```
Color RayTracer::LiveRenderer::traceRay (
    RayTracer::Ray ray,
    double tMin,
    double tMax,
    int recursion_depth )
```

## 6.21.3 Field Documentation

### 6.21.3.1 canWrite\_

```
bool RayTracer::LiveRenderer::canWrite_ { false } [private]
```

### 6.21.3.2 done\_

```
bool RayTracer::LiveRenderer::done_ { false } [private]
```

### 6.21.3.3 index

```
std::size_t RayTracer::LiveRenderer::index = 0 [private]
```

### 6.21.3.4 result\_

```
std::vector<std::vector<Color> > RayTracer::LiveRenderer::result_ [private]
```

#### 6.21.3.5 scene\_

```
Scene RayTracer::LiveRenderer::scene_ {} [private]
```

#### 6.21.3.6 x

```
double RayTracer::LiveRenderer::x [private]
```

#### 6.21.3.7 y

```
double RayTracer::LiveRenderer::y [private]
```

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

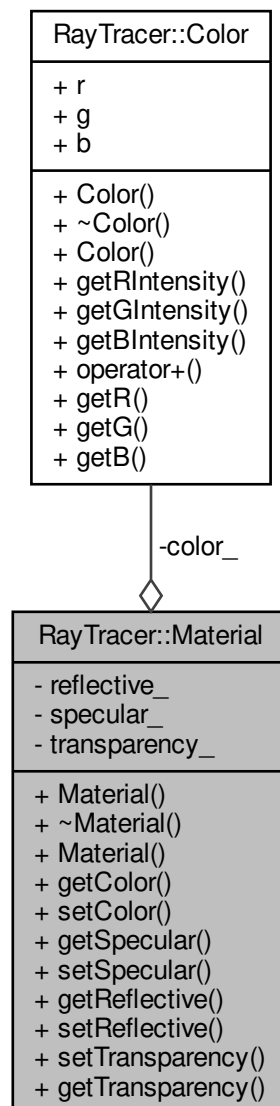
- src/plugins/renderer/liveRenderer/[Renderer.hpp](#)
- src/plugins/renderer/liveRenderer/[Renderer.cpp](#)

## 6.22 RayTracer::Material Class Reference

[Material](#) class representing the properties of a surface of an object.

```
#include <Material.hpp>
```

Collaboration diagram for RayTracer::Material:



## Public Member Functions

- [Material](#) ()=default
- [~Material](#) ()=default
- [Material](#) (double reflect, int specular, [Color](#) color, double transparency)  
*Construct a new [Material](#) object.*
- [Color](#) [getColor](#) () const  
*Get the color of the material.*
- void [setColor](#) (double r, double g, double b)  
*Set the color of the material.*
- int [getSpecular](#) () const

- Get the specular exponent of the material.*
  - void [setSpecular](#) (int spec)
- Set the specular exponent of the material.*
  - double [getReflective](#) () const
- Get the reflection coefficient of the material.*
  - void [setReflective](#) (double reflect)
- Set the reflection coefficient of the material.*
  - void [setTransparency](#) (double transparency)
- Set the transparency of the material.*
  - double [getTransparency](#) ()
- Get the transparency of the material.*

## Private Attributes

- double [reflective\\_](#) {0}
- int [specular\\_](#) {-1}
- [Color](#) [color\\_](#) {0, 0, 0}
- double [transparency\\_](#) {0}

### 6.22.1 Detailed Description

[Material](#) class representing the properties of a surface of an object.

### 6.22.2 Constructor & Destructor Documentation

#### 6.22.2.1 [Material\(\)](#) [1/2]

```
RayTracer::Material::Material ( ) [default]
```

#### 6.22.2.2 [~Material\(\)](#)

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

#### 6.22.2.3 [Material\(\)](#) [2/2]

```
RayTracer::Material::Material (
    double reflect,
    int specular,
    Color color,
    double transparency )
```

Construct a new [Material](#) object.

## Parameters

<i>reflect</i>	the reflection coefficient of the material
<i>specular</i>	the specular exponent of the material
<i>color</i>	the color of the material
<i>transparency</i>	the transparency of the material

## 6.22.3 Member Function Documentation

### 6.22.3.1 getColor()

```
RayTracer::Color RayTracer::Material::getColor ( ) const
```

Get the color of the material.

## Returns

the color of the material

### 6.22.3.2 getReflective()

```
double RayTracer::Material::getReflective ( ) const
```

Get the reflection coefficient of the material.

## Returns

the reflection coefficient of the material

### 6.22.3.3 getSpecular()

```
int RayTracer::Material::getSpecular ( ) const
```

Get the specular exponent of the material.

## Returns

the specular exponent of the material

#### 6.22.3.4 getTransparency()

```
double RayTracer::Material::getTransparency ( )
```

Get the transparency of the material.

##### Returns

the transparency of the material

#### 6.22.3.5 setColor()

```
void RayTracer::Material::setColor (
    double r,
    double g,
    double b )
```

Set the color of the material.

##### Parameters

<i>r</i>	the red component of the color
<i>g</i>	the green component of the color
<i>b</i>	the blue component of the color

#### 6.22.3.6 setReflective()

```
void RayTracer::Material::setReflective (
    double reflect )
```

Set the reflection coefficient of the material.

##### Parameters

<i>reflect</i>	the reflection coefficient to be set
----------------	--------------------------------------

#### 6.22.3.7 setSpecular()

```
void RayTracer::Material::setSpecular (
    int spec )
```

Set the specular exponent of the material.

## Parameters

<i>spec</i>	the specular exponent to be set
-------------	---------------------------------

### 6.22.3.8 setTransparency()

```
void RayTracer::Material::setTransparency (
    double transparency )
```

Set the transparency of the material.

## Parameters

<i>transparency</i>	the transparency to be set
---------------------	----------------------------

## 6.22.4 Field Documentation

### 6.22.4.1 color\_

```
Color RayTracer::Material::color_ {0, 0, 0} [private]
```

### 6.22.4.2 reflective\_

```
double RayTracer::Material::reflective_ {0} [private]
```

### 6.22.4.3 specular\_

```
int RayTracer::Material::specular_ {-1} [private]
```

### 6.22.4.4 transparency\_

```
double RayTracer::Material::transparency_ {0} [private]
```

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

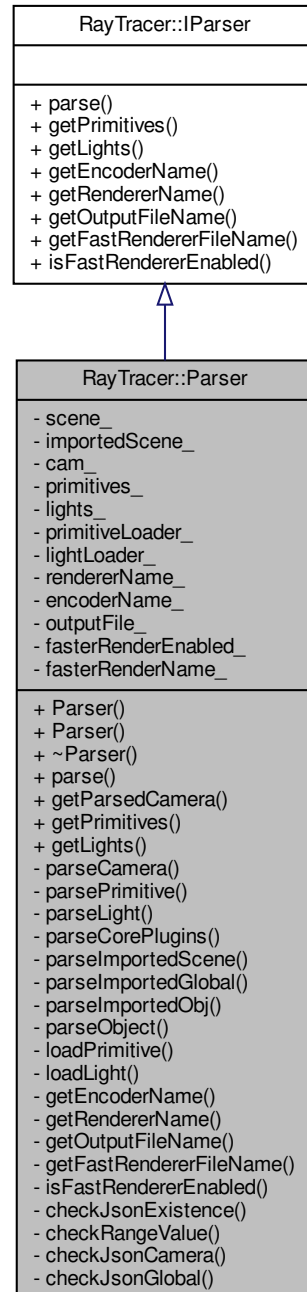
- include/material/[Material.hpp](#)
- src/utis/[Material.cpp](#)

## 6.23 RayTracer::Parser Class Reference

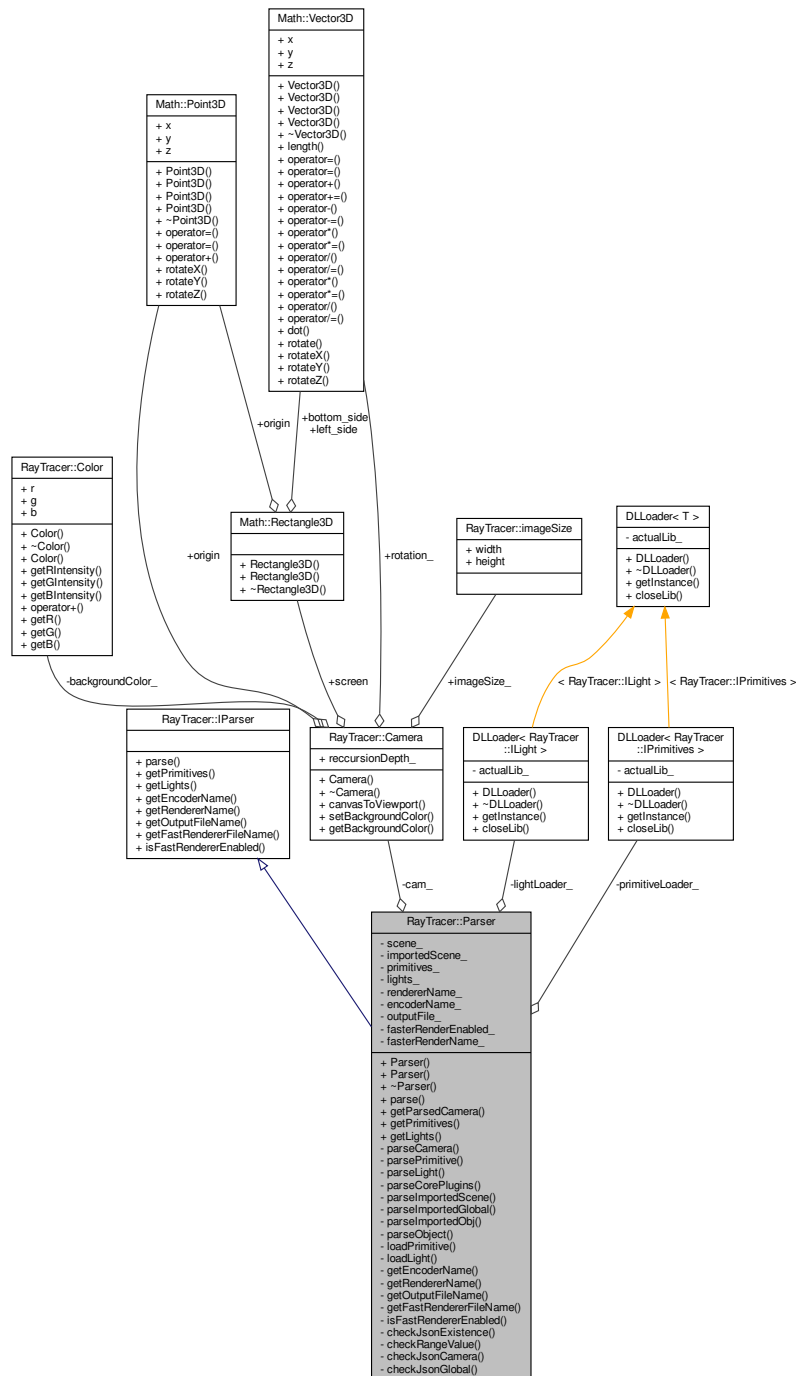
Class for the parser.

```
#include <Parser.hpp>
```

Inheritance diagram for RayTracer::Parser:







- `Parser()`=default
- `Parser(const std::string &path)`
- `~Parser()`=default
- `void parse(RayTracer::Camera &cam) final`

Generated by Doxygen

- [RayTracer::Camera getParsedCamera \(\)](#)  
*Returns the parsed camera object.*
- `std::vector< std::unique_ptr< RayTracer::IPrimitives > > & getPrimitives ()` final  
*Returns a reference to the vector of parsed primitives.*
- `std::vector< std::unique_ptr< RayTracer::ILight > > & getLights ()` final  
*Returns a reference to the vector of parsed lights.*

## Private Member Functions

- void [parseCamera](#) ([RayTracer::Camera](#) &cam)  
*Parses the camera information from the scene file.*
- void [parsePrimitive](#) ()  
*Parses primitive objects from the scene file.*
- void [parseLight](#) ()  
*Parses light objects from the scene file.*
- void [parseCorePlugins](#) ()  
*Parses the core plugins from the scene file.*
- void [parseImportedScene](#) ()  
*Parses elements from the imported scene.*
- void [parseImportedGlobal](#) ()  
*Parses the "global" element from the imported scene.*
- void [parseImportedObj](#) (const std::string &name)  
*Parses objects from the imported scene.*
- void [parseObject](#) (const std::string &name, [json](#) object, const std::string &key)  
*Parses a nested object from the imported scene.*
- `std::unique_ptr< IPrimitives > loadPrimitive (const std::string &key)  
Loads a primitive object based on the given key.`
- `std::unique_ptr< ILight > loadLight (const std::string &key)  
Loads a light object based on the given key.`
- std::string [getEncoderName](#) () const final  
*Gets the name of the encoder.*
- std::string [getRendererName](#) () const final  
*Gets the name of the renderer.*
- std::string [getOutputFileName](#) () const final  
*Gets the name of the output file.*
- std::string [getFastRendererFileName](#) () const final  
*Gets the name of the fast renderer.*
- bool [isFastRendererEnabled](#) () const final  
*Checks if the fast renderer is enabled.*
- void [checkJsonExistence](#) (const [json](#) &scene, const std::string &field\_name)  
*Checks if the given key exists in the scene file.*
- void [checkRangeValue](#) (const [json](#) &scene, const std::string &field\_name, const std::string &comparison\_↵  
sign, double value)  
*Checks if the value of the given key is within the specified range.*
- void [checkJsonCamera](#) ([json](#) scene)  
*Checks if the camera exists in the scene file.*
- void [checkJsonGlobal](#) ([json](#) scene)  
*Checks if the global exist in the scene file.*

## Private Attributes

- [json scene\\_](#) {}
- [json importedScene\\_](#) {}
- [RayTracer::Camera cam\\_](#)
- [std::vector< std::unique\\_ptr< RayTracer::IPrimitives > > primitives\\_](#)
- [std::vector< std::unique\\_ptr< RayTracer::ILight > > lights\\_](#)
- [DLLoader< IPrimitives > primitiveLoader\\_](#)
- [DLLoader< ILight > lightLoader\\_](#)
- [std::string rendererName\\_](#)
- [std::string encoderName\\_](#)
- [std::string outputFile\\_](#)
- [bool fasterRenderEnabled\\_](#) { false }
- [std::string fasterRenderName\\_](#)

### 6.23.1 Detailed Description

Class for the parser.

### 6.23.2 Constructor & Destructor Documentation

#### 6.23.2.1 Parser() [1/2]

```
RayTracer::Parser::Parser ( ) [default]
```

#### 6.23.2.2 Parser() [2/2]

```
RayTracer::Parser::Parser (
    const std::string & path ) [explicit]
```

#### 6.23.2.3 ~Parser()

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

### 6.23.3 Member Function Documentation

#### 6.23.3.1 checkJsonCamera()

```
void RayTracer::Parser::checkJsonCamera (
    json scene ) [private]
```

Checks if the camera exists in the scene file.

## Parameters

<i>scene</i>	the scene file
--------------	----------------

**6.23.3.2 checkJsonExistence()**

```
void RayTracer::Parser::checkJsonExistence (
    const json & scene,
    const std::string & field_name ) [private]
```

Checks if the given key exists in the scene file.

## Parameters

<i>scene</i>	the scene file
<i>field_name</i>	the key to check

**6.23.3.3 checkJsonGlobal()**

```
void RayTracer::Parser::checkJsonGlobal (
    json scene ) [private]
```

Checks if the global exist in the scene file.

## Parameters

<i>scene</i>	the scene file
--------------	----------------

**6.23.3.4 checkRangeValue()**

```
void RayTracer::Parser::checkRangeValue (
    const json & scene,
    const std::string & field_name,
    const std::string & comparison_sign,
    double value ) [private]
```

Checks if the value of the given key is within the specified range.

## Parameters

<i>scene</i>	the scene file
<i>field_name</i>	the key to check
<i>comparison_sign</i>	the comparison sign
<i>value</i>	the value to compare with

### 6.23.3.5 getEncoderName()

```
std::string RayTracer::Parser::getEncoderName ( ) const [final], [private], [virtual]
```

Gets the name of the encoder.

#### Returns

the name of the encoder

Implements [RayTracer::IParser](#).

### 6.23.3.6 getFastRendererFileName()

```
std::string RayTracer::Parser::getFastRendererFileName ( ) const [final], [private], [virtual]
```

Gets the name of the fast renderer.

#### Returns

the name of the fast renderer

Implements [RayTracer::IParser](#).

### 6.23.3.7 getLights()

```
std::vector< std::unique_ptr< RayTracer::ILight > > & RayTracer::Parser::getLights ( ) [final], [virtual]
```

Returns a reference to the vector of parsed lights.

#### Returns

the vector of parsed lights

Implements [RayTracer::IParser](#).

#### 6.23.3.8 getOutputFileName()

```
std::string RayTracer::Parser::getOutputFileName ( ) const [final], [private], [virtual]
```

Gets the name of the output file.

##### Returns

the name of the output file

Implements [RayTracer::IParser](#).

#### 6.23.3.9 getParsedCamera()

```
RayTracer::Camera RayTracer::Parser::getParsedCamera ( )
```

Returns the parsed camera object.

##### Returns

the parsed camera object

#### 6.23.3.10 getPrimitives()

```
std::vector< std::unique_ptr< RayTracer::IPrimitives > > & RayTracer::Parser::getPrimitives ( ) [final], [virtual]
```

Returns a reference to the vector of parsed primitives.

##### Returns

the vector of parsed primitives

Implements [RayTracer::IParser](#).

#### 6.23.3.11 getRendererName()

```
std::string RayTracer::Parser::getRendererName ( ) const [final], [private], [virtual]
```

Gets the name of the renderer.

##### Returns

the name of the renderer

Implements [RayTracer::IParser](#).

### 6.23.3.12 isFastRendererEnabled()

```
bool RayTracer::Parser::isFastRendererEnabled ( ) const [final], [private], [virtual]
```

Checks if the fast renderer is enabled.

#### Returns

true if the fast renderer is enabled, false otherwise

Implements [RayTracer::IParser](#).

### 6.23.3.13 loadLight()

```
std::unique_ptr< ILight > RayTracer::Parser::loadLight (
    const std::string & key ) [private]
```

Loads a light object based on the given key.

#### Parameters

<i>key</i>	the key for the light object
------------	------------------------------

#### Returns

a unique pointer to the parsed light object

### 6.23.3.14 loadPrimitive()

```
std::unique_ptr< IPrimitives > RayTracer::Parser::loadPrimitive (
    const std::string & key ) [private]
```

Loads a primitive object based on the given key.

#### Parameters

<i>key</i>	the key for the primitive object
------------	----------------------------------

#### Returns

a unique pointer to the parsed primitive object

### 6.23.3.15 parse()

```
void RayTracer::Parser::parse (
    RayTracer::Camera & cam ) [final], [virtual]
```

Parses the scene file and extracts the camera information.

#### Parameters

<i>cam</i>	the camera object to store the parsed data
------------	--

Implements [RayTracer::IParser](#).

### 6.23.3.16 parseCamera()

```
void RayTracer::Parser::parseCamera (
    RayTracer::Camera & cam ) [private]
```

Parses the camera information from the scene file.

#### Parameters

<i>cam</i>	the camera object to store the parsed data
------------	--

### 6.23.3.17 parseCorePlugins()

```
void RayTracer::Parser::parseCorePlugins ( ) [private]
```

Parses the core plugins from the scene file.

### 6.23.3.18 parseImportedGlobal()

```
void RayTracer::Parser::parseImportedGlobal ( ) [private]
```

Parses the "global" element from the imported scene.

### 6.23.3.19 parseImportedObj()

```
void RayTracer::Parser::parseImportedObj (
    const std::string & name ) [private]
```

Parses objects from the imported scene.



## Parameters

<i>name</i>	the name of the object
-------------	------------------------

**6.23.3.20 parseImportedScene()**

```
void RayTracer::Parser::parseImportedScene ( ) [private]
```

Parses elements from the imported scene.

**6.23.3.21 parseLight()**

```
void RayTracer::Parser::parseLight ( ) [private]
```

Parses light objects from the scene file.

**6.23.3.22 parseObject()**

```
void RayTracer::Parser::parseObject (
    const std::string & name,
    json object,
    const std::string & key ) [private]
```

Parses a nested object from the imported scene.

## Parameters

<i>name</i>	the name of the object
<i>object</i>	the object to parse
<i>key</i>	the key of the object

**6.23.3.23 parsePrimitive()**

```
void RayTracer::Parser::parsePrimitive ( ) [private]
```

Parses primitive objects from the scene file.

**6.23.4 Field Documentation**

#### 6.23.4.1 cam\_

```
RayTracer::Camera RayTracer::Parser::cam_ [private]
```

#### 6.23.4.2 encoderName\_

```
std::string RayTracer::Parser::encoderName_ [private]
```

#### 6.23.4.3 fasterRenderEnabled\_

```
bool RayTracer::Parser::fasterRenderEnabled_ { false } [private]
```

#### 6.23.4.4 fasterRenderName\_

```
std::string RayTracer::Parser::fasterRenderName_ [private]
```

#### 6.23.4.5 importedScene\_

```
json RayTracer::Parser::importedScene_ {} [private]
```

#### 6.23.4.6 lightLoader\_

```
DLLoader<ILight> RayTracer::Parser::lightLoader_ [private]
```

#### 6.23.4.7 lights\_

```
std::vector<std::unique_ptr<RayTracer::ILight> > RayTracer::Parser::lights_ [private]
```

#### 6.23.4.8 outputFile\_

```
std::string RayTracer::Parser::outputFile_ [private]
```

**6.23.4.9 primitiveLoader\_**

```
DLLoader<IPrimitives> RayTracer::Parser::primitiveLoader_ [private]
```

**6.23.4.10 primitives\_**

```
std::vector<std::unique_ptr<RayTracer::IPrimitives> > RayTracer::Parser::primitives_ [private]
```

**6.23.4.11 rendererName\_**

```
std::string RayTracer::Parser::rendererName_ [private]
```

**6.23.4.12 scene\_**

```
json RayTracer::Parser::scene_ {} [private]
```

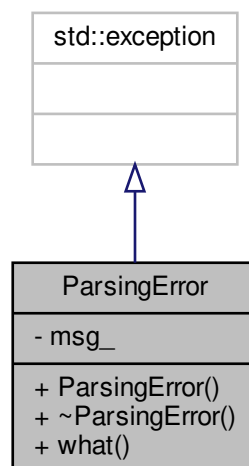
The documentation for this class was generated from the following files:

- include/parser/Parser.hpp
- src/parser/CheckJsonExistence.cpp
- src/parser/ParseImportedScene.cpp
- src/parser/ParseLight.cpp
- src/parser/ParsePrimitive.cpp
- src/parser/Parser.cpp

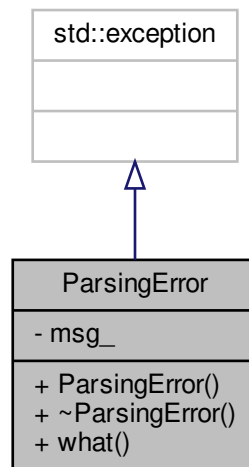
**6.24 ParsingError Class Reference**

```
#include <Error.hpp>
```

Inheritance diagram for ParsingError:



Collaboration diagram for ParsingError:



## Public Member Functions

- [ParsingError](#) (const char \*msg)
- [~ParsingError](#) () override=default
- const char \* [what](#) () const noexcept override

## Private Attributes

- const char \* [msg\\_](#)

## 6.24.1 Constructor & Destructor Documentation

### 6.24.1.1 ParsingError()

```
ParsingError::ParsingError (
    const char * msg )
```

### 6.24.1.2 ~ParsingError()

```
ParsingError::~~ParsingError ( ) [override], [default]
```

## 6.24.2 Member Function Documentation

### 6.24.2.1 what()

```
const char * ParsingError::what ( ) const [override], [noexcept]
```

## 6.24.3 Field Documentation

### 6.24.3.1 msg\_

```
const char* ParsingError::msg_ [private]
```

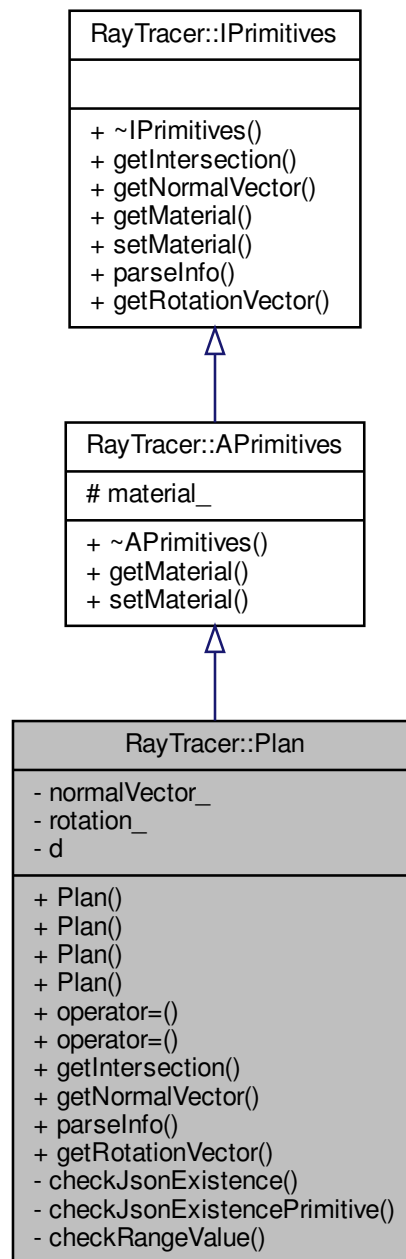
The documentation for this class was generated from the following files:

- [include/Error.hpp](#)
- [src/parser/Error.cpp](#)

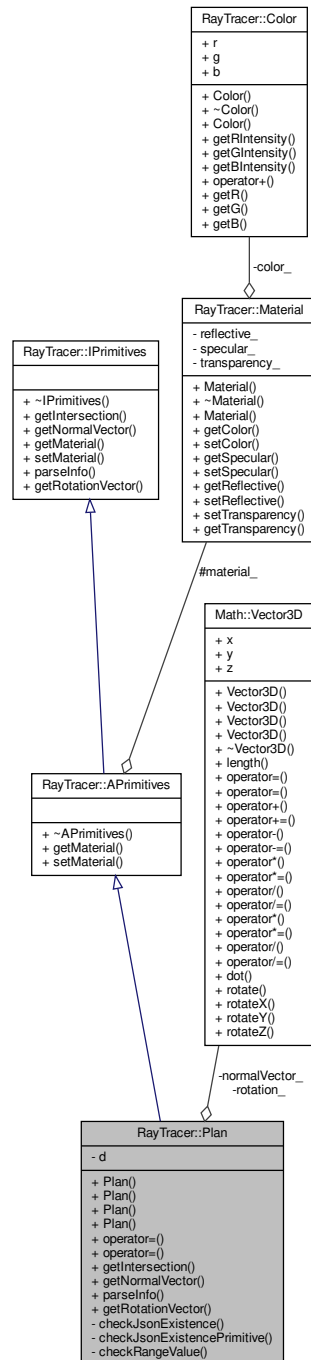
## 6.25 RayTracer::Plan Class Reference

```
#include <Plan.hpp>
```

Inheritance diagram for RayTracer::Plan:



Collaboration diagram for RayTracer::Plan:



## Public Member Functions

- `Plan ()`=default
- `Plan (Math::Vector3D normalVector, double d)`
- `Plan (Plan &)=default`
- `Plan (Plan &&)=default`
- `Plan & operator= (const Plan &)=default`

- `Plan & operator= (Plan &&)=default`
- `std::vector< double > getIntersection (RayTracer::Ray ray) final`
- `Math::Vector3D getNormalVector (Math::Point3D point) final`
- `void parseInfo (json object) final`
- `Math::Vector3D getRotationVector ()`

## Private Member Functions

- `void checkJsonExistence (const json &scene, const std::string &field_name)`
- `void checkJsonExistencePrimitive (const json &scene)`
- `void checkRangeValue (const json &scene, const std::string &field_name, const std::string &comparison_↵  
sign, double value)`

## Private Attributes

- `Math::Vector3D normalVector_`
- `Math::Vector3D rotation_`
- `double d`

## Additional Inherited Members

### 6.25.1 Constructor & Destructor Documentation

#### 6.25.1.1 Plan() [1/4]

```
RayTracer::Plan::Plan ( ) [default]
```

#### 6.25.1.2 Plan() [2/4]

```
RayTracer::Plan::Plan (
    Math::Vector3D normalVector,
    double d )
```

#### 6.25.1.3 Plan() [3/4]

```
RayTracer::Plan::Plan (
    Plan & ) [default]
```



#### 6.25.1.4 Plan() [4/4]

```
RayTracer::Plan::Plan (  
    Plan && ) [default]
```

### 6.25.2 Member Function Documentation

#### 6.25.2.1 checkJsonExistence()

```
void RayTracer::Plan::checkJsonExistence (  
    const json & scene,  
    const std::string & field_name ) [private]
```

#### 6.25.2.2 checkJsonExistencePrimitive()

```
void RayTracer::Plan::checkJsonExistencePrimitive (  
    const json & scene ) [private]
```

#### 6.25.2.3 checkRangeValue()

```
void RayTracer::Plan::checkRangeValue (  
    const json & scene,  
    const std::string & field_name,  
    const std::string & comparison_sign,  
    double value ) [private]
```

#### 6.25.2.4 getIntersection()

```
std::vector< double > RayTracer::Plan::getIntersection (  
    RayTracer::Ray ray ) [final], [virtual]
```

Implements [RayTracer::IPrimitives](#).

#### 6.25.2.5 getNormalVector()

```
Math::Vector3D RayTracer::Plan::getNormalVector (  
    Math::Point3D point ) [final], [virtual]
```

Implements [RayTracer::IPrimitives](#).

#### 6.25.2.6 getRotationVector()

```
Math::Vector3D RayTracer::Plan::getRotationVector ( ) [virtual]
```

Implements [RayTracer::IPrimitives](#).

#### 6.25.2.7 operator=() [1/2]

```
Plan& RayTracer::Plan::operator= (
    const Plan & ) [default]
```

#### 6.25.2.8 operator=() [2/2]

```
Plan& RayTracer::Plan::operator= (
    Plan && ) [default]
```

#### 6.25.2.9 parseInfo()

```
void RayTracer::Plan::parseInfo (
    json object ) [final], [virtual]
```

Implements [RayTracer::IPrimitives](#).

### 6.25.3 Field Documentation

#### 6.25.3.1 d

```
double RayTracer::Plan::d [private]
```

#### 6.25.3.2 normalVector\_

```
Math::Vector3D RayTracer::Plan::normalVector_ [private]
```

### 6.25.3.3 rotation\_

`Math::Vector3D RayTracer::Plan::rotation_ [private]`

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

- `src/plugins/primitives/plan/Plan.hpp`
- `src/plugins/primitives/plan/Plan.cpp`

## 6.26 Math::Point3D Class Reference

Class for the 3D point.

```
#include <Point3D.hpp>
```

Collaboration diagram for Math::Point3D:

Math::Point3D
+ x + y + z
+ Point3D() + Point3D() + Point3D() + Point3D() + ~Point3D() + operator=() + operator=() + operator+() + rotateX() + rotateY() + rotateZ()

### Public Member Functions

- `Point3D ()`=default
- `Point3D (double x, double y, double z)`  
*Constructs a point with the given x, y, and z coordinates.*
- `Point3D (Point3D &)=default`  
*Copy constructor, constructs a new point by copying the coordinates of the given point.*
- `Point3D (Point3D &&)=default`  
*Move constructor, constructs a new point by moving the coordinates of the given point.*
- `~Point3D ()`=default
- `Point3D & operator= (const Point3D &)=default`

- *Copy assignment operator, assigns the coordinates of the given point to this point.*  
• `Point3D & operator= (Point3D &&)=default`  
*Move assignment operator, moves the coordinates of the given point to this point.*
- `Point3D operator+ (const Vector3D &)`  
*Adds a vector to this point and returns the result as a new point.*
- `void rotateX (double x)`  
*Rotate the point on the x axis.*
- `void rotateY (double y)`  
*Rotate the point on the y axis.*
- `void rotateZ (double z)`  
*Rotate the point on the z axis.*

## Data Fields

- double `x` = 0
- double `y` = 0
- double `z` = 0

## 6.26.1 Detailed Description

Class for the 3D point.

## 6.26.2 Constructor & Destructor Documentation

### 6.26.2.1 Point3D() [1/4]

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

### 6.26.2.2 Point3D() [2/4]

```
Math::Point3D::Point3D (
    double x,
    double y,
    double z )
```

Constructs a point with the given x, y, and z coordinates.

#### Parameters

<code>x</code>	The x coordinate of the point.
<code>y</code>	The y coordinate of the point.
<code>z</code>	The z coordinate of the point.

### 6.26.2.3 Point3D() [3/4]

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

Copy constructor, constructs a new point by copying the coordinates of the given point.

#### Parameters

<i>point</i>	The point to copy.
--------------	--------------------

### 6.26.2.4 Point3D() [4/4]

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

Move constructor, constructs a new point by moving the coordinates of the given point.

#### Parameters

<i>point</i>	The point to move.
--------------	--------------------

### 6.26.2.5 ~Point3D()

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

## 6.26.3 Member Function Documentation

### 6.26.3.1 operator+()

```
Point3D Math::Point3D::operator+ (
    const Vector3D & vect )
```

Adds a vector to this point and returns the result as a new point.

#### Parameters

<i>vect</i>	The vector to add.
-------------	--------------------

**Returns**

The resulting point after adding the vector.

**6.26.3.2 operator=() [1/2]**

```
Point3D& Math::Point3D::operator= (
    const Point3D & ) [default]
```

Copy assignment operator, assigns the coordinates of the given point to this point.

**Parameters**

<i>point</i>	The point to copy.
--------------	--------------------

**Returns**

A reference to this point.

**6.26.3.3 operator=() [2/2]**

```
Point3D& Math::Point3D::operator= (
    Point3D && ) [default]
```

Move assignment operator, moves the coordinates of the given point to this point.

**Parameters**

<i>point</i>	The point to move.
--------------	--------------------

**Returns**

A reference to this point.

**6.26.3.4 rotateX()**

```
void Math::Point3D::rotateX (
    double x )
```

Rotate the point on the x axis.

## Parameters

<i>x</i>	The x angle in radian
----------	-----------------------

**6.26.3.5 rotateY()**

```
void Math::Point3D::rotateY (
    double y )
```

Rotate the point on the y axis.

## Parameters

<i>y</i>	The y angle in radian
----------	-----------------------

**6.26.3.6 rotateZ()**

```
void Math::Point3D::rotateZ (
    double z )
```

Rotate the point on the z axis.

## Parameters

<i>z</i>	The z angle in radian
----------	-----------------------

**6.26.4 Field Documentation****6.26.4.1 x**

```
double Math::Point3D::x = 0
```

**6.26.4.2 y**

```
double Math::Point3D::y = 0
```

#### 6.26.4.3 z

```
double Math::Point3D::z = 0
```

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

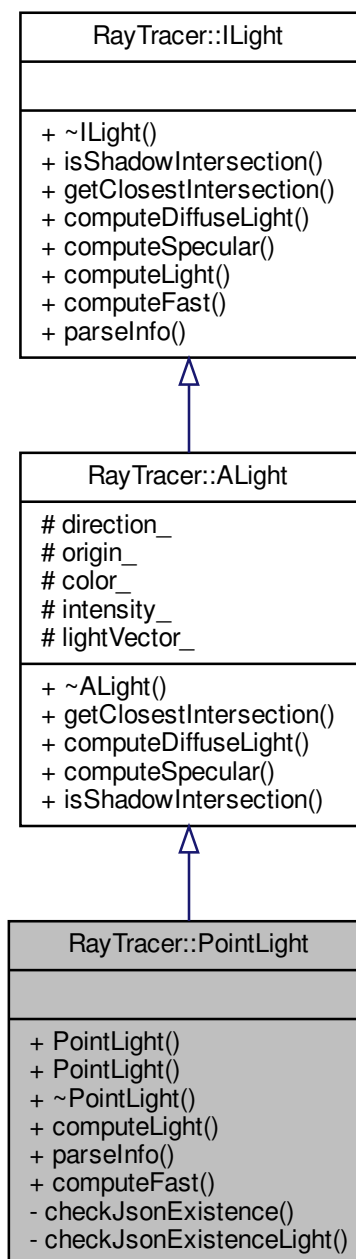
- include/maths/[Point3D.hpp](#)
- src/maths/[Point3D.cpp](#)

## 6.27 RayTracer::PointLight Class Reference

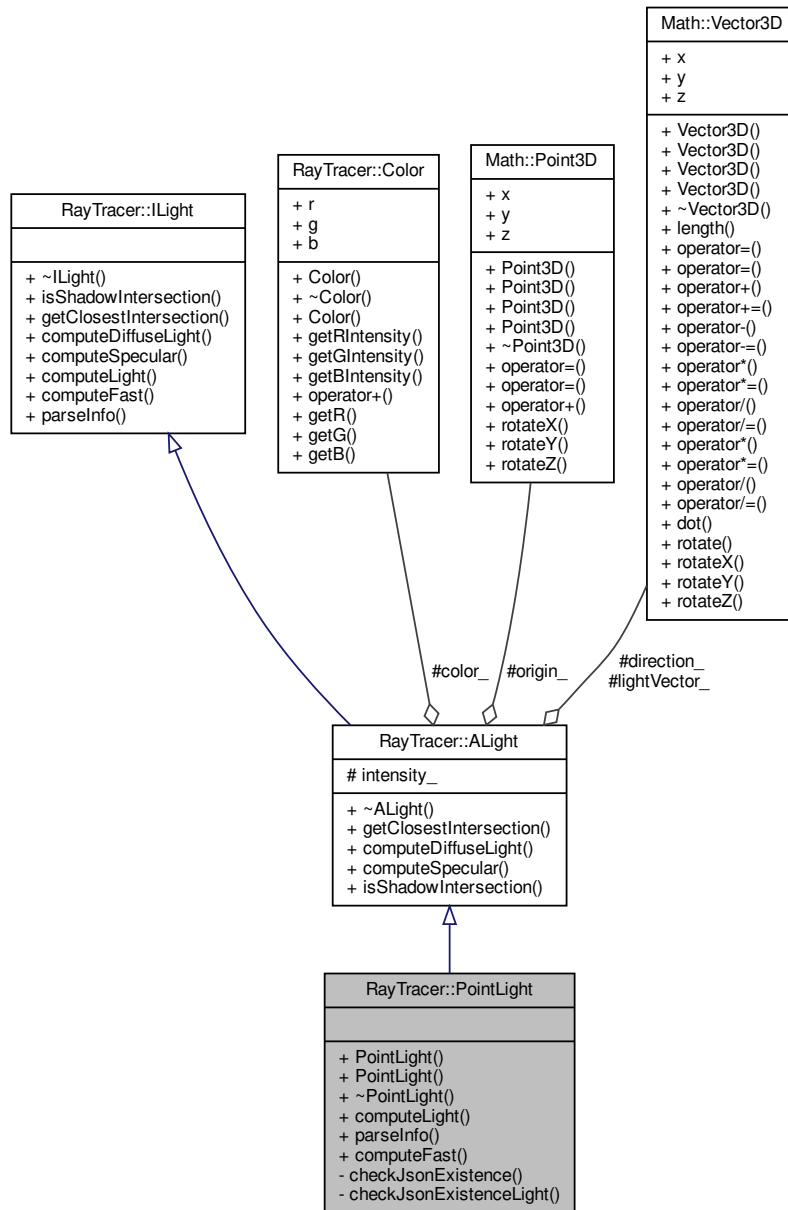
```
#include <PointLight.hpp>
```



Inheritance diagram for RayTracer::PointLight:



Collaboration diagram for RayTracer::PointLight:



## Public Member Functions

- `PointLight` (`Color` color, double intensity, `Math::Point3D` origin)
- `PointLight` ()=default
- `~PointLight` () override=default
- `Color computeLight` (`Math::Vector3D` normalVector, int spec, `Math::Point3D` intersectionPoint, `Math::Vector3D` rayDir, `std::vector< std::unique_ptr< IPrimitives >> &primitives`) override
- void `parseInfo` (json object) final  
*Information parser to create the light object.*
- `Color computeFast` (`Math::Vector3D` normalVector, int spec, `Math::Point3D` intersectionPoint, `Math::Vector3D` rayDir, `std::vector< std::unique_ptr< IPrimitives >> &primitives`) final

## Private Member Functions

- void [checkJsonExistence](#) (const [json](#) &scene, const std::string &field\_name)
- void [checkJsonExistenceLight](#) (const [json](#) &scene)

## Additional Inherited Members

### 6.27.1 Constructor & Destructor Documentation

#### 6.27.1.1 PointLight() [1/2]

```
RayTracer::PointLight::PointLight (
    Color color,
    double intensity,
    Math::Point3D origin )
```

#### 6.27.1.2 PointLight() [2/2]

```
RayTracer::PointLight::PointLight ( ) [default]
```

#### 6.27.1.3 ~PointLight()

```
RayTracer::PointLight::~~PointLight ( ) [override], [default]
```

### 6.27.2 Member Function Documentation

#### 6.27.2.1 checkJsonExistence()

```
void RayTracer::PointLight::checkJsonExistence (
    const json & scene,
    const std::string & field_name ) [private]
```

### 6.27.2.2 checkJsonExistenceLight()

```
void RayTracer::PointLight::checkJsonExistenceLight (
    const json & scene ) [private]
```

### 6.27.2.3 computeFast()

```
Color RayTracer::PointLight::computeFast (
    Math::Vector3D normalVector,
    int spec,
    Math::Point3D intersectionPoint,
    Math::Vector3D rayDir,
    std::vector< std::unique_ptr< IPrimitives >> & primitives ) [final], [virtual]
```

Implements [RayTracer::ILight](#).

### 6.27.2.4 computeLight()

```
Color RayTracer::PointLight::computeLight (
    Math::Vector3D normalVector,
    int spec,
    Math::Point3D intersectionPoint,
    Math::Vector3D rayDir,
    std::vector< std::unique_ptr< IPrimitives >> & primitives ) [override], [virtual]
```

Implements [RayTracer::ILight](#).

### 6.27.2.5 parseInfo()

```
void RayTracer::PointLight::parseInfo (
    json object ) [final], [virtual]
```

Information parser to create the light object.

#### Parameters

<i>object</i>	the json object containing light info
---------------	---------------------------------------

Implements [RayTracer::ILight](#).

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

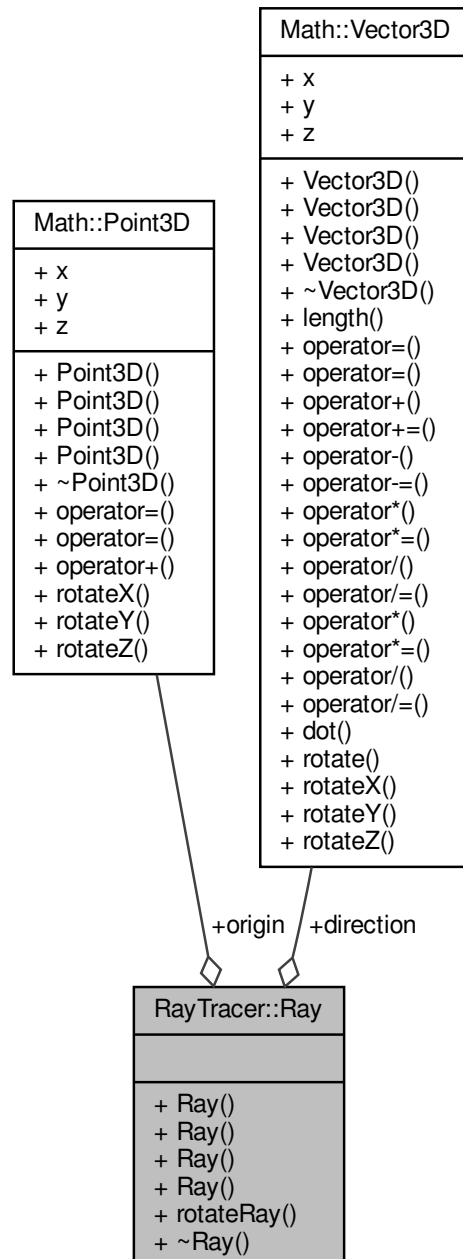
- [src/plugins/lights/point/PointLight.hpp](#)
- [src/plugins/lights/point/PointLight.cpp](#)

## 6.28 RayTracer::Ray Class Reference

Class for the ray.

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

Collaboration diagram for RayTracer::Ray:



## Public Member Functions

- [Ray](#) ()=default
- [Ray](#) ([Math::Point3D](#), [Math::Vector3D](#))  
*Constructs a new 3D ray with the given origin point and direction vector.*
- [Ray](#) ([Ray](#) &)=default  
*Constructs a new 3D ray by copying or moving the contents of the given ray.*
- [Ray](#) ([Ray](#) &&)=default  
*Constructs a new 3D ray by copying or moving the contents of the given ray.*
- void [rotateRay](#) (double x, double y, double z)  
*Rotate the ray (origin and direction)*
- [~Ray](#) ()=default

## Data Fields

- [Math::Point3D](#) origin
- [Math::Vector3D](#) direction

### 6.28.1 Detailed Description

Class for the ray.

### 6.28.2 Constructor & Destructor Documentation

#### 6.28.2.1 [Ray\(\)](#) [1/4]

```
RayTracer::Ray::Ray ( ) [default]
```

#### 6.28.2.2 [Ray\(\)](#) [2/4]

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

Constructs a new 3D ray with the given origin point and direction vector.

#### Parameters

<i>origin</i>	The origin point of the ray.
<i>direction</i>	A vector representing the direction of the ray.

### 6.28.2.3 Ray() [3/4]

```
RayTracer::Ray::Ray (
    Ray & ) [default]
```

Constructs a new 3D ray by copying or moving the contents of the given ray.

#### Parameters

<i>ray</i>	The ray to copy.
------------	------------------

### 6.28.2.4 Ray() [4/4]

```
RayTracer::Ray::Ray (
    Ray && ) [default]
```

Constructs a new 3D ray by copying or moving the contents of the given ray.

#### Parameters

<i>ray</i>	The ray to move.
------------	------------------

### 6.28.2.5 ~Ray()

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

## 6.28.3 Member Function Documentation

### 6.28.3.1 rotateRay()

```
void RayTracer::Ray::rotateRay (
    double x,
    double y,
    double z )
```

Rotate the ray (origin and direction)

#### Parameters

<i>x</i>	The x angle in radian
<i>y</i>	The y angle in radian
<i>z</i>	The z angle in radian

## 6.28.4 Field Documentation

### 6.28.4.1 direction

`Math::Vector3D RayTracer::Ray::direction`

### 6.28.4.2 origin

`Math::Point3D RayTracer::Ray::origin`

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

- `include/maths/Ray.hpp`
- `src/maths/Ray.cpp`

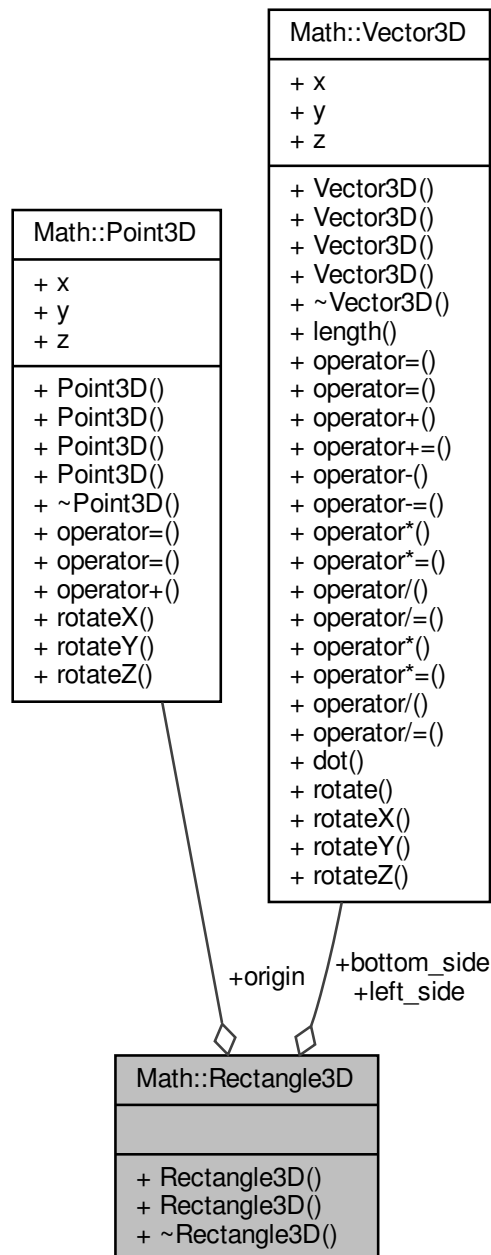
## 6.29 Math::Rectangle3D Class Reference

Class for the 3D rectangle.

```
#include <Rectangle3D.hpp>
```



Collaboration diagram for Math::Rectangle3D:



## Public Member Functions

- [Rectangle3D \(\)](#)=default
- [Rectangle3D \(Math::Point3D origin, Math::Vector3D bottom\\_side, Math::Vector3D left\\_side\)](#)  
*Constructs a new 3D rectangle with the given origin, bottom side vector, and left side vector.*
- [~Rectangle3D \(\)](#)=default

## Data Fields

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

### 6.29.1 Detailed Description

Class for the 3D rectangle.

### 6.29.2 Constructor & Destructor Documentation

#### 6.29.2.1 `Rectangle3D()` [1/2]

```
Math::Rectangle3D::Rectangle3D ( ) [default]
```

#### 6.29.2.2 `Rectangle3D()` [2/2]

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

Constructs a new 3D rectangle with the given origin, bottom side vector, and left side vector.

#### Parameters

<i>origin</i>	The origin point of the rectangle.
<i>bottom_side</i>	A vector representing the length and direction of the bottom side of the rectangle.
<i>left_side</i>	A vector representing the length and direction of the left side of the rectangle.

#### 6.29.2.3 `~Rectangle3D()`

```
Math::Rectangle3D::~~Rectangle3D ( ) [default]
```

### 6.29.3 Field Documentation

### 6.29.3.1 bottom\_side

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

### 6.29.3.2 left\_side

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

### 6.29.3.3 origin

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

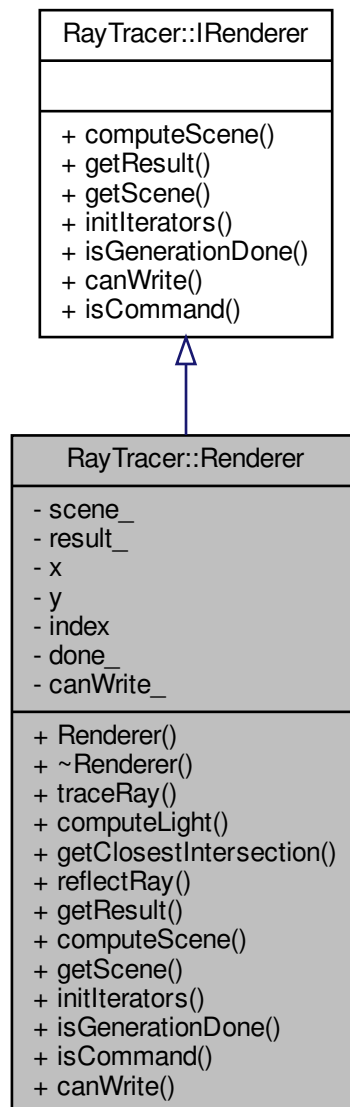
The documentation for this class was generated from the following files:

- include/maths/[Rectangle3D.hpp](#)
- src/maths/[Rectangle3D.cpp](#)

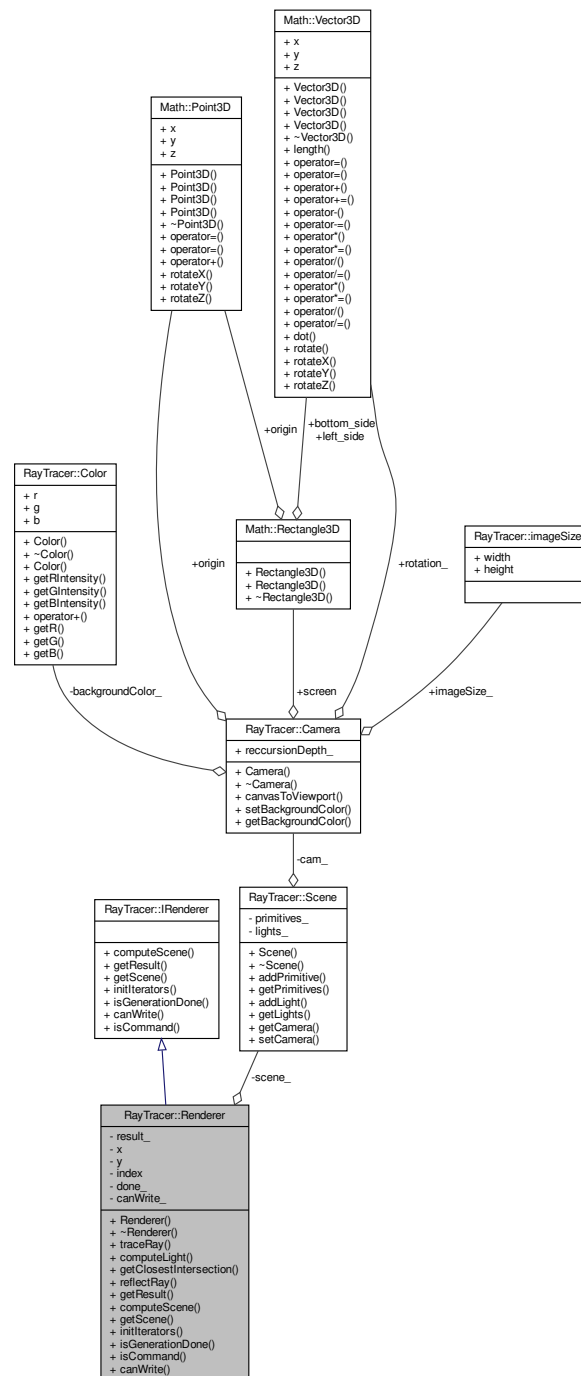
## 6.30 RayTracer::Renderer Class Reference

```
#include <Renderer.hpp>
```

Inheritance diagram for RayTracer::Renderer:



Collaboration diagram for RayTracer::Renderer:



## Public Member Functions

- `Renderer()`=default
- `~Renderer()`=default
- `Color traceRay (RayTracer::Ray ray, double tMin, double tMax, int recursion_depth)`
- `Color computeLight (Math::Point3D intersectionPoint, Math::Vector3D normalVector, Math::Vector3D rayDir, int spec)`

- `intersection getClosestIntersection` (`RayTracer::Ray` ray, double tMin, double tMax)
- `Math::Vector3D reflectRay` (`Math::Vector3D` normalVector, `Math::Vector3D` ray)
- `std::vector< std::vector< Color > > getResult` () override  
*Get the result of the rendering as a 2D vector of colors.*
- void `computeScene` () override  
*Compute the scene and render the result.*
- `Scene & getScene` () override  
*Get the scene used for rendering.*
- void `initIterators` () override  
*Initializes the iterators for generation.*
- bool `isGenerationDone` () override  
*Checks if the generation process is done.*
- bool `isCommand` () override  
*Checks if the current action is a command.*
- bool `canWrite` () override  
*Checks if writing is allowed.*

## Private Attributes

- `Scene scene_` {}
- `std::vector< std::vector< Color > > result_`
- double `x`
- double `y`
- `std::size_t index` = 0
- bool `done_` { false }
- bool `canWrite_` { false }

## 6.30.1 Constructor & Destructor Documentation

### 6.30.1.1 Renderer()

```
RayTracer::Renderer::Renderer ( ) [default]
```

### 6.30.1.2 ~Renderer()

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

## 6.30.2 Member Function Documentation

### 6.30.2.1 canWrite()

```
bool RayTracer::Renderer::canWrite ( ) [override], [virtual]
```

Checks if writing is allowed.

#### Returns

True if writing is allowed, False otherwise

Implements [RayTracer::IRenderer](#).

### 6.30.2.2 computeLight()

```
Color RayTracer::Renderer::computeLight (
    Math::Point3D intersectionPoint,
    Math::Vector3D normalVector,
    Math::Vector3D rayDir,
    int spec )
```

### 6.30.2.3 computeScene()

```
void RayTracer::Renderer::computeScene ( ) [override], [virtual]
```

Compute the scene and render the result.

Implements [RayTracer::IRenderer](#).

### 6.30.2.4 getClosestIntersection()

```
intersection RayTracer::Renderer::getClosestIntersection (
    RayTracer::Ray ray,
    double tMin,
    double tMax )
```

### 6.30.2.5 getResult()

```
std::vector< std::vector< Color > > RayTracer::Renderer::getResult ( ) [override], [virtual]
```

Get the result of the rendering as a 2D vector of colors.

#### Returns

The result of the rendering as a 2D vector of colors.

Implements [RayTracer::IRenderer](#).

#### 6.30.2.6 `getScene()`

```
Scene & RayTracer::Renderer::getScene ( ) [override], [virtual]
```

Get the scene used for rendering.

##### Returns

The scene used for rendering.

Implements [RayTracer::IRenderer](#).

#### 6.30.2.7 `initIterators()`

```
void RayTracer::Renderer::initIterators ( ) [override], [virtual]
```

Initializes the iterators for generation.

Implements [RayTracer::IRenderer](#).

#### 6.30.2.8 `isCommand()`

```
bool RayTracer::Renderer::isCommand ( ) [override], [virtual]
```

Checks if the current action is a command.

##### Returns

True if the current action is a command, False otherwise

Implements [RayTracer::IRenderer](#).

#### 6.30.2.9 `isGenerationDone()`

```
bool RayTracer::Renderer::isGenerationDone ( ) [override], [virtual]
```

Checks if the generation process is done.

##### Returns

True if the generation is done, False otherwise

Implements [RayTracer::IRenderer](#).



#### 6.30.2.10 reflectRay()

```
Math::Vector3D RayTracer::Renderer::reflectRay (
    Math::Vector3D normalVector,
    Math::Vector3D ray )
```

#### 6.30.2.11 traceRay()

```
Color RayTracer::Renderer::traceRay (
    RayTracer::Ray ray,
    double tMin,
    double tMax,
    int recursion_depth )
```

### 6.30.3 Field Documentation

#### 6.30.3.1 canWrite\_

```
bool RayTracer::Renderer::canWrite_ { false } [private]
```

#### 6.30.3.2 done\_

```
bool RayTracer::Renderer::done_ { false } [private]
```

#### 6.30.3.3 index

```
std::size_t RayTracer::Renderer::index = 0 [private]
```

#### 6.30.3.4 result\_

```
std::vector<std::vector<Color> > RayTracer::Renderer::result_ [private]
```

#### 6.30.3.5 scene\_

```
Scene RayTracer::Renderer::scene_ {} [private]
```

#### 6.30.3.6 x

```
double RayTracer::Renderer::x [private]
```

#### 6.30.3.7 y

```
double RayTracer::Renderer::y [private]
```

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

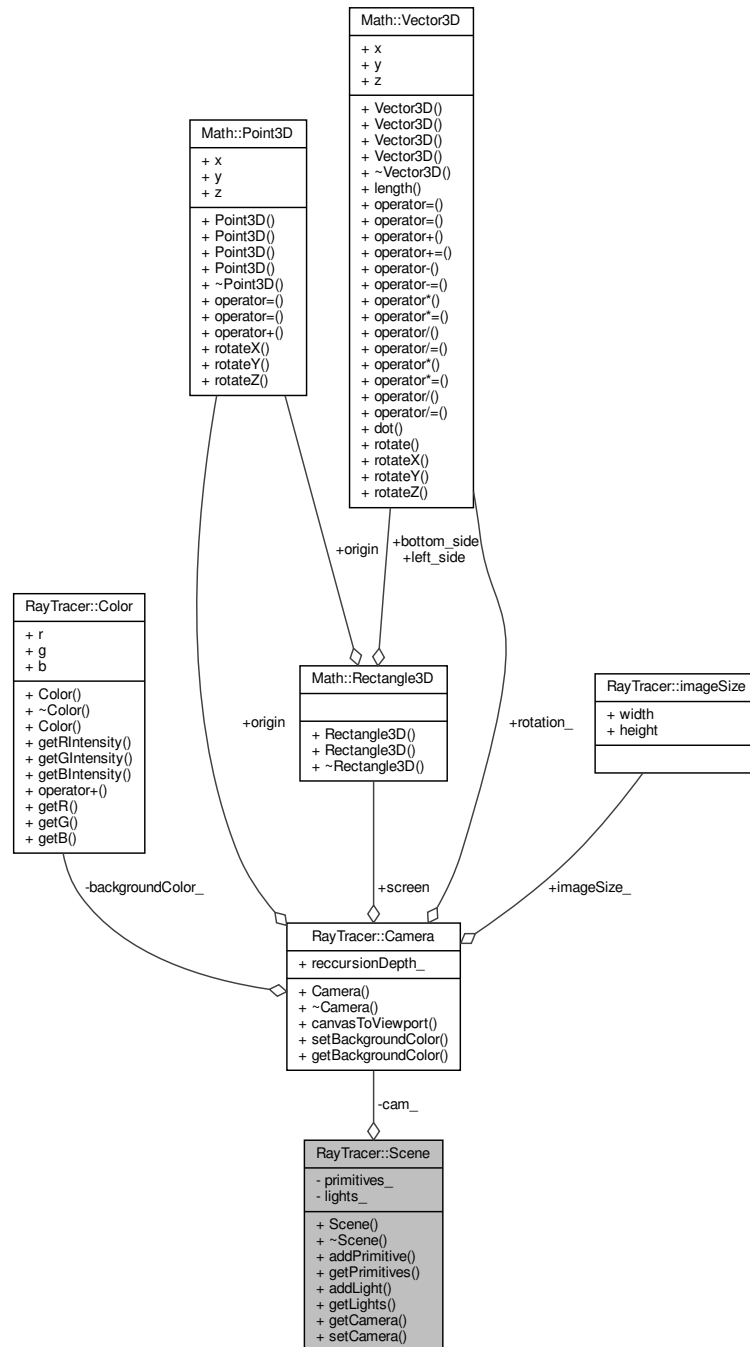
- src/plugins/renderer/baseRenderer/[Renderer.hpp](#)
- src/plugins/renderer/baseRenderer/[Renderer.cpp](#)

## 6.31 RayTracer::Scene Class Reference

Class for the scene.

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

Collaboration diagram for RayTracer::Scene:



## Public Member Functions

- `Scene()`=default
- `~Scene()`=default
- `void addPrimitive (std::unique_ptr< IPrimitives > prim)`  
*Adds a new primitive object to the scene.*
- `std::vector< std::unique_ptr< IPrimitives > > & getPrimitives ()`

- Returns a reference to the vector of primitives in the scene.*
- void `addLight` (std::unique\_ptr< [ILight](#) > light)
- Adds a new light source to the scene.*
- std::vector< std::unique\_ptr< [ILight](#) > > & `getLights` ()
- Returns a reference to the vector of light sources in the scene.*
- [Camera](#) & `getCamera` ()
- Returns a reference to the camera object in the scene.*
- void `setCamera` ([Camera](#) cam)
- Sets the camera object in the scene.*

## Private Attributes

- std::vector< std::unique\_ptr< [IPrimitives](#) > > `primitives_`
- std::vector< std::unique\_ptr< [ILight](#) > > `lights_`
- [Camera](#) `cam_` {}

### 6.31.1 Detailed Description

Class for the scene.

### 6.31.2 Constructor & Destructor Documentation

#### 6.31.2.1 Scene()

```
RayTracer::Scene::Scene ( ) [default]
```

#### 6.31.2.2 ~Scene()

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

### 6.31.3 Member Function Documentation

#### 6.31.3.1 addLight()

```
void RayTracer::Scene::addLight (
    std::unique_ptr< ILight > light )
```

Adds a new light source to the scene.

## Parameters

<i>light</i>	A unique pointer to the light source to add.
--------------	--

**6.31.3.2 addPrimitive()**

```
void RayTracer::Scene::addPrimitive (
    std::unique_ptr< IPrimitives > prim )
```

Adds a new primitive object to the scene.

## Parameters

<i>prim</i>	A unique pointer to the primitive object to add.
-------------	--

**6.31.3.3 getCamera()**

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

Returns a reference to the camera object in the scene.

## Returns

A reference to the camera object in the scene.

**6.31.3.4 getLights()**

```
std::vector< std::unique_ptr< ILight > > & RayTracer::Scene::getLights ( )
```

Returns a reference to the vector of light sources in the scene.

## Returns

A reference to the vector of unique pointers to the light sources in the scene.

### 6.31.3.5 getPrimitives()

```
std::vector< std::unique_ptr< IPrimitives > > & RayTracer::Scene::getPrimitives ( )
```

Returns a reference to the vector of primitives in the scene.

#### Returns

A reference to the vector of unique pointers to the primitives in the scene.

### 6.31.3.6 setCamera()

```
void RayTracer::Scene::setCamera (
    RayTracer::Camera cam )
```

Sets the camera object in the scene.

#### Parameters

<i>cam</i>	The new camera object to set.
------------	-------------------------------

## 6.31.4 Field Documentation

### 6.31.4.1 cam\_

```
Camera RayTracer::Scene::cam_ {} [private]
```

### 6.31.4.2 lights\_

```
std::vector<std::unique_ptr<ILight> > RayTracer::Scene::lights_ [private]
```

### 6.31.4.3 primitives\_

```
std::vector<std::unique_ptr<IPrimitives> > RayTracer::Scene::primitives_ [private]
```

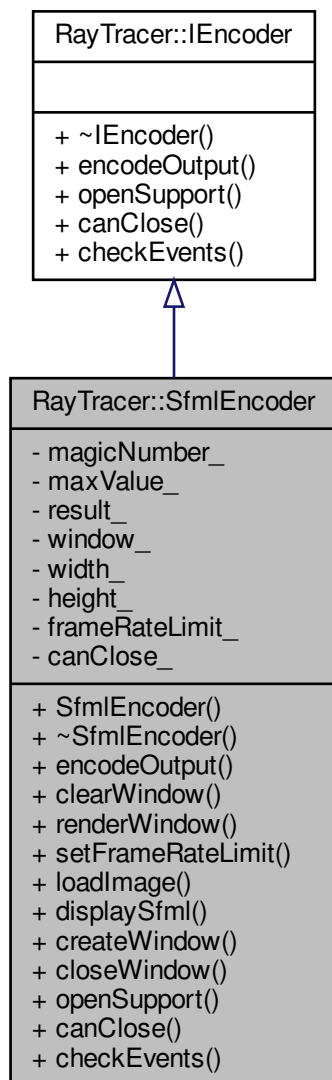
The documentation for this class was generated from the following files:

- include/Scene.hpp
- src/utils/Scene.cpp

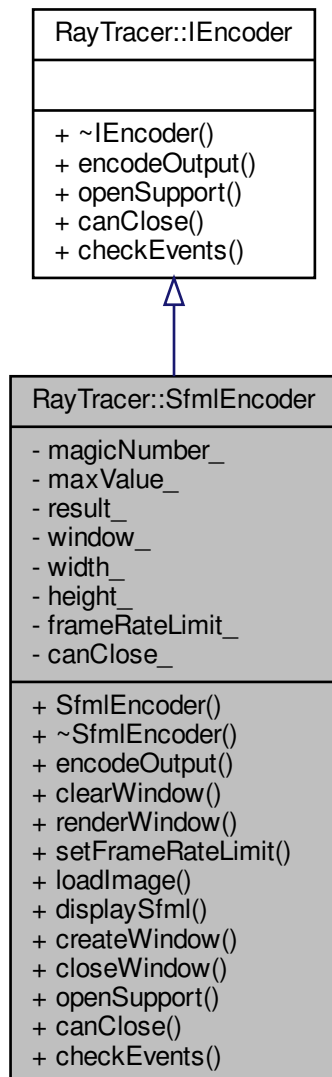
## 6.32 RayTracer::SfmlEncoder Class Reference

```
#include <Encoder.hpp>
```

Inheritance diagram for RayTracer::SfmlEncoder:



Collaboration diagram for RayTracer::SfmlEncoder:



## Data Structures

- struct [Image](#)

## Public Member Functions

- [SfmlEncoder](#) ()=default
- [~SfmlEncoder](#) () override=default
- void [encodeOutput](#) (std::vector< std::vector< [Color](#) >> [result\\_](#), const std::string &filename) override  
*Encode the output into a ppm image.*
- void [clearWindow](#) ()



- *Clear the window.*
- void `renderWindow ()`
- *Render the window.*
- void `setFrameRateLimit ()`
- *Set the Frame Rate Limit object.*
- sf::Image `loadImage (std::vector< std::vector< Color >> result)`
- *Draw the image.*
- void `displaySfml (std::vector< std::vector< Color >> result)`
- *Display the image.*
- void `createWindow (const std::string &name)`
- *Create a Window object.*
- void `closeWindow ()`
- *Close the window.*
- void `openSupport (const std::string &name, imageSize size) override`
- *Opens the necessary support for encoding.*
- bool `canClose () override`
- *Checks if the encoder can be closed.*
- void `checkEvents () override`
- *Checks for any pending events or actions.*

## Private Attributes

- std::string `magicNumber_ { HEADER_MAGIC_NUMBER "\n" }`
- *Magic number (P3) indicates that the colors are in ASCII and are in RGB format.*
- unsigned int `maxValue_ { MAX_VALUE }`
- *Maximum value of a color.*
- std::vector< std::vector< Color > > `result_`
- *Result of the raytracer calculations.*
- sf::RenderWindow `window_`
- long unsigned int `width_`
- long unsigned int `height_`
- unsigned int `frameRateLimit_ { FRAME_RATE_LIMIT }`
- bool `canClose_ { false }`

## 6.32.1 Constructor & Destructor Documentation

### 6.32.1.1 SfmlEncoder()

```
RayTracer::SfmlEncoder::SfmlEncoder ( ) [default]
```

### 6.32.1.2 ~SfmlEncoder()

```
RayTracer::SfmlEncoder::~~SfmlEncoder ( ) [override], [default]
```

## 6.32.2 Member Function Documentation

### 6.32.2.1 canClose()

```
bool RayTracer::SfmlEncoder::canClose ( ) [override], [virtual]
```

Checks if the encoder can be closed.

#### Returns

True if the encoder can be closed, False otherwise

Implements [RayTracer::IEncoder](#).

### 6.32.2.2 checkEvents()

```
void RayTracer::SfmlEncoder::checkEvents ( ) [override], [virtual]
```

Checks for any pending events or actions.

Implements [RayTracer::IEncoder](#).

### 6.32.2.3 clearWindow()

```
void RayTracer::SfmlEncoder::clearWindow ( )
```

Clear the window.

### 6.32.2.4 closeWindow()

```
void RayTracer::SfmlEncoder::closeWindow ( )
```

Close the window.

### 6.32.2.5 createWindow()

```
void RayTracer::SfmlEncoder::createWindow (
    const std::string & name )
```

Create a Window object.

### 6.32.2.6 displaySfml()

```
void RayTracer::SfmlEncoder::displaySfml (
    std::vector< std::vector< Color >> result )
```

Display the image.

## Parameters

<i>filename</i>	
-----------------	--

**6.32.2.7 encodeOutput()**

```
void RayTracer::SfmlEncoder::encodeOutput (
    std::vector< std::vector< Color >> result_,
    const std::string & filename ) [override], [virtual]
```

Encode the output into a ppm image.

## Parameters

<i>result_↩</i>	
<i>—</i>	
<i>filename</i>	

Implements [RayTracer::IEncoder](#).

**6.32.2.8 loadImage()**

```
sf::Image RayTracer::SfmlEncoder::loadImage (
    std::vector< std::vector< Color >> result )
```

Draw the image.

## Parameters

<i>filename</i>	
-----------------	--

**6.32.2.9 openSupport()**

```
void RayTracer::SfmlEncoder::openSupport (
    const std::string & name,
    imageSize size ) [override], [virtual]
```

Opens the necessary support for encoding.

## Parameters

<i>name</i>	The name of the support to be opened
<i>size</i>	The size of the image to be encoded

Implements [RayTracer::IEncoder](#).

#### 6.32.2.10 renderWindow()

```
void RayTracer::SfmlEncoder::renderWindow ( )
```

Render the window.

#### 6.32.2.11 setFrameRateLimit()

```
void RayTracer::SfmlEncoder::setFrameRateLimit ( )
```

Set the Frame Rate Limit object.

### 6.32.3 Field Documentation

#### 6.32.3.1 canClose\_

```
bool RayTracer::SfmlEncoder::canClose_ { false } [private]
```

#### 6.32.3.2 frameRateLimit\_

```
unsigned int RayTracer::SfmlEncoder::frameRateLimit_ { FRAME_RATE_LIMIT } [private]
```

#### 6.32.3.3 height\_

```
long unsigned int RayTracer::SfmlEncoder::height_ [private]
```

#### 6.32.3.4 magicNumber\_

```
std::string RayTracer::SfmlEncoder::magicNumber_ { HEADER_MAGIC_NUMBER "\n" } [private]
```

Magic number (P3) indicates that the colors are in ASCII and are in RGB format.

#### 6.32.3.5 maxValue\_

```
unsigned int RayTracer::SfmlEncoder::maxValue_ { MAX_VALUE } [private]
```

Maximum value of a color.

#### 6.32.3.6 result\_

```
std::vector<std::vector<Color> > RayTracer::SfmlEncoder::result_ [private]
```

Result of the raytracer calculations.

#### 6.32.3.7 width\_

```
long unsigned int RayTracer::SfmlEncoder::width_ [private]
```

#### 6.32.3.8 window\_

```
sf::RenderWindow RayTracer::SfmlEncoder::window_ [private]
```

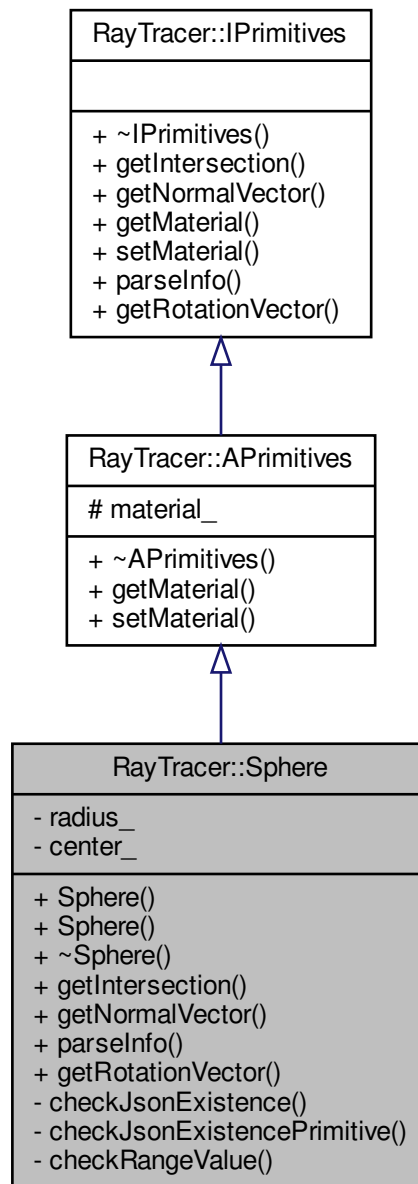
The documentation for this class was generated from the following files:

- src/plugins/encoder/sfmlEncoder/[Encoder.hpp](#)
- src/plugins/encoder/sfmlEncoder/[Encoder.cpp](#)

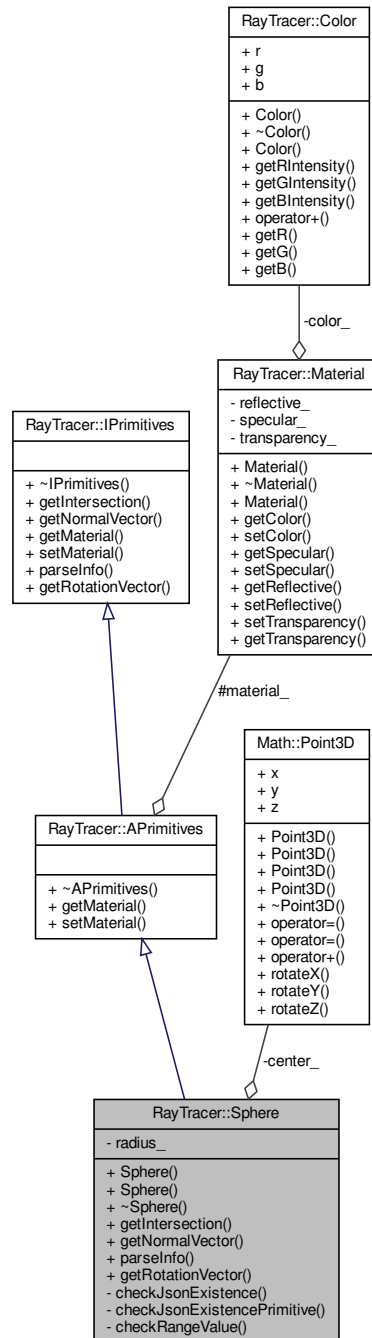
## 6.33 RayTracer::Sphere Class Reference

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

Inheritance diagram for RayTracer::Sphere:



Collaboration diagram for RayTracer::Sphere:



## Public Member Functions

- [Sphere](#) (double radius, [Math::Point3D](#) center)
- [Sphere](#) ()=default
- [~Sphere](#) () override=default
- `std::vector< double > getIntersection` ([RayTracer::Ray](#) ray) final
- [Math::Vector3D](#) `getNormalVector` ([Math::Point3D](#) point) final

- void [parseInfo](#) ([json](#) object) final
- [Math::Vector3D](#) [getRotationVector](#) ()

## Private Member Functions

- void [checkJsonExistence](#) (const [json](#) &scene, const std::string &field\_name)
- void [checkJsonExistencePrimitive](#) (const [json](#) &scene)
- void [checkRangeValue](#) (const [json](#) &scene, const std::string &field\_name, const std::string &comparison\_↔ sign, double value)

## Private Attributes

- double [radius\\_](#) { 0 }
- [Math::Point3D](#) [center\\_](#)

## Additional Inherited Members

### 6.33.1 Constructor & Destructor Documentation

#### 6.33.1.1 Sphere() [1/2]

```
RayTracer::Sphere::Sphere (
    double radius,
    Math::Point3D center )
```

#### 6.33.1.2 Sphere() [2/2]

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

#### 6.33.1.3 ~Sphere()

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

### 6.33.2 Member Function Documentation



### 6.33.2.1 checkJsonExistence()

```
void RayTracer::Sphere::checkJsonExistence (
    const json & scene,
    const std::string & field_name ) [private]
```

### 6.33.2.2 checkJsonExistencePrimitive()

```
void RayTracer::Sphere::checkJsonExistencePrimitive (
    const json & scene ) [private]
```

### 6.33.2.3 checkRangeValue()

```
void RayTracer::Sphere::checkRangeValue (
    const json & scene,
    const std::string & field_name,
    const std::string & comparison_sign,
    double value ) [private]
```

### 6.33.2.4 getIntersection()

```
std::vector< double > RayTracer::Sphere::getIntersection (
    RayTracer::Ray ray ) [final], [virtual]
```

Implements [RayTracer::IPrimitives](#).

### 6.33.2.5 getNormalVector()

```
Math::Vector3D RayTracer::Sphere::getNormalVector (
    Math::Point3D point ) [final], [virtual]
```

Implements [RayTracer::IPrimitives](#).

### 6.33.2.6 getRotationVector()

```
Math::Vector3D RayTracer::Sphere::getRotationVector ( ) [virtual]
```

Implements [RayTracer::IPrimitives](#).

### 6.33.2.7 parseInfo()

```
void RayTracer::Sphere::parseInfo (
    json object ) [final], [virtual]
```

Implements [RayTracer::IPrimitives](#).

## 6.33.3 Field Documentation

### 6.33.3.1 center\_

```
Math::Point3D RayTracer::Sphere::center_ [private]
```

### 6.33.3.2 radius\_

```
double RayTracer::Sphere::radius_ { 0 } [private]
```

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

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

## 6.34 Math::Vector3D Class Reference

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

Collaboration diagram for Math::Vector3D:

Math::Vector3D
+ x + y + z
+ Vector3D() + Vector3D() + Vector3D() + Vector3D() + ~Vector3D() + length() + operator=() + operator=() + operator+() + operator+=() + operator-() + operator-=() + operator*() + operator*=() + operator/() + operator/=() + operator*() + operator*=() + operator/() + operator/=() + dot() + rotate() + rotateX() + rotateY() + rotateZ()

## Public Member Functions

- [Vector3D](#) ()=default
- [Vector3D](#) (double [x](#), double [y](#), double [z](#))
- [Vector3D](#) ([Vector3D](#) &)=default
- [Vector3D](#) ([Vector3D](#) &&)=default
- [~Vector3D](#) ()=default
- double [length](#) () const
- [Vector3D](#) & [operator=](#) (const [Vector3D](#) &)=default
- [Vector3D](#) & [operator=](#) ([Vector3D](#) &&)=default
- [Vector3D](#) [operator+](#) (const [Vector3D](#) &)  
*Addition operator for vectors.*
- [Vector3D](#) & [operator+=](#) (const [Vector3D](#) &)  
*Addition operator for vectors with assignment.*
- [Vector3D](#) [operator-](#) (const [Vector3D](#) &)  
*Subtraction operator for vectors.*

- `Vector3D & operator-=` (const `Vector3D` &)  
*Subtraction operator for vectors with assignment.*
- `Vector3D operator*` (const `Vector3D` &)  
*Multiplication operator for vectors.*
- `Vector3D & operator*='` (const `Vector3D` &)  
*Multiplication operator for vectors with assignment.*
- `Vector3D operator/` (const `Vector3D` &)  
*Division operator for vectors.*
- `Vector3D & operator/=` (const `Vector3D` &)  
*Division operator for vectors with assignment.*
- `Vector3D operator*` (double)  
*Multiplication operator for scalar values.*
- `Vector3D & operator*='` (double)  
*Multiplication operator for scalar values with assignment.*
- `Vector3D operator/` (double)  
*Division operator for scalar values.*
- `Vector3D & operator/=` (double)  
*Division operator for scalar values with assignment.*
- `double dot` (const `Vector3D` &)  
*Calculates the dot product of this vector and another vector.*
- `void rotate` (double `x`, double `y`, double `z`)  
*Rotate the vector in the 3 directions.*
- `void rotateX` (double `x`)  
*Rotate the vector (origin and direction) on the x axis.*
- `void rotateY` (double `y`)  
*Rotate the vector (origin and direction) on the y axis.*
- `void rotateZ` (double `z`)  
*Rotate the vector (origin and direction) on the z axis.*

## Data Fields

- `double x` = 0
- `double y` = 0
- `double z` = 0

## 6.34.1 Constructor & Destructor Documentation

### 6.34.1.1 `Vector3D()` [1/4]

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

**6.34.1.2 Vector3D()** [2/4]

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

**6.34.1.3 Vector3D()** [3/4]

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

**6.34.1.4 Vector3D()** [4/4]

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

**6.34.1.5 ~Vector3D()**

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

**6.34.2 Member Function Documentation****6.34.2.1 dot()**

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

Calculates the dot product of this vector and another vector.

**Parameters**

<i>other</i>	The other vector to calculate the dot product with.
--------------	---

**Returns**

The dot product of the two vectors as a double value.

### 6.34.2.2 length()

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

### 6.34.2.3 operator\*() [1/2]

```
Vector3D Math::Vector3D::operator* (
    const Vector3D & vect )
```

Multiplication operator for vectors.

#### Parameters

<i>vector</i>	The vector to multiply.
---------------	-------------------------

#### Returns

A new vector resulting from the multiplication of the two vectors.

### 6.34.2.4 operator\*() [2/2]

```
Vector3D Math::Vector3D::operator* (
    double value )
```

Multiplication operator for scalar values.

#### Parameters

<i>scalar</i>	The scalar value to multiply.
---------------	-------------------------------

#### Returns

A new vector resulting from the multiplication of the vector by the scalar.

### 6.34.2.5 operator\*=( ) [1/2]

```
Vector3D & Math::Vector3D::operator*= (
    const Vector3D & vect )
```

Multiplication operator for vectors with assignment.

## Parameters

<i>vector</i>	The vector to multiply.
---------------	-------------------------

## Returns

A reference to the current vector after multiplying the vector passed as a parameter.

**6.34.2.6 operator\*=( ) [2/2]**

```
Vector3D & Math::Vector3D::operator*= (
    double value )
```

Multiplication operator for scalar values with assignment.

## Parameters

<i>scalar</i>	The scalar value to multiply.
---------------	-------------------------------

## Returns

A reference to the current vector after multiplying the vector by the scalar value.

**6.34.2.7 operator+( )**

```
Vector3D Math::Vector3D::operator+ (
    const Vector3D & vect )
```

Addition operator for vectors.

## Parameters

<i>vector</i>	The vector to add.
---------------	--------------------

## Returns

A new vector resulting from the addition of the two vectors.

**6.34.2.8 operator+=( )**

```
Vector3D & Math::Vector3D::operator+= (
    const Vector3D & vect )
```

Addition operator for vectors with assignment.



## Parameters

<i>vector</i>	The vector to add.
---------------	--------------------

## Returns

A reference to the current vector after adding the vector passed as a parameter.

**6.34.2.9 operator-()**

```
Vector3D Math::Vector3D::operator- (
    const Vector3D & vect )
```

Subtraction operator for vectors.

## Parameters

<i>vector</i>	The vector to subtract.
---------------	-------------------------

## Returns

A new vector resulting from the subtraction of the two vectors.

**6.34.2.10 operator-=()**

```
Vector3D & Math::Vector3D::operator-= (
    const Vector3D & vect )
```

Subtraction operator for vectors with assignment.

## Parameters

<i>vector</i>	The vector to subtract.
---------------	-------------------------

## Returns

A reference to the current vector after subtracting the vector passed as a parameter.

**6.34.2.11 operator/()** [1/2]

```
Vector3D Math::Vector3D::operator/ (
    const Vector3D & vect )
```

Division operator for vectors.

## Parameters

<i>vector</i>	The vector to divide.
---------------	-----------------------

## Returns

A new vector resulting from the division of the two vectors.

**6.34.2.12 operator/()** [2/2]

```
Vector3D Math::Vector3D::operator/ (
    double value )
```

Division operator for scalar values.

## Parameters

<i>scalar</i>	The scalar value to divide.
---------------	-----------------------------

## Returns

A new vector resulting from the division of the vector by the scalar.

**6.34.2.13 operator/=()** [1/2]

```
Vector3D & Math::Vector3D::operator/= (
    const Vector3D & vect )
```

Division operator for vectors with assignment.

## Parameters

<i>vector</i>	The vector to divide.
---------------	-----------------------

## Returns

A reference to the current vector after dividing the vector passed as a parameter.

**6.34.2.14 operator/=()** [2/2]

```
Vector3D & Math::Vector3D::operator/= (
    double value )
```

Division operator for scalar values with assignment.

## Parameters

<i>scalar</i>	The scalar value to divide.
---------------	-----------------------------

## Returns

A reference to the current vector after dividing the vector by the scalar value.

**6.34.2.15 operator=()** [1/2]

```
Vector3D& Math::Vector3D::operator= (
    const Vector3D & ) [default]
```

**6.34.2.16 operator=()** [2/2]

```
Vector3D& Math::Vector3D::operator= (
    Vector3D && ) [default]
```

**6.34.2.17 rotate()**

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

Rotate the vector in the 3 directions.

## Parameters

<i>x</i>	The x angle in radian
<i>y</i>	The y angle in radian
<i>z</i>	The z angle in radian

**6.34.2.18 rotateX()**

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

Rotate the vector (origin and direction) on the x axis.

**Parameters**

<i>x</i>	The x angle in radian
----------	-----------------------

**6.34.2.19 rotateY()**

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

Rotate the vector (origin and direction) on the y axis.

**Parameters**

<i>y</i>	The y angle in radian
----------	-----------------------

**6.34.2.20 rotateZ()**

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

Rotate the vector (origin and direction) on the z axis.

**Parameters**

<i>z</i>	The z angle in radian
----------	-----------------------

**6.34.3 Field Documentation****6.34.3.1 x**

```
double Math::Vector3D::x = 0
```

**6.34.3.2 y**

```
double Math::Vector3D::y = 0
```

### 6.34.3.3 z

```
double Math::Vector3D::z = 0
```

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

- [include/maths/Vector3D.hpp](#)
- [src/maths/Vector3D.cpp](#)





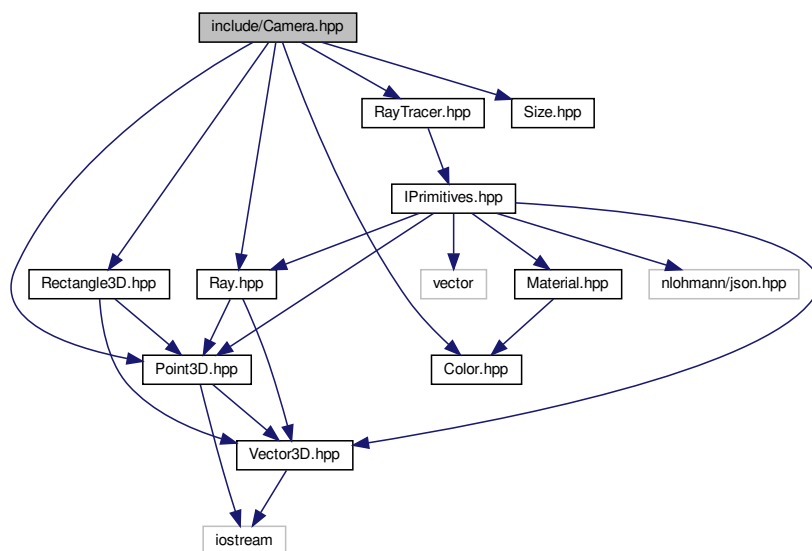
## Chapter 7

# File Documentation

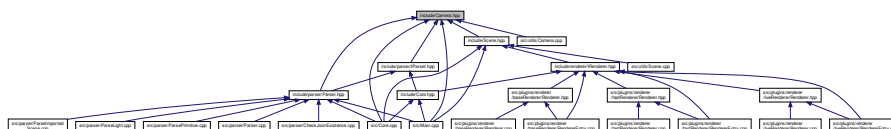
### 7.1 include/Camera.hpp File Reference

```
#include "Color.hpp"  
#include "Point3D.hpp"  
#include "Ray.hpp"  
#include "RayTracer.hpp"  
#include "Rectangle3D.hpp"  
#include "Size.hpp"
```

Include dependency graph for Camera.hpp:

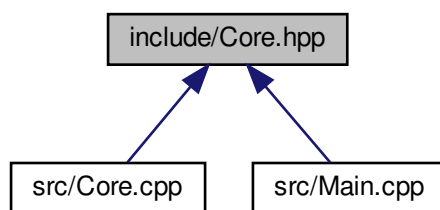


This graph shows which files directly or indirectly include this file:





This graph shows which files directly or indirectly include this file:



## Data Structures

- class [RayTracer::Core](#)

## Namespaces

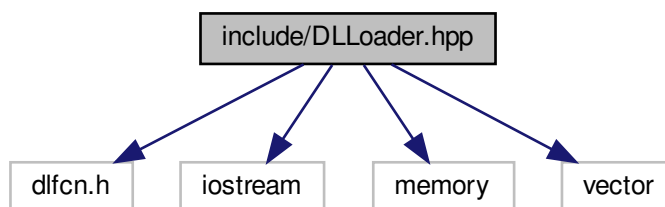
- [RayTracer](#)

*Namespace for the raytracer.*

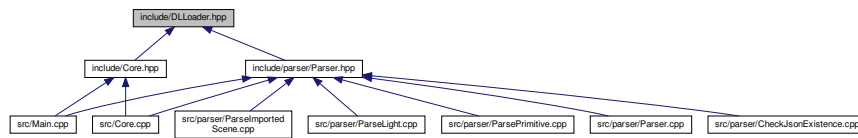
## 7.3 include/DLLoader.hpp File Reference

```
#include <dlfcn.h>
#include <iostream>
#include <memory>
#include <vector>
```

Include dependency graph for `DLLoader.hpp`:



This graph shows which files directly or indirectly include this file:



## Data Structures

- class [DLLoader< T >](#)

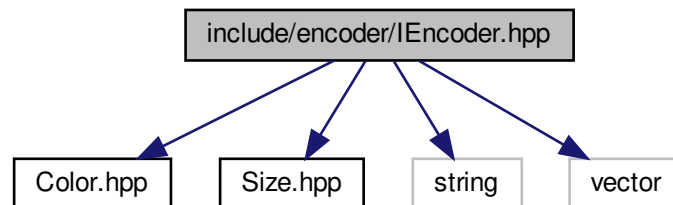
## 7.4 include/encoder/IEncoder.hpp File Reference

```

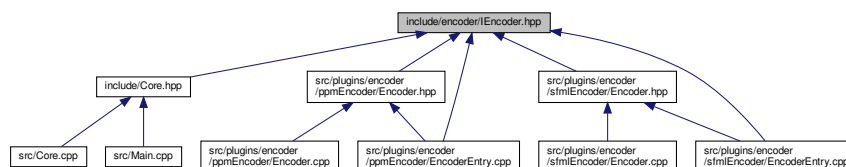
#include "Color.hpp"
#include "Size.hpp"
#include <string>
#include <vector>

```

Include dependency graph for IEncoder.hpp:



This graph shows which files directly or indirectly include this file:



## Data Structures

- class [RayTracer::IEncoder](#)

## Namespaces

- [RayTracer](#)

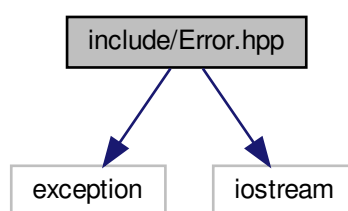
*Namespace for the raytracer.*

## 7.5 include/Error.hpp File Reference

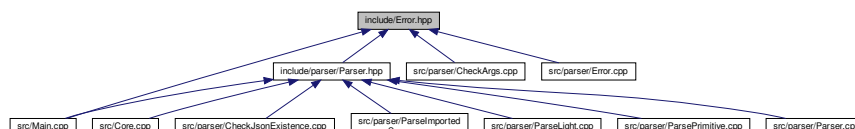
```
#include <exception>
```

```
#include <iostream>
```

Include dependency graph for Error.hpp:



This graph shows which files directly or indirectly include this file:



## Data Structures

- class [ParsingError](#)

## Macros

- `#define` [EPITECH\\_ERROR](#) 84

### 7.5.1 Macro Definition Documentation

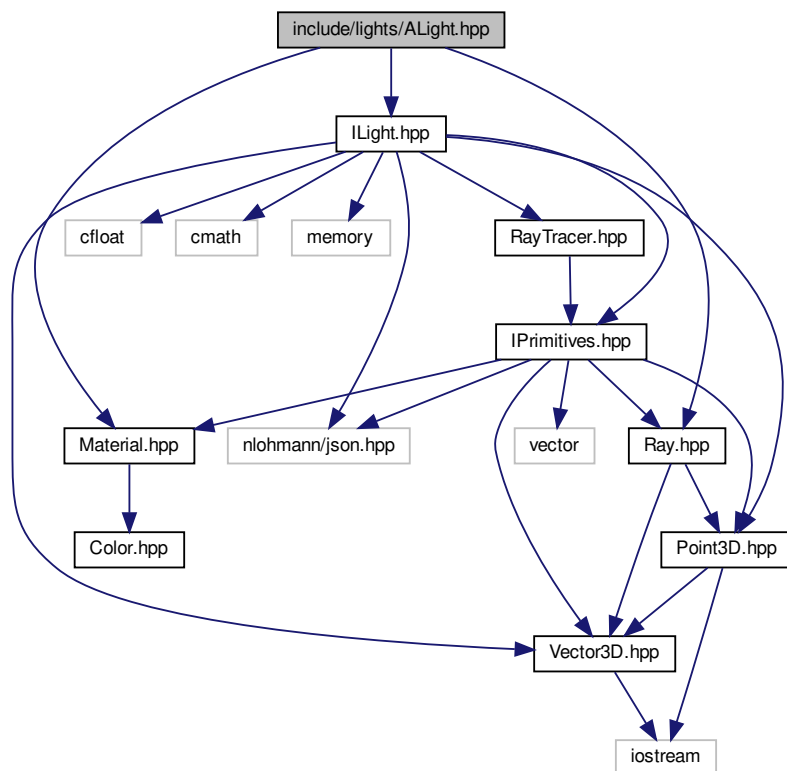
### 7.5.1.1 EPITECH\_ERROR

```
#define EPITECH_ERROR 84
```

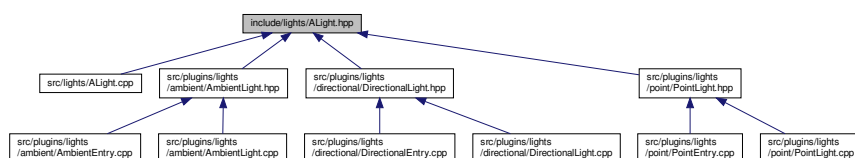
## 7.6 include/lights/ALight.hpp File Reference

```
#include "ILight.hpp"
#include "Material.hpp"
#include "Ray.hpp"
```

Include dependency graph for ALight.hpp:



This graph shows which files directly or indirectly include this file:



## Data Structures

- class [RayTracer::ALight](#)

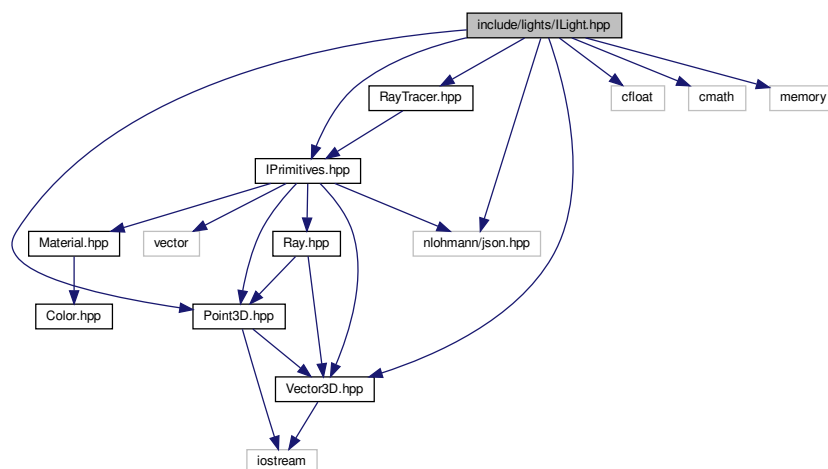
## Namespaces

- [RayTracer](#)

*Namespace for the raytracer.*

## 7.7 include/lights/ILight.hpp File Reference

```
#include "IPrimitives.hpp"
#include "Point3D.hpp"
#include "RayTracer.hpp"
#include "Vector3D.hpp"
#include <cfloat>
#include <cmath>
#include <memory>
#include <nlohmann/json.hpp>
Include dependency graph for ILight.hpp:
```



This graph shows which files directly or indirectly include this file:



## Data Structures

- class [RayTracer::ILight](#)

## Namespaces

- [RayTracer](#)

*Namespace for the raytracer.*

## Typedefs

- using [json](#) = `nlohmann::json`

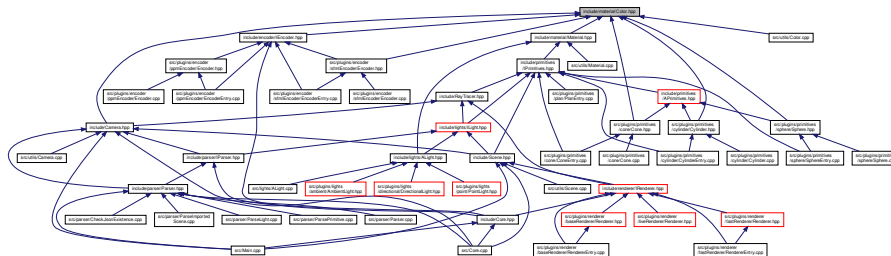
### 7.7.1 Typedef Documentation

#### 7.7.1.1 json

```
using json = nlohmann::json
```

## 7.8 include/material/Color.hpp File Reference

This graph shows which files directly or indirectly include this file:



## Data Structures

- class [RayTracer::Color](#)

## Namespaces

- [RayTracer](#)

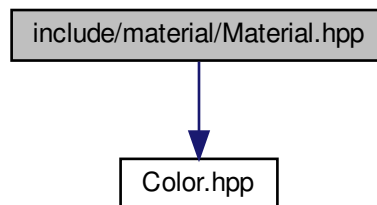
*Namespace for the raytracer.*



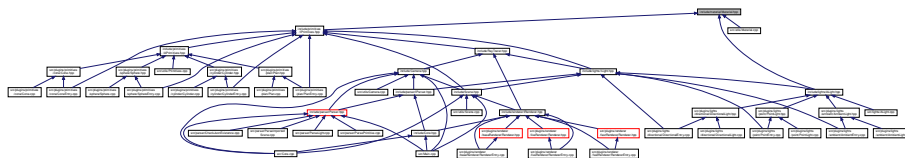
## 7.9 include/material/Material.hpp File Reference

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

Include dependency graph for Material.hpp:



This graph shows which files directly or indirectly include this file:



### Data Structures

- class [RayTracer::Material](#)  
*Material class representing the properties of a surface of an object.*

### Namespaces

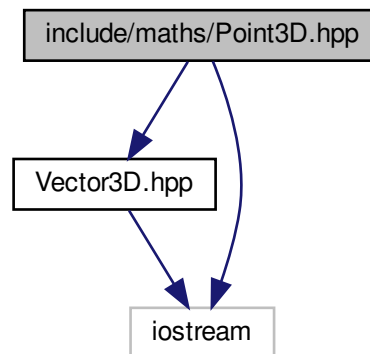
- [RayTracer](#)  
*Namespace for the raytracer.*

## 7.10 include/maths/Point3D.hpp File Reference

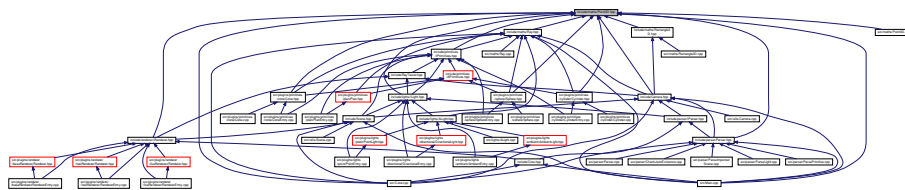
```
#include "Vector3D.hpp"
```

```
#include <iostream>
```

Include dependency graph for Point3D.hpp:



This graph shows which files directly or indirectly include this file:



## Data Structures

- class [Math::Point3D](#)  
*Class for the 3D point.*

## Namespaces

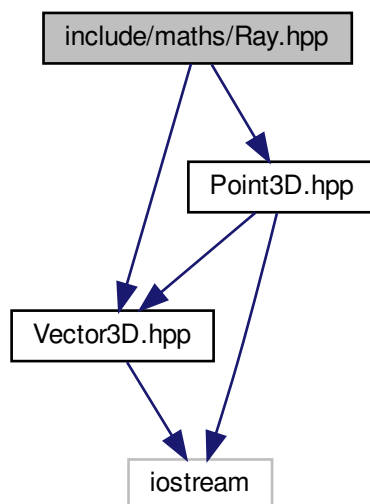
- [Math](#)  
*Namespace for the math functions.*

## Functions

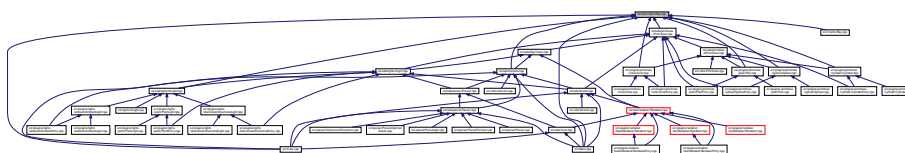
- `std::ostream & Math::operator<< (std::ostream &os, const Point3D &vect)`  
*Outputs a human-readable representation of the given point to the given output stream.*

## 7.11 include/maths/Ray.hpp File Reference

```
#include "Point3D.hpp"
#include "Vector3D.hpp"
Include dependency graph for Ray.hpp:
```



This graph shows which files directly or indirectly include this file:



### Data Structures

- class [RayTracer::Ray](#)  
*Class for the ray.*

### Namespaces

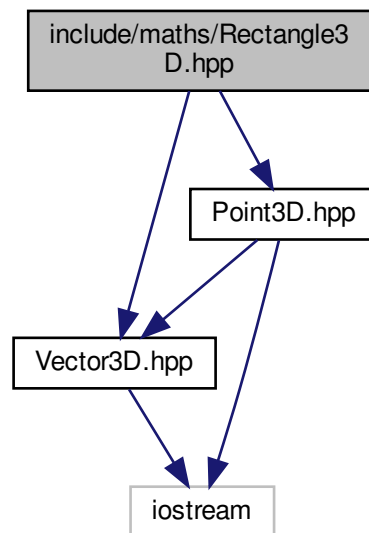
- [RayTracer](#)  
*Namespace for the raytracer.*

## 7.12 include/maths/Rectangle3D.hpp File Reference

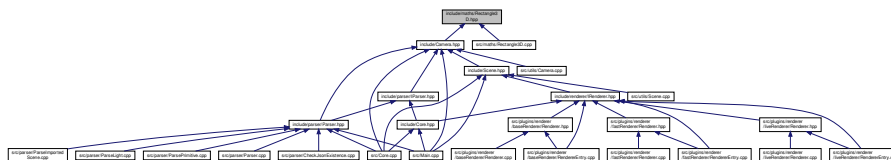
```
#include "Point3D.hpp"
```

```
#include "Vector3D.hpp"
```

Include dependency graph for Rectangle3D.hpp:



This graph shows which files directly or indirectly include this file:



### Data Structures

- class [Math::Rectangle3D](#)  
*Class for the 3D rectangle.*

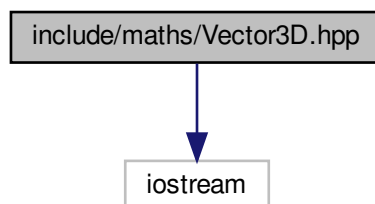
### Namespaces

- [Math](#)  
*Namespace for the math functions.*

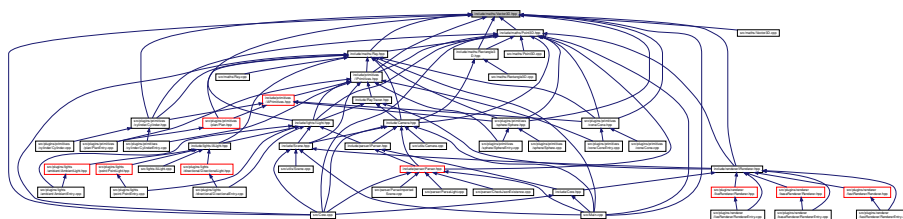
## 7.13 include/maths/Vector3D.hpp File Reference

```
#include <iostream>
```

Include dependency graph for Vector3D.hpp:



This graph shows which files directly or indirectly include this file:



### Data Structures

- class [Math::Vector3D](#)

### Namespaces

- [Math](#)  
*Namespace for the math functions.*

### Functions

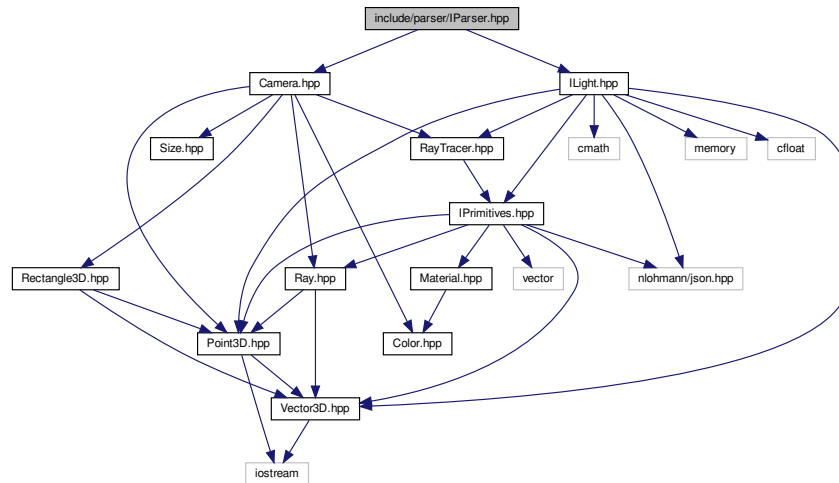
- `std::ostream & Math::operator<< (std::ostream &os, const Vector3D &vect)`  
*Calculates the cross product of two vectors.*

## 7.14 include/parser/IParser.hpp File Reference

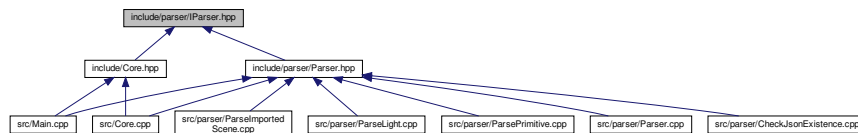
```
#include "Camera.hpp"
```

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

Include dependency graph for IParser.hpp:



This graph shows which files directly or indirectly include this file:



## Data Structures

- class [RayTracer::IParser](#)

## Namespaces

- [RayTracer](#)

*Namespace for the raytracer.*

## 7.15 include/parser/Parser.hpp File Reference

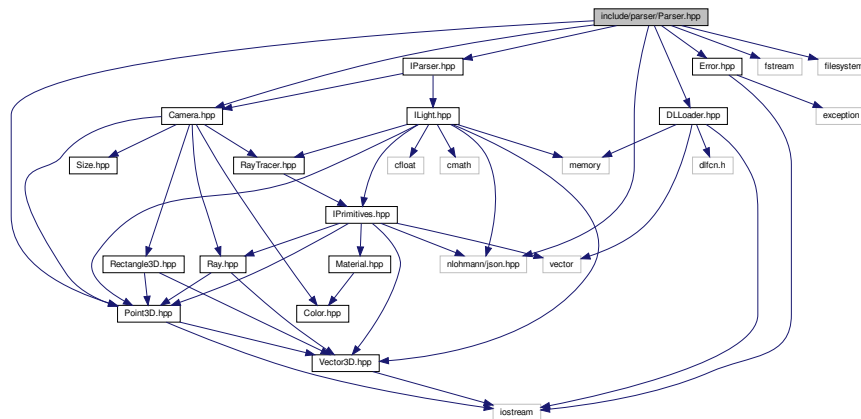
```
#include "Camera.hpp"
```

```
#include "DLoader.hpp"
```

```
#include "Error.hpp"
```

```
#include "IParser.hpp"
```

```
#include "Point3D.hpp"
#include "fstream"
#include <filesystem>
#include <nlohmann/json.hpp>
Include dependency graph for Parser.hpp:
```



This graph shows which files directly or indirectly include this file:



## Data Structures

- class `RayTracer::Parser`  
*Class for the parser.*

## Namespaces

- RayTracer
  - Namespace for the raytracer.*

## Typedefs

- using `json` = `nlohmann::json`

### 7.15.1 Typedef Documentation

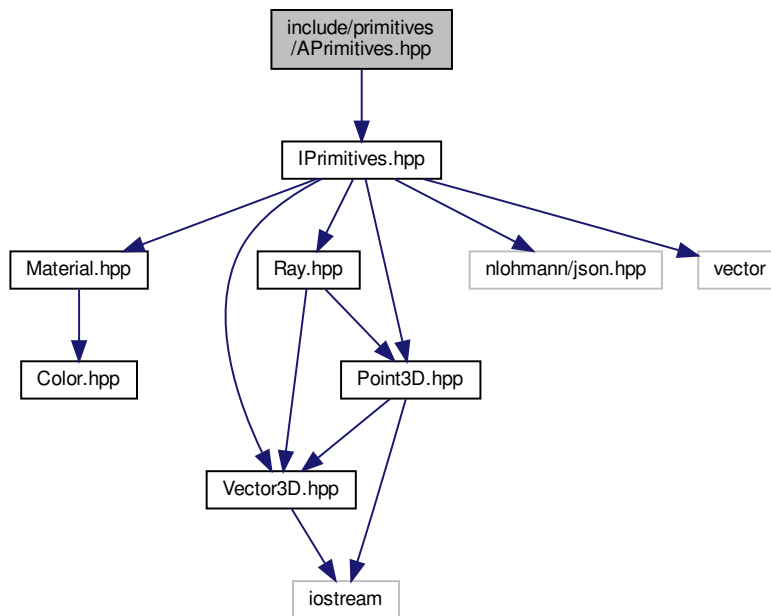
### 7.15.1.1 json

```
using json = nlohmann::json
```

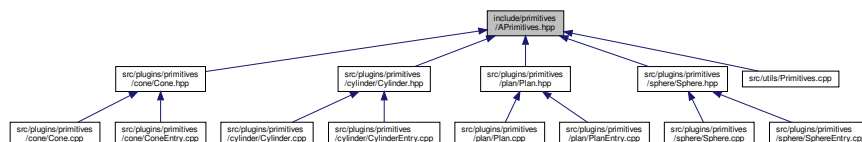
## 7.16 include/primitives/APrimitives.hpp File Reference

```
#include "IPrimitives.hpp"
```

Include dependency graph for APrimitives.hpp:



This graph shows which files directly or indirectly include this file:



## Data Structures

- class `RayTracer::APrimitives`

## Namespaces

- `RayTracer`

*Namespace for the raytracer.*



## Macros

- `#define PI_RAD 3.14159265 / 180`

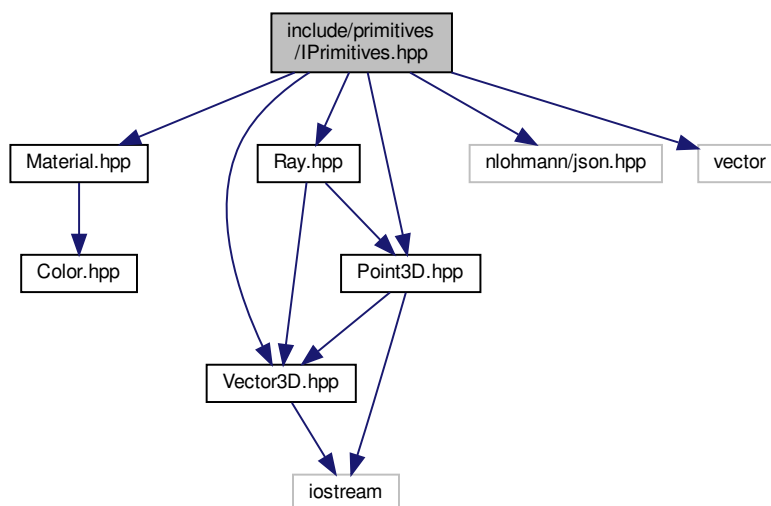
### 7.16.1 Macro Definition Documentation

#### 7.16.1.1 PI\_RAD

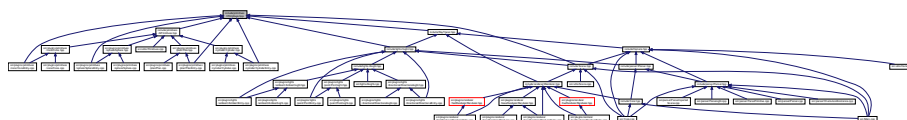
```
#define PI_RAD 3.14159265 / 180
```

## 7.17 include/primitives/IPrimitives.hpp File Reference

```
#include "Material.hpp"
#include "Point3D.hpp"
#include "Ray.hpp"
#include "Vector3D.hpp"
#include <nlohmann/json.hpp>
#include <vector>
Include dependency graph for IPrimitives.hpp:
```



This graph shows which files directly or indirectly include this file:



## Data Structures

- class [RayTracer::IPrimitives](#)

## Namespaces

- [RayTracer](#)  
*Namespace for the raytracer.*

## Typedefs

- using [json](#) = nlohmann::json

### 7.17.1 Typedef Documentation

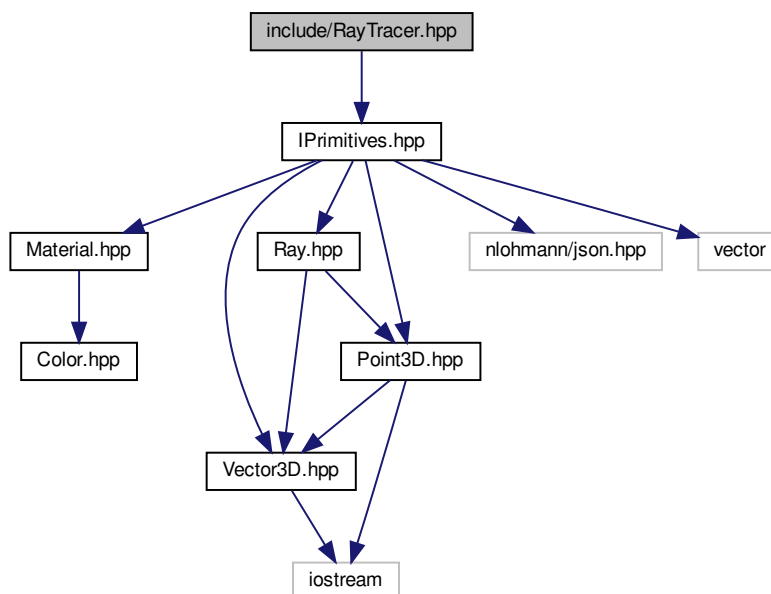
#### 7.17.1.1 json

```
using json = nlohmann::json
```

## 7.18 include/RayTracer.hpp File Reference

```
#include "IPrimitives.hpp"
```

Include dependency graph for RayTracer.hpp:

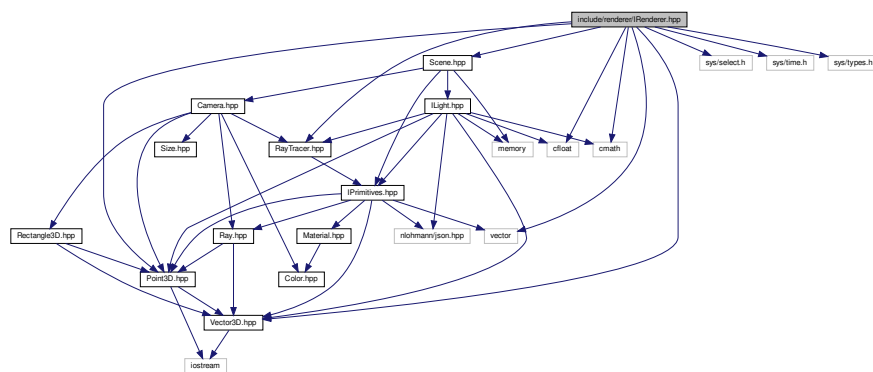


- struct RayTracer::intersection  
*Struct for the intersection.*

- RayTracer
  - Namespace for the raytracer.*

```
#include "Point3D.hpp"
#include "RayTracer.hpp"
#include "Scene.hpp"
#include "Vector3D.hpp"
#include <cfloat>
#include <cmath>
#include <sys/select.h>
#include <sys/time.h>
#include <sys/types.h>
#include <vector>
```

**Include dependency graph for IRenderer.hpp:**

[illegible]

## Data Structures

- class [RayTracer::IRenderer](#)

## Namespaces

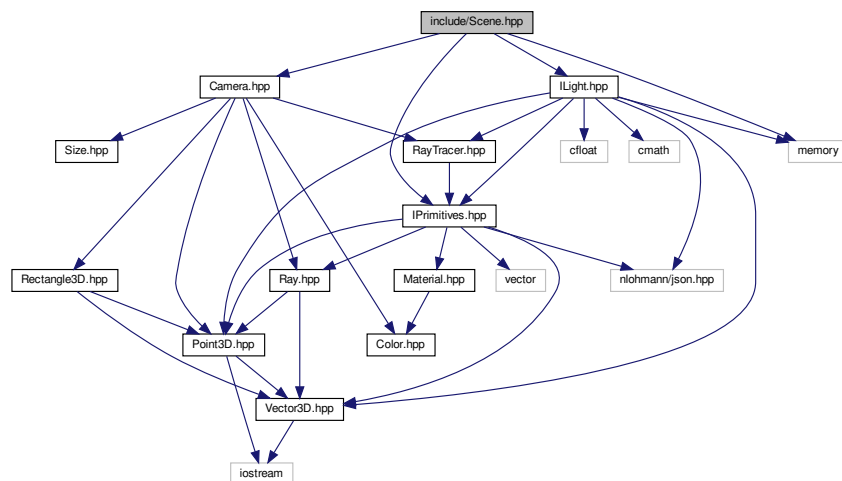
- [RayTracer](#)

*Namespace for the raytracer.*

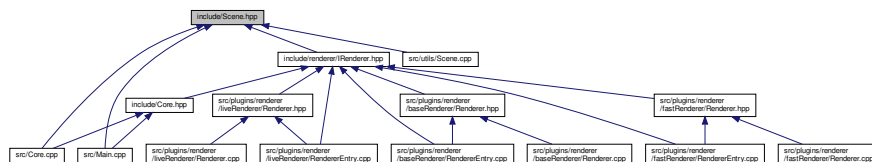
## 7.20 include/Scene.hpp File Reference

```
#include "Camera.hpp"
#include "IPrimitives.hpp"
#include "ILight.hpp"
#include <memory>
```

Include dependency graph for Scene.hpp:



This graph shows which files directly or indirectly include this file:



## Data Structures

- class [RayTracer::Scene](#)  
*Class for the scene.*



## Namespaces

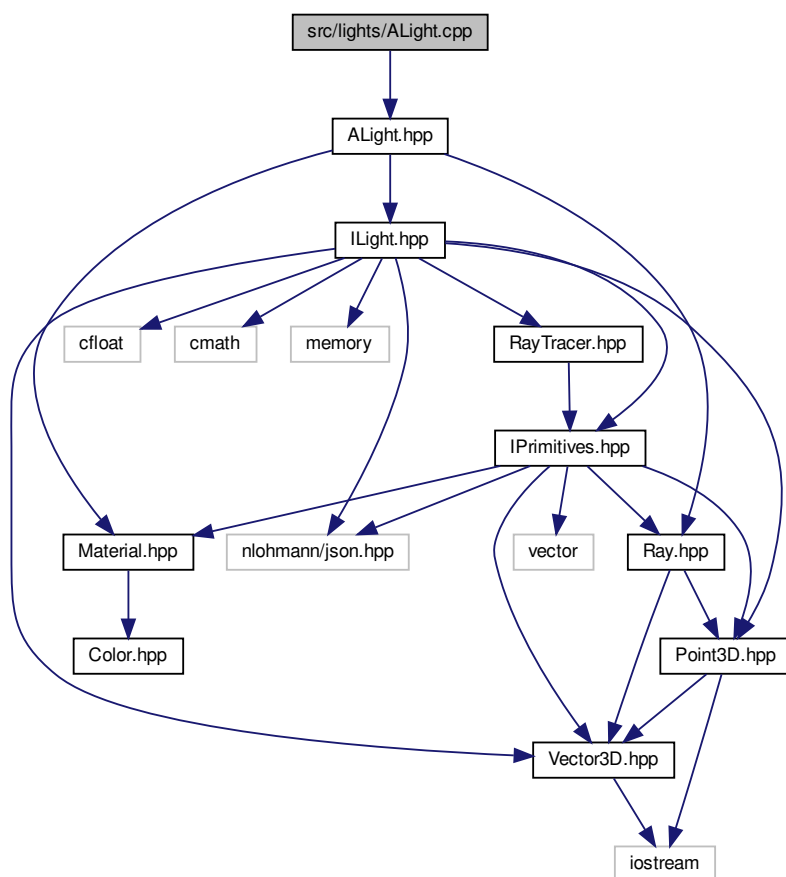
- [RayTracer](#)

*Namespace for the raytracer.*

## 7.23 src/lights/ALight.cpp File Reference

```
#include "ALight.hpp"
```

Include dependency graph for ALight.cpp:



## Namespaces

- [RayTracer](#)

*Namespace for the raytracer.*

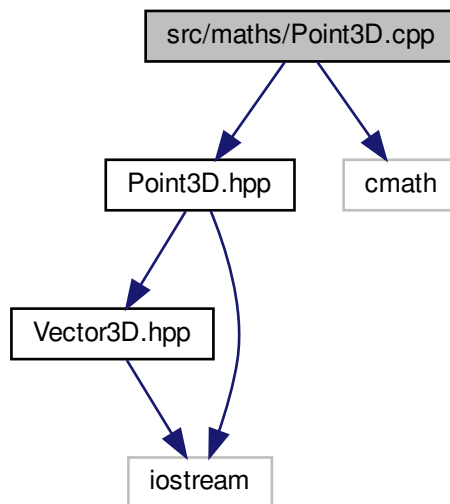


## 7.25 src/maths/Point3D.cpp File Reference

```
#include "Point3D.hpp"
```

```
#include <cmath>
```

Include dependency graph for Point3D.cpp:



### Namespaces

- [Math](#)

*Namespace for the math functions.*

### Functions

- `std::ostream & Math::operator<< (std::ostream &os, const Point3D &vect)`

*Outputs a human-readable representation of the given point to the given output stream.*

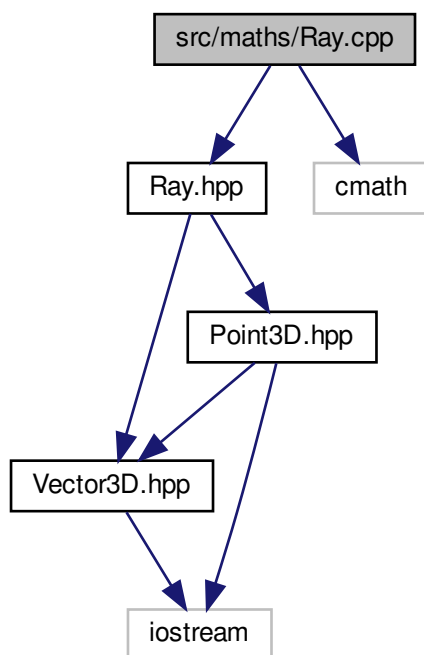
## 7.26 src/maths/Ray.cpp File Reference

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

```
#include <cmath>
```



Include dependency graph for Ray.cpp:



## Namespaces

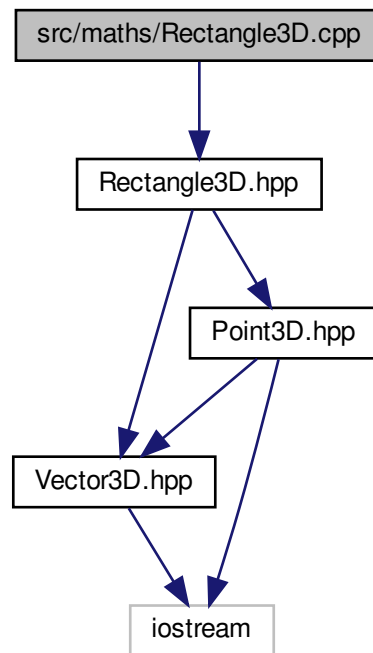
- [RayTracer](#)

*Namespace for the raytracer.*

## 7.27 src/maths/Rectangle3D.cpp File Reference

```
#include "Rectangle3D.hpp"
```

Include dependency graph for Rectangle3D.cpp:



## Namespaces

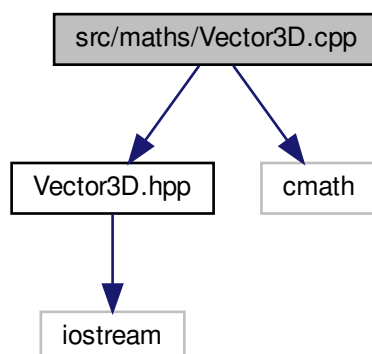
- [Math](#)

*Namespace for the math functions.*

## 7.28 src/maths/Vector3D.cpp File Reference

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

Include dependency graph for Vector3D.cpp:



## Namespaces

- [Math](#)

*Namespace for the math functions.*

## Functions

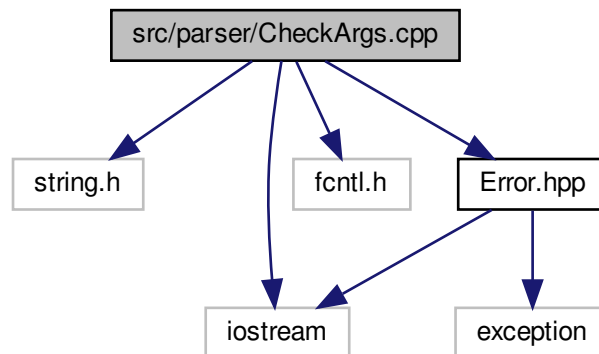
- `std::ostream & Math::operator<< (std::ostream &os, const Vector3D &vect)`

*Calculates the cross product of two vectors.*

## 7.29 src/parser/CheckArgs.cpp File Reference

```
#include <string.h>
#include <iostream>
#include <fcntl.h>
#include "Error.hpp"
```

Include dependency graph for CheckArgs.cpp:



## Functions

- void `displayUsage` ()
- int `checkArgs` (int ac, const char \*av[])

### 7.29.1 Function Documentation

#### 7.29.1.1 `checkArgs()`

```
int checkArgs (  
    int ac,  
    const char * av[] )
```

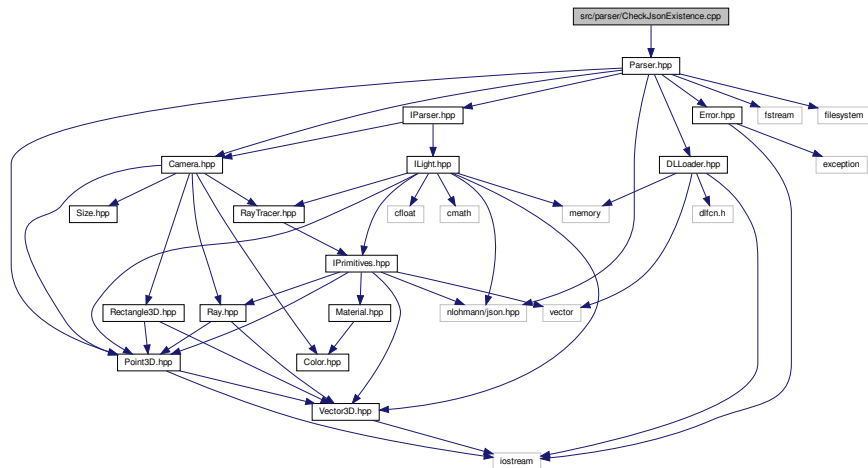
#### 7.29.1.2 `displayUsage()`

```
void displayUsage ( )
```

## 7.30 src/parser/CheckJsonExistence.cpp File Reference

```
#include "Parser.hpp"
```

Include dependency graph for CheckJsonExistence.cpp:



### Namespaces

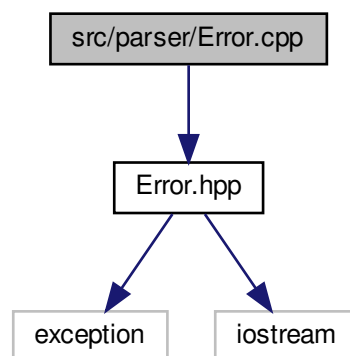
- [RayTracer](#)

*Namespace for the raytracer.*

## 7.31 src/parser/Error.cpp File Reference

```
#include "Error.hpp"
```

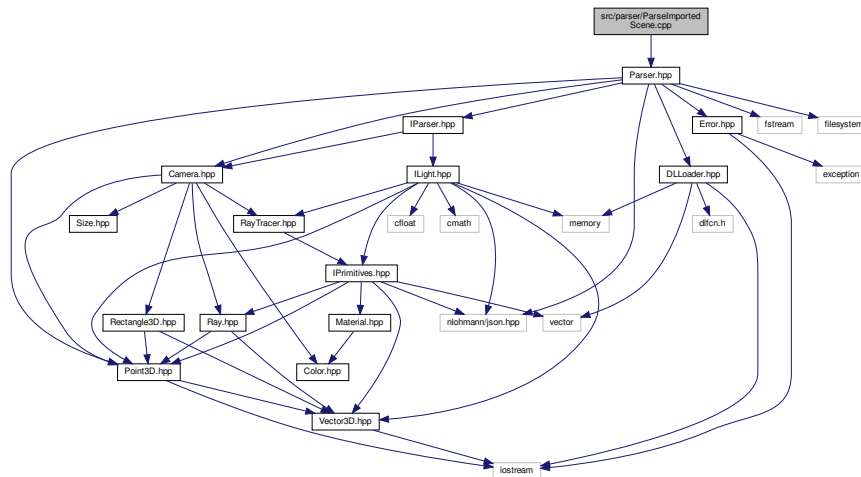
Include dependency graph for Error.cpp:



## 7.32 src/parser/ParseImportedScene.cpp File Reference

```
#include "Parser.hpp"
```

Include dependency graph for ParseImportedScene.cpp:



### Namespaces

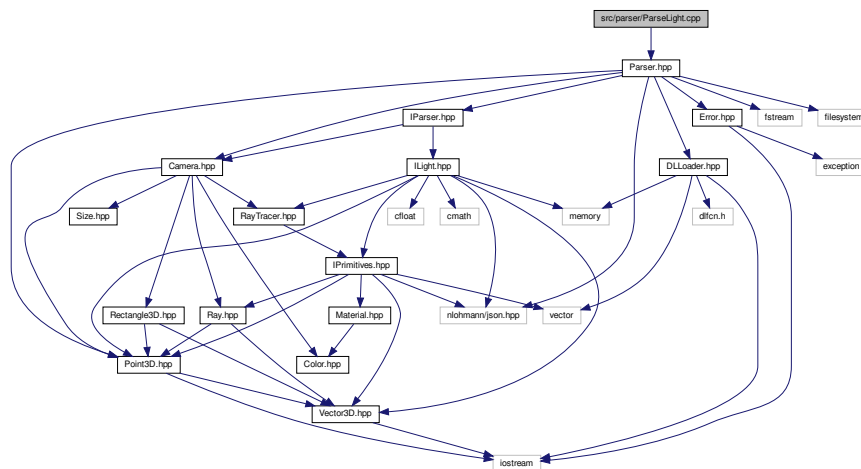
- [RayTracer](#)

*Namespace for the raytracer.*

## 7.33 src/parser/ParseLight.cpp File Reference

```
#include "Parser.hpp"
```

Include dependency graph for ParseLight.cpp:



## Namespaces

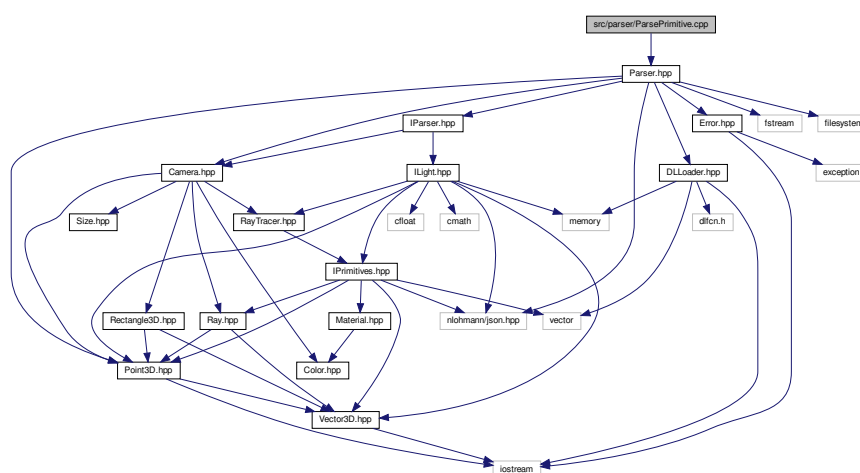
- [RayTracer](#)

*Namespace for the raytracer.*

## 7.34 src/parser/ParsePrimitive.cpp File Reference

```
#include "Parser.hpp"
```

Include dependency graph for ParsePrimitive.cpp:



## Namespaces

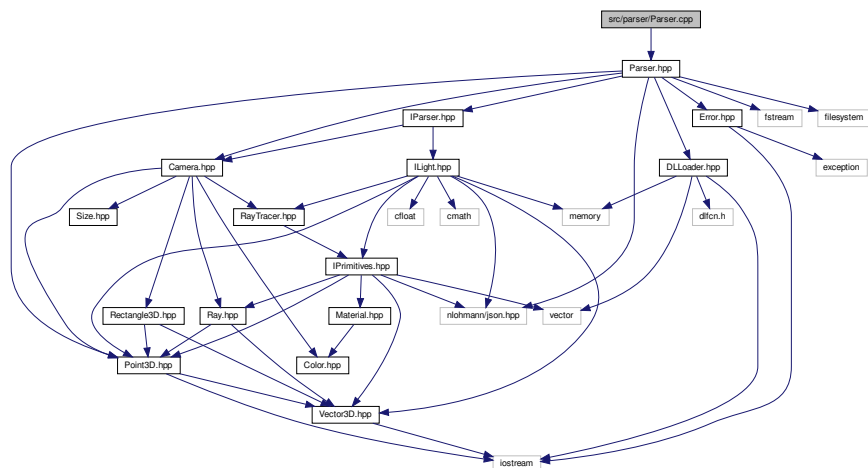
- [RayTracer](#)

*Namespace for the raytracer.*

## 7.35 src/parser/Parser.cpp File Reference

```
#include "Parser.hpp"
```

Include dependency graph for Parser.cpp:



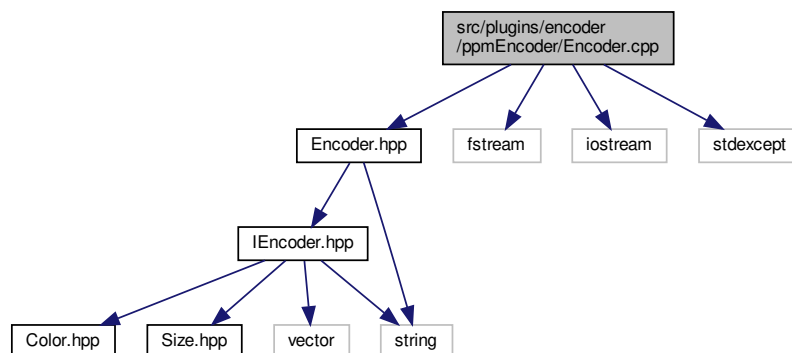
## Namespaces

- [RayTracer](#)

*Namespace for the raytracer.*

## 7.36 src/plugins/encoder/ppmEncoder/Encoder.cpp File Reference

```
#include "Encoder.hpp"
#include <fstream>
#include <iostream>
#include <stdexcept>
Include dependency graph for Encoder.cpp:
```



## Namespaces

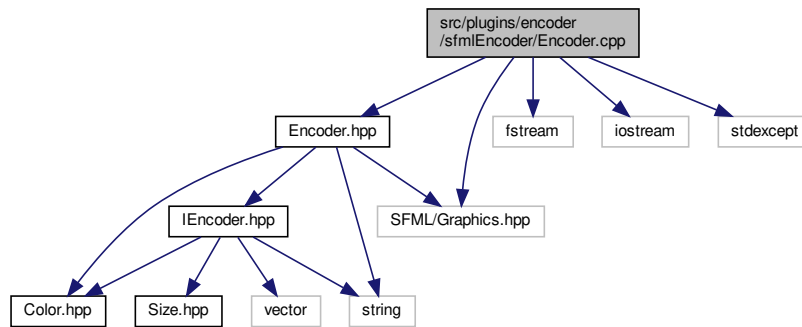
- [RayTracer](#)

*Namespace for the raytracer.*



## 7.37 src/plugins/encoder/sfmlEncoder/Encoder.cpp File Reference

```
#include "Encoder.hpp"
#include "SFML/Graphics.hpp"
#include <fstream>
#include <iostream>
#include <stdexcept>
Include dependency graph for Encoder.cpp:
```



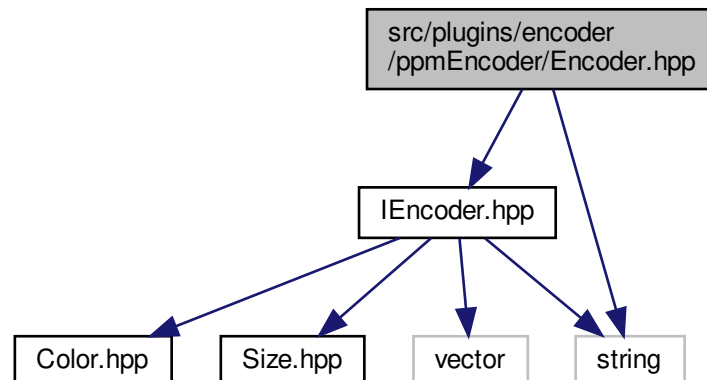
### Namespaces

- [RayTracer](#)

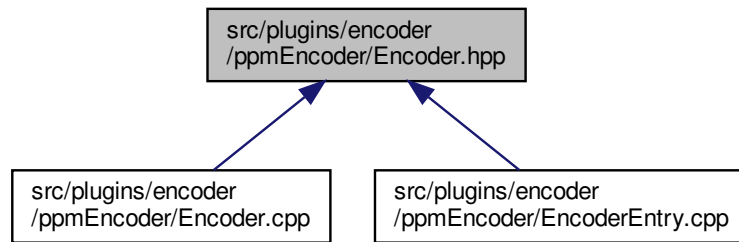
*Namespace for the raytracer.*

## 7.38 src/plugins/encoder/ppmEncoder/Encoder.hpp File Reference

```
#include "IEncoder.hpp"
#include <string>
Include dependency graph for Encoder.hpp:
```



This graph shows which files directly or indirectly include this file:



## Data Structures

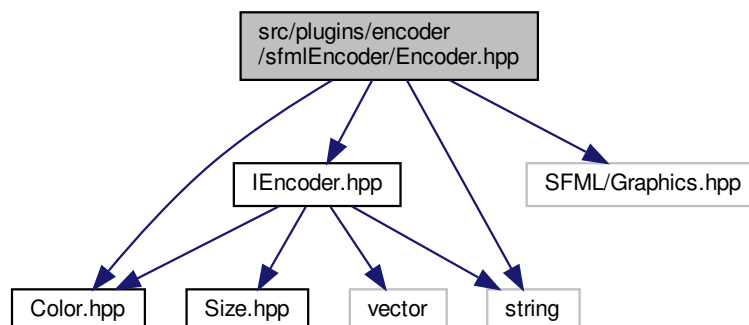
- class [RayTracer::Encoder](#)  
*Encoder class.*

## Namespaces

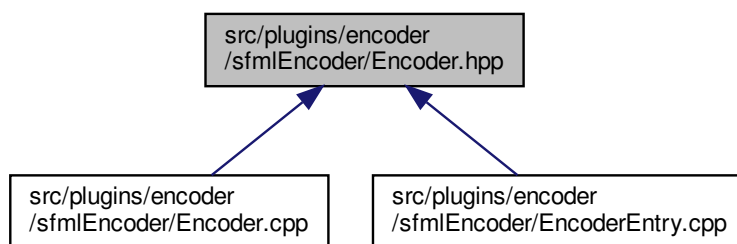
- [RayTracer](#)  
*Namespace for the raytracer.*

## 7.39 src/plugins/encoder/sfmlEncoder/Encoder.hpp File Reference

```
#include "Color.hpp"
#include "IEncoder.hpp"
#include <SFML/Graphics.hpp>
#include <string>
Include dependency graph for Encoder.hpp:
```



This graph shows which files directly or indirectly include this file:



## Data Structures

- class [RayTracer::SfmlEncoder](#)
- struct [RayTracer::SfmlEncoder::Image](#)

## Namespaces

- [RayTracer](#)  
*Namespace for the raytracer.*

## Macros

- `#define` [FRAME\\_RATE\\_LIMIT](#) 60
- `#define` [MAX\\_VALUE](#) 255
- `#define` [BYTE\\_OF\\_RGBA\\_FORMAT](#) 4
- `#define` [HEADER\\_MAGIC\\_NUMBER](#) "P3"

### 7.39.1 Macro Definition Documentation

#### 7.39.1.1 BYTE\_OF\_RGBA\_FORMAT

```
#define BYTE_OF_RGBA_FORMAT 4
```

#### 7.39.1.2 FRAME\_RATE\_LIMIT

```
#define FRAME_RATE_LIMIT 60
```

### 7.39.1.3 HEADER\_MAGIC\_NUMBER

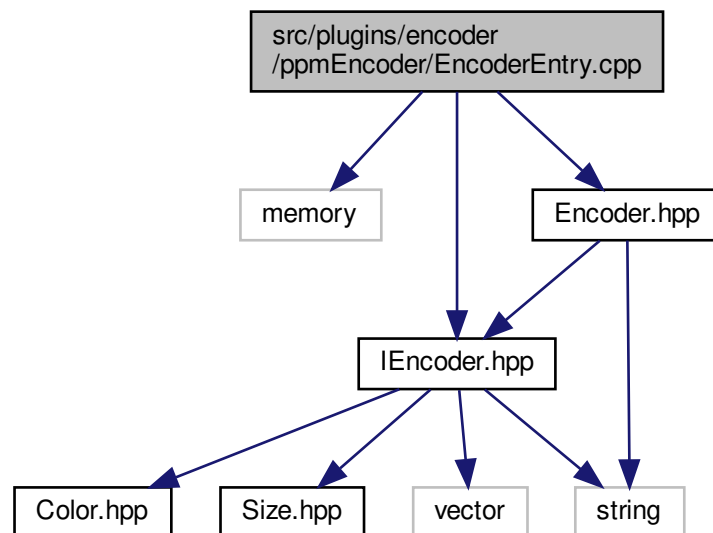
```
#define HEADER_MAGIC_NUMBER "P3"
```

### 7.39.1.4 MAX\_VALUE

```
#define MAX_VALUE 255
```

## 7.40 src/plugins/encoder/ppmEncoder/EncoderEntry.cpp File Reference

```
#include <memory>
#include "IEncoder.hpp"
#include "Encoder.hpp"
Include dependency graph for EncoderEntry.cpp:
```



## Functions

- `std::unique_ptr< RayTracer::IEncoder > entryPoint ()`

### 7.40.1 Function Documentation

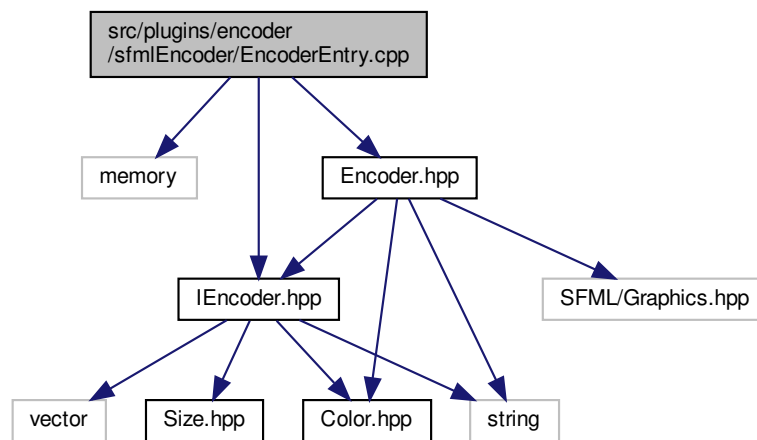
7.40.1.1 `entryPoint()`

```
std::unique_ptr<RayTracer::IEncoder> entryPoint ( )
```

## 7.41 src/plugins/encoder/sfmlEncoder/EncoderEntry.cpp File Reference

```
#include <memory>
#include "IEncoder.hpp"
#include "Encoder.hpp"
```

Include dependency graph for EncoderEntry.cpp:



## Functions

- `std::unique_ptr< RayTracer::IEncoder > entryPoint ( )`

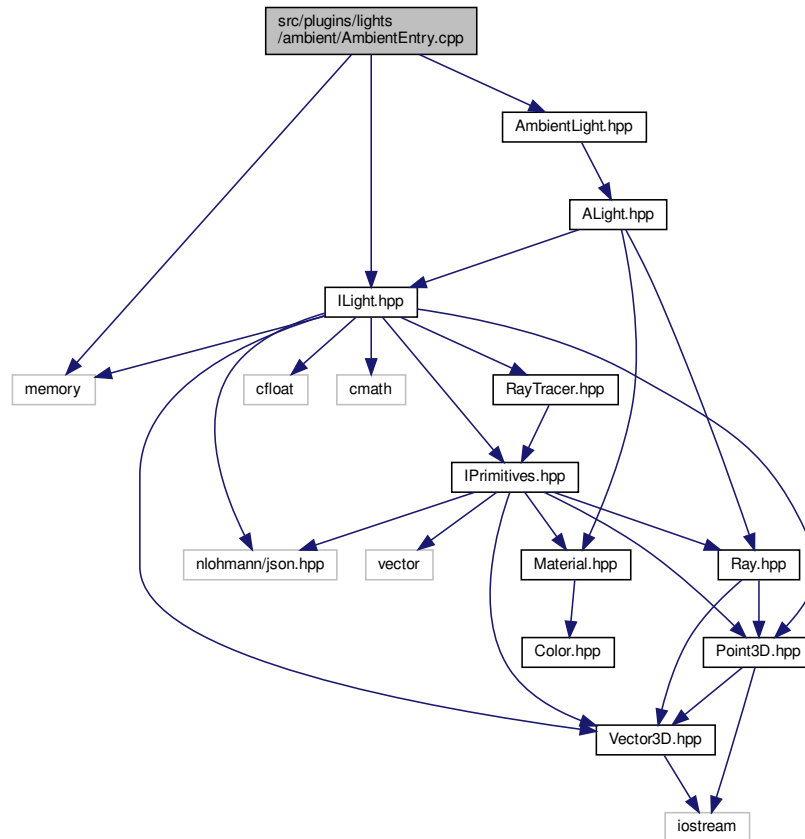
## 7.41.1 Function Documentation

7.41.1.1 `entryPoint()`

```
std::unique_ptr<RayTracer::IEncoder> entryPoint ( )
```

## 7.42 src/plugins/lights/ambient/AmbientEntry.cpp File Reference

```
#include <memory>
#include "ILight.hpp"
#include "AmbientLight.hpp"
Include dependency graph for AmbientEntry.cpp:
```



### Functions

- `std::unique_ptr< RayTracer::ILight > entryPoint ()`

### 7.42.1 Function Documentation

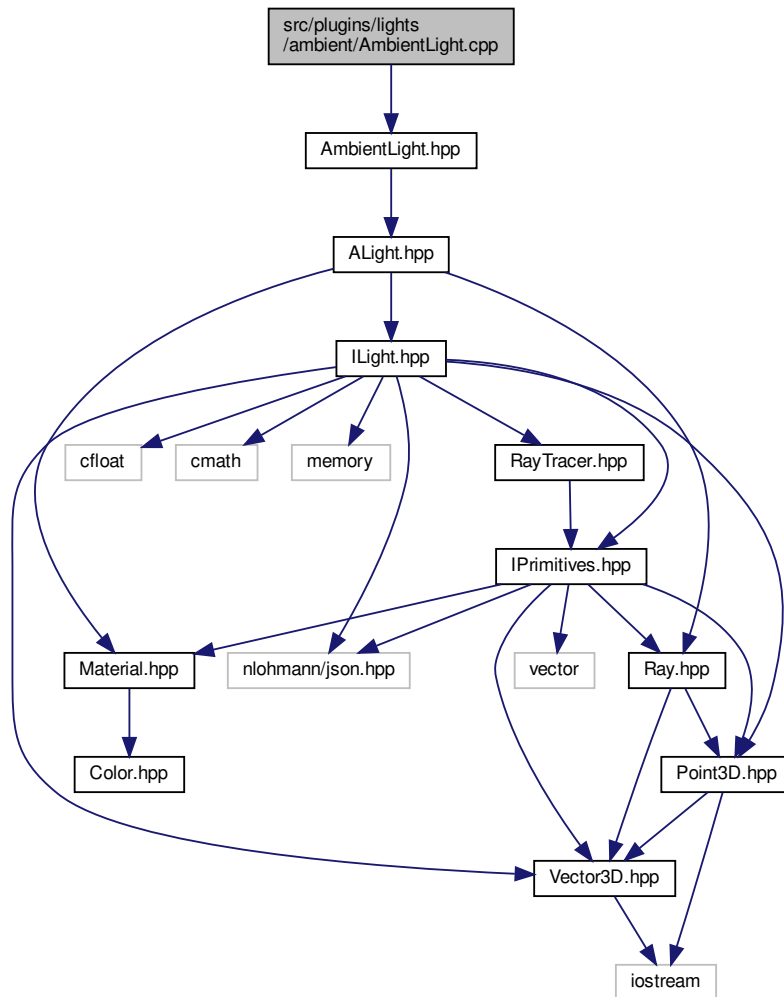
#### 7.42.1.1 entryPoint()

```
std::unique_ptr<RayTracer::ILight> entryPoint ( )
```

## 7.43 src/plugins/lights/ambient/AmbientLight.cpp File Reference

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

Include dependency graph for AmbientLight.cpp:



### Namespaces

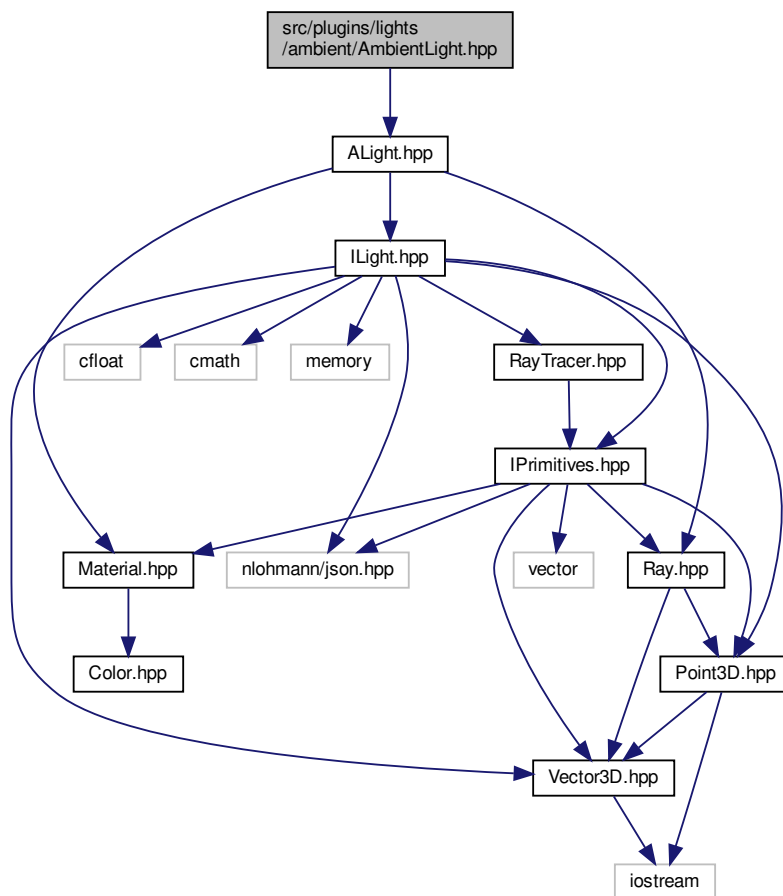
- [RayTracer](#)

*Namespace for the raytracer.*

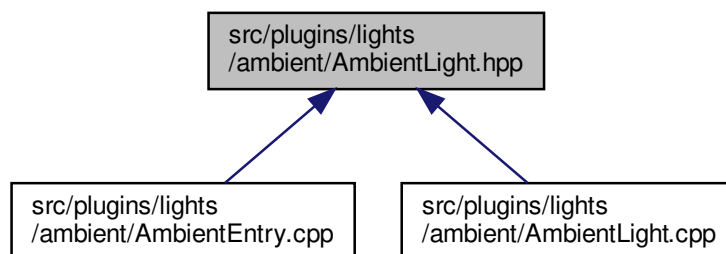
## 7.44 src/plugins/lights/ambient/AmbientLight.hpp File Reference

```
#include "ALight.hpp"
```

Include dependency graph for AmbientLight.hpp:



This graph shows which files directly or indirectly include this file:



## Data Structures

- class [RayTracer::AmbientLight](#)



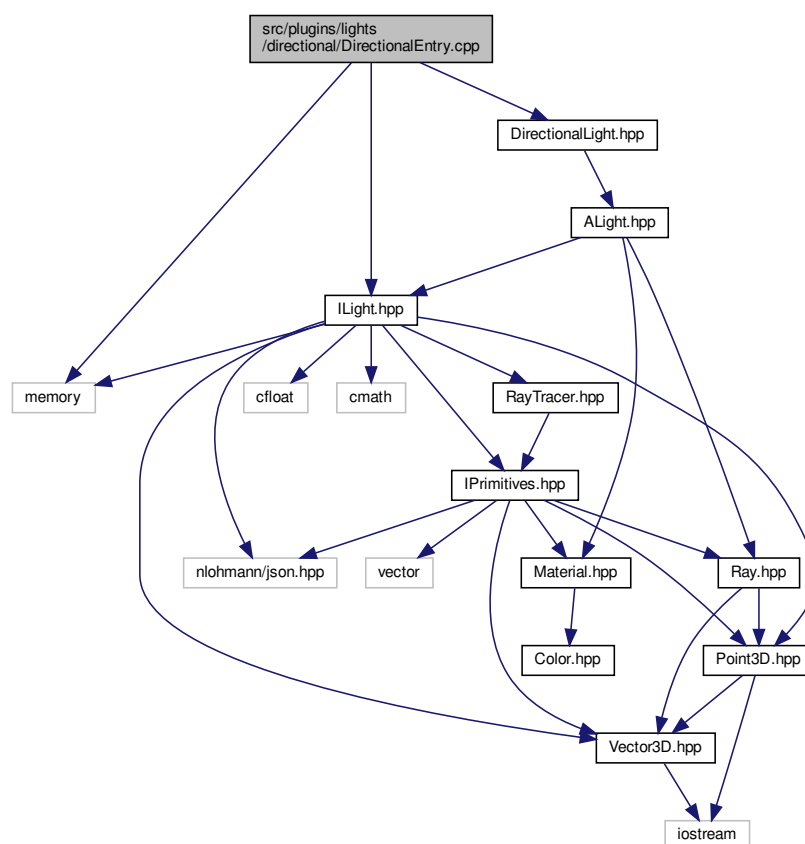
## Namespaces

- [RayTracer](#)

*Namespace for the raytracer.*

## 7.45 src/plugins/lights/directional/DirectionalEntry.cpp File Reference

```
#include <memory>
#include "ILight.hpp"
#include "DirectionalLight.hpp"
Include dependency graph for DirectionalEntry.cpp:
```



## Functions

- `std::unique_ptr< RayTracer::ILight > entryPoint ()`

### 7.45.1 Function Documentation

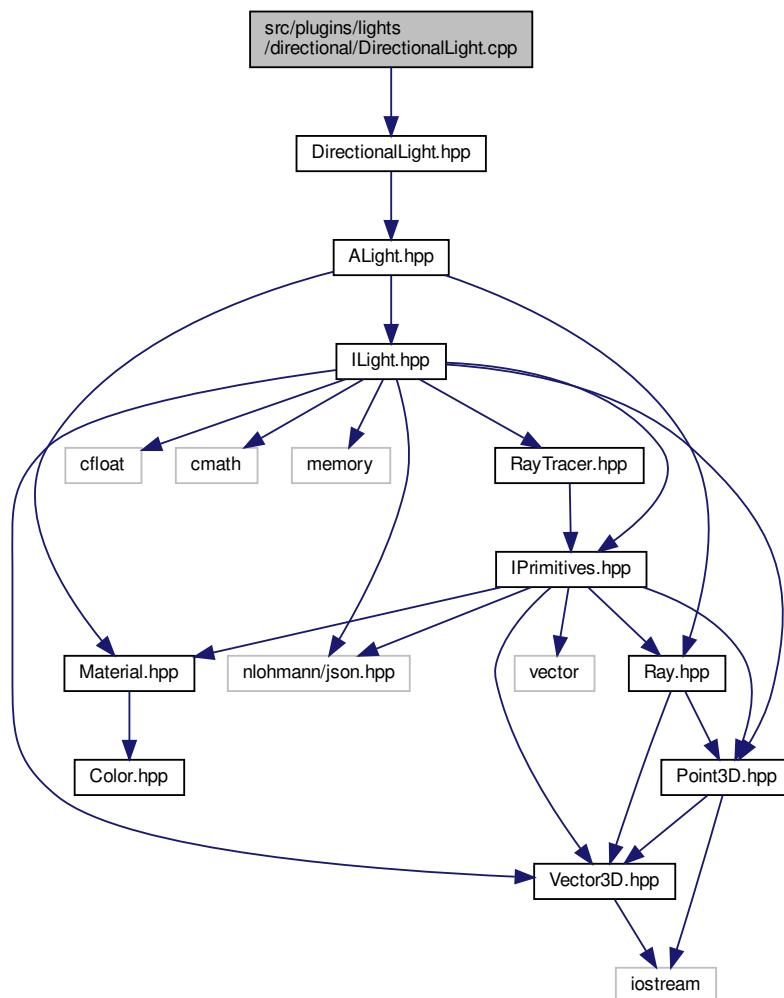
### 7.45.1.1 `entryPoint()`

```
std::unique_ptr<RayTracer::ILight> entryPoint ( )
```

## 7.46 `src/plugins/lights/directional/DirectionalLight.cpp` File Reference

```
#include "DirectionalLight.hpp"
```

Include dependency graph for `DirectionalLight.cpp`:



## Namespaces

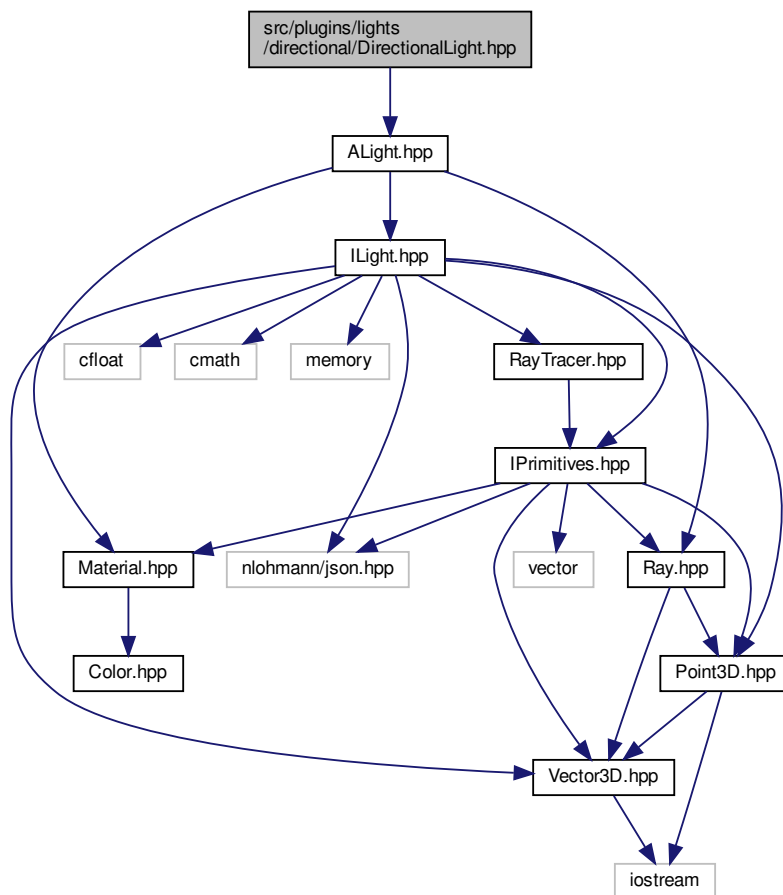
- [RayTracer](#)

*Namespace for the raytracer.*

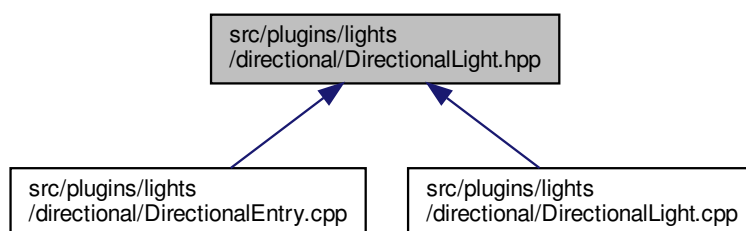
## 7.47 src/plugins/lights/directional/DirectionalLight.hpp File Reference

```
#include "ALight.hpp"
```

Include dependency graph for DirectionalLight.hpp:



This graph shows which files directly or indirectly include this file:



## Data Structures

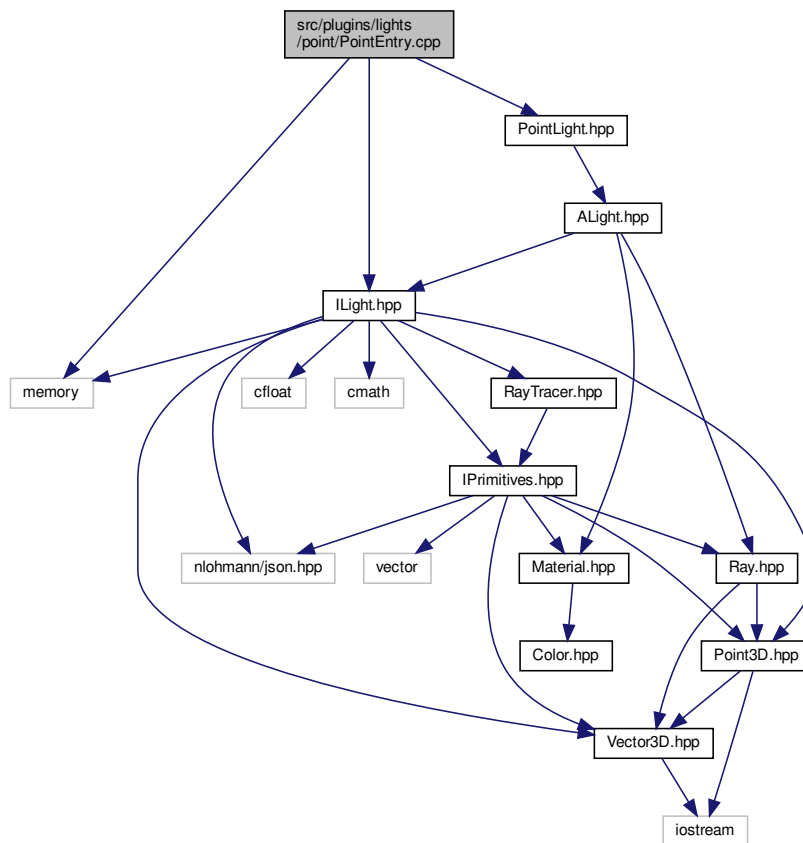
- class [RayTracer::DirectionalLight](#)

## Namespaces

- [RayTracer](#)  
*Namespace for the raytracer.*

## 7.48 src/plugins/lights/point/PointEntry.cpp File Reference

```
#include <memory>
#include "ILight.hpp"
#include "PointLight.hpp"
Include dependency graph for PointEntry.cpp:
```



## Functions

- `std::unique_ptr< RayTracer::ILight > entryPoint ()`

## 7.48.1 Function Documentation

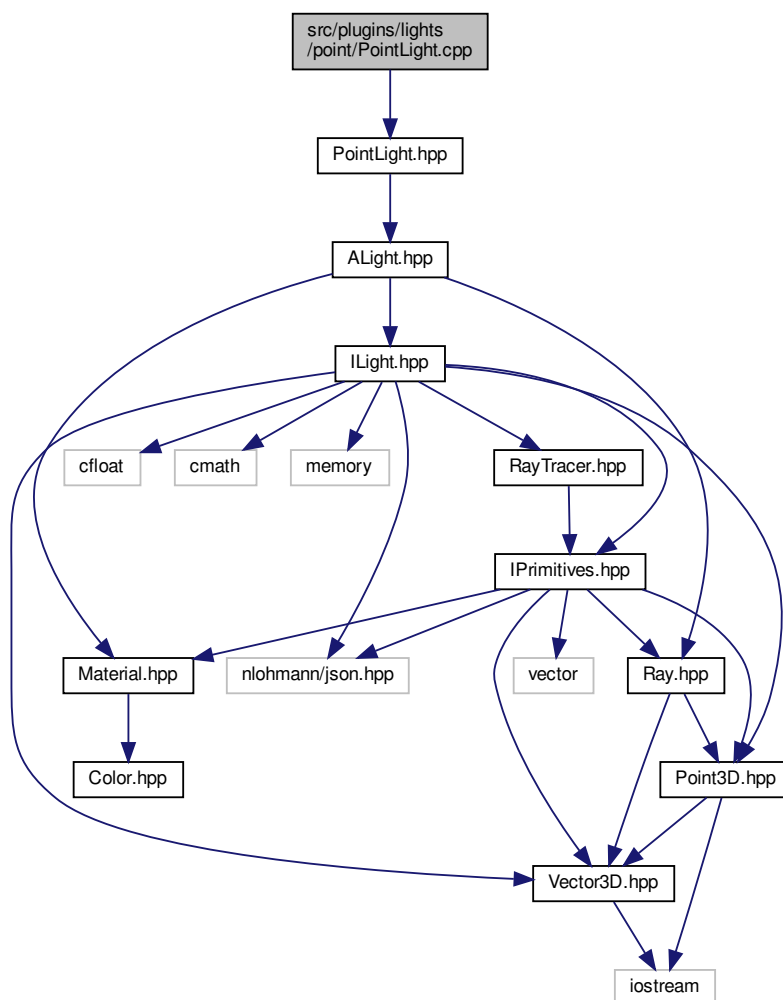
### 7.48.1.1 `entryPoint()`

```
std::unique_ptr<RayTracer::ILight> entryPoint ( )
```

## 7.49 src/plugins/lights/point/PointLight.cpp File Reference

```
#include "PointLight.hpp"
```

Include dependency graph for PointLight.cpp:



## Namespaces

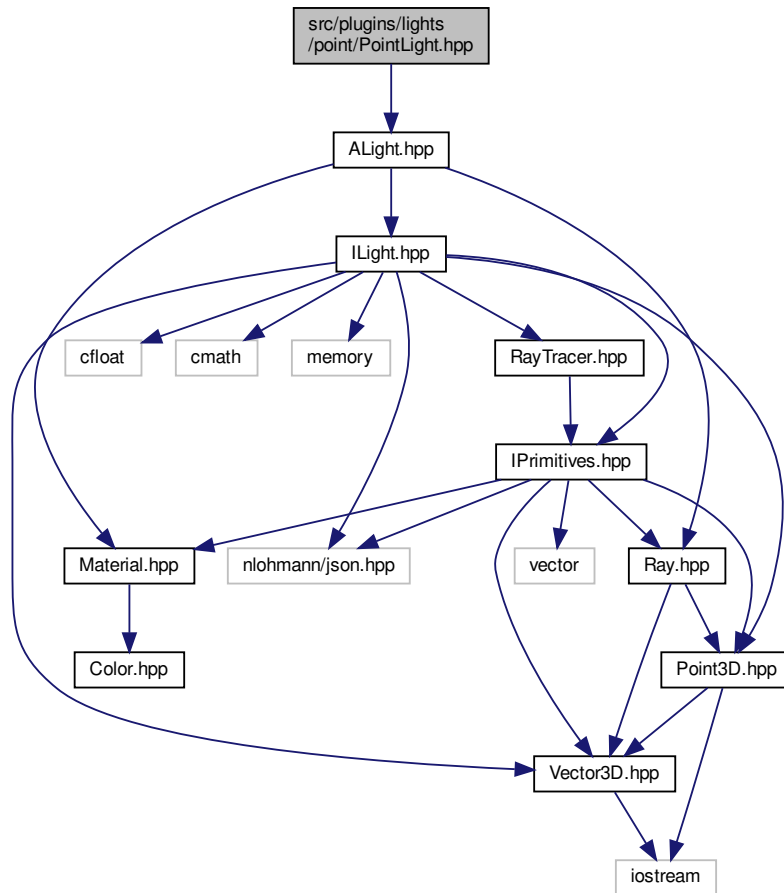
- [RayTracer](#)

*Namespace for the raytracer.*

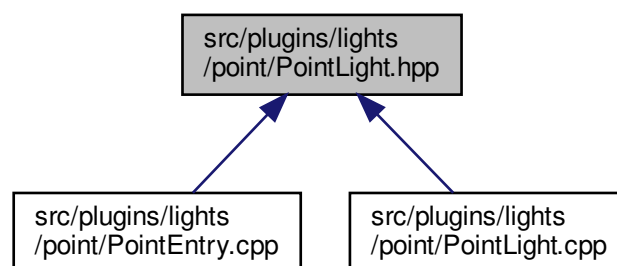
## 7.50 src/plugins/lights/point/PointLight.hpp File Reference

```
#include "ALight.hpp"
```

Include dependency graph for PointLight.hpp:



This graph shows which files directly or indirectly include this file:



## Data Structures

- class [RayTracer::PointLight](#)

## Namespaces

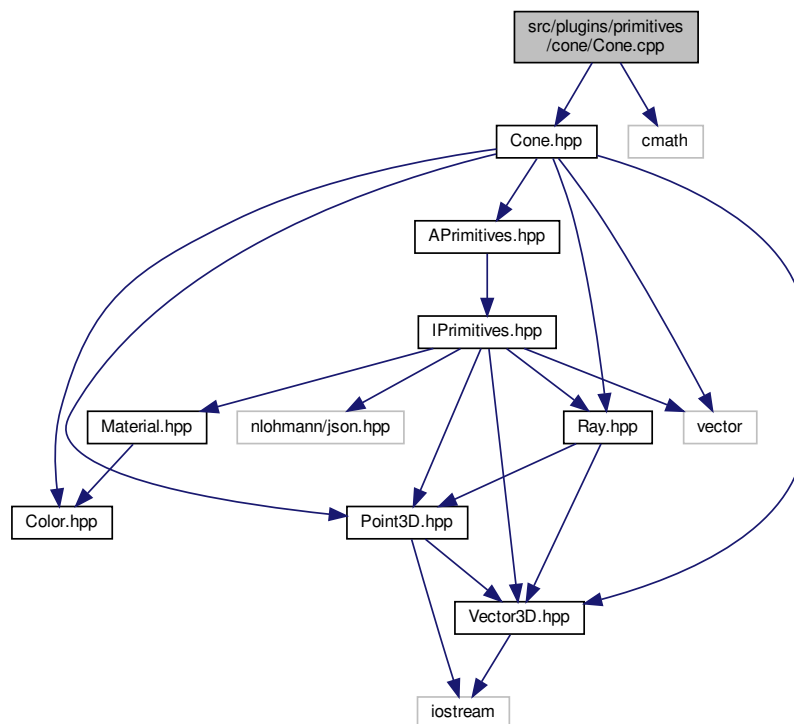
- [RayTracer](#)

*Namespace for the raytracer.*

## 7.51 src/plugins/primitives/cone/Cone.cpp File Reference

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

Include dependency graph for Cone.cpp:



## Namespaces

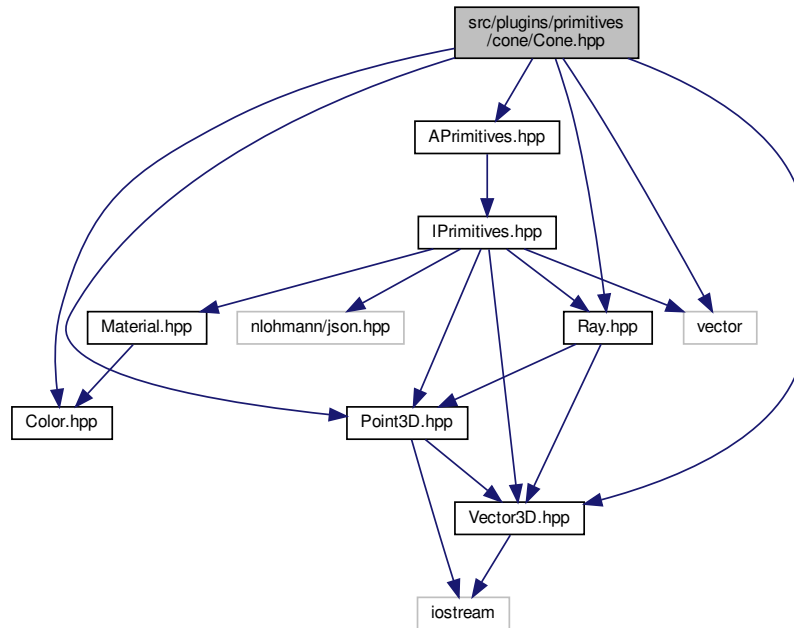
- [RayTracer](#)

*Namespace for the raytracer.*

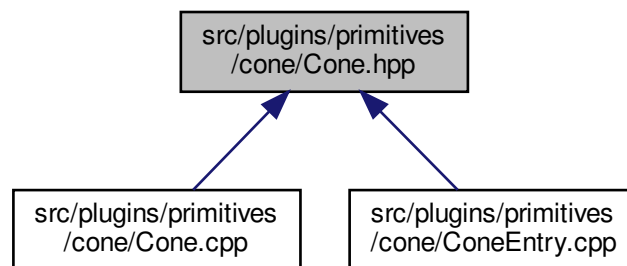
## 7.52 src/plugins/primitives/cone/Cone.hpp File Reference

```
#include "APrimitives.hpp"
#include "Color.hpp"
#include "Point3D.hpp"
#include "Ray.hpp"
#include "Vector3D.hpp"
#include <vector>
```

Include dependency graph for Cone.hpp:



This graph shows which files directly or indirectly include this file:



### Data Structures

- class [RayTracer::Cone](#)



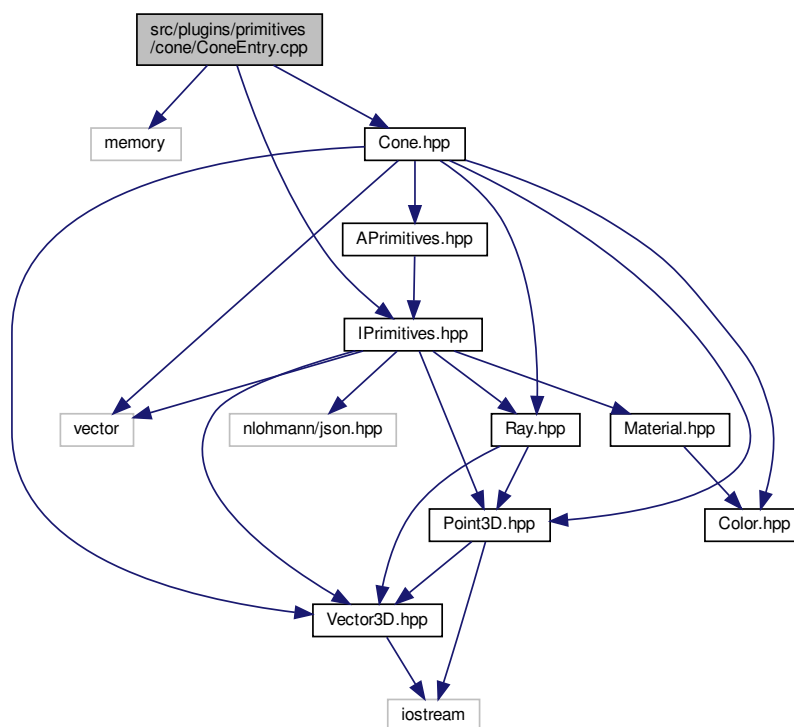
## Namespaces

- [RayTracer](#)

*Namespace for the raytracer.*

## 7.53 src/plugins/primitives/cone/ConeEntry.cpp File Reference

```
#include <memory>
#include "IPrimitives.hpp"
#include "Cone.hpp"
Include dependency graph for ConeEntry.cpp:
```



## Functions

- `std::unique_ptr<RayTracer::IPrimitives> entryPoint()`

### 7.53.1 Function Documentation

#### 7.53.1.1 entryPoint()

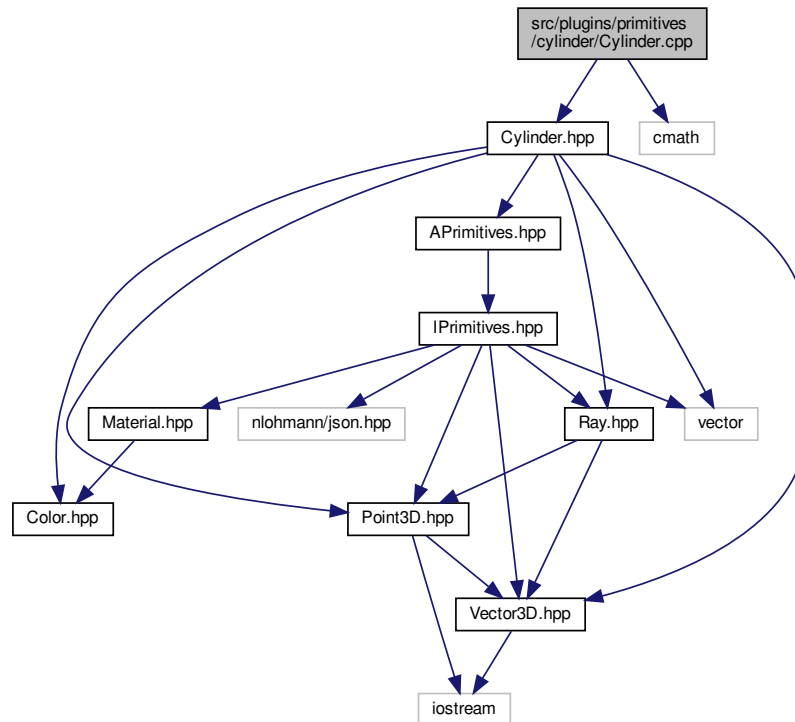
```
std::unique_ptr<RayTracer::IPrimitives> entryPoint ( )
```

## 7.54 src/plugins/primitives/cylinder/Cylinder.cpp File Reference

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

```
#include <cmath>
```

Include dependency graph for Cylinder.cpp:



## Namespaces

- [RayTracer](#)

*Namespace for the raytracer.*

## 7.55 src/plugins/primitives/cylinder/Cylinder.hpp File Reference

```
#include "APrimitives.hpp"
```

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

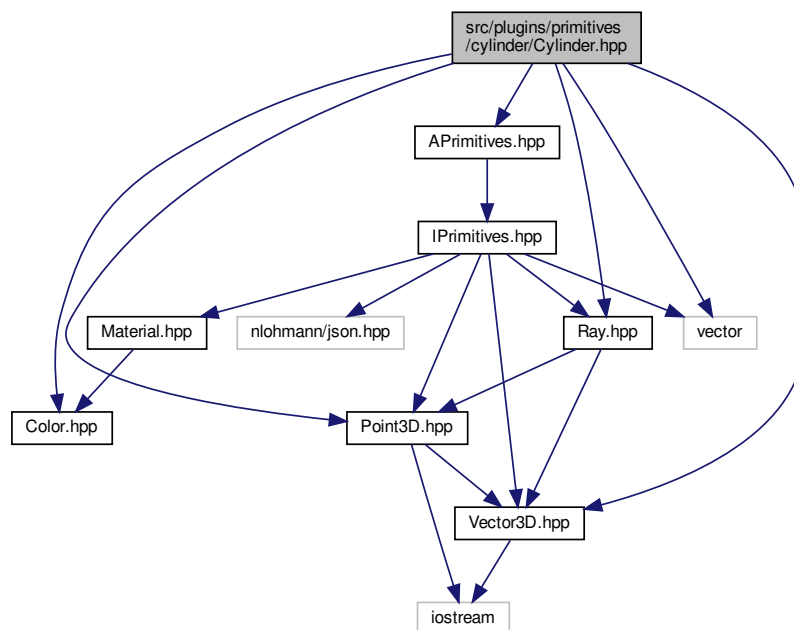
```
#include "Point3D.hpp"
```

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

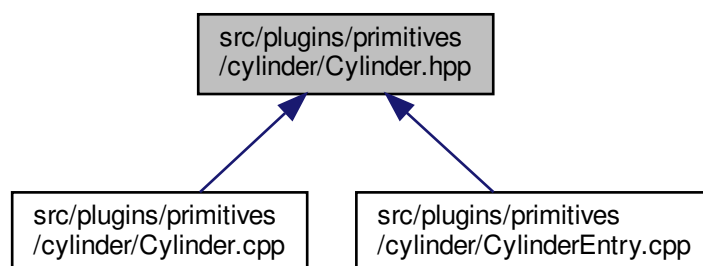
```
#include "Vector3D.hpp"
```

```
#include <vector>
```

Include dependency graph for Cylinder.hpp:



This graph shows which files directly or indirectly include this file:



## Data Structures

- class [RayTracer::Cylinder](#)

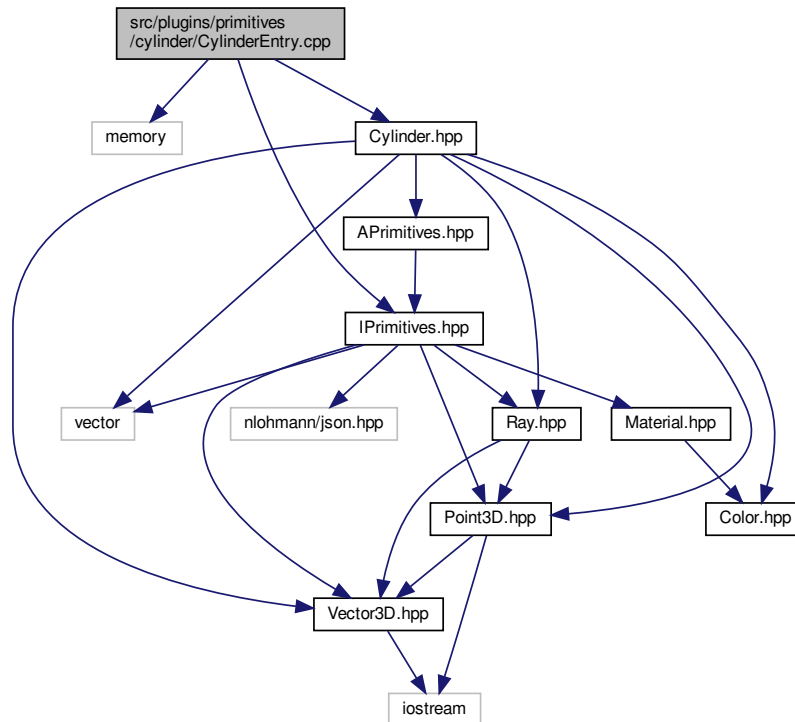
## Namespaces

- [RayTracer](#)

*Namespace for the raytracer.*

## 7.56 src/plugins/primitives/cylinder/CylinderEntry.cpp File Reference

```
#include <memory>
#include "IPrimitives.hpp"
#include "Cylinder.hpp"
Include dependency graph for CylinderEntry.cpp:
```



### Functions

- `std::unique_ptr< RayTracer::IPrimitives > entryPoint ()`

### 7.56.1 Function Documentation

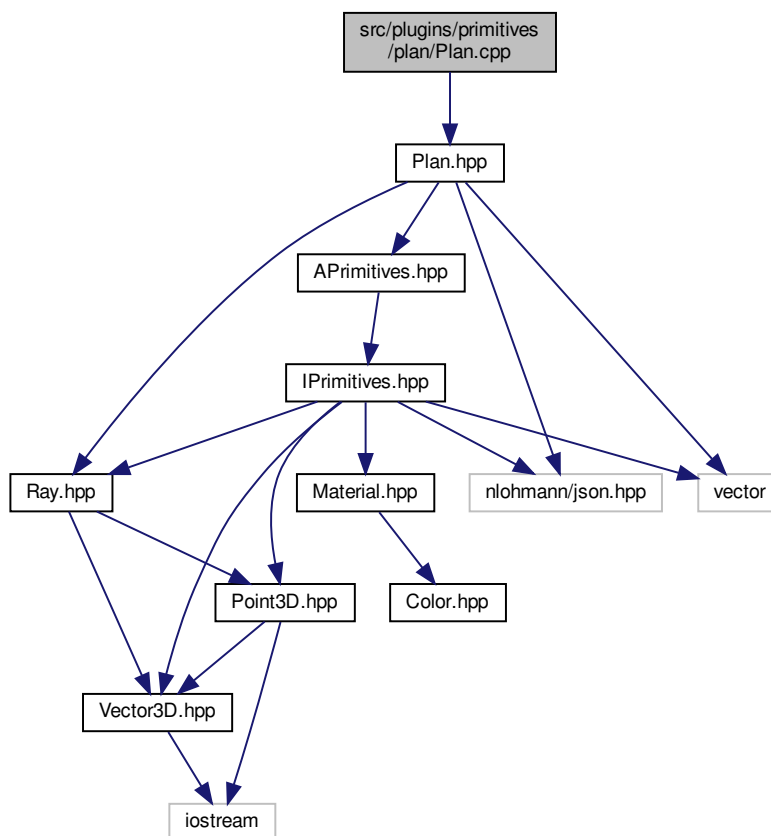
#### 7.56.1.1 entryPoint()

```
std::unique_ptr<RayTracer::IPrimitives> entryPoint ( )
```

## 7.57 src/plugins/primitives/plan/Plan.cpp File Reference

```
#include "Plan.hpp"
```

Include dependency graph for Plan.cpp:



### Namespaces

- [RayTracer](#)

*Namespace for the raytracer.*

## 7.58 src/plugins/primitives/plan/Plan.hpp File Reference

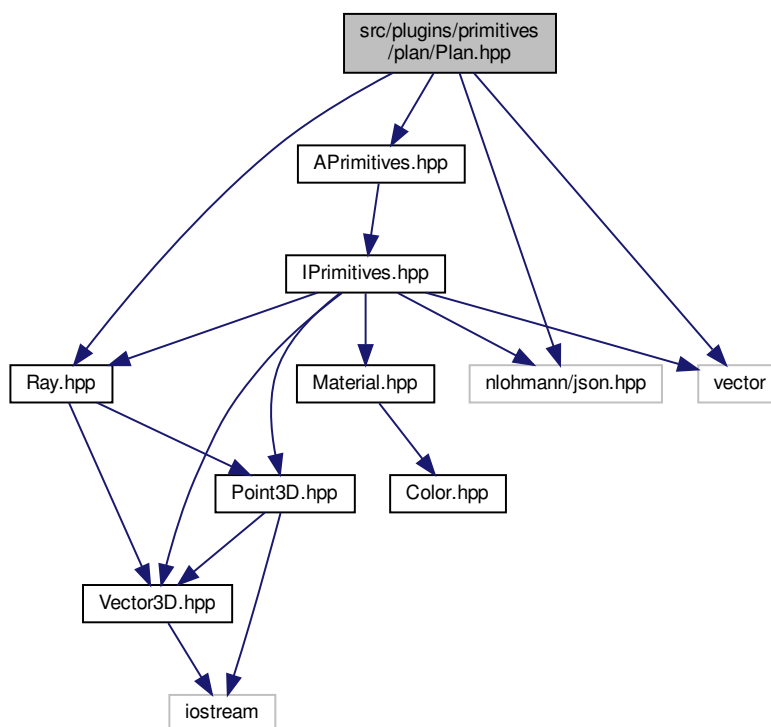
```
#include "APrimitives.hpp"
```

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

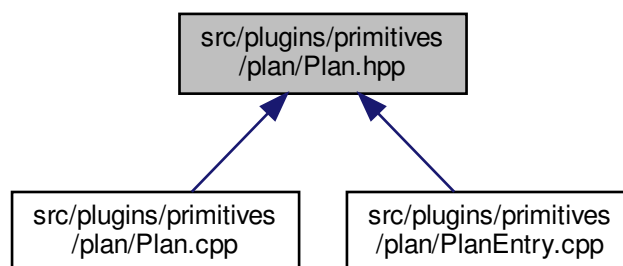
```
#include <nlohmann/json.hpp>
```

```
#include <vector>
```

Include dependency graph for Plan.hpp:



This graph shows which files directly or indirectly include this file:



## Data Structures

- class [RayTracer::Plan](#)

## Namespaces

- [RayTracer](#)

*Namespace for the raytracer.*

## Typedefs

- using [json](#) = nlohmann::json

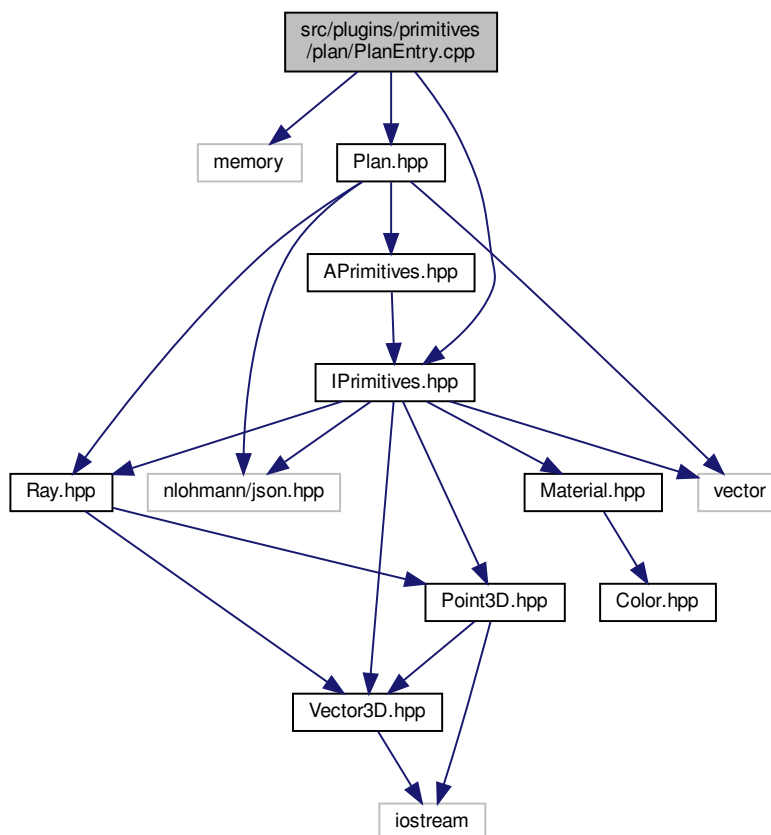
### 7.58.1 Typedef Documentation

#### 7.58.1.1 json

```
using json = nlohmann::json
```

## 7.59 src/plugins/primitives/plan/PlanEntry.cpp File Reference

```
#include <memory>
#include "IPrimitives.hpp"
#include "Plan.hpp"
Include dependency graph for PlanEntry.cpp:
```



## Functions

- `std::unique_ptr< RayTracer::IPrimitives > entryPoint ()`

### 7.59.1 Function Documentation

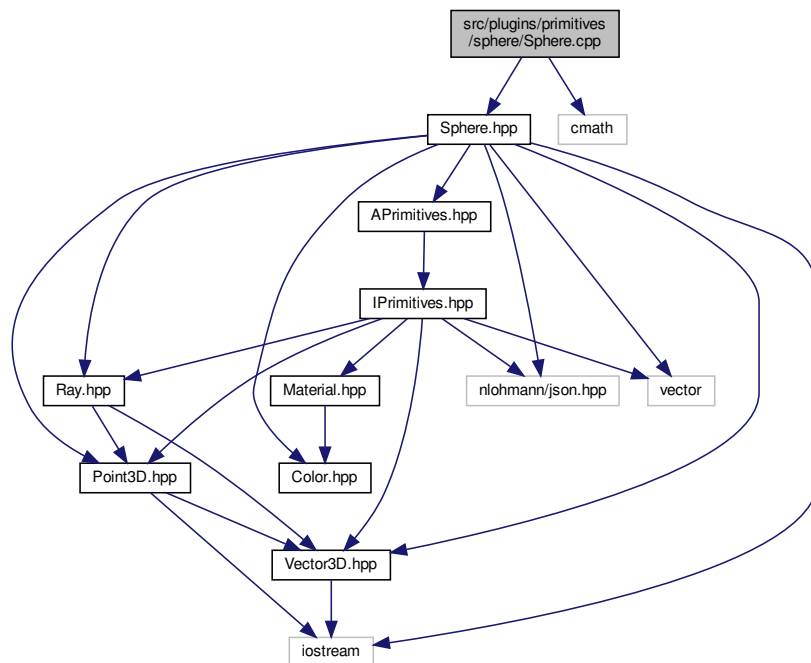
#### 7.59.1.1 entryPoint()

```
std::unique_ptr<RayTracer::IPrimitives> entryPoint ( )
```

## 7.60 src/plugins/primitives/sphere/Sphere.cpp File Reference

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

Include dependency graph for Sphere.cpp:



## Namespaces

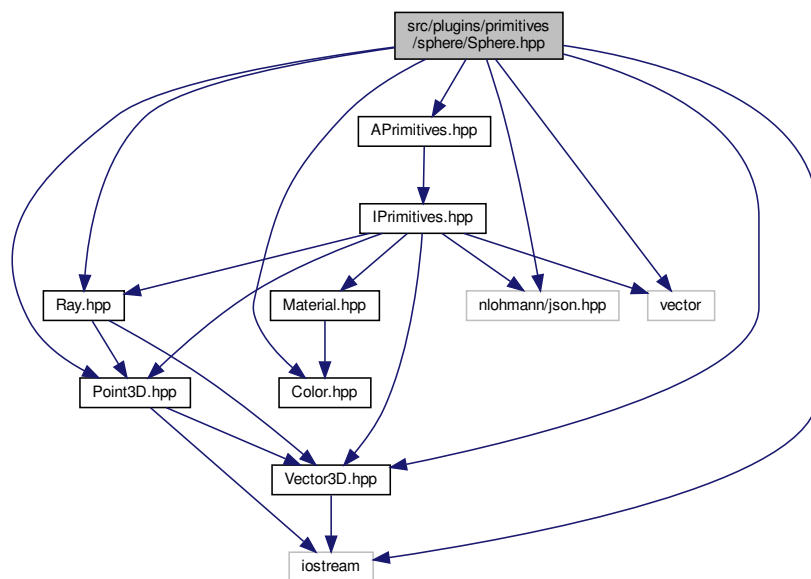
- `RayTracer`  
*Namespace for the raytracer.*



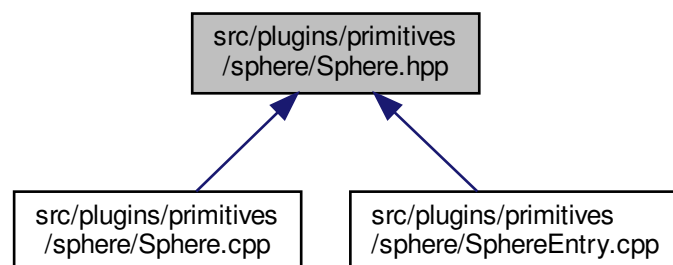
## 7.61 src/plugins/primitives/sphere/Sphere.hpp File Reference

```
#include "APrimitives.hpp"
#include "Color.hpp"
#include "Point3D.hpp"
#include "Ray.hpp"
#include "Vector3D.hpp"
#include <iostream>
#include <nlohmann/json.hpp>
#include <vector>
```

Include dependency graph for Sphere.hpp:



This graph shows which files directly or indirectly include this file:



### Data Structures

- class [RayTracer::Sphere](#)

## Namespaces

- [RayTracer](#)

*Namespace for the raytracer.*

## Typedefs

- using [json](#) = nlohmann::json

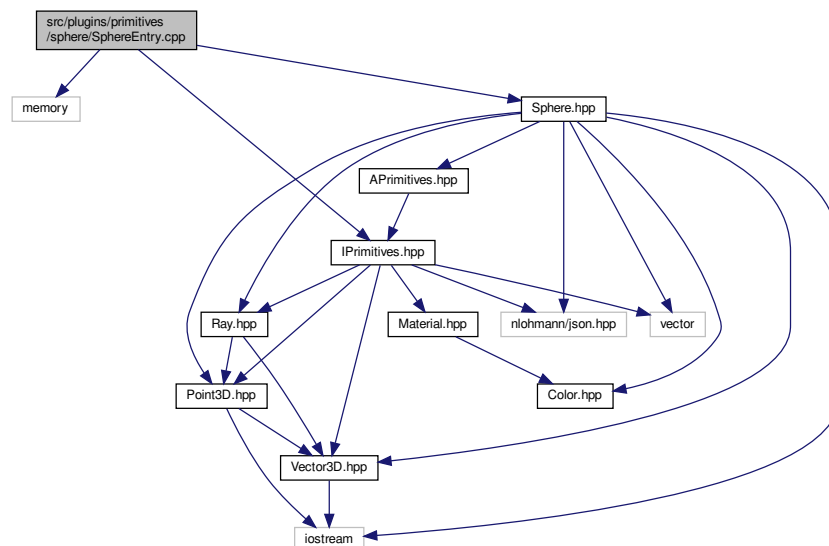
## 7.61.1 Typedef Documentation

### 7.61.1.1 json

```
using json = nlohmann::json
```

## 7.62 src/plugins/primitives/sphere/SphereEntry.cpp File Reference

```
#include <memory>
#include "IPrimitives.hpp"
#include "Sphere.hpp"
Include dependency graph for SphereEntry.cpp:
```



## Functions

- `std::unique_ptr< RayTracer::IPrimitives > entryPoint ()`

## 7.62.1 Function Documentation

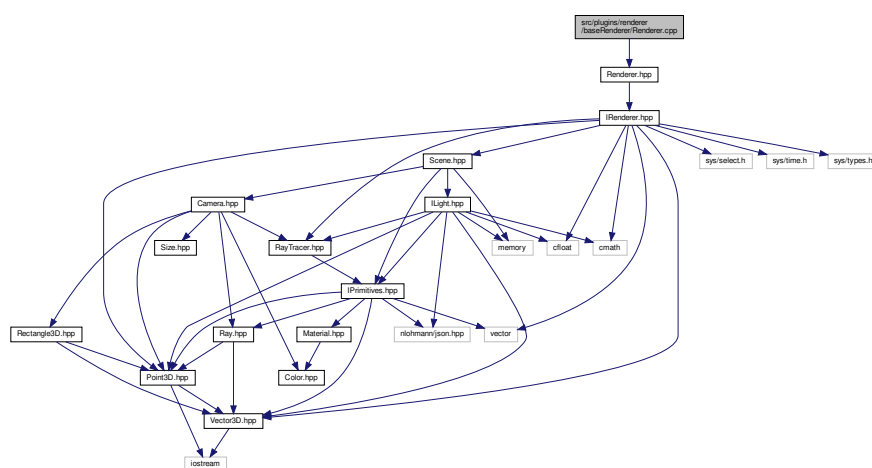
### 7.62.1.1 `entryPoint()`

```
std::unique_ptr<RayTracer::IPrimitives> entryPoint ( )
```

## 7.63 src/plugins/renderer/baseRenderer/Renderer.cpp File Reference

```
#include "Renderer.hpp"
```

Include dependency graph for `Renderer.cpp`:



## Namespaces

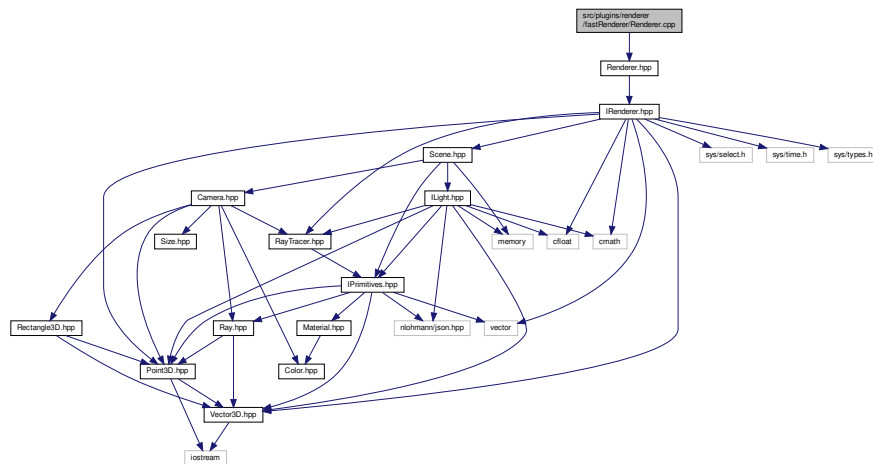
- [RayTracer](#)

*Namespace for the raytracer.*

## 7.64 src/plugins/renderer/fastRenderer/Renderer.cpp File Reference

```
#include "Renderer.hpp"
```

Include dependency graph for `Renderer.cpp`:



## Namespaces

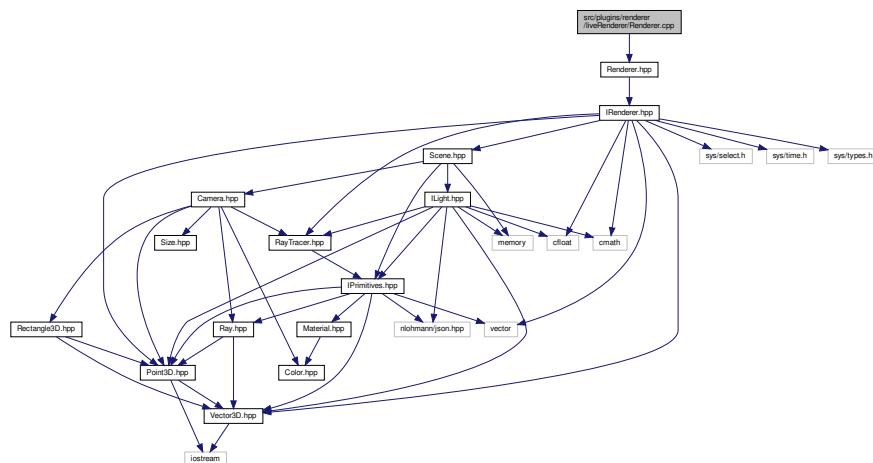
- [RayTracer](#)

*Namespace for the raytracer.*

## 7.65 src/plugins/renderer/liveRenderer/Renderer.cpp File Reference

```
#include "Renderer.hpp"
```

Include dependency graph for `Renderer.cpp`:



## Namespaces

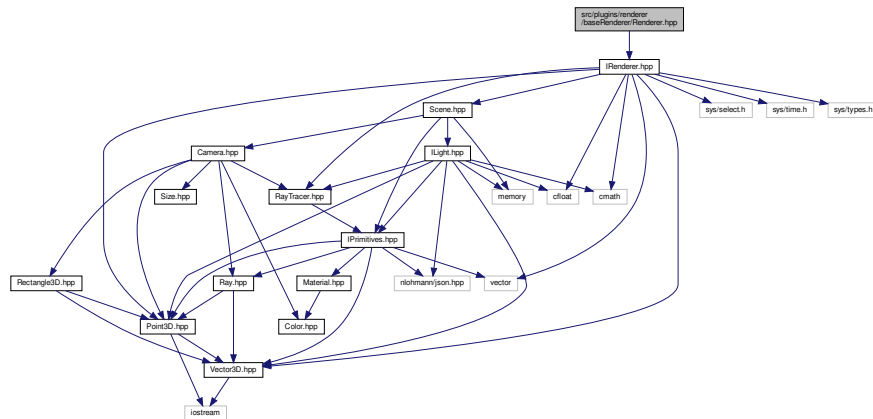
- [RayTracer](#)

*Namespace for the raytracer.*

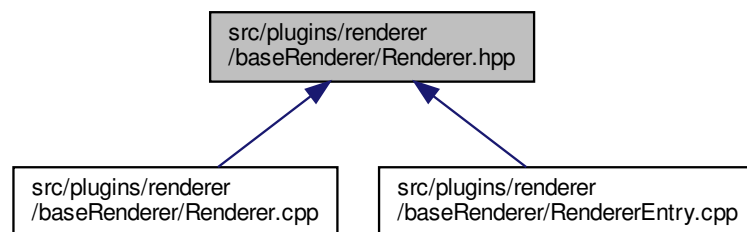
## 7.66 src/plugins/renderer/baseRenderer/Renderer.hpp File Reference

```
#include "IRenderer.hpp"
```

Include dependency graph for Renderer.hpp:



This graph shows which files directly or indirectly include this file:



### Data Structures

- class [RayTracer::Renderer](#)

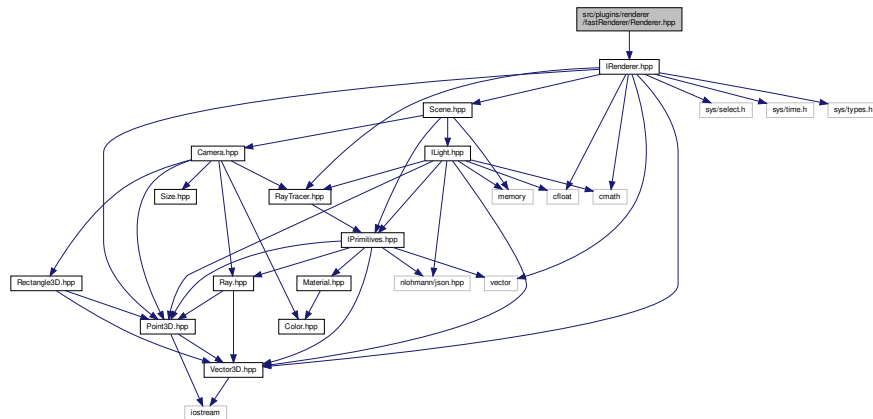
### Namespaces

- [RayTracer](#)  
*Namespace for the raytracer.*

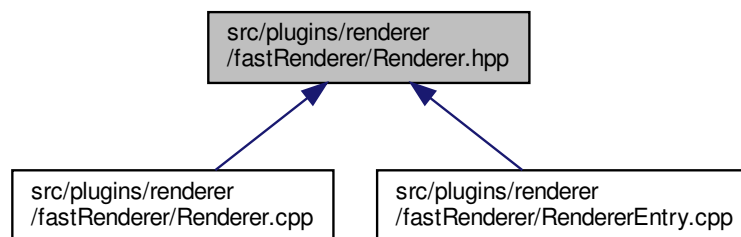
## 7.67 src/plugins/renderer/fastRenderer/Renderer.hpp File Reference

```
#include "IRenderer.hpp"
```

Include dependency graph for Renderer.hpp:



This graph shows which files directly or indirectly include this file:



### Data Structures

- class [RayTracer::FastRenderer](#)

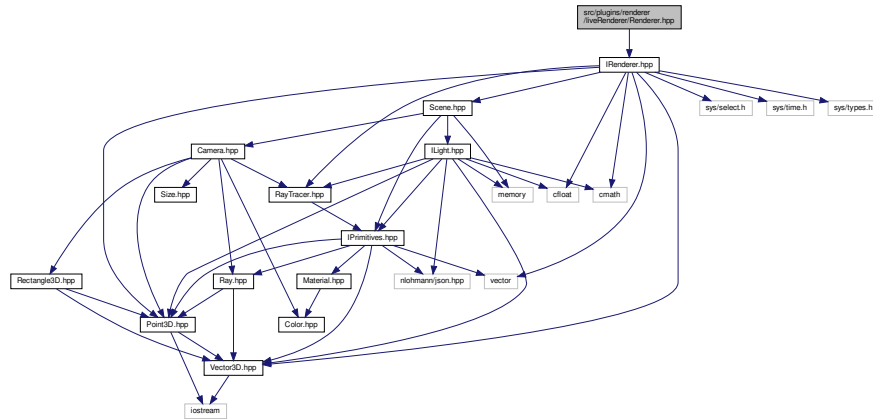
### Namespaces

- [RayTracer](#)  
*Namespace for the raytracer.*

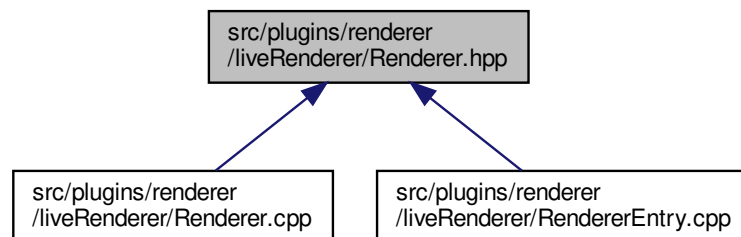
## 7.68 src/plugins/renderer/liveRenderer/Renderer.hpp File Reference

```
#include "IRenderer.hpp"
```

Include dependency graph for Renderer.hpp:



This graph shows which files directly or indirectly include this file:



### Data Structures

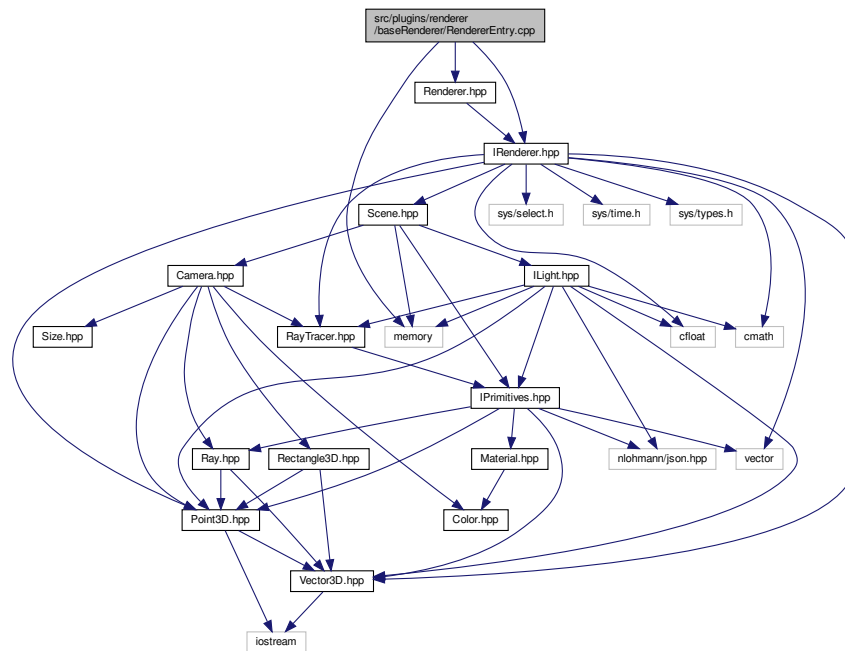
- class [RayTracer::LiveRenderer](#)

### Namespaces

- [RayTracer](#)  
*Namespace for the raytracer.*

## 7.69 src/plugins/renderer/baseRenderer/RendererEntry.cpp File Reference

```
#include <memory>
#include "IRenderer.hpp"
#include "Renderer.hpp"
Include dependency graph for RendererEntry.cpp:
```



### Functions

- `std::unique_ptr< RayTracer::IRenderer > entryPoint ()`

### 7.69.1 Function Documentation

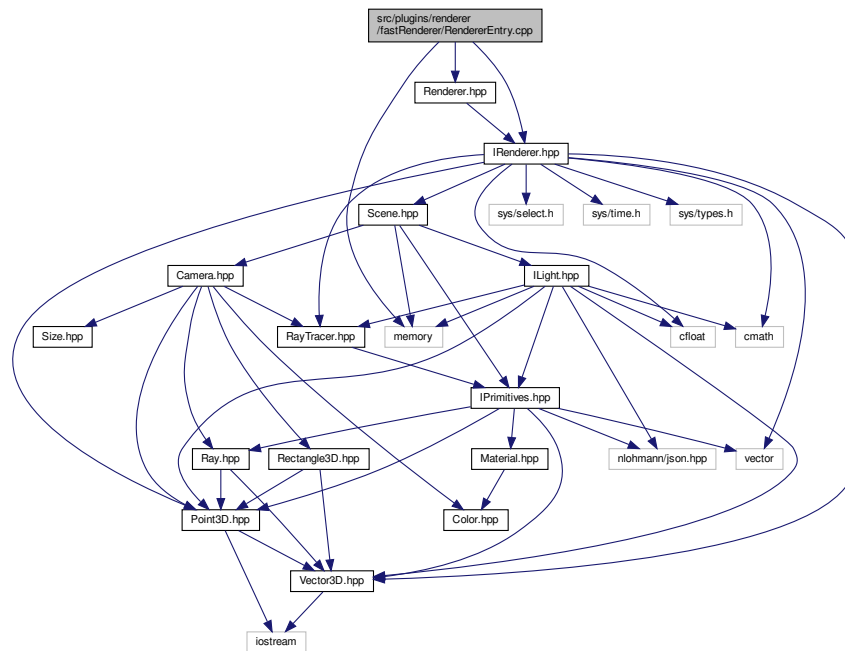
#### 7.69.1.1 entryPoint()

```
std::unique_ptr<RayTracer::IRenderer> entryPoint ( )
```



## 7.70 src/plugins/renderer/fastRenderer/RendererEntry.cpp File Reference

```
#include <memory>
#include "IRenderer.hpp"
#include "Renderer.hpp"
Include dependency graph for RendererEntry.cpp:
```



### Functions

- `std::unique_ptr< RayTracer::IRenderer > entryPoint ()`

### 7.70.1 Function Documentation

#### 7.70.1.1 `entryPoint()`

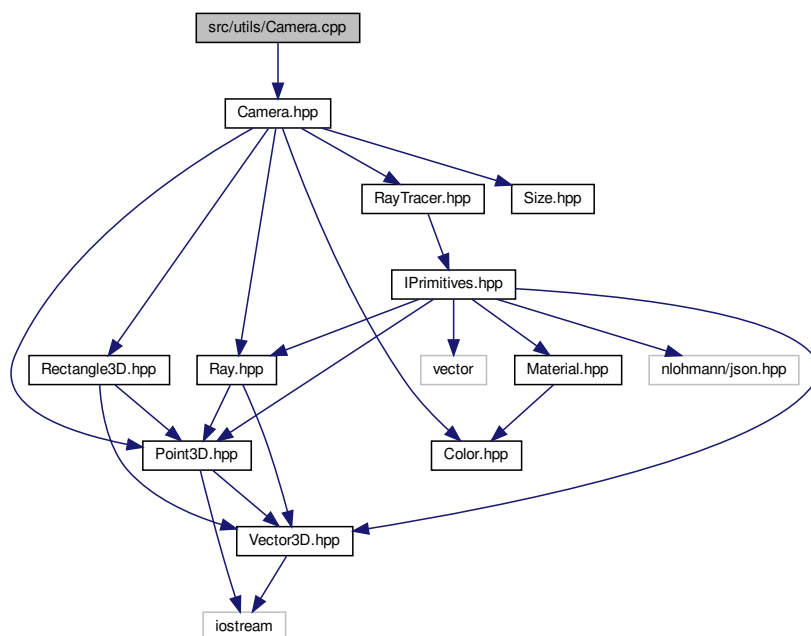
```
std::unique_ptr<RayTracer::IRenderer> entryPoint ( )
```



## 7.72 src/utls/Camera.cpp File Reference

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

Include dependency graph for Camera.cpp:



### Namespaces

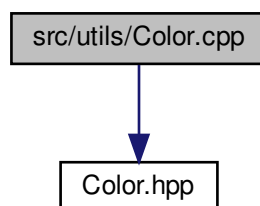
- [RayTracer](#)

*Namespace for the raytracer.*

## 7.73 src/utls/Color.cpp File Reference

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

Include dependency graph for Color.cpp:



## Namespaces

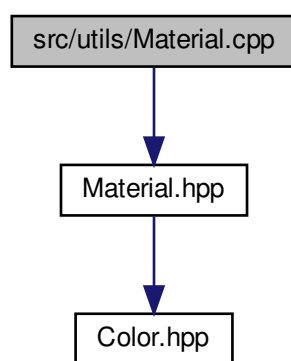
- [RayTracer](#)

*Namespace for the raytracer.*

### 7.74 src/utils/Material.cpp File Reference

```
#include "Material.hpp"
```

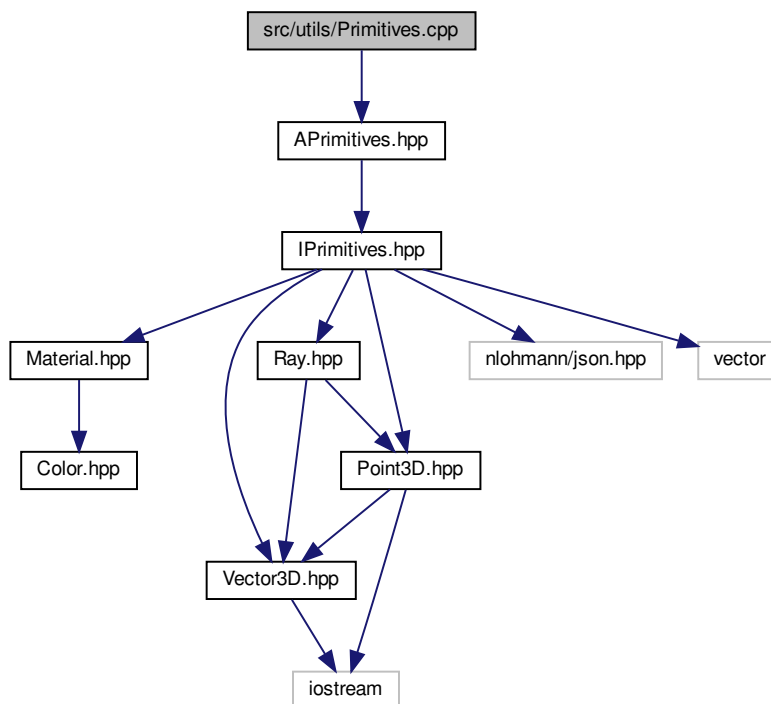
Include dependency graph for Material.cpp:



### 7.75 src/utils/Primitives.cpp File Reference

```
#include "APrimitives.hpp"
```

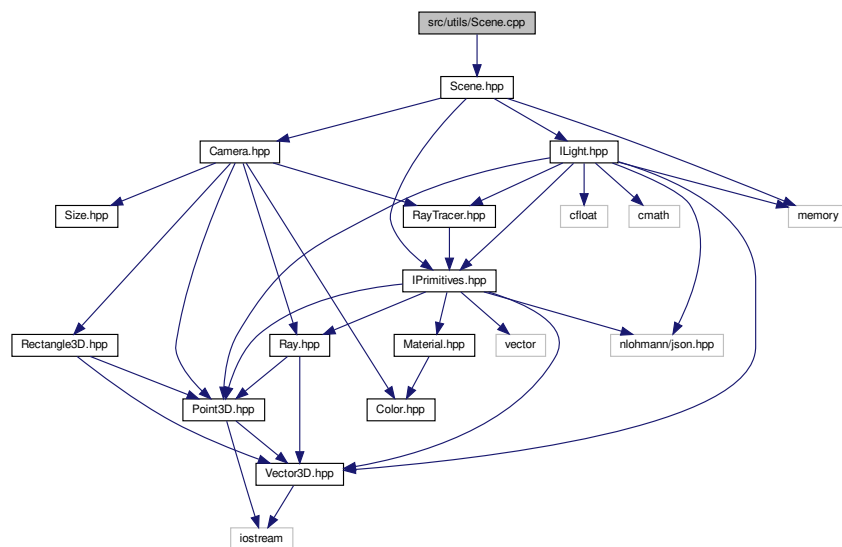
Include dependency graph for Primitives.cpp:



## 7.76 src/utls/Scene.cpp File Reference

```
#include "Scene.hpp"
```

Include dependency graph for Scene.cpp:



## Namespaces

- [RayTracer](#)

*Namespace for the raytracer.*

# Index

- ~ALight
  - RayTracer::ALight, [15](#)
- ~APrimitives
  - RayTracer::APrimitives, [23](#)
- ~Camera
  - RayTracer::Camera, [25](#)
- ~Color
  - RayTracer::Color, [28](#)
- ~Core
  - RayTracer::Core, [38](#)
- ~DLLoader
  - DLLoader< T >, [54](#)
- ~Encoder
  - RayTracer::Encoder, [57](#)
- ~FastRenderer
  - RayTracer::FastRenderer, [62](#)
- ~IEncoder
  - RayTracer::IEncoder, [68](#)
- ~ILight
  - RayTracer::ILight, [71](#)
- ~IPrimitives
  - RayTracer::IPrimitives, [83](#)
- ~LiveRenderer
  - RayTracer::LiveRenderer, [90](#)
- ~Material
  - RayTracer::Material, [96](#)
- ~Parser
  - RayTracer::Parser, [103](#)
- ~ParsingError
  - ParsingError, [112](#)
- ~Point3D
  - Math::Point3D, [121](#)
- ~PointLight
  - RayTracer::PointLight, [127](#)
- ~Ray
  - RayTracer::Ray, [131](#)
- ~Rectangle3D
  - Math::Rectangle3D, [134](#)
- ~Renderer
  - RayTracer::Renderer, [138](#)
- ~Scene
  - RayTracer::Scene, [144](#)
- ~SfmlEncoder
  - RayTracer::SfmlEncoder, [149](#)
- ~Sphere
  - RayTracer::Sphere, [156](#)
- ~Vector3D
  - Math::Vector3D, [161](#)
- actualLib\_
  - DLLoader< T >, [54](#)
- addLight
  - RayTracer::Scene, [144](#)
- addPrimitive
  - RayTracer::Scene, [145](#)
- AmbientEntry.cpp
  - entryPoint, [210](#)
- AmbientLight
  - RayTracer::AmbientLight, [19](#)
- angle\_
  - RayTracer::Cone, [36](#)
- APrimitives.hpp
  - PI\_RAD, [189](#)
- b
  - RayTracer::Color, [30](#)
- backgroundColor\_
  - RayTracer::Camera, [26](#)
- bottom\_side
  - Math::Rectangle3D, [134](#)
- BYTE\_OF\_RGBA\_FORMAT
  - Encoder.hpp, [207](#)
- cam\_
  - RayTracer::Parser, [109](#)
  - RayTracer::Scene, [146](#)
- Camera
  - RayTracer::Camera, [25](#)
- Camera.hpp
  - PI\_RAD, [174](#)
- canClose
  - RayTracer::Encoder, [57](#)
  - RayTracer::IEncoder, [68](#)
  - RayTracer::SfmlEncoder, [150](#)
- canClose\_
  - RayTracer::Encoder, [59](#)
  - RayTracer::SfmlEncoder, [152](#)
- canvasToViewport
  - RayTracer::Camera, [25](#)
- canWrite
  - RayTracer::FastRenderer, [62](#)
  - RayTracer::IRenderer, [85](#)
  - RayTracer::LiveRenderer, [90](#)
  - RayTracer::Renderer, [138](#)
- canWrite\_
  - RayTracer::FastRenderer, [65](#)
  - RayTracer::LiveRenderer, [93](#)
  - RayTracer::Renderer, [141](#)
- center\_
  - RayTracer::Sphere, [158](#)

- checkArgs
  - CheckArgs.cpp, 200
  - Main.cpp, 195
- CheckArgs.cpp
  - checkArgs, 200
  - displayUsage, 200
- checkEvents
  - RayTracer::Encoder, 57
  - RayTracer::IEncoder, 69
  - RayTracer::SfmlEncoder, 150
- checkJsonCamera
  - RayTracer::Parser, 103
- checkJsonExistence
  - RayTracer::AmbientLight, 19
  - RayTracer::Cone, 34
  - RayTracer::Cylinder, 46
  - RayTracer::DirectionalLight, 51
  - RayTracer::Parser, 104
  - RayTracer::Plan, 117
  - RayTracer::PointLight, 127
  - RayTracer::Sphere, 156
- checkJsonExistenceLight
  - RayTracer::AmbientLight, 19
  - RayTracer::DirectionalLight, 51
  - RayTracer::PointLight, 127
- checkJsonExistencePrimitive
  - RayTracer::Cone, 34
  - RayTracer::Cylinder, 46
  - RayTracer::Plan, 117
  - RayTracer::Sphere, 157
- checkJsonGlobal
  - RayTracer::Parser, 104
- checkRangeValue
  - RayTracer::Cone, 35
  - RayTracer::Cylinder, 47
  - RayTracer::Parser, 104
  - RayTracer::Plan, 117
  - RayTracer::Sphere, 157
- clearWindow
  - RayTracer::SfmlEncoder, 150
- closeLib
  - DLLoader< T >, 54
- closestPrim
  - RayTracer::intersection, 76
- closestT
  - RayTracer::intersection, 77
- closeWindow
  - RayTracer::SfmlEncoder, 150
- Color
  - RayTracer::Color, 28
- color\_
  - RayTracer::ALight, 16
  - RayTracer::Material, 99
- computeDiffuseLight
  - RayTracer::ALight, 15
  - RayTracer::ILight, 71
- computeFast
  - RayTracer::AmbientLight, 19
- RayTracer::DirectionalLight, 51
- RayTracer::ILight, 72
- RayTracer::PointLight, 128
- computeLight
  - RayTracer::AmbientLight, 20
  - RayTracer::DirectionalLight, 52
  - RayTracer::FastRenderer, 63
  - RayTracer::ILight, 72
  - RayTracer::LiveRenderer, 91
  - RayTracer::PointLight, 128
  - RayTracer::Renderer, 139
- computeScene
  - RayTracer::FastRenderer, 63
  - RayTracer::IRenderer, 85
  - RayTracer::LiveRenderer, 91
  - RayTracer::Renderer, 139
- computeSpecular
  - RayTracer::ALight, 15
  - RayTracer::ILight, 72
- Cone
  - RayTracer::Cone, 34
- ConeEntry.cpp
  - entryPoint, 221
- Core
  - RayTracer::Core, 38
- createWindow
  - RayTracer::SfmlEncoder, 150
- crossingPoint\_
  - RayTracer::Cone, 36
  - RayTracer::Cylinder, 48
- Cylinder
  - RayTracer::Cylinder, 46
- CylinderEntry.cpp
  - entryPoint, 224
- d
  - RayTracer::Plan, 118
- direction
  - RayTracer::Ray, 132
- direction\_
  - RayTracer::ALight, 16
- DirectionalEntry.cpp
  - entryPoint, 213
- DirectionalLight
  - RayTracer::DirectionalLight, 51
- displaySfml
  - RayTracer::SfmlEncoder, 150
- displayUsage
  - CheckArgs.cpp, 200
- DLLoader
  - DLLoader< T >, 53
- DLLoader< T >, 52
  - ~DLLoader, 54
  - actualLib\_, 54
  - closeLib, 54
  - DLLoader, 53
  - getInstance, 54
- done\_
  - RayTracer::FastRenderer, 65



- RayTracer::LiveRenderer, 93
- RayTracer::Renderer, 141
- dot
  - Math::Vector3D, 161
- encodeOutput
  - RayTracer::Encoder, 57
  - RayTracer::IEncoder, 69
  - RayTracer::SfmlEncoder, 151
- Encoder
  - RayTracer::Encoder, 57
- Encoder.hpp
  - BYTE\_OF\_RGBA\_FORMAT, 207
  - FRAME\_RATE\_LIMIT, 207
  - HEADER\_MAGIC\_NUMBER, 207
  - MAX\_VALUE, 208
- encoder\_
  - RayTracer::Core, 42
- EncoderEntry.cpp
  - entryPoint, 208, 209
- encoderLoader\_
  - RayTracer::Core, 42
- encoderName\_
  - RayTracer::Parser, 110
- entryPoint
  - AmbientEntry.cpp, 210
  - ConeEntry.cpp, 221
  - CylinderEntry.cpp, 224
  - DirectionalEntry.cpp, 213
  - EncoderEntry.cpp, 208, 209
  - PlanEntry.cpp, 228
  - PointEntry.cpp, 217
  - RendererEntry.cpp, 236–238
  - SphereEntry.cpp, 231
- EPITECH\_ERROR
  - Error.hpp, 177
- Error.hpp
  - EPITECH\_ERROR, 177
- fasterRenderEnabled\_
  - RayTracer::Parser, 110
- fasterRenderName\_
  - RayTracer::Parser, 110
- FastRenderer
  - RayTracer::FastRenderer, 62
- FRAME\_RATE\_LIMIT
  - Encoder.hpp, 207
- frameRateLimit\_
  - RayTracer::SfmlEncoder, 152
- g
  - RayTracer::Color, 30
- getB
  - RayTracer::Color, 29
- getBackgroundColor
  - RayTracer::Camera, 26
- getBIntensity
  - RayTracer::Color, 29
- getCamera
  - RayTracer::Scene, 145
- getClosestIntersection
  - RayTracer::ALight, 15
  - RayTracer::FastRenderer, 63
  - RayTracer::ILight, 72
  - RayTracer::LiveRenderer, 91
  - RayTracer::Renderer, 139
- getColor
  - RayTracer::Material, 97
- getEncoder
  - RayTracer::Core, 38
- getEncoderName
  - RayTracer::IParser, 79
  - RayTracer::Parser, 105
- getFastRendererFileName
  - RayTracer::IParser, 79
  - RayTracer::Parser, 105
- getG
  - RayTracer::Color, 29
- getGIntensity
  - RayTracer::Color, 29
- getInstance
  - DLoader< T >, 54
- getIntersection
  - RayTracer::Cone, 35
  - RayTracer::Cylinder, 47
  - RayTracer::IPrimitives, 83
  - RayTracer::Plan, 117
  - RayTracer::Sphere, 157
- getLights
  - RayTracer::IParser, 80
  - RayTracer::Parser, 105
  - RayTracer::Scene, 145
- getMaterial
  - RayTracer::APrimitives, 23
  - RayTracer::IPrimitives, 83
- getNormalVector
  - RayTracer::Cone, 35
  - RayTracer::Cylinder, 47
  - RayTracer::IPrimitives, 83
  - RayTracer::Plan, 117
  - RayTracer::Sphere, 157
- getOutputFileName
  - RayTracer::IParser, 80
  - RayTracer::Parser, 105
- getParsedCamera
  - RayTracer::Parser, 106
- getParser
  - RayTracer::Core, 39
- getPrimitives
  - RayTracer::IParser, 80
  - RayTracer::Parser, 106
  - RayTracer::Scene, 145
- getR
  - RayTracer::Color, 29
- getReflective
  - RayTracer::Material, 97
- getRenderer

- RayTracer::Core, 39
- getRendererName
  - RayTracer::IParser, 80
  - RayTracer::Parser, 106
- getResult
  - RayTracer::FastRenderer, 63
  - RayTracer::IRenderer, 86
  - RayTracer::LiveRenderer, 91
  - RayTracer::Renderer, 139
- getRIntensity
  - RayTracer::Color, 30
- getRotationVector
  - RayTracer::Cone, 35
  - RayTracer::Cylinder, 47
  - RayTracer::IPrimitives, 83
  - RayTracer::Plan, 117
  - RayTracer::Sphere, 157
- getScene
  - RayTracer::FastRenderer, 63
  - RayTracer::IRenderer, 86
  - RayTracer::LiveRenderer, 91
  - RayTracer::Renderer, 139
- getSpecular
  - RayTracer::Material, 97
- getTransparency
  - RayTracer::Material, 97
- handleCommand
  - RayTracer::Core, 39
- handleRealPreview
  - RayTracer::Core, 39
- HEADER\_MAGIC\_NUMBER
  - Encoder.hpp, 207
- height
  - RayTracer::imageSize, 75
- height\_
  - RayTracer::SfmlEncoder, 152
- ILight.hpp
  - json, 180
- imageSize\_
  - RayTracer::Camera, 26
- importedScene\_
  - RayTracer::Parser, 110
- include/Camera.hpp, 173
- include/Core.hpp, 174
- include/DLLoader.hpp, 175
- include/encoder/IEncoder.hpp, 176
- include/Error.hpp, 177
- include/lights/ALight.hpp, 178
- include/lights/ILight.hpp, 179
- include/material/Color.hpp, 180
- include/material/Material.hpp, 181
- include/maths/Point3D.hpp, 181
- include/maths/Ray.hpp, 183
- include/maths/Rectangle3D.hpp, 184
- include/maths/Vector3D.hpp, 185
- include/parser/IParser.hpp, 186
- include/parser/Parser.hpp, 186
- include/primitives/APrimitives.hpp, 188
- include/primitives/IPrimitives.hpp, 189
- include/RayTracer.hpp, 190
- include/renderer/IRenderer.hpp, 191
- include/Scene.hpp, 192
- include/Size.hpp, 193
- index
  - RayTracer::FastRenderer, 65
  - RayTracer::LiveRenderer, 93
  - RayTracer::Renderer, 141
- initIterators
  - RayTracer::FastRenderer, 64
  - RayTracer::IRenderer, 86
  - RayTracer::LiveRenderer, 92
  - RayTracer::Renderer, 140
- intensity\_
  - RayTracer::ALight, 16
- IPrimitives.hpp
  - json, 190
- isCommand
  - RayTracer::FastRenderer, 64
  - RayTracer::IRenderer, 86
  - RayTracer::LiveRenderer, 92
  - RayTracer::Renderer, 140
- isFastRendererEnabled
  - RayTracer::IParser, 81
  - RayTracer::Parser, 106
- isGenerationDone
  - RayTracer::FastRenderer, 64
  - RayTracer::IRenderer, 87
  - RayTracer::LiveRenderer, 92
  - RayTracer::Renderer, 140
- isShadowIntersection
  - RayTracer::ALight, 15
  - RayTracer::ILight, 73
- json
  - ILight.hpp, 180
  - IPrimitives.hpp, 190
  - Parser.hpp, 187
  - Plan.hpp, 227
  - Sphere.hpp, 230
- left\_side
  - Math::Rectangle3D, 135
- length
  - Math::Vector3D, 161
- lightLoader\_
  - RayTracer::Parser, 110
- lights\_
  - RayTracer::Parser, 110
  - RayTracer::Scene, 146
- lightVector\_
  - RayTracer::ALight, 16
- LiveRenderer
  - RayTracer::LiveRenderer, 90
- loadEncoder
  - RayTracer::Core, 39
- loadImage

- RayTracer::SfmlEncoder, 151
- loadLight
  - RayTracer::Parser, 107
- loadParser
  - RayTracer::Core, 40
- loadPrimitive
  - RayTracer::Parser, 107
- loadRenderer
  - RayTracer::Core, 40
- loadScene
  - RayTracer::Core, 40
- magicNumber\_
  - RayTracer::SfmlEncoder, 152
- main
  - Main.cpp, 195
- Main.cpp
  - checkArgs, 195
  - main, 195
- Material
  - RayTracer::Material, 96
- material\_
  - RayTracer::APrimitives, 23
- Math, 9
  - operator<<, 9, 10
- Math::Point3D, 119
  - ~Point3D, 121
  - operator+, 121
  - operator=, 122
  - Point3D, 120, 121
  - rotateX, 122
  - rotateY, 123
  - rotateZ, 123
  - x, 123
  - y, 123
  - z, 123
- Math::Rectangle3D, 132
  - ~Rectangle3D, 134
  - bottom\_side, 134
  - left\_side, 135
  - origin, 135
  - Rectangle3D, 134
- Math::Vector3D, 158
  - ~Vector3D, 161
  - dot, 161
  - length, 161
  - operator\*, 162
  - operator\*=, 162, 163
  - operator+, 163
  - operator+=, 163
  - operator-, 165
  - operator-=, 165
  - operator/, 165, 167
  - operator/=: 167
  - operator=, 169
  - rotate, 169
  - rotateX, 169
  - rotateY, 170
  - rotateZ, 170
- Vector3D, 160, 161
  - x, 170
  - y, 170
  - z, 170
- MAX\_VALUE
  - Encoder.hpp, 208
- maxValue\_
  - RayTracer::SfmlEncoder, 152
- msg\_
  - ParsingError, 113
- normalVector\_
  - RayTracer::Plan, 118
- openSupport
  - RayTracer::Encoder, 58
  - RayTracer::IEncoder, 69
  - RayTracer::SfmlEncoder, 151
- operator<<
  - Math, 9, 10
- operator\*
  - Math::Vector3D, 162
- operator\*=
  - Math::Vector3D, 162, 163
- operator+
  - Math::Point3D, 121
  - Math::Vector3D, 163
  - RayTracer::Color, 30
- operator+=
  - Math::Vector3D, 163
- operator-
  - Math::Vector3D, 165
- operator-=
  - Math::Vector3D, 165
- operator/
  - Math::Vector3D, 165, 167
- operator/=
  - Math::Vector3D, 167
- operator=
  - Math::Point3D, 122
  - Math::Vector3D, 169
  - RayTracer::Plan, 118
- origin
  - Math::Rectangle3D, 135
  - RayTracer::Camera, 26
  - RayTracer::Ray, 132
- origin\_
  - RayTracer::ALight, 16
- outputFile\_
  - RayTracer::Parser, 110
- parse
  - RayTracer::IParser, 81
  - RayTracer::Parser, 107
- parseCamera
  - RayTracer::Parser, 108
- parseCorePlugins
  - RayTracer::Parser, 108
- parseImportedGlobal

- RayTracer::Parser, 108
- parseImportedObj
  - RayTracer::Parser, 108
- parseImportedScene
  - RayTracer::Parser, 109
- parseInfo
  - RayTracer::AmbientLight, 20
  - RayTracer::Cone, 35
  - RayTracer::Cylinder, 47
  - RayTracer::DirectionalLight, 52
  - RayTracer::ILight, 73
  - RayTracer::IPrimitives, 83
  - RayTracer::Plan, 118
  - RayTracer::PointLight, 128
  - RayTracer::Sphere, 157
- parseLight
  - RayTracer::Parser, 109
- parseObject
  - RayTracer::Parser, 109
- parsePrimitive
  - RayTracer::Parser, 109
- Parser
  - RayTracer::Parser, 103
- Parser.hpp
  - json, 187
- parser\_
  - RayTracer::Core, 42
- parserLoader\_
  - RayTracer::Core, 42
- ParsingError, 111
  - ~ParsingError, 112
  - msg\_, 113
  - ParsingError, 112
  - what, 113
- PI\_RAD
  - APrimitives.hpp, 189
  - Camera.hpp, 174
- Plan
  - RayTracer::Plan, 116
- Plan.hpp
  - json, 227
- PlanEntry.cpp
  - entryPoint, 228
- Point3D
  - Math::Point3D, 120, 121
- PointEntry.cpp
  - entryPoint, 217
- PointLight
  - RayTracer::PointLight, 127
- primitiveLoader\_
  - RayTracer::Parser, 110
- primitives\_
  - RayTracer::Parser, 111
  - RayTracer::Scene, 146
- r
  - RayTracer::Color, 30
- radius\_
  - RayTracer::Cylinder, 48
- RayTracer::Sphere, 158
- Ray
  - RayTracer::Ray, 130, 131
- RayTracer, 10
- RayTracer::ALight, 13
  - ~ALight, 15
  - color\_, 16
  - computeDiffuseLight, 15
  - computeSpecular, 15
  - direction\_, 16
  - getClosestIntersection, 15
  - intensity\_, 16
  - isShadowIntersection, 15
  - lightVector\_, 16
  - origin\_, 16
- RayTracer::AmbientLight, 17
  - AmbientLight, 19
  - checkJsonExistence, 19
  - checkJsonExistenceLight, 19
  - computeFast, 19
  - computeLight, 20
  - parseInfo, 20
- RayTracer::APrimitives, 20
  - ~APrimitives, 23
  - getMaterial, 23
  - material\_, 23
  - setMaterial, 23
- RayTracer::Camera, 24
  - ~Camera, 25
  - backgroundColor\_, 26
  - Camera, 25
  - canvasToViewport, 25
  - getBackgroundColor, 26
  - imageSize\_, 26
  - origin, 26
  - reccursionDepth\_, 27
  - rotation\_, 27
  - screen, 27
  - setBackgroundColor, 26
- RayTracer::Color, 27
  - ~Color, 28
  - b, 30
  - Color, 28
  - g, 30
  - getB, 29
  - getBIntensity, 29
  - getG, 29
  - getGIntensity, 29
  - getR, 29
  - getRIntensity, 30
  - operator+, 30
  - r, 30
- RayTracer::Cone, 31
  - angle\_, 36
  - checkJsonExistence, 34
  - checkJsonExistencePrimitive, 34
  - checkRangeValue, 35
  - Cone, 34

- crossingPoint\_, 36
- getIntersection, 35
- getNormalVector, 35
- getRotationVector, 35
- parseInfo, 35
- rotation\_, 36
- RayTracer::Core, 36
  - ~Core, 38
  - Core, 38
  - encoder\_, 42
  - encoderLoader\_, 42
  - getEncoder, 38
  - getParser, 39
  - getRenderer, 39
  - handleCommand, 39
  - handleRealPreview, 39
  - loadEncoder, 39
  - loadParser, 40
  - loadRenderer, 40
  - loadScene, 40
  - parser\_, 42
  - parserLoader\_, 42
  - renderer\_, 42
  - rendererLoader\_, 42
  - renderImage, 40
  - resetPrimitives, 41
  - setEncoder, 41
  - setParser, 41
  - setRenderer, 41
  - startCli, 42
- RayTracer::Cylinder, 43
  - checkJsonExistence, 46
  - checkJsonExistencePrimitive, 46
  - checkRangeValue, 47
  - crossingPoint\_, 48
  - Cylinder, 46
  - getIntersection, 47
  - getNormalVector, 47
  - getRotationVector, 47
  - parseInfo, 47
  - radius\_, 48
  - rotation\_, 48
- RayTracer::DirectionalLight, 48
  - checkJsonExistence, 51
  - checkJsonExistenceLight, 51
  - computeFast, 51
  - computeLight, 52
  - DirectionalLight, 51
  - parseInfo, 52
- RayTracer::Encoder, 55
  - ~Encoder, 57
  - canClose, 57
  - canClose\_, 59
  - checkEvents, 57
  - encodeOutput, 57
  - Encoder, 57
  - openSupport, 58
  - result\_, 59
  - writeHeaderToPpm, 58
  - writePixelDataToPpm, 58
- RayTracer::FastRenderer, 59
  - ~FastRenderer, 62
  - canWrite, 62
  - canWrite\_, 65
  - computeLight, 63
  - computeScene, 63
  - done\_, 65
  - FastRenderer, 62
  - getClosestIntersection, 63
  - getResult, 63
  - getScene, 63
  - index, 65
  - initIterators, 64
  - isCommand, 64
  - isGenerationDone, 64
  - reflectRay, 64
  - result\_, 65
  - scene\_, 65
  - traceRay, 65
  - x, 66
  - y, 66
- RayTracer::IEncoder, 66
  - ~IEncoder, 68
  - canClose, 68
  - checkEvents, 69
  - encodeOutput, 69
  - openSupport, 69
- RayTracer::ILight, 70
  - ~ILight, 71
  - computeDiffuseLight, 71
  - computeFast, 72
  - computeLight, 72
  - computeSpecular, 72
  - getClosestIntersection, 72
  - isShadowIntersection, 73
  - parseInfo, 73
- RayTracer::imageSize, 75
  - height, 75
  - width, 75
- RayTracer::intersection, 76
  - closestPrim, 76
  - closestT, 77
- RayTracer::IParser, 77
  - getEncoderName, 79
  - getFastRendererFileName, 79
  - getLights, 80
  - getOutputFileName, 80
  - getPrimitives, 80
  - getRendererName, 80
  - isFastRendererEnabled, 81
  - parse, 81
- RayTracer::IPrimitives, 81
  - ~IPrimitives, 83
  - getIntersection, 83
  - getMaterial, 83
  - getNormalVector, 83

- getRotationVector, 83
  - parseInfo, 83
  - setMaterial, 84
- RayTracer::IRenderer, 84
  - canWrite, 85
  - computeScene, 85
  - getResult, 86
  - getScene, 86
  - initIterators, 86
  - isCommand, 86
  - isGenerationDone, 87
- RayTracer::LiveRenderer, 87
  - ~LiveRenderer, 90
  - canWrite, 90
  - canWrite\_, 93
  - computeLight, 91
  - computeScene, 91
  - done\_, 93
  - getClosestIntersection, 91
  - getResult, 91
  - getScene, 91
  - index, 93
  - initIterators, 92
  - isCommand, 92
  - isGenerationDone, 92
  - LiveRenderer, 90
  - reflectRay, 92
  - result\_, 93
  - scene\_, 93
  - traceRay, 93
  - x, 94
  - y, 94
- RayTracer::Material, 94
  - ~Material, 96
  - color\_, 99
  - getColor, 97
  - getReflective, 97
  - getSpecular, 97
  - getTransparency, 97
  - Material, 96
  - reflective\_, 99
  - setColor, 98
  - setReflective, 98
  - setSpecular, 98
  - setTransparency, 99
  - specular\_, 99
  - transparency\_, 99
- RayTracer::Parser, 100
  - ~Parser, 103
  - cam\_, 109
  - checkJsonCamera, 103
  - checkJsonExistence, 104
  - checkJsonGlobal, 104
  - checkRangeValue, 104
  - encoderName\_, 110
  - fasterRenderEnabled\_, 110
  - fasterRenderName\_, 110
  - getEncoderName, 105
  - getFastRendererFileName, 105
  - getLights, 105
  - getOutputFileName, 105
  - getParsedCamera, 106
  - getPrimitives, 106
  - getRendererName, 106
  - importedScene\_, 110
  - isFastRendererEnabled, 106
  - lightLoader\_, 110
  - lights\_, 110
  - loadLight, 107
  - loadPrimitive, 107
  - outputFile\_, 110
  - parse, 107
  - parseCamera, 108
  - parseCorePlugins, 108
  - parseImportedGlobal, 108
  - parseImportedObj, 108
  - parseImportedScene, 109
  - parseLight, 109
  - parseObject, 109
  - parsePrimitive, 109
  - Parser, 103
  - primitiveLoader\_, 110
  - primitives\_, 111
  - rendererName\_, 111
  - scene\_, 111
- RayTracer::Plan, 113
  - checkJsonExistence, 117
  - checkJsonExistencePrimitive, 117
  - checkRangeValue, 117
  - d, 118
  - getIntersection, 117
  - getNormalVector, 117
  - getRotationVector, 117
  - normalVector\_, 118
  - operator=, 118
  - parseInfo, 118
  - Plan, 116
  - rotation\_, 118
- RayTracer::PointLight, 124
  - ~PointLight, 127
  - checkJsonExistence, 127
  - checkJsonExistenceLight, 127
  - computeFast, 128
  - computeLight, 128
  - parseInfo, 128
  - PointLight, 127
- RayTracer::Ray, 129
  - ~Ray, 131
  - direction, 132
  - origin, 132
  - Ray, 130, 131
  - rotateRay, 131
- RayTracer::Renderer, 135
  - ~Renderer, 138
  - canWrite, 138
  - canWrite\_, 141

- computeLight, 139
- computeScene, 139
- done\_, 141
- getClosestIntersection, 139
- getResult, 139
- getScene, 139
- index, 141
- initIterators, 140
- isCommand, 140
- isGenerationDone, 140
- reflectRay, 140
- Renderer, 138
- result\_, 141
- scene\_, 141
- traceRay, 141
- x, 142
- y, 142
- RayTracer::Scene, 142
  - ~Scene, 144
  - addLight, 144
  - addPrimitive, 145
  - cam\_, 146
  - getCamera, 145
  - getLights, 145
  - getPrimitives, 145
  - lights\_, 146
  - primitives\_, 146
  - Scene, 144
  - setCamera, 146
- RayTracer::SfmlEncoder, 147
  - ~SfmlEncoder, 149
  - canClose, 150
  - canClose\_, 152
  - checkEvents, 150
  - clearWindow, 150
  - closeWindow, 150
  - createWindow, 150
  - displaySfml, 150
  - encodeOutput, 151
  - frameRateLimit\_, 152
  - height\_, 152
  - loadImage, 151
  - magicNumber\_, 152
  - maxValue\_, 152
  - openSupport, 151
  - renderWindow, 152
  - result\_, 153
  - setFrameRateLimit, 152
  - SfmlEncoder, 149
  - width\_, 153
  - window\_, 153
- RayTracer::SfmlEncoder::Image, 74
  - sprite, 74
  - texture, 74
- RayTracer::Sphere, 153
  - ~Sphere, 156
  - center\_, 158
  - checkJsonExistence, 156
  - checkJsonExistencePrimitive, 157
  - checkRangeValue, 157
  - getIntersection, 157
  - getNormalVector, 157
  - getRotationVector, 157
  - parseInfo, 157
  - radius\_, 158
  - Sphere, 156
- reccursionDepth\_
  - RayTracer::Camera, 27
- Rectangle3D
  - Math::Rectangle3D, 134
- reflective\_
  - RayTracer::Material, 99
- reflectRay
  - RayTracer::FastRenderer, 64
  - RayTracer::LiveRenderer, 92
  - RayTracer::Renderer, 140
- Renderer
  - RayTracer::Renderer, 138
- renderer\_
  - RayTracer::Core, 42
- RendererEntry.cpp
  - entryPoint, 236–238
- rendererLoader\_
  - RayTracer::Core, 42
- rendererName\_
  - RayTracer::Parser, 111
- renderImage
  - RayTracer::Core, 40
- renderWindow
  - RayTracer::SfmlEncoder, 152
- resetPrimitives
  - RayTracer::Core, 41
- result\_
  - RayTracer::Encoder, 59
  - RayTracer::FastRenderer, 65
  - RayTracer::LiveRenderer, 93
  - RayTracer::Renderer, 141
  - RayTracer::SfmlEncoder, 153
- rotate
  - Math::Vector3D, 169
- rotateRay
  - RayTracer::Ray, 131
- rotateX
  - Math::Point3D, 122
  - Math::Vector3D, 169
- rotateY
  - Math::Point3D, 123
  - Math::Vector3D, 170
- rotateZ
  - Math::Point3D, 123
  - Math::Vector3D, 170
- rotation\_
  - RayTracer::Camera, 27
  - RayTracer::Cone, 36
  - RayTracer::Cylinder, 48
  - RayTracer::Plan, 118

- Scene
  - RayTracer::Scene, 144
- scene\_
  - RayTracer::FastRenderer, 65
  - RayTracer::LiveRenderer, 93
  - RayTracer::Parser, 111
  - RayTracer::Renderer, 141
- screen
  - RayTracer::Camera, 27
- setBackgroundColor
  - RayTracer::Camera, 26
- setCamera
  - RayTracer::Scene, 146
- setColor
  - RayTracer::Material, 98
- setEncoder
  - RayTracer::Core, 41
- setFrameRateLimit
  - RayTracer::SfmlEncoder, 152
- setMaterial
  - RayTracer::APrimitives, 23
  - RayTracer::IPrimitives, 84
- setParser
  - RayTracer::Core, 41
- setReflective
  - RayTracer::Material, 98
- setRenderer
  - RayTracer::Core, 41
- setSpecular
  - RayTracer::Material, 98
- setTransparency
  - RayTracer::Material, 99
- SfmlEncoder
  - RayTracer::SfmlEncoder, 149
- specular\_
  - RayTracer::Material, 99
- Sphere
  - RayTracer::Sphere, 156
- Sphere.hpp
  - json, 230
- SphereEntry.cpp
  - entryPoint, 231
- sprite
  - RayTracer::SfmlEncoder::Image, 74
- src/Core.cpp, 193
- src/lights/ALight.cpp, 194
- src/Main.cpp, 195
- src/math/Point3D.cpp, 196
- src/math/Ray.cpp, 196
- src/math/Rectangle3D.cpp, 197
- src/math/Vector3D.cpp, 198
- src/parser/CheckArgs.cpp, 199
- src/parser/CheckJsonExistence.cpp, 201
- src/parser/Error.cpp, 201
- src/parser/ParseImportedScene.cpp, 202
- src/parser/ParseLight.cpp, 202
- src/parser/ParsePrimitive.cpp, 203
- src/parser/Parser.cpp, 203
- src/plugins/encoder/ppmEncoder/Encoder.cpp, 204
- src/plugins/encoder/ppmEncoder/Encoder.hpp, 205
- src/plugins/encoder/ppmEncoder/EncoderEntry.cpp, 208
- src/plugins/encoder/sfmlEncoder/Encoder.cpp, 205
- src/plugins/encoder/sfmlEncoder/Encoder.hpp, 206
- src/plugins/encoder/sfmlEncoder/EncoderEntry.cpp, 209
- src/plugins/lights/ambient/AmbientEntry.cpp, 210
- src/plugins/lights/ambient/AmbientLight.cpp, 211
- src/plugins/lights/ambient/AmbientLight.hpp, 211
- src/plugins/lights/directional/DirectionalEntry.cpp, 213
- src/plugins/lights/directional/DirectionalLight.cpp, 214
- src/plugins/lights/directional/DirectionalLight.hpp, 215
- src/plugins/lights/point/PointEntry.cpp, 216
- src/plugins/lights/point/PointLight.cpp, 217
- src/plugins/lights/point/PointLight.hpp, 218
- src/plugins/primitives/cone/Cone.cpp, 219
- src/plugins/primitives/cone/Cone.hpp, 220
- src/plugins/primitives/cone/ConeEntry.cpp, 221
- src/plugins/primitives/cylinder/Cylinder.cpp, 222
- src/plugins/primitives/cylinder/Cylinder.hpp, 222
- src/plugins/primitives/cylinder/CylinderEntry.cpp, 224
- src/plugins/primitives/plan/Plan.cpp, 225
- src/plugins/primitives/plan/Plan.hpp, 225
- src/plugins/primitives/plan/PlanEntry.cpp, 227
- src/plugins/primitives/sphere/Sphere.cpp, 228
- src/plugins/primitives/sphere/Sphere.hpp, 229
- src/plugins/primitives/sphere/SphereEntry.cpp, 230
- src/plugins/renderer/baseRenderer/Renderer.cpp, 231
- src/plugins/renderer/baseRenderer/Renderer.hpp, 233
- src/plugins/renderer/baseRenderer/RendererEntry.cpp, 236
- src/plugins/renderer/fastRenderer/Renderer.cpp, 231
- src/plugins/renderer/fastRenderer/Renderer.hpp, 234
- src/plugins/renderer/fastRenderer/RendererEntry.cpp, 237
- src/plugins/renderer/liveRenderer/Renderer.cpp, 232
- src/plugins/renderer/liveRenderer/Renderer.hpp, 235
- src/plugins/renderer/liveRenderer/RendererEntry.cpp, 238
- src/Utils/Camera.cpp, 239
- src/Utils/Color.cpp, 239
- src/Utils/Material.cpp, 240
- src/Utils/Primitives.cpp, 240
- src/Utils/Scene.cpp, 241
- startCli
  - RayTracer::Core, 42
- texture
  - RayTracer::SfmlEncoder::Image, 74
- traceRay
  - RayTracer::FastRenderer, 65
  - RayTracer::LiveRenderer, 93
  - RayTracer::Renderer, 141
- transparency\_
  - RayTracer::Material, 99
- Vector3D



- Math::Vector3D, [160](#), [161](#)
- what
  - ParsingError, [113](#)
- width
  - RayTracer::imageSize, [75](#)
- width\_
  - RayTracer::SfmlEncoder, [153](#)
- window\_
  - RayTracer::SfmlEncoder, [153](#)
- writeHeaderToPpm
  - RayTracer::Encoder, [58](#)
- writePixelDataToPpm
  - RayTracer::Encoder, [58](#)
- x
  - Math::Point3D, [123](#)
  - Math::Vector3D, [170](#)
  - RayTracer::FastRenderer, [66](#)
  - RayTracer::LiveRenderer, [94](#)
  - RayTracer::Renderer, [142](#)
- y
  - Math::Point3D, [123](#)
  - Math::Vector3D, [170](#)
  - RayTracer::FastRenderer, [66](#)
  - RayTracer::LiveRenderer, [94](#)
  - RayTracer::Renderer, [142](#)
- z
  - Math::Point3D, [123](#)
  - Math::Vector3D, [170](#)