

## Simple Analyzer

Erzeugt von Doxygen 1.8.6

Son Mai 25 2014 18:26:45



# Inhaltsverzeichnis

<b>1</b>	<b>SimpleAnalyzer</b>	<b>1</b>
<b>2</b>	<b>SimpleAnalyzer</b>	<b>3</b>
<b>3</b>	<b>Verzeichnis der Namensbereiche</b>	<b>5</b>
3.1	Liste aller Namensbereiche . . . . .	5
<b>4</b>	<b>Hierarchie-Verzeichnis</b>	<b>7</b>
4.1	Klassenhierarchie . . . . .	7
<b>5</b>	<b>Klassen-Verzeichnis</b>	<b>9</b>
5.1	Auflistung der Klassen . . . . .	9
<b>6</b>	<b>Datei-Verzeichnis</b>	<b>11</b>
6.1	Auflistung der Dateien . . . . .	11
<b>7</b>	<b>Dokumentation der Namensbereiche</b>	<b>13</b>
7.1	std-Namensbereichsreferenz . . . . .	13
7.2	Utils-Namensbereichsreferenz . . . . .	13
7.2.1	Ausführliche Beschreibung . . . . .	14
7.2.2	Dokumentation der Aufzählungstypen . . . . .	14
7.2.2.1	PIM_algorithm . . . . .	14
7.2.3	Dokumentation der Funktionen . . . . .	14
7.2.3.1	clampHue . . . . .	14
7.2.3.2	copySensorPoint . . . . .	15
7.2.3.3	floattostr . . . . .	15
7.2.3.4	floattowxstr . . . . .	16
7.2.3.5	floattowxstr . . . . .	17
7.2.3.6	getPointValue . . . . .	18
7.2.3.7	hsvToRgb . . . . .	19
7.2.3.8	nextCombination . . . . .	20
7.2.3.9	pointInsideMesh . . . . .	21
7.2.3.10	pointInsideTetrahedron . . . . .	22
7.2.3.11	pointInsideTetrahedron . . . . .	23

7.2.3.12	pointInsideTetrahedron	23
7.2.3.13	rayIntersectsTriangle	24
7.2.3.14	sqr	25
<b>8</b>	<b>Klassen-Dokumentation</b>	<b>27</b>
8.1	Analyzer Klassenreferenz	27
8.1.1	Ausführliche Beschreibung	28
8.1.2	Beschreibung der Konstruktoren und Destruktoren	28
8.1.2.1	Analyzer	28
8.1.2.2	~Analyzer	28
8.1.3	Dokumentation der Elementfunktionen	28
8.1.3.1	analyzeObject	28
8.1.3.2	analyzePoint	29
8.1.4	Freundbeziehungen und Funktionsdokumentation	30
8.1.4.1	operator<<	30
8.2	Analyzer::AnalyzerData_dataset Strukturreferenz	31
8.2.1	Ausführliche Beschreibung	31
8.2.2	Dokumentation der Datenelemente	31
8.2.2.1	heat_energy	31
8.2.2.2	mat_data	32
8.2.2.3	name	32
8.3	Analyzer::AnalyzerData_material Strukturreferenz	32
8.3.1	Ausführliche Beschreibung	32
8.3.2	Dokumentation der Datenelemente	32
8.3.2.1	heat_energy	32
8.3.2.2	name	32
8.3.2.3	volume	33
8.4	Analyzer::AnalyzerData_object Strukturreferenz	33
8.4.1	Ausführliche Beschreibung	34
8.4.2	Dokumentation der Datenelemente	34
8.4.2.1	data_sets	34
8.4.2.2	volume	34
8.5	Analyzer::AnalyzerData_point Strukturreferenz	34
8.5.1	Ausführliche Beschreibung	34
8.5.2	Dokumentation der Datenelemente	35
8.5.2.1	extrapolated	35
8.5.2.2	value	35
8.6	CsvToSdConverter Klassenreferenz	35
8.6.1	Ausführliche Beschreibung	36
8.6.2	Dokumentation der Elementfunktionen	36

8.6.2.1	<a href="#">contains</a>	36
8.6.2.2	<a href="#">contains</a>	37
8.6.2.3	<a href="#">convert</a>	37
8.6.2.4	<a href="#">getTextBlock</a>	37
8.6.2.5	<a href="#">parseArguments</a>	38
8.6.2.6	<a href="#">parseLine</a>	38
8.6.2.7	<a href="#">readConfiguration</a>	38
8.6.2.8	<a href="#">readInputFile</a>	39
8.6.2.9	<a href="#">readSensorDefinitions</a>	39
8.6.2.10	<a href="#">replaceAll</a>	40
8.6.2.11	<a href="#">writeOutputFile</a>	41
8.6.3	<a href="#">Dokumentation der Datenelemente</a>	41
8.6.3.1	<a href="#">configpaths</a>	41
8.6.3.2	<a href="#">NUMBEROFPATHS</a>	41
8.6.3.3	<a href="#">opts</a>	41
8.7	<a href="#">Utils::CutRender_info Strukturreferenz</a>	42
8.7.1	<a href="#">Ausführliche Beschreibung</a>	42
8.7.2	<a href="#">Dokumentation der Datenelemente</a>	42
8.7.2.1	<a href="#">img_height</a>	42
8.7.2.2	<a href="#">img_width</a>	43
8.7.2.3	<a href="#">in_volume_algorithm</a>	43
8.7.2.4	<a href="#">mmperpixel</a>	43
8.7.2.5	<a href="#">tri</a>	43
8.8	<a href="#">Exporter Klassenreferenz</a>	43
8.8.1	<a href="#">Ausführliche Beschreibung</a>	44
8.8.2	<a href="#">Beschreibung der Konstruktoren und Destruktoren</a>	44
8.8.2.1	<a href="#">Exporter</a>	44
8.8.2.2	<a href="#">~Exporter</a>	44
8.8.3	<a href="#">Dokumentation der Elementfunktionen</a>	44
8.8.3.1	<a href="#">ExportCutCSV</a>	44
8.8.3.2	<a href="#">ExportLegacyVTK</a>	45
8.8.4	<a href="#">Dokumentation der Datenelemente</a>	45
8.8.4.1	<a href="#">CSV_SEPARATOR</a>	45
8.9	<a href="#">GUIAnalyzeOutputWindow Klassenreferenz</a>	46
8.9.1	<a href="#">Ausführliche Beschreibung</a>	47
8.9.2	<a href="#">Beschreibung der Konstruktoren und Destruktoren</a>	47
8.9.2.1	<a href="#">GUIAnalyzeOutputWindow</a>	47
8.9.2.2	<a href="#">~GUIAnalyzeOutputWindow</a>	47
8.9.3	<a href="#">Dokumentation der Elementfunktionen</a>	47
8.9.3.1	<a href="#">OnKeyPress</a>	47

8.9.3.2	SelectAll	48
8.9.3.3	ToClipboard	48
8.9.3.4	Update	48
8.9.4	Dokumentation der Datenelemente	48
8.9.4.1	table	48
8.10	GUIAnalyzePointWindow Klassenreferenz	49
8.10.1	Ausführliche Beschreibung	50
8.10.2	Beschreibung der Konstruktoren und Destruktoren	50
8.10.2.1	GUIAnalyzePointWindow	50
8.10.2.2	~GUIAnalyzePointWindow	50
8.10.3	Dokumentation der Elementfunktionen	50
8.10.3.1	analyzePoint	50
8.10.4	Dokumentation der Datenelemente	51
8.10.4.1	calcbt	51
8.10.4.2	interpolationModeLabel	51
8.10.4.3	interpolationModeList	51
8.10.4.4	label	51
8.10.4.5	xedit	52
8.10.4.6	yedit	52
8.10.4.7	zedit	52
8.11	GUIColorScalePanel Klassenreferenz	52
8.11.1	Ausführliche Beschreibung	54
8.11.2	Dokumentation der Aufzählungstypen	54
8.11.2.1	ScaleMode	54
8.11.3	Beschreibung der Konstruktoren und Destruktoren	54
8.11.3.1	GUIColorScalePanel	54
8.11.3.2	~GUIColorScalePanel	54
8.11.4	Dokumentation der Elementfunktionen	54
8.11.4.1	fitBounds	54
8.11.4.2	getDisplayArea	55
8.11.4.3	getFontSize	55
8.11.4.4	getImage	55
8.11.4.5	getMode	55
8.11.4.6	getStepWidth	56
8.11.4.7	getTextColor	56
8.11.4.8	getX	56
8.11.4.9	getY	56
8.11.4.10	handleMouse	56
8.11.4.11	mouseOnDisplayArea	57
8.11.4.12	paintTo	57

8.11.4.13 refresh	58
8.11.4.14 setFontSize	59
8.11.4.15 setMode	59
8.11.4.16 setStepWidth	59
8.11.4.17 setTextColor	60
8.11.5 Dokumentation der Datenelemente	60
8.11.5.1 current_mx	60
8.11.5.2 current_my	60
8.11.5.3 font_size	60
8.11.5.4 height	60
8.11.5.5 image	61
8.11.5.6 mode	61
8.11.5.7 prev_mouse_down	61
8.11.5.8 scaling	61
8.11.5.9 step_width	61
8.11.5.10 text_color	61
8.11.5.11 transforming	61
8.11.5.12 width	61
8.11.5.13 x	61
8.11.5.14 y	62
8.12 GUICutRenderWindow Klassenreferenz	62
8.12.1 Ausführliche Beschreibung	65
8.12.2 Beschreibung der Konstruktoren und Destruktoren	65
8.12.2.1 GUICutRenderWindow	65
8.12.2.2 ~GUICutRenderWindow	65
8.12.3 Dokumentation der Elementfunktionen	66
8.12.3.1 DECLARE_EVENT_TABLE	66
8.12.3.2 getCutRenderProperties	66
8.12.3.3 OnColorScaleChanged	66
8.12.3.4 OnColorScaleChanged_spin	67
8.12.3.5 OnCSColorBtClick	68
8.12.3.6 OnCutPropsChanged	68
8.12.3.7 OnExportCSV	69
8.12.3.8 OnExportImage	69
8.12.3.9 OnResize	70
8.12.3.10 OnSCutPropsChanged_spin	70
8.12.3.11 refreshVisualisation	71
8.12.3.12 renderCutBtClick	72
8.12.3.13 renderImage	73
8.12.4 Dokumentation der Datenelemente	74

8.12.4.1	calcbt	74
8.12.4.2	canvas	75
8.12.4.3	core_count	75
8.12.4.4	export_csv_bt	75
8.12.4.5	export_img_bt	75
8.12.4.6	image	75
8.12.4.7	imgHeightEdit	75
8.12.4.8	imgWidthEdit	75
8.12.4.9	mmperpixedit	75
8.12.4.10	mmperpixellabel	75
8.12.4.11	optionslbl	76
8.12.4.12	p1label	76
8.12.4.13	p1xedit	76
8.12.4.14	p1yedit	76
8.12.4.15	p1zedit	76
8.12.4.16	p2label	76
8.12.4.17	p2xedit	76
8.12.4.18	p2yedit	76
8.12.4.19	p2zedit	76
8.12.4.20	p3label	77
8.12.4.21	p3xedit	77
8.12.4.22	p3yedit	77
8.12.4.23	p3zedit	77
8.12.4.24	scalefontcolorbt	77
8.12.4.25	scalefontpropslbl	77
8.12.4.26	scalefontsizeedit	77
8.12.4.27	scaletbl	77
8.12.4.28	scalemodecb	77
8.12.4.29	scalemodelbl	78
8.12.4.30	scalestepedit	78
8.12.4.31	scroll_pane	78
8.12.4.32	threadcountedit	78
8.12.4.33	threadcountlbl	78
8.12.4.34	trilabel	78
8.12.4.35	value_img	78
8.12.4.36	widthHeightlbl	78
8.13	GUIGLCanvas Klassenreferenz	79
8.13.1	Ausführliche Beschreibung	80
8.13.2	Beschreibung der Konstruktoren und Destruktoren	80
8.13.2.1	GUIGLCanvas	80



8.13.2.2	<a href="#">~GUIGLCanvas</a>	80
8.13.3	<a href="#">Dokumentation der Elementfunktionen</a>	81
8.13.3.1	<a href="#">getRenderer</a>	81
8.13.3.2	<a href="#">OnMouseMove</a>	81
8.13.3.3	<a href="#">OnMouseWheel</a>	82
8.13.3.4	<a href="#">OnPaint</a>	82
8.13.3.5	<a href="#">OnResize</a>	82
8.13.3.6	<a href="#">refresh</a>	82
8.13.3.7	<a href="#">setRenderObject</a>	83
8.13.4	<a href="#">Dokumentation der Datenelemente</a>	83
8.13.4.1	<a href="#">do_refresh</a>	83
8.13.4.2	<a href="#">is_initialized</a>	83
8.13.4.3	<a href="#">prev_mouse_x</a>	83
8.13.4.4	<a href="#">prev_mouse_y</a>	83
8.13.4.5	<a href="#">renderer</a>	84
8.14	<a href="#">GUIMainWindow Klassenreferenz</a>	84
8.14.1	<a href="#">Ausführliche Beschreibung</a>	87
8.14.2	<a href="#">Beschreibung der Konstruktoren und Destruktoren</a>	87
8.14.2.1	<a href="#">GUIMainWindow</a>	87
8.14.2.2	<a href="#">~GUIMainWindow</a>	88
8.14.3	<a href="#">Dokumentation der Elementfunktionen</a>	88
8.14.3.1	<a href="#">addObject</a>	88
8.14.3.2	<a href="#">assignCurrentObjectProps</a>	88
8.14.3.3	<a href="#">assignViewProps</a>	89
8.14.3.4	<a href="#">getGLCanvas</a>	89
8.14.3.5	<a href="#">OnActiveObjectChange</a>	89
8.14.3.6	<a href="#">OnActiveObjectChangePopup</a>	89
8.14.3.7	<a href="#">OnActiveObjectDelete</a>	90
8.14.3.8	<a href="#">OnAnalyze</a>	90
8.14.3.9	<a href="#">OnAnalyzeMarkerChange</a>	90
8.14.3.10	<a href="#">OnAnalyzePoint</a>	90
8.14.3.11	<a href="#">OnAutoUpdateChange</a>	90
8.14.3.12	<a href="#">OnExportViewportImage</a>	90
8.14.3.13	<a href="#">OnExportVTK</a>	90
8.14.3.14	<a href="#">OnFindMaxTSD</a>	91
8.14.3.15	<a href="#">OnGeneralPropChange</a>	91
8.14.3.16	<a href="#">OnImmediateUpdatePropChange</a>	91
8.14.3.17	<a href="#">OnMaterialSelect</a>	91
8.14.3.18	<a href="#">OnMenuFileQuit</a>	91
8.14.3.19	<a href="#">OnMenuHelpAbout</a>	92

8.14.3.20 OnMenuImportObj . . . . .	92
8.14.3.21 OnMenuImportSD . . . . .	92
8.14.3.22 OnMenuImportTSD . . . . .	92
8.14.3.23 OnMenuOpenManual . . . . .	93
8.14.3.24 OnRecalcBtClick . . . . .	93
8.14.3.25 OnRenderCut . . . . .	93
8.14.3.26 OnResize . . . . .	93
8.14.3.27 OnSDTimelineChange . . . . .	94
8.14.3.28 OnSDTLMarkerClear . . . . .	94
8.14.3.29 OnSDTLNextMarker . . . . .	94
8.14.3.30 OnSDTLPrevMarker . . . . .	94
8.14.3.31 OnSensorDataChange . . . . .	94
8.14.3.32 OnViewPropChange . . . . .	94
8.14.3.33 OnViewPropSpinChange . . . . .	94
8.14.3.34 setActiveObject . . . . .	94
8.14.3.35 setAnalyzeWindowStatus . . . . .	94
8.14.3.36 setCutRenderWindowStatus . . . . .	95
8.14.3.37 updateObjectPropGUI . . . . .	95
8.14.3.38 updateViewPropGUI . . . . .	96
8.14.4 Dokumentation der Datenelemente . . . . .	96
8.14.4.1 analyze_window_valid . . . . .	96
8.14.4.2 analyzerframe . . . . .	97
8.14.4.3 configpaths . . . . .	97
8.14.4.4 data_directory . . . . .	97
8.14.4.5 gl_context . . . . .	97
8.14.4.6 mwAnalyzeMenu . . . . .	97
8.14.4.7 mwEditMenu . . . . .	97
8.14.4.8 mwExportMenu . . . . .	97
8.14.4.9 mwFileMenu . . . . .	97
8.14.4.10 mwHelpMenu . . . . .	98
8.14.4.11 mwImportMenu . . . . .	98
8.14.4.12 mwMenuBar . . . . .	98
8.14.4.13 NUMBEROFPATHS . . . . .	98
8.14.4.14 prop_scroll_win . . . . .	98
8.14.4.15 propbox . . . . .	98
8.14.4.16 render_cut_window_valid . . . . .	98
8.14.4.17 rendercutwindow . . . . .	98
8.14.4.18 toolbar . . . . .	98
8.14.4.19 updating . . . . .	99
8.14.4.20 view_scroll_win . . . . .	99

8.14.4.21	viewbox	99
8.15	GUIRenderCutCanvas Klassenreferenz	99
8.15.1	Ausführliche Beschreibung	101
8.15.2	Beschreibung der Konstruktoren und Destruktoren	101
8.15.2.1	GUIRenderCutCanvas	101
8.15.2.2	~GUIRenderCutCanvas	101
8.15.3	Dokumentation der Elementfunktionen	101
8.15.3.1	getScalePanel	101
8.15.3.2	onCanvasPaint	102
8.15.3.3	OnMouseDown	102
8.15.3.4	OnMouseMove	103
8.15.3.5	OnMouseWheel	103
8.15.3.6	OnResize	103
8.15.3.7	setImage	104
8.15.3.8	setValueImg	104
8.15.4	Dokumentation der Datenelemente	104
8.15.4.1	current_mx	104
8.15.4.2	current_my	104
8.15.4.3	deltaX	104
8.15.4.4	deltaY	104
8.15.4.5	image	105
8.15.4.6	mouse_to_scalepanel	105
8.15.4.7	scalepanel	105
8.15.4.8	value_img	105
8.15.4.9	zoom	105
8.16	GUITimeline Klassenreferenz	105
8.16.1	Ausführliche Beschreibung	108
8.16.2	Dokumentation der Aufzählungstypen	108
8.16.2.1	GUI_TIMELINE_STYLE	108
8.16.3	Beschreibung der Konstruktoren und Destruktoren	108
8.16.3.1	GUITimeline	108
8.16.3.2	~GUITimeline	109
8.16.4	Dokumentation der Elementfunktionen	109
8.16.4.1	calcStepWidth	109
8.16.4.2	clearMarkers	109
8.16.4.3	findMaxValue	110
8.16.4.4	getMarkers	110
8.16.4.5	getMaxValue	110
8.16.4.6	getMinValue	111
8.16.4.7	getValue	111

8.16.4.8	isMarked	111
8.16.4.9	OnKeyDown	112
8.16.4.10	OnMouseDown	112
8.16.4.11	OnMouseMove	112
8.16.4.12	OnMouseWheel	113
8.16.4.13	OnPaint	113
8.16.4.14	OnResize	113
8.16.4.15	posToVal	113
8.16.4.16	sendTimelineEvent	114
8.16.4.17	setMarked	114
8.16.4.18	setMarkerList	114
8.16.4.19	setMarkers	115
8.16.4.20	setMaxValue	115
8.16.4.21	setMinValue	115
8.16.4.22	setNameList	116
8.16.4.23	setValue	116
8.16.5	Dokumentation der Datenelemente	117
8.16.5.1	delta_v_view	117
8.16.5.2	markers	117
8.16.5.3	maxdigits	117
8.16.5.4	maxvalue	117
8.16.5.5	minvalue	117
8.16.5.6	names	117
8.16.5.7	prev_mouse_x	117
8.16.5.8	value	118
8.16.5.9	zoom	118
8.17	Importer Klassenreferenz	118
8.17.1	Ausführliche Beschreibung	118
8.17.2	Beschreibung der Konstruktoren und Destruktoren	118
8.17.2.1	Importer	118
8.17.2.2	~Importer	119
8.17.3	Dokumentation der Elementfunktionen	119
8.17.3.1	ImportObj	119
8.17.3.2	LoadSensorData	119
8.17.3.3	LoadTimedData	120
8.18	Interpolator Klassenreferenz	121
8.18.1	Ausführliche Beschreibung	122
8.18.2	Dokumentation der Aufzählungstypen	122
8.18.2.1	InterpolationMode	122
8.18.3	Beschreibung der Konstruktoren und Destruktoren	122

8.18.3.1	Interpolator	122
8.18.3.2	~Interpolator	122
8.18.4	Dokumentation der Elementfunktionen	123
8.18.4.1	interpolateTet	123
8.18.4.2	interpolateTri	124
8.18.4.3	setMode	125
8.18.5	Dokumentation der Datenelemente	126
8.18.5.1	mode	126
8.19	ObjectData::MaterialData Strukturreferenz	126
8.19.1	Ausführliche Beschreibung	127
8.19.2	Dokumentation der Datenelemente	127
8.19.2.1	color	127
8.19.2.2	density	127
8.19.2.3	extrapolated	127
8.19.2.4	interpolation_mode	127
8.19.2.5	name	127
8.19.2.6	specificheatcapacity	127
8.19.2.7	tetgeninput	127
8.19.2.8	tetgenoutput	128
8.19.2.9	visible	128
8.20	Matrix3D Klassenreferenz	128
8.20.1	Ausführliche Beschreibung	128
8.20.2	Beschreibung der Konstruktoren und Destruktoren	129
8.20.2.1	Matrix3D	129
8.20.2.2	Matrix3D	129
8.20.3	Dokumentation der Elementfunktionen	129
8.20.3.1	mult	129
8.20.3.2	mult	130
8.20.3.3	print	131
8.20.3.4	rotateX	131
8.20.3.5	rotateY	131
8.20.3.6	rotateZ	132
8.20.3.7	transpose	132
8.20.4	Dokumentation der Datenelemente	132
8.20.4.1	elements	132
8.21	MeshProcessor Klassenreferenz	133
8.21.1	Ausführliche Beschreibung	133
8.21.2	Beschreibung der Konstruktoren und Destruktoren	133
8.21.2.1	MeshProcessor	133
8.21.2.2	~MeshProcessor	133

8.21.3	Dokumentation der Elementfunktionen	133
8.21.3.1	process	133
8.22	ObjectData Klassenreferenz	134
8.22.1	Ausführliche Beschreibung	136
8.22.2	Dokumentation der Aufzählungstypen	136
8.22.2.1	ObjectDataStatus	136
8.22.3	Beschreibung der Konstruktoren und Destruktoren	137
8.22.3.1	ObjectData	137
8.22.3.2	~ObjectData	137
8.22.4	Dokumentation der Elementfunktionen	137
8.22.4.1	addSensorData	137
8.22.4.2	addTimedData	138
8.22.4.3	calculateIO	138
8.22.4.4	getCurrentSensorIndex	139
8.22.4.5	getMaterials	140
8.22.4.6	getMaxvolume	140
8.22.4.7	getName	141
8.22.4.8	getQuality	141
8.22.4.9	getSensorDataList	142
8.22.4.10	loadFromFile	142
8.22.4.11	setCurrentSensorIndex	143
8.22.4.12	setMaxvolume	143
8.22.4.13	setName	143
8.22.4.14	setQuality	144
8.22.5	Dokumentation der Datenelemente	144
8.22.5.1	current_sensor_index	144
8.22.5.2	materials	144
8.22.5.3	maxvolume	144
8.22.5.4	name	145
8.22.5.5	quality	145
8.22.5.6	sensorDataList	145
8.23	OdisiToSdConverter Klassenreferenz	145
8.23.1	Ausführliche Beschreibung	146
8.23.2	Dokumentation der Elementfunktionen	147
8.23.2.1	contains	147
8.23.2.2	contains	147
8.23.2.3	convert	147
8.23.2.4	floattostr	148
8.23.2.5	getTextBlock	148
8.23.2.6	parseArguments	148

8.23.2.7	parseLine	148
8.23.2.8	readConfiguration	149
8.23.2.9	readInputFile	149
8.23.2.10	readSensorDefinitions	150
8.23.2.11	replaceAll	150
8.23.2.12	writeOutputFile	151
8.23.3	Dokumentation der Datenelemente	151
8.23.3.1	configpaths	151
8.23.3.2	NUMBEROFPATHS	151
8.23.3.3	opts	151
8.24	CsvToSdConverter::Options Strukturreferenz	152
8.24.1	Ausführliche Beschreibung	152
8.24.2	Dokumentation der Datenelemente	152
8.24.2.1	max_time	152
8.24.2.2	min_time	152
8.24.2.3	namecol	152
8.24.2.4	replace_comma_with_point	153
8.24.2.5	separator	153
8.24.2.6	start_col	153
8.24.2.7	time_step_delta	153
8.24.2.8	timecol	153
8.25	TsdMerger::Options Strukturreferenz	153
8.25.1	Ausführliche Beschreibung	153
8.25.2	Dokumentation der Datenelemente	154
8.25.2.1	auto_delta	154
8.25.2.2	delta	154
8.25.2.3	max_dt	154
8.25.2.4	offset	154
8.26	OdisiToSdConverter::Options Strukturreferenz	154
8.26.1	Ausführliche Beschreibung	155
8.26.2	Dokumentation der Datenelemente	155
8.26.2.1	basetemp	155
8.26.2.2	error_threshold	155
8.26.2.3	fiber_step_delta	155
8.26.2.4	flipobj	155
8.26.2.5	height	155
8.26.2.6	max_time	156
8.26.2.7	maxfwcount	156
8.26.2.8	min_time	156
8.26.2.9	objwidth	156

8.26.2.10	replace_comma_with_point	156
8.26.2.11	separator	156
8.26.2.12	startrow	156
8.26.2.13	tab_space_count	156
8.26.2.14	time_step_delta	156
8.26.2.15	timecol	157
8.27	PropertiesBox Klassenreferenz	157
8.27.1	Ausführliche Beschreibung	160
8.27.2	Beschreibung der Konstruktoren und Destruktoren	160
8.27.2.1	PropertiesBox	160
8.27.2.2	~PropertiesBox	160
8.27.3	Dokumentation der Elementfunktionen	160
8.27.3.1	getAnalyzeMarkerCheckBox	160
8.27.3.2	getAutoUpdateCeckBox	160
8.27.3.3	getClearAnalyzeMarkerBt	161
8.27.3.4	getCurrentMaterial	161
8.27.3.5	getDensityEdit	161
8.27.3.6	getFindMaxBt	161
8.27.3.7	getInterpolationModeList	161
8.27.3.8	getMatListBox	161
8.27.3.9	getMatNameEdit	161
8.27.3.10	getMatPropBox	161
8.27.3.11	getMaxVolumeEdit	161
8.27.3.12	getNextMarkerBt	162
8.27.3.13	getObjNameEdit	162
8.27.3.14	getPrevMarkerBt	162
8.27.3.15	getQualityEdit	162
8.27.3.16	getRecalcButton	162
8.27.3.17	getSdTimeline	162
8.27.3.18	getSensorDataList	162
8.27.3.19	getSpecificHeatCapEdit	162
8.27.3.20	getUpToDateLbl	163
8.27.3.21	resize	163
8.27.3.22	setCurrentMaterial	163
8.27.4	Dokumentation der Datenelemente	163
8.27.4.1	analyzeMarkerCheckBox	163
8.27.4.2	autoUpdateCeckBox	163
8.27.4.3	clearAnalyzeMarkerBt	164
8.27.4.4	current_material	164
8.27.4.5	densityEdit	164



8.27.4.6	densityLbl	164
8.27.4.7	findMaxBt	164
8.27.4.8	interpolationModeLbl	164
8.27.4.9	interpolationModeList	164
8.27.4.10	matListBox	164
8.27.4.11	matListBoxLbl	164
8.27.4.12	matNameEdit	165
8.27.4.13	matNameLbl	165
8.27.4.14	matPropBox	165
8.27.4.15	maxVolumeEdit	165
8.27.4.16	maxVolumeLbl	165
8.27.4.17	nextMarkerBt	165
8.27.4.18	objNameEdit	165
8.27.4.19	objNameLbl	165
8.27.4.20	prevMarkerBt	165
8.27.4.21	qualityEdit	166
8.27.4.22	qualityLbl	166
8.27.4.23	recalcButton	166
8.27.4.24	sdTimeline	166
8.27.4.25	sensorDataLbl	166
8.27.4.26	sensorDataList	166
8.27.4.27	specificHeatCapEdit	166
8.27.4.28	specificHeatCapLbl	166
8.27.4.29	upToDateLbl	166
8.28	Renderer Klassenreferenz	167
8.28.1	Ausführliche Beschreibung	168
8.28.2	Dokumentation der Aufzählungstypen	168
8.28.2.1	RenderMode	168
8.28.3	Beschreibung der Konstruktoren und Destruktoren	169
8.28.3.1	Renderer	169
8.28.3.2	~Renderer	169
8.28.4	Dokumentation der Elementfunktionen	169
8.28.4.1	getViewport	169
8.28.4.2	getViewportImage	169
8.28.4.3	initGL	169
8.28.4.4	render	170
8.28.4.5	renderMaterial	171
8.28.4.6	renderSensorData	172
8.28.4.7	renderTetrahedra	172
8.28.4.8	resize	173

8.28.4.9	setCutRenderInfo	174
8.28.4.10	setObject	174
8.28.5	Dokumentation der Datenelemente	175
8.28.5.1	cut_visualisation_info	175
8.28.5.2	displayList	175
8.28.5.3	object	175
8.28.5.4	viewport	175
8.29	Utils::SensorData Strukturreferenz	175
8.29.1	Ausführliche Beschreibung	176
8.29.2	Dokumentation der Datenelemente	176
8.29.2.1	current_time_index	176
8.29.2.2	data	177
8.29.2.3	markers	177
8.29.2.4	name	177
8.29.2.5	subnames	177
8.29.2.6	timed	177
8.29.2.7	timestamps	177
8.30	Utils::SensorPoint Strukturreferenz	177
8.30.1	Ausführliche Beschreibung	178
8.30.2	Dokumentation der Datenelemente	178
8.30.2.1	coords	178
8.30.2.2	temperature	178
8.31	Utils::SensorPointComparator Strukturreferenz	178
8.31.1	Ausführliche Beschreibung	178
8.31.2	Dokumentation der Elementfunktionen	179
8.31.2.1	getDistance_d	179
8.31.2.2	operator()	180
8.31.3	Dokumentation der Datenelemente	180
8.31.3.1	meshpoint	180
8.32	SimpleAnalyzerApp Klassenreferenz	180
8.32.1	Ausführliche Beschreibung	182
8.32.2	Beschreibung der Konstruktoren und Destruktoren	182
8.32.2.1	~SimpleAnalyzerApp	182
8.32.3	Dokumentation der Elementfunktionen	182
8.32.3.1	addObject	182
8.32.3.2	getActiveObject	183
8.32.3.3	getCurrentDataObjectIndex	183
8.32.3.4	getDataObjects	183
8.32.3.5	getVisualizationInfo	183
8.32.3.6	OnInit	183

8.32.3.7	<a href="#">removeCurrentObject</a>	183
8.32.3.8	<a href="#">setCurrentDataObjectIndex</a>	183
8.32.4	<a href="#">Dokumentation der Datenelemente</a>	184
8.32.4.1	<a href="#">current_data_object_index</a>	184
8.32.4.2	<a href="#">data_objects</a>	184
8.32.4.3	<a href="#">visualization_info</a>	184
8.33	<a href="#">Utils::SortStruct Strukturreferenz</a>	184
8.33.1	<a href="#">Ausführliche Beschreibung</a>	184
8.33.2	<a href="#">Dokumentation der Datenelemente</a>	185
8.33.2.1	<a href="#">distance</a>	185
8.33.2.2	<a href="#">pointIndex</a>	185
8.34	<a href="#">Tetrahedron Klassenreferenz</a>	185
8.34.1	<a href="#">Ausführliche Beschreibung</a>	186
8.34.2	<a href="#">Beschreibung der Konstruktoren und Destruktoren</a>	186
8.34.2.1	<a href="#">Tetrahedron</a>	186
8.34.3	<a href="#">Dokumentation der Elementfunktionen</a>	186
8.34.3.1	<a href="#">getV1</a>	186
8.34.3.2	<a href="#">getV2</a>	186
8.34.3.3	<a href="#">getV3</a>	187
8.34.3.4	<a href="#">getV4</a>	187
8.34.3.5	<a href="#">getVert</a>	187
8.34.4	<a href="#">Dokumentation der Datenelemente</a>	188
8.34.4.1	<a href="#">verts</a>	188
8.35	<a href="#">Triangle Klassenreferenz</a>	188
8.35.1	<a href="#">Ausführliche Beschreibung</a>	189
8.35.2	<a href="#">Beschreibung der Konstruktoren und Destruktoren</a>	189
8.35.2.1	<a href="#">Triangle</a>	189
8.35.2.2	<a href="#">~Triangle</a>	189
8.35.3	<a href="#">Dokumentation der Elementfunktionen</a>	189
8.35.3.1	<a href="#">getNormal</a>	189
8.35.3.2	<a href="#">getV1</a>	190
8.35.3.3	<a href="#">getV2</a>	190
8.35.3.4	<a href="#">getV3</a>	191
8.35.3.5	<a href="#">getVert</a>	191
8.35.3.6	<a href="#">print</a>	192
8.35.4	<a href="#">Dokumentation der Datenelemente</a>	192
8.35.4.1	<a href="#">verts</a>	192
8.36	<a href="#">TsdMerger Klassenreferenz</a>	192
8.36.1	<a href="#">Ausführliche Beschreibung</a>	193
8.36.2	<a href="#">Dokumentation der Elementfunktionen</a>	193

8.36.2.1	getTextBlock	193
8.36.2.2	merge	193
8.36.2.3	parseArguments	194
8.36.2.4	parseFile	194
8.36.2.5	writeOutputFile	195
8.36.3	Dokumentation der Datenelemente	195
8.36.3.1	opts	195
8.37	std::vector< T > Template-Klassenreferenz	195
8.37.1	Ausführliche Beschreibung	196
8.37.2	Dokumentation der Datenelemente	196
8.37.2.1	element	196
8.38	Vector3D Klassenreferenz	196
8.38.1	Ausführliche Beschreibung	198
8.38.2	Beschreibung der Konstruktoren und Destruktoren	198
8.38.2.1	Vector3D	198
8.38.2.2	Vector3D	198
8.38.2.3	Vector3D	198
8.38.2.4	~Vector3D	199
8.38.3	Dokumentation der Elementfunktionen	199
8.38.3.1	add	199
8.38.3.2	copy	199
8.38.3.3	crossProduct	200
8.38.3.4	dotProduct	201
8.38.3.5	equals	201
8.38.3.6	getAngleTo	202
8.38.3.7	getDistanceTo	203
8.38.3.8	getLength	204
8.38.3.9	getX	205
8.38.3.10	getXYZ	205
8.38.3.11	getY	206
8.38.3.12	getZ	206
8.38.3.13	mult	207
8.38.3.14	normalize	207
8.38.3.15	print	208
8.38.3.16	printTo	208
8.38.3.17	sub	208
8.38.4	Freundbeziehungen und Funktionsdokumentation	209
8.38.4.1	operator<<	209
8.38.5	Dokumentation der Datenelemente	209
8.38.5.1	coords	209

8.39	Renderer::Viewport_info Strukturreferenz	210
8.39.1	Ausführliche Beschreibung	211
8.39.2	Dokumentation der Datenelemente	211
8.39.2.1	cameraPosition	211
8.39.2.2	cut	211
8.39.2.3	height	211
8.39.2.4	invertcut	211
8.39.2.5	rotationX	211
8.39.2.6	rotationY	211
8.39.2.7	scale	212
8.39.2.8	show_extrapolated	212
8.39.2.9	show_sensordata	212
8.39.2.10	showEdges	212
8.39.2.11	showFaces	212
8.39.2.12	showPoints	212
8.39.2.13	width	212
8.39.2.14	zoom	212
8.40	ViewpropBox Klassenreferenz	212
8.40.1	Ausführliche Beschreibung	214
8.40.2	Beschreibung der Konstruktoren und Destruktoren	214
8.40.2.1	ViewpropBox	214
8.40.2.2	~ViewpropBox	215
8.40.3	Dokumentation der Elementfunktionen	215
8.40.3.1	getColorRangeMaxEdit	215
8.40.3.2	getColorRangeMinEdit	215
8.40.3.3	getEdgesCheckBox	215
8.40.3.4	getFacesCheckBox	215
8.40.3.5	getMatVisibilityListBox	215
8.40.3.6	getPointsCheckBox	215
8.40.3.7	getShowExtrapolatedCheckBox	215
8.40.3.8	getShowShowSensorData	216
8.40.3.9	getViewScaleEdit	216
8.40.3.10	resize	216
8.40.4	Dokumentation der Datenelemente	216
8.40.4.1	colorRangeLbl	216
8.40.4.2	colorRangeMaxEdit	216
8.40.4.3	colorRangeMinEdit	216
8.40.4.4	edgesCheckBox	216
8.40.4.5	facesCheckBox	216
8.40.4.6	matVisibilityListBox	216

8.40.4.7	matVisualizationLbl	217
8.40.4.8	pointsCheckBox	217
8.40.4.9	showExtrapolatedCheckBox	217
8.40.4.10	showShowSensorData	217
8.40.4.11	viewScaleEdit	217
8.40.4.12	viewScaleLbl	217
8.41	Utils::Visualization_info Strukturreferenz	217
8.41.1	Ausführliche Beschreibung	218
8.41.2	Dokumentation der Datenelemente	218
8.41.2.1	max_visualisation_temp	218
8.41.2.2	min_visualisation_temp	218
<b>9</b>	<b>Datei-Dokumentation</b>	<b>219</b>
9.1	/daten/Projekte/eclipse_workspace/csvtosd/main.cpp-Dateireferenz	219
9.1.1	Dokumentation der Funktionen	219
9.1.1.1	main	219
9.2	/daten/Projekte/eclipse_workspace/odisitosd/main.cpp-Dateireferenz	220
9.2.1	Dokumentation der Funktionen	220
9.2.1.1	main	220
9.3	doxygen_dep_dummy.h-Dateireferenz	221
9.4	/daten/Projekte/eclipse_workspace/mergetsd/src/mergetsd.cpp-Dateireferenz	221
9.4.1	Dokumentation der Funktionen	222
9.4.1.1	main	222
9.5	/daten/Projekte/eclipse_workspace/README.md-Dateireferenz	222
9.6	/daten/Projekte/eclipse_workspace/simpleanalyzer-gui/src/fileIO/Exporter.cpp-Dateireferenz	222
9.6.1	Variablen-Dokumentation	223
9.6.1.1	tetface_indices	223
9.7	/daten/Projekte/eclipse_workspace/simpleanalyzer-gui/src/fileIO/Exporter.h-Dateireferenz	223
9.8	/daten/Projekte/eclipse_workspace/simpleanalyzer-gui/src/fileIO/Importer.cpp-Dateireferenz	224
9.8.1	Makro-Dokumentation	224
9.8.1.1	PATH_SEPARATOR	224
9.8.2	Dokumentation der Funktionen	224
9.8.2.1	getFaceIndex	224
9.8.2.2	getTextBlock	225
9.9	/daten/Projekte/eclipse_workspace/simpleanalyzer-gui/src/fileIO/Importer.h-Dateireferenz	225
9.10	/daten/Projekte/eclipse_workspace/simpleanalyzer-gui/src/GUI/constants.h-Dateireferenz	226
9.10.1	Dokumentation der Aufzählungstypen	227
9.10.1.1	EventID	228
9.10.2	Variablen-Dokumentation	229
9.10.2.1	INTERPOLATION_MODE_STRINGS	229

9.10.2.2	NUMBER_OF_INTERPOLATION_MODES . . . . .	229
9.11	/daten/Projekte/eclipse_workspace/simpleanalyzer-gui/src/GUI/GUIAnalyzeOutputWindow.cpp-- Dateireferenz . . . . .	229
9.12	/daten/Projekte/eclipse_workspace/simpleanalyzer-gui/src/GUI/GUIAnalyzeOutputWindow.h-- Dateireferenz . . . . .	229
9.13	/daten/Projekte/eclipse_workspace/simpleanalyzer-gui/src/GUI/GUIAnalyzePointWindow.cpp-- Dateireferenz . . . . .	230
9.14	/daten/Projekte/eclipse_workspace/simpleanalyzer-gui/src/GUI/GUIAnalyzePointWindow.h-Dateireferenz	231
9.15	/daten/Projekte/eclipse_workspace/simpleanalyzer-gui/src/GUI/GUIColorScalePanel.cpp-Dateireferenz	232
9.15.1	Makro-Dokumentation . . . . .	232
9.15.1.1	MIN_HEIGHT . . . . .	232
9.15.1.2	MIN_WIDTH . . . . .	232
9.16	/daten/Projekte/eclipse_workspace/simpleanalyzer-gui/src/GUI/GUIColorScalePanel.h-Dateireferenz	233
9.17	/daten/Projekte/eclipse_workspace/simpleanalyzer-gui/src/GUI/GUICutRenderWindow.cpp-- Dateireferenz . . . . .	233
9.17.1	Dokumentation der Funktionen . . . . .	234
9.17.1.1	render_thread . . . . .	234
9.18	/daten/Projekte/eclipse_workspace/simpleanalyzer-gui/src/GUI/GUICutRenderWindow.h-Dateireferenz	236
9.19	/daten/Projekte/eclipse_workspace/simpleanalyzer-gui/src/GUI/GUIGLCanvas.cpp-Dateireferenz . . . . .	237
9.19.1	Variablen-Dokumentation . . . . .	238
9.19.1.1	attrib_list . . . . .	238
9.20	/daten/Projekte/eclipse_workspace/simpleanalyzer-gui/src/GUI/GUIGLCanvas.h-Dateireferenz . . . . .	238
9.21	/daten/Projekte/eclipse_workspace/simpleanalyzer-gui/src/GUI/GUIMainWindow.cpp-Dateireferenz	239
9.21.1	Makro-Dokumentation . . . . .	239
9.21.1.1	PATH_SEPARATOR . . . . .	239
9.21.1.2	PROPBOXWIDTH . . . . .	239
9.21.1.3	VIEWBOXWIDTH . . . . .	239
9.22	/daten/Projekte/eclipse_workspace/simpleanalyzer-gui/src/GUI/GUIMainWindow.h-Dateireferenz . . . . .	240
9.23	/daten/Projekte/eclipse_workspace/simpleanalyzer-gui/src/GUI/GUIRenderCutCanvas.cpp-Dateireferenz	240
9.24	/daten/Projekte/eclipse_workspace/simpleanalyzer-gui/src/GUI/GUIRenderCutCanvas.h-Dateireferenz	241
9.25	/daten/Projekte/eclipse_workspace/simpleanalyzer-gui/src/GUI/GUITimeline.cpp-Dateireferenz . . . . .	242
9.25.1	Makro-Dokumentation . . . . .	243
9.25.1.1	SCALE_REFINE_STEPS . . . . .	243
9.25.2	Variablen-Dokumentation . . . . .	243
9.25.2.1	refine_factors . . . . .	243
9.25.2.2	wxEVT_TIMELINE_CHANGE . . . . .	243
9.26	/daten/Projekte/eclipse_workspace/simpleanalyzer-gui/src/GUI/GUITimeline.h-Dateireferenz . . . . .	243
9.26.1	Variablen-Dokumentation . . . . .	244
9.26.1.1	wxEVT_TIMELINE_CHANGE . . . . .	244
9.27	/daten/Projekte/eclipse_workspace/simpleanalyzer-gui/src/GUI/PropertiesBox.cpp-Dateireferenz . . . . .	244
9.27.1	Variablen-Dokumentation . . . . .	245

9.27.1.1	<a href="#">sdfilestring</a> . . . . .	245
9.28	<a href="#">/daten/Projekte/eclipse_workspace/simpleanalyzer-gui/src/GUI/PropertiesBox.h-Dateireferenz</a> . . . .	245
9.29	<a href="#">/daten/Projekte/eclipse_workspace/simpleanalyzer-gui/src/GUI/Renderer.cpp-Dateireferenz</a> . . . .	246
9.29.1	Dokumentation der Funktionen . . . . .	247
9.29.1.1	<a href="#">drawCutRenderInfo</a> . . . . .	247
9.29.1.2	<a href="#">drawVector</a> . . . . .	248
9.29.1.3	<a href="#">pointBehindCut</a> . . . . .	249
9.29.1.4	<a href="#">renderGrid</a> . . . . .	250
9.30	<a href="#">/daten/Projekte/eclipse_workspace/simpleanalyzer-gui/src/GUI/Renderer.h-Dateireferenz</a> . . . .	251
9.31	<a href="#">/daten/Projekte/eclipse_workspace/simpleanalyzer-gui/src/GUI/ViewpropBox.cpp-Dateireferenz</a> . .	252
9.31.1	Variablen-Dokumentation . . . . .	253
9.31.1.1	<a href="#">renderchoices</a> . . . . .	253
9.32	<a href="#">/daten/Projekte/eclipse_workspace/simpleanalyzer-gui/src/GUI/ViewpropBox.h-Dateireferenz</a> . . . .	253
9.33	<a href="#">/daten/Projekte/eclipse_workspace/simpleanalyzer-gui/src/libraries/interpolate/GeometryClasses.cpp-Dateireferenz</a> . . . . .	254
9.33.1	Makro-Dokumentation . . . . .	255
9.33.1.1	<a href="#">EPSILON</a> . . . . .	255
9.33.2	Dokumentation der Funktionen . . . . .	255
9.33.2.1	<a href="#">operator&lt;&lt;</a> . . . . .	255
9.33.2.2	<a href="#">sqr</a> . . . . .	256
9.34	<a href="#">/daten/Projekte/eclipse_workspace/simpleanalyzer-gui/src/libraries/interpolate/GeometryClasses.h-Dateireferenz</a> . . . . .	256
9.34.1	Dokumentation der Funktionen . . . . .	257
9.34.1.1	<a href="#">operator&lt;&lt;</a> . . . . .	257
9.35	<a href="#">/daten/Projekte/eclipse_workspace/simpleanalyzer-gui/src/libraries/interpolate/Interpolator.cpp--Dateireferenz</a> . . . . .	257
9.35.1	Makro-Dokumentation . . . . .	258
9.35.1.1	<a href="#">PI</a> . . . . .	258
9.35.2	Dokumentation der Funktionen . . . . .	258
9.35.2.1	<a href="#">getSign</a> . . . . .	258
9.35.2.2	<a href="#">sqr</a> . . . . .	259
9.36	<a href="#">/daten/Projekte/eclipse_workspace/simpleanalyzer-gui/src/libraries/interpolate/Interpolator.h--Dateireferenz</a> . . . . .	259
9.37	<a href="#">/daten/Projekte/eclipse_workspace/simpleanalyzer-gui/src/processing/Analyzer.cpp-Dateireferenz</a> .	260
9.37.1	Dokumentation der Funktionen . . . . .	261
9.37.1.1	<a href="#">operator&lt;&lt;</a> . . . . .	261
9.38	<a href="#">/daten/Projekte/eclipse_workspace/simpleanalyzer-gui/src/processing/Analyzer.h-Dateireferenz</a> . .	261
9.39	<a href="#">/daten/Projekte/eclipse_workspace/simpleanalyzer-gui/src/processing/MeshProcessor.cpp-Dateireferenz</a> 262	
9.39.1	Dokumentation der Funktionen . . . . .	263
9.39.1.1	<a href="#">interpolatePoint</a> . . . . .	263
9.40	<a href="#">/daten/Projekte/eclipse_workspace/simpleanalyzer-gui/src/processing/MeshProcessor.h-Dateireferenz</a> 264	



9.41	/daten/Projekte/eclipse_workspace/simpleanalyzer-gui/src/processing/ObjectData.cpp-Dateireferenz	265
9.41.1	Makro-Dokumentation	266
9.41.1.1	PATH_SEPARATOR	266
9.42	/daten/Projekte/eclipse_workspace/simpleanalyzer-gui/src/processing/ObjectData.h-Dateireferenz	266
9.42.1	Makro-Dokumentation	267
9.42.1.1	NUMBEROFSENSORATTRIBUTES	267
9.43	/daten/Projekte/eclipse_workspace/simpleanalyzer-gui/src/processing/Utils.cpp-Dateireferenz	267
9.43.1	Makro-Dokumentation	268
9.43.1.1	EPSILON	268
9.44	/daten/Projekte/eclipse_workspace/simpleanalyzer-gui/src/processing/Utils.h-Dateireferenz	268
9.45	/daten/Projekte/eclipse_workspace/simpleanalyzer-gui/src/SimpleAnalyzerApp.cpp-Dateireferenz	270
9.46	/daten/Projekte/eclipse_workspace/simpleanalyzer-gui/src/SimpleAnalyzerApp.h-Dateireferenz	271
<b>Index</b>		<b>273</b>



# Kapitel 1

## SimpleAnalyzer

Dies ist die Dokumentation der Programmquellen des SimpleAnalyzer-Programmpakets. Für Informationen über die Verwendung der Programme konsultieren Sie bitte das Handbuch der Software.

Alle in <https://github.com/vroland/SimpleAnalyzer> zur Für das Simpleanalyzer-Softwarepaket gelten die Lizenzbestimmungen der GNU Affero General Public License. Genauere Informationen sind der LIC-ENSE-Datei zu entnehmen.

### Autor

Valentin Roland



## Kapitel 2

# SimpleAnalyzer

Das SimpleAnalyzer-Softwarepaket enthält Programme zur Auswertung physikalischer Versuche für debianbasierte Betriebssysteme. Mithilfe der enthaltenen Software sind Sie im Stande, Temperaturmessdaten aus einer .csv-Datei oder Messwerte des ODiSI-Instruments von Luna in ein einheitliches Format umzuwandeln und zusammenzuführen.

Über eine grafische Oberfläche ist es möglich, mithilfe der so aufbereiteten Daten Auswertungen wie eine Temperaturverteilung über ein dreidimensionales Modell oder das Bestimmen des Wärmegehalts vorzunehmen und den Versuch zu visualisieren.

Zur weiteren Nutzung der Ergebnisse können diese, beispielsweise als VTK-Datei oder PNG-Grafik, exportiert werden.

Quelltext, Handbuch, Dokumentation und Beispiele sowie Binärdateien finden Sie unter <https://github.com/vroland/SimpleAnalyzer>.

### Lizenz

Für das Simpleanalyzer-Softwarepaket gelten die Lizenzbestimmungen der GNU Affero General Public License. Genauere Informationen sind der LICENSE-Datei zu entnehmen.

### Handbuch

Im Handbuch zum Programm finden Sie Informationen zur Installation und Bedienung der Programme. Es liegt im pdf-Format unter simpleanalyzer-gui/Debug/simpleanalyzer-man.pdf vor und kann über das Hilfemenü in SimpleAnalyzer-GUI aufgerufen werden.

### Dokumentation

Eine Dokumentation für die Weiterentwicklung der Software befindet sich im Verzeichnis doc/html.



## Kapitel 3

# Verzeichnis der Namensbereiche

### 3.1 Liste aller Namensbereiche

Liste aller Namensbereiche mit Kurzbeschreibung:

<a href="#">std</a> . . . . .	13
<a href="#">Utils</a>	
Allgemeine Funktionen und Typen . . . . .	13





## Kapitel 4

# Hierarchie-Verzeichnis

### 4.1 Klassenhierarchie

Die Liste der Ableitungen ist -mit Einschränkungen- alphabetisch sortiert:

Analyzer	27
Analyzer::AnalyzerData_dataset	31
Analyzer::AnalyzerData_material	32
Analyzer::AnalyzerData_object	33
Analyzer::AnalyzerData_point	34
CsvToSdConverter	35
Utils::CutRender_info	42
Exporter	43
GUIColorScalePanel	52
Importer	118
Interpolator	121
ObjectData::MaterialData	126
Matrix3D	128
MeshProcessor	133
ObjectData	134
OdisiToSdConverter	145
CsvToSdConverter::Options	152
TsdMerger::Options	153
OdisiToSdConverter::Options	154
Renderer	167
Utils::SensorData	175
Utils::SensorPoint	177
Utils::SensorPointComparator	178
Utils::SortStruct	184
Tetrahedron	185
Triangle	188
TsdMerger	192
std::vector< T >	195
Vector3D	196
std::vector< Analyzer::AnalyzerData_dataset >	195
std::vector< Analyzer::AnalyzerData_material >	195
std::vector< int >	195
std::vector< ObjectData * >	195
std::vector< ObjectData::MaterialData >	195
std::vector< SensorData >	195
std::vector< string >	195
std::vector< vector< Utils::SensorPoint > >	195
Renderer::Viewport_info	210

Utils::Visualization_info . . . . .	217
wxApp	
SimpleAnalyzerApp . . . . .	180
wxDialog	
GUIAnalyzePointWindow . . . . .	49
wxFrame	
GUIAnalyzeOutputWindow . . . . .	46
GUICutRenderWindow . . . . .	62
GUIMainWindow . . . . .	84
wxGLCanvas	
GUIGLCanvas . . . . .	79
wxPanel	
GUIRenderCutCanvas . . . . .	99
GUITimeline . . . . .	105
wxStaticBox	
PropertiesBox . . . . .	157
ViewpropBox . . . . .	212

## Kapitel 5

# Klassen-Verzeichnis

### 5.1 Auflistung der Klassen

Hier folgt die Aufzählung aller Klassen, Strukturen, Varianten und Schnittstellen mit einer Kurzbeschreibung:

<a href="#">Analyzer</a>	Ermittelt Daten aus der Temperaturverteilung . . . . .	27
<a href="#">Analyzer::AnalyzerData_dataset</a>	Analyseergebnisse für einen Sensordatensatz . . . . .	31
<a href="#">Analyzer::AnalyzerData_material</a>	Analyseergebnisse für ein Material . . . . .	32
<a href="#">Analyzer::AnalyzerData_object</a>	Analyseergebnisse für ein Objekt . . . . .	33
<a href="#">Analyzer::AnalyzerData_point</a>	Analyseergebnisse für einen Punkt . . . . .	34
<a href="#">CsvToSdConverter</a>	Konverter von .csv zu .tsd . . . . .	35
<a href="#">Utils::CutRender_info</a>	Daten zur Darstellung einer 2D-Temperaturverteilungs-Ebene . . . . .	42
<a href="#">Exporter</a>	Export der gewonnenen Daten . . . . .	43
<a href="#">GUIAnalyzeOutputWindow</a>	Übersichtsfenster über die Analysedaten . . . . .	46
<a href="#">GUIAnalyzePointWindow</a>	Analysefenster für einen Punkt . . . . .	49
<a href="#">GUIColorScalePanel</a>	Farbige Temperaturskala für zweidimensionale Temperaturverteilung . . . . .	52
<a href="#">GUICutRenderWindow</a>	Fenster zum erstellen zweidimensionaler Temperaturverteilungen . . . . .	62
<a href="#">GUIGLCanvas</a>	Zeichenfläche für das 3D-Fenster . . . . .	79
<a href="#">GUIMainWindow</a>	Hauptfenster mit Hauptmenü und Zugriff auf die einzelnen Programmfunktionen . . . . .	84
<a href="#">GUIRenderCutCanvas</a>	Zeichenfläche für die 2D-Temperaturverteilung . . . . .	99
<a href="#">GUITimeline</a>	Eine Zeitleistenkomponente . . . . .	105
<a href="#">Importer</a>	Importieren von 3D-Modell (.obj) und Sensordaten (.tsd oder .sd) . . . . .	118
<a href="#">Interpolator</a>	2- und 3-dimensionale Inter-/Extrapolation . . . . .	121
<a href="#">ObjectData::MaterialData</a>	Die Daten eines Materials . . . . .	126

<a href="#">Matrix3D</a>	3x3-Matrixklasse mit Operationen . . . . .	128
<a href="#">MeshProcessor</a>	Errechnet die Temperaturverteilung für ein Objekt . . . . .	133
<a href="#">ObjectData</a>	Die Daten eines Versuchsobjekts . . . . .	134
<a href="#">OdisiToSdConverter</a>	Konverter von ODiSI zu .tsd . . . . .	145
<a href="#">CsvToSdConverter::Options</a>	Struktur für die Programmeinstellungen . . . . .	152
<a href="#">TsdMerger::Options</a>	Struktur für die Programmeinstellungen . . . . .	153
<a href="#">OdisiToSdConverter::Options</a>	Struktur für die Programmeinstellungen . . . . .	154
<a href="#">PropertiesBox</a>	Oberfläche zum Verändern/Anzeigen der Eigenschaften eines Objekts . . . . .	157
<a href="#">Renderer</a>	Zeichnet den Inhalt der 3D-Fensters . . . . .	167
<a href="#">Utils::SensorData</a>	Ein Sensordatensatz . . . . .	175
<a href="#">Utils::SensorPoint</a>	Daten eines Sensordatenpunktes . . . . .	177
<a href="#">Utils::SensorPointComparator</a>	Hilfsstruktur zum Vergleichen des Abstands von Messpunkten . . . . .	178
<a href="#">SimpleAnalyzerApp</a>	Regelt den allgemeinen Ablauf des Programms . . . . .	180
<a href="#">Utils::SortStruct</a>	Hilfsstruktur zum Sortieren von Punkten nach dem Abstand zu einem anderen Punkt . . . . .	184
<a href="#">Tetrahedron</a>	Ein durch 4 Ortsvektoren beschriebener Tetraeder . . . . .	185
<a href="#">Triangle</a>	Ein durch 3 Ortsvektoren beschriebenes Dreieck . . . . .	188
<a href="#">TsdMerger</a>	Zusammenführen zweier .tsd-Dateien . . . . .	192
<a href="#">std::vector&lt; T &gt;</a>	. . . . .	195
<a href="#">Vector3D</a>	3D-Vektorklasse mit nützlichen Operationen . . . . .	196
<a href="#">Renderer::Viewport_info</a>	Informationen über die Ansicht des Modells (Virtuelle Kamera) und welche Elemente dargestellt werden . . . . .	210
<a href="#">ViewpropBox</a>	Oberfläche zum Verändern/Anzeigen der Visualisierungsoptionen . . . . .	212
<a href="#">Utils::Visualization_info</a>	Informationen über die Farbgebung bei der Visualisierung . . . . .	217

# Kapitel 6

## Datei-Verzeichnis

### 6.1 Auflistung der Dateien

Hier folgt die Aufzählung aller Dateien mit einer Kurzbeschreibung:

/daten/Projekte/eclipse_workspace/csvtosd/main.cpp	219
doxygen_dep_dummy.h	221
/daten/Projekte/eclipse_workspace/mergetsd/src/mergetsd.cpp	221
/daten/Projekte/eclipse_workspace/odisitosd/main.cpp	220
/daten/Projekte/eclipse_workspace/simpleanalyzer-gui/src/SimpleAnalyzerApp.cpp	270
/daten/Projekte/eclipse_workspace/simpleanalyzer-gui/src/SimpleAnalyzerApp.h	271
/daten/Projekte/eclipse_workspace/simpleanalyzer-gui/src/fileIO/Exporter.cpp	222
/daten/Projekte/eclipse_workspace/simpleanalyzer-gui/src/fileIO/Exporter.h	223
/daten/Projekte/eclipse_workspace/simpleanalyzer-gui/src/fileIO/Importer.cpp	224
/daten/Projekte/eclipse_workspace/simpleanalyzer-gui/src/fileIO/Importer.h	225
/daten/Projekte/eclipse_workspace/simpleanalyzer-gui/src/GUI/constants.h	226
/daten/Projekte/eclipse_workspace/simpleanalyzer-gui/src/GUI/GUIAnalyzeOutputWindow.cpp	229
/daten/Projekte/eclipse_workspace/simpleanalyzer-gui/src/GUI/GUIAnalyzeOutputWindow.h	229
/daten/Projekte/eclipse_workspace/simpleanalyzer-gui/src/GUI/GUIAnalyzePointWindow.cpp	230
/daten/Projekte/eclipse_workspace/simpleanalyzer-gui/src/GUI/GUIAnalyzePointWindow.h	231
/daten/Projekte/eclipse_workspace/simpleanalyzer-gui/src/GUI/GUIColorScalePanel.cpp	232
/daten/Projekte/eclipse_workspace/simpleanalyzer-gui/src/GUI/GUIColorScalePanel.h	233
/daten/Projekte/eclipse_workspace/simpleanalyzer-gui/src/GUI/GUICutRenderWindow.cpp	233
/daten/Projekte/eclipse_workspace/simpleanalyzer-gui/src/GUI/GUICutRenderWindow.h	236
/daten/Projekte/eclipse_workspace/simpleanalyzer-gui/src/GUI/GUIGLCanvas.cpp	237
/daten/Projekte/eclipse_workspace/simpleanalyzer-gui/src/GUI/GUIGLCanvas.h	238
/daten/Projekte/eclipse_workspace/simpleanalyzer-gui/src/GUI/GUIMainWindow.cpp	239
/daten/Projekte/eclipse_workspace/simpleanalyzer-gui/src/GUI/GUIMainWindow.h	240
/daten/Projekte/eclipse_workspace/simpleanalyzer-gui/src/GUI/GUIRenderCutCanvas.cpp	240
/daten/Projekte/eclipse_workspace/simpleanalyzer-gui/src/GUI/GUIRenderCutCanvas.h	241
/daten/Projekte/eclipse_workspace/simpleanalyzer-gui/src/GUI/GUITimeline.cpp	242
/daten/Projekte/eclipse_workspace/simpleanalyzer-gui/src/GUI/GUITimeline.h	243
/daten/Projekte/eclipse_workspace/simpleanalyzer-gui/src/GUI/PropertiesBox.cpp	244
/daten/Projekte/eclipse_workspace/simpleanalyzer-gui/src/GUI/PropertiesBox.h	245
/daten/Projekte/eclipse_workspace/simpleanalyzer-gui/src/GUI/Renderer.cpp	246
/daten/Projekte/eclipse_workspace/simpleanalyzer-gui/src/GUI/Renderer.h	251
/daten/Projekte/eclipse_workspace/simpleanalyzer-gui/src/GUI/ViewpropBox.cpp	252
/daten/Projekte/eclipse_workspace/simpleanalyzer-gui/src/GUI/ViewpropBox.h	253
/daten/Projekte/eclipse_workspace/simpleanalyzer-gui/src/libraries/interpolate/GeometryClasses.cpp	254
/daten/Projekte/eclipse_workspace/simpleanalyzer-gui/src/libraries/interpolate/GeometryClasses.h	256
/daten/Projekte/eclipse_workspace/simpleanalyzer-gui/src/libraries/interpolate/Interpolator.cpp	257
/daten/Projekte/eclipse_workspace/simpleanalyzer-gui/src/libraries/interpolate/Interpolator.h	259
/daten/Projekte/eclipse_workspace/simpleanalyzer-gui/src/processing/Analyzer.cpp	260

/daten/Projekte/eclipse_workspace/simpleanalyzer-gui/src/processing/Analyzer.h . . . . .	261
/daten/Projekte/eclipse_workspace/simpleanalyzer-gui/src/processing/MeshProcessor.cpp . . . . .	262
/daten/Projekte/eclipse_workspace/simpleanalyzer-gui/src/processing/MeshProcessor.h . . . . .	264
/daten/Projekte/eclipse_workspace/simpleanalyzer-gui/src/processing/ObjectData.cpp . . . . .	265
/daten/Projekte/eclipse_workspace/simpleanalyzer-gui/src/processing/ObjectData.h . . . . .	266
/daten/Projekte/eclipse_workspace/simpleanalyzer-gui/src/processing/Utils.cpp . . . . .	267
/daten/Projekte/eclipse_workspace/simpleanalyzer-gui/src/processing/Utils.h . . . . .	268

# Kapitel 7

## Dokumentation der Namensbereiche

### 7.1 std-Namensbereichsreferenz

#### Klassen

- class `vector`

### 7.2 Utils-Namensbereichsreferenz

allgemeine Funktionen und Typen.

#### Klassen

- struct `Visualization_info`  
*Informationen über die Farbgebung bei der Visualisierung.*
- struct `SortStruct`  
*Hilfsstruktur zum Sortieren von Punkten nach dem Abstand zu einem anderen Punkt.*
- struct `SensorPoint`  
*Daten eines Sensordatenpunktes.*
- struct `CutRender_info`  
*Daten zur Darstellung einer 2D-Temperaturverteilungs-Ebene.*
- struct `SensorData`  
*Ein Sensordatensatz.*
- struct `SensorPointComparator`  
*Hilfsstruktur zum Vergleichen des Abstands von Messpunkten.*

#### Aufzählungen

- enum `PIM_algorithm` { `ALGORITHM_TETRAHEDRONS` = 0, `ALGORITHM_RAY` }  
*Zum Punkt-in-Volumen Testen verwendeter Algorithmus.*

#### Funktionen

- double `sqr` (double d)  
*Quadriert eine Zahl.*
- float `clampHue` (float h)

- Begrenzt einen Wert auf den Bereich 0..1.*
  - string `floattostr` (double val)
    - Hilfsfunktion zur Umwandlung einer Zahl in einen String.*
  - wxString `floattowxstr` (double val)
    - Wandelt eine Fließkommazahl in einen wxWidgets-String um.*
  - wxString `floattowxstr` (double val, int digits)
    - Wandelt eine Fließkommazahl in einen wxWidgets-String um.*
  - int `rayIntersectsTriangle` (Vector3D \*p, Vector3D \*direction, Triangle \*tri, double \*depth)
    - Testet, ob ein Strahl ein Dreieck schneidet.*
  - int `pointInsideMesh` (Vector3D \*p, tetgenio \*io, PIM\_algorithm algorithm)
    - Testet, ob sich ein Punkt innerhalb eines Körpers befindet.*
  - int `pointInsideTetrahedron` (Vector3D \*pges, Vector3D \*v1, Vector3D \*v2, Vector3D \*v3, Vector3D \*v4)
    - Testet, ob sich ein Punkt innerhalb eines Tetraeders befindet.*
  - int `pointInsideTetrahedron` (double \*pges, double \*v1, double \*v2, double \*v3, double \*v4)
    - Testet, ob sich ein Punkt innerhalb eines Tetraeders befindet.*
  - int `pointInsideTetrahedron` (double \*p, vector< SensorPoint \* > \*tet)
    - Testet, ob sich ein Punkt innerhalb eines Tetraeders befindet.*
  - void `nextCombination` (vector< int > \*indices, int depth, int dataPointCount)
    - Ermöglicht das generieren aller möglichen Verteilungen von 4 Elementen auf dataPointCount Plätze.*
  - double `getPointValue` (int &status, vector< SensorPoint \* > \*sensorpoints, double \*p, Interpolator \*interpolator, vector< SensorPoint \* > \*prev\_tet=NULL, vector< SensorPoint \* > \*current\_tet=NULL)
    - Gibt den inter/extrapolierten Wert eines Punktes zurück.*
  - float \* `hsvToRgb` (float h, float s, float v)
    - Wandelt eine Farbe im HSV-Format ins RGB-Format um.*
  - void `copySensorPoint` (SensorPoint \*from, SensorPoint \*to)
    - Kopiert die Eigenschaften eines Sensorpunktes in einen Anderen.*

## 7.2.1 Ausführliche Beschreibung

allgemeine Funktionen und Typen.

## 7.2.2 Dokumentation der Aufzählungstypen

### 7.2.2.1 enum Utils::PIM\_algorithm

Zum Punkt-in-Volumen Testen verwendeter Algorithmus.

Dies wird bei ALGORITHM\_TETRAHEDRONS über alle Tetraeder des Objekts und deren Flächennormalen ermittelt. Bei ALGORITHM\_RAY werden die Schnittpunkte aller Außenflächen mit einem Strahl gezählt (Aktuell nicht verwendet).

Aufzählungswerte

**ALGORITHM\_TETRAHEDRONS**  
**ALGORITHM\_RAY**

Definiert in Zeile 31 der Datei utils.h.

## 7.2.3 Dokumentation der Funktionen

### 7.2.3.1 float Utils::clampHue ( float h )

Begrenzt einen Wert auf den Bereich 0..1.



## Parameter

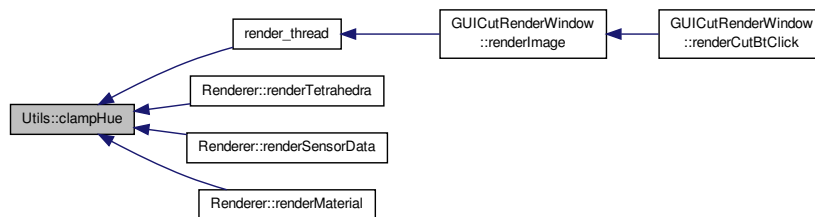
<i>h</i>	Die zu begrenzende Zahl.
----------	--------------------------

## Rückgabe

Der den Grenzen entsprechende Wert.

Definiert in Zeile 39 der Datei utils.cpp.

Hier ist ein Graph der zeigt, wo diese Funktion aufgerufen wird:



## 7.2.3.2 void Utils::copySensorPoint ( SensorPoint \* from, SensorPoint \* to )

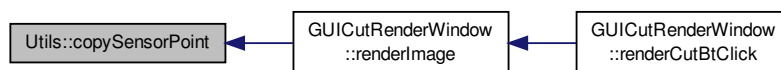
Kopiert die Eigenschaften eines Sensorpunktes in einen Anderen.

## Parameter

<i>from</i>	Quelle.
<i>to</i>	Ziel.

Definiert in Zeile 97 der Datei utils.cpp.

Hier ist ein Graph der zeigt, wo diese Funktion aufgerufen wird:



## 7.2.3.3 string Utils::floattostr ( double val ) [inline]

Hilfsfunktion zur Umwandlung einer Zahl in einen String.

## Parameter

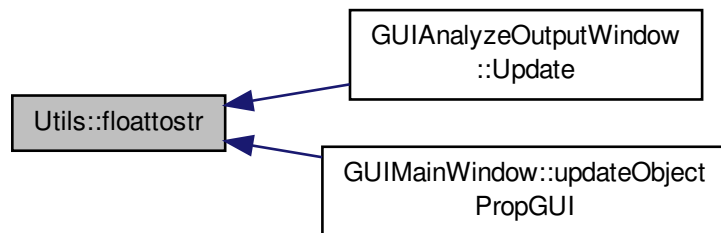
<i>val</i>	Die umzuwandelnde Zahl.
------------	-------------------------

**Rückgabe**

Der resultierende String.

Definiert in Zeile 130 der Datei utils.h.

Hier ist ein Graph der zeigt, wo diese Funktion aufgerufen wird:

**7.2.3.4 wxString Utils::floattowxstr ( double val )**

Wandelt eine Fließkommazahl in einen wxWidgets-String um.

**Parameter**

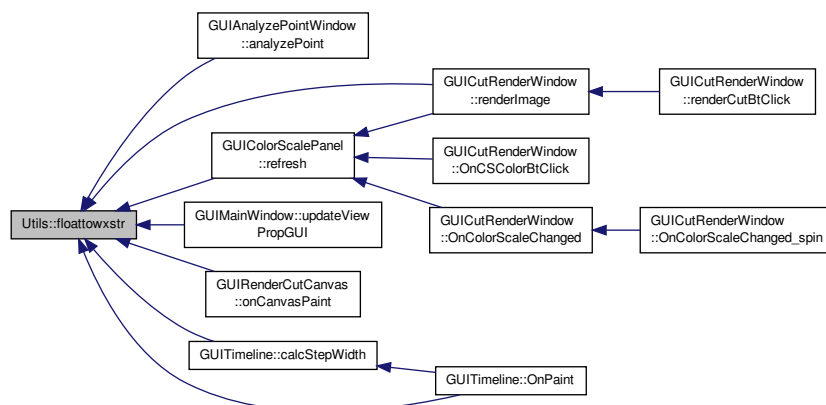
<i>val</i>	Die umzuwandelnde Zahl.
------------	-------------------------

**Rückgabe**

Der entstandene String.

Definiert in Zeile 49 der Datei utils.cpp.

Hier ist ein Graph der zeigt, wo diese Funktion aufgerufen wird:



### 7.2.3.5 wxString Utils::floattowxstr ( double *val*, int *digits* )

Wandelt eine Fließkommazahl in einen wxWidgets-String um.

**Parameter**

<i>val</i>	Die umzuwandelnde Zahl.
<i>digits</i>	Anzahl der zu übernehmenden Stellen.

**Rückgabe**

Der entstandene String.

Definiert in Zeile 55 der Datei utils.cpp.

**7.2.3.6** `double Utils::getPointValue ( int & status, vector< SensorPoint > * sensorpoints, double * p, Interpolator * interpolator, vector< SensorPoint * > * prev_tet = NULL, vector< SensorPoint * > * current_tet = NULL )`

Gibt den inter/extrapolierten Wert eines Punktes zurück.

**Parameter**

<i>status</i>	Rückgabeveriable. 1: Punkt wurde extrapoliert 0: Punkt wurde interpoliert. -1: Alle Sensorpunkte sind komplanar.
<i>sensorpoints</i>	Die zu verwendenden Senosorpunkte.
<i>p</i>	Die Koordinaten des gesuchten Punktes.
<i>interpolator</i>	Das zu verwendende Interpolatorobjekt.
<i>prev_tet</i>	Zuerst zu Testender Tetraeder (optional, NULL zum Nichtverwenden).
<i>current_tet</i>	Rückgabeveriable für den zuletzt verwendeten Tetraeder (optional, NULL zum Nichtverwenden).

**Rückgabe**

Temperatur des gesuchten Punktes.

Definiert in Zeile 354 der Datei utils.cpp.



**Parameter**

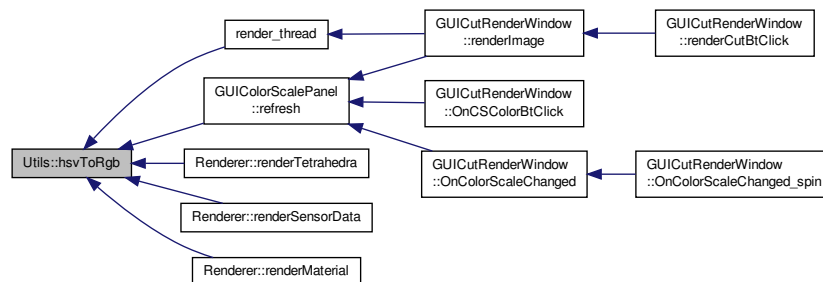
<i>h</i>	H-Komponente der Farbe.
<i>s</i>	S-Komponente der Farbe.
<i>v</i>	V-Komponente der Farbe.

**Rückgabe**

RGB-Farbe als Liste mit 3 Werten im Bereich 0..1. Muss manuell mit delete[] freigegeben werden!

Definiert in Zeile 61 der Datei utils.cpp.

Hier ist ein Graph der zeigt, wo diese Funktion aufgerufen wird:



### 7.2.3.8 void Utils::nextCombination ( vector< int > \* indices, int depth, int dataPointCount )

Ermöglicht das generieren aller möglichen Verteilungen von 4 Elementen auf dataPointCount Plätze.

Die Indices der Plätze, die die Elemente jeweils besetzten stehen in indices. Verschiedene Reihenfolgen der selben Indices werden dabei nicht generiert. Diese Funktion generiert aus der vorherigen Anordnung die Nächste, indem die Indices bis zum überlauf hochgezählt wird, woraufhin der vorhergehende erhöht wird, z.b. für dataPointCount = 8:

```

0 1 2 3
0 1 2 4
0 1 2 5
0 1 2 6
0 1 2 7
0 1 2 8 -> Umschlag
0 1 3 4
0 1 3 5
0 1 3 6
  
```

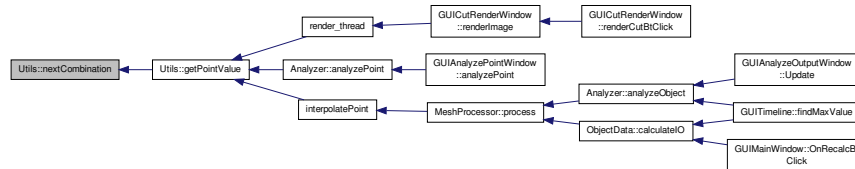
**Parameter**

<i>indices</i>	Liste der Indices der Elemente.
<i>depth</i>	Index des in diesem Funktionsaufruf verarbeiteten Elements. Beim ersten Aufruf also 3.

<code>dataPointCount</code>	Anzahl der Plätze.
-----------------------------	--------------------

Definiert in Zeile 16 der Datei `utils.cpp`.

Hier ist ein Graph der zeigt, wo diese Funktion aufgerufen wird:



### 7.2.3.9 `int Utils::pointInsideMesh ( Vector3D * p, tetgenio * io, PIM_algorithm algorithm )`

Testet, ob sich ein Punkt innerhalb eines Körpers befindet.

Parameter

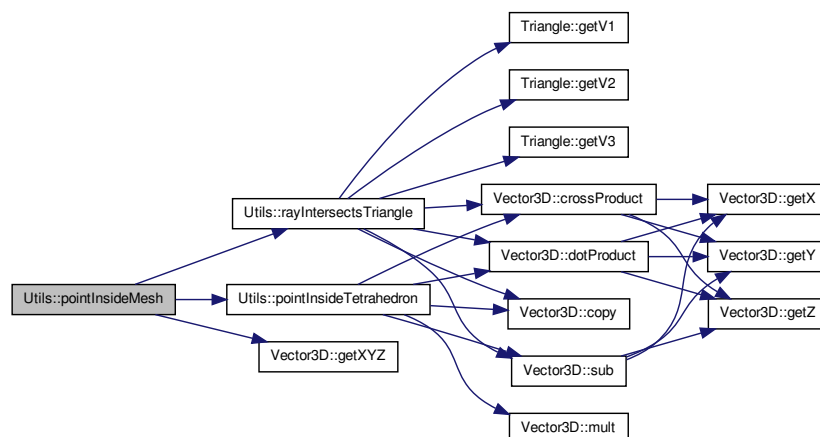
<i>p</i>	Der zu testende Punkt.
<i>io</i>	Der zu testende Körper als Tetgen-Daten (s. Tetgen Dokumentation).
<i>algorithm</i>	Der zu verwendende Testalgorithmus (Empfohlen und ausschließlich verwendet: ALGORITHM_TETRAHEDRONS).

Rückgabe

1 Wenn innerhalb, 0 wenn außerhalb. Bei einer falschen Algorithmuskonstante -1.

Definiert in Zeile 170 der Datei `utils.cpp`.

Hier ist ein Graph, der zeigt, was diese Funktion aufruft:



Hier ist ein Graph der zeigt, wo diese Funktion aufgerufen wird:



**7.2.3.10** `int Utils::pointInsideTetrahedron ( Vector3D * pges, Vector3D * v1, Vector3D * v2, Vector3D * v3, Vector3D * v4 )`

Testet, ob sich ein Punkt innerhalb eines Tetraeders befindet.

**Parameter**

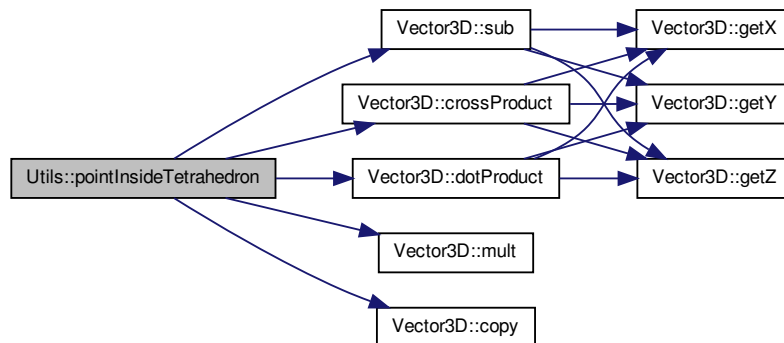
<i>pges</i>	Der zu testende Punkt.
<i>v1</i>	Der 1. Punkt des Tetraeders.
<i>v2</i>	Der 2. Punkt des Tetraeders.
<i>v3</i>	Der 3. Punkt des Tetraeders.
<i>v4</i>	Der 4. Punkt des Tetraeders.

**Rückgabe**

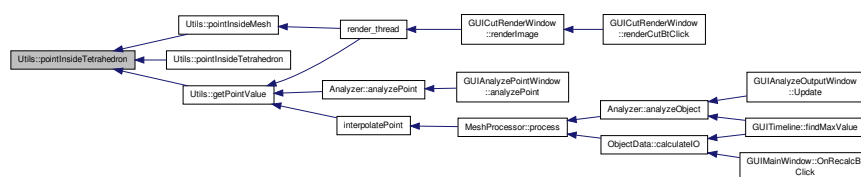
1 Wenn innerhalb, 0 wenn außerhalb. -1, wenn der Tetraeder komplanar ist.

Definiert in Zeile 249 der Datei utils.cpp.

Hier ist ein Graph, der zeigt, was diese Funktion aufruft:



Hier ist ein Graph der zeigt, wo diese Funktion aufgerufen wird:





### 7.2.3.11 `int Utils::pointInsideTetrahedron ( double * pges, double * v1, double * v2, double * v3, double * v4 )`

Testet, ob sich ein Punkt innerhalb eines Tetraeders befindet.

Parameter

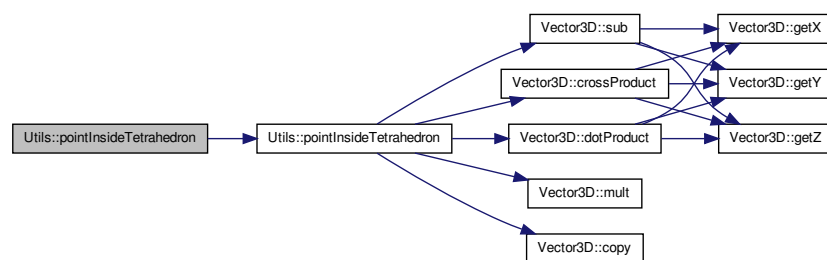
<i>pges</i>	Koordinaten des zu testenden Punktes.
<i>v1</i>	Koordinaten des 1. Punktes des Tetraeders.
<i>v2</i>	Koordinaten des 2. Punktes des Tetraeders.
<i>v3</i>	Koordinaten des 3. Punktes des Tetraeders.
<i>v4</i>	Koordinaten des 4. Punktes des Tetraeders.

Rückgabe

1 Wenn innerhalb, 0 wenn außerhalb. -1, wenn der Tetraeder komplanar ist.

Definiert in Zeile 343 der Datei `utils.cpp`.

Hier ist ein Graph, der zeigt, was diese Funktion aufruft:



### 7.2.3.12 `int Utils::pointInsideTetrahedron ( double * p, vector< SensorPoint * > * tet )`

Testet, ob sich ein Punkt innerhalb eines Tetraeders befindet.

Parameter

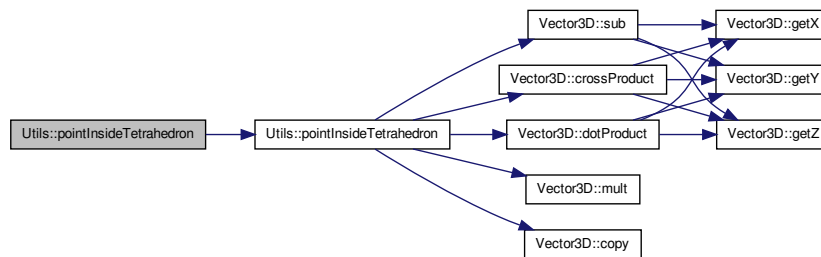
<i>p</i>	Koordinaten des zu testenden Punktes.
<i>tet</i>	Der zu untersuchende Tetraeder als Liste von Sensordaten.

**Rückgabe**

1 Wenn innerhalb, 0 wenn außerhalb. -1, wenn der Tetraeder komplanar ist.

Definiert in Zeile 333 der Datei utils.cpp.

Hier ist ein Graph, der zeigt, was diese Funktion aufruft:



### 7.2.3.13 `int Utils::rayIntersectsTriangle ( Vector3D * p, Vector3D * direction, Triangle * tri, double * depth )`

Testet, ob ein Strahl ein Dreieck schneidet.

Gefunden unter [http://en.wikipedia.org/wiki/M%C3%B6ller%E2%80%93Trumbore\\_intersection\\_algorithm](http://en.wikipedia.org/wiki/M%C3%B6ller%E2%80%93Trumbore_intersection_algorithm) am 4.9.13 und auf C++ und eigene Datentypen portiert.

**Parameter**

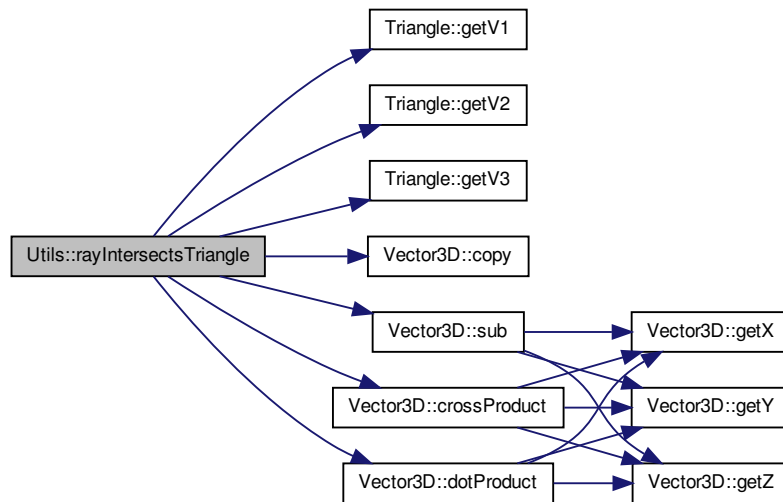
<i>p</i>	Ortsvektor zum Ausgangspunkt des Strahls.
<i>direction</i>	Richtung des Strahls.
<i>tri</i>	Das zu testende Dreieck.
<i>depth</i>	Ausgabevariable, ein Maß für den Abstand von Ausgangspunkt zu Schnittpunkt.

**Rückgabe**

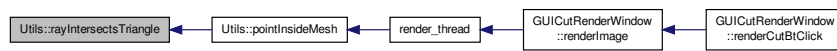
Gibt 1 zurück, wenn es einen Schnittpunkt gibt, ansonsten 0.

Definiert in Zeile 105 der Datei utils.cpp.

Hier ist ein Graph, der zeigt, was diese Funktion aufruft:



Hier ist ein Graph der zeigt, wo diese Funktion aufgerufen wird:



#### 7.2.3.14 double Utils::sqr ( double d ) [inline]

Quadriert eine Zahl.

Parameter

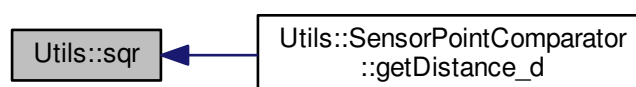
<code>d</code>	Die zu quadrierende Zahl.
----------------	---------------------------

Rückgabe

$d^2$ .

Definiert in Zeile 40 der Datei `utils.h`.

Hier ist ein Graph der zeigt, wo diese Funktion aufgerufen wird:





# Kapitel 8

## Klassen-Dokumentation

### 8.1 Analyzer Klassenreferenz

Ermittelt Daten aus der Temperaturverteilung.

```
#include <Analyzer.h>
```

#### Klassen

- struct [AnalyzerData\\_dataset](#)  
*Analyseergebnisse für einen Sensordatensatz.*
- struct [AnalyzerData\\_material](#)  
*Analyseergebnisse für ein Material.*
- struct [AnalyzerData\\_object](#)  
*Analyseergebnisse für ein Objekt.*
- struct [AnalyzerData\\_point](#)  
*Analyseergebnisse für einen Punkt.*

#### Öffentliche Methoden

- [Analyzer](#) ()  
*Der Konstruktor.*
- void [analyzeObject](#) ([ObjectData](#) \*obj, [AnalyzerData\\_object](#) \*out, bool use\_markers=true, int sdindex=-1)  
*Ermittelt Daten für ein Objekt.*
- void [analyzePoint](#) ([ObjectData](#) \*obj, [Vector3D](#) \*point, [AnalyzerData\\_point](#) \*point\_data, [Interpolator](#) \*interpolator)  
*Ermittelt Daten für einen Punkt am aktuell ausgewählten Zeitpunkt.*
- virtual [~Analyzer](#) ()  
*Der Destruktor.*

#### Freundbeziehungen

- std::ostream & [operator<<](#) (std::ostream &out, const [AnalyzerData\\_object](#) &data)  
*Operator zum Ausgeben der Analysedaten für ein Objekt in einem Stream.*

### 8.1.1 Ausführliche Beschreibung

Ermittelt Daten aus der Temperaturverteilung.

Definiert in Zeile 21 der Datei Analyzer.h.

### 8.1.2 Beschreibung der Konstruktoren und Destrukturen

#### 8.1.2.1 Analyzer::Analyzer ( )

Der Konstruktor.

Definiert in Zeile 16 der Datei Analyzer.cpp.

#### 8.1.2.2 Analyzer::~~Analyzer ( ) [virtual]

Der Destruktor.

Definiert in Zeile 191 der Datei Analyzer.cpp.

### 8.1.3 Dokumentation der Elementfunktionen

#### 8.1.3.1 void Analyzer::analyzeObject ( ObjectData \* *obj*, AnalyzerData\_object \* *out*, bool *use\_markers* = *true*, int *sdindex* = -1 )

Ermittelt Daten für ein Objekt.

Objekt zum Vergleichen von Messpunkten hinsichtlich der Temperatur.

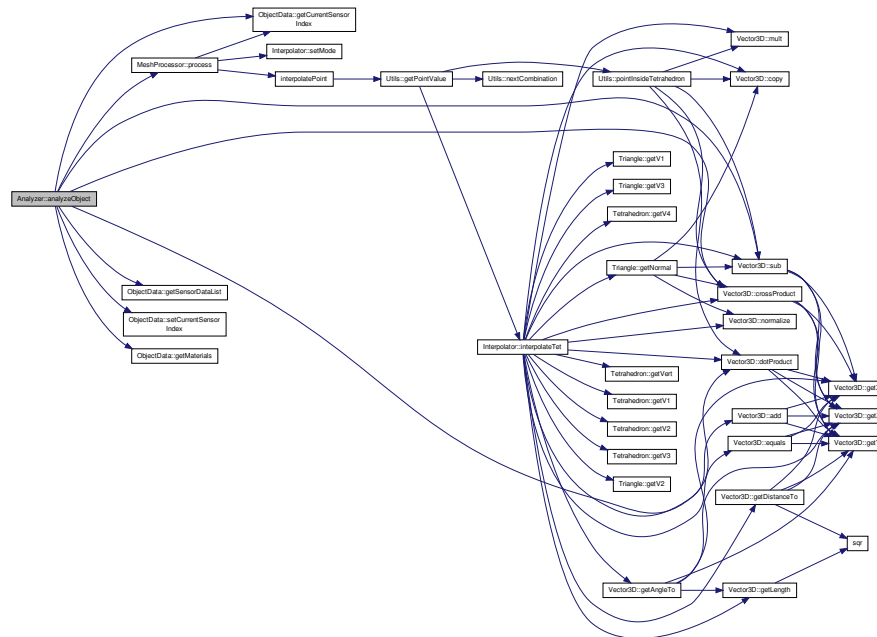
Parameter

<i>obj</i>	Das zu analysierende Objekt.
<i>out</i>	Referenz auf die <a href="#">AnalyzerData_object</a> -Struktur in der die Analyseergebnisse gespeichert werden sollen.
<i>use_markers</i>	Die markierten Zeitpunkte eines Sensordatensatzes analysieren. Wenn false wird nur der aktuell ausgewählte Zeitpunkt analysiert.
<i>sdindex</i>	Nur den Sensordatensatz mit diesem Index analysieren.

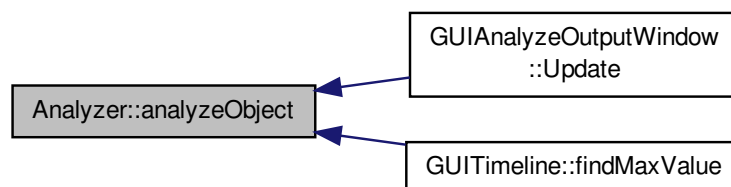
Wird von s

Definiert in Zeile 26 der Datei Analyzer.cpp.

Hier ist ein Graph, der zeigt, was diese Funktion aufruft:



Hier ist ein Graph der zeigt, wo diese Funktion aufgerufen wird:



**8.1.3.2 void Analyzer::analyzePoint ( ObjectData \* obj, Vector3D \* point, AnalyzerData\_point \* point\_data, Interpolator \* interpolator )**

Ermittelt Daten für einen Punkt am aktuell ausgewählten Zeitpunkt.

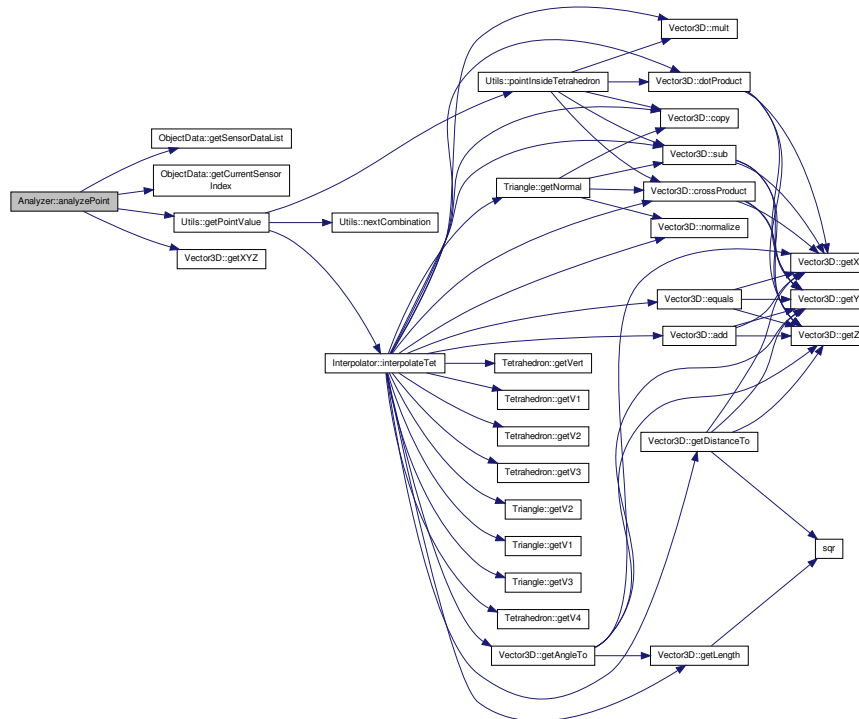
**Parameter**

<i>obj</i>	Das zu analysierende Objekt.
<i>point</i>	Der Ortsvektor zum zu analysierenden Punkt.
<i>point_data</i>	Referenz auf die <a href="#">AnalyzerData_point</a> -Struktur in der die Analyseergebnisse gespeichert werden sollen.

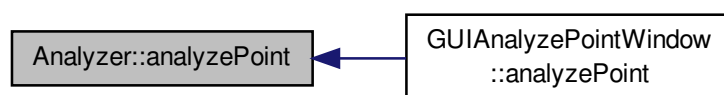
<i>interpolator</i>	Das zu verwendende Interpolatorobjekt.
---------------------	--

Definiert in Zeile 147 der Datei Analyzer.cpp.

Hier ist ein Graph, der zeigt, was diese Funktion aufruft:



Hier ist ein Graph der zeigt, wo diese Funktion aufgerufen wird:



## 8.1.4 Freundbeziehungen und Funktionsdokumentation

8.1.4.1 `std::ostream& operator<< ( std::ostream & out, const AnalyzerData_object & data ) [friend]`

Operator zum Ausgeben der Analysedaten für ein Objekt in einem Stream.

Definiert in Zeile 164 der Datei Analyzer.cpp.

Die Dokumentation für diese Klasse wurde erzeugt aufgrund der Dateien:

- /daten/Projekte/eclipse\_workspace/simpleanalyzer-gui/src/processing/Analyzer.h
- /daten/Projekte/eclipse\_workspace/simpleanalyzer-gui/src/processing/Analyzer.cpp

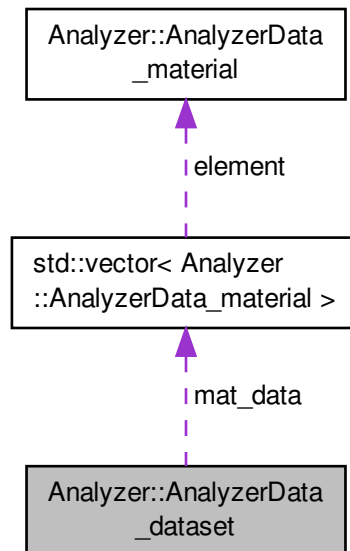


## 8.2 Analyzer::AnalyzerData\_dataset Strukturreferenz

Analyseergebnisse für einen Sensordatensatz.

```
#include <Analyzer.h>
```

Zusammengehörigkeiten von Analyzer::AnalyzerData\_dataset:



### Öffentliche Attribute

- string `name`  
*Der Name des Sensordatensatzes<.*
- double `heat_energy`  
*Die Wärmeenergie, die das Objekt für diesen Datensatz enthält.*
- vector< `AnalyzerData_material` > `mat_data`  
*Die Analyseergebnisse für die Einzelnen Materialien.*

### 8.2.1 Ausführliche Beschreibung

Analyseergebnisse für einen Sensordatensatz.

Definiert in Zeile 35 der Datei Analyzer.h.

### 8.2.2 Dokumentation der Datenelemente

#### 8.2.2.1 double Analyzer::AnalyzerData\_dataset::heat\_energy

Die Wärmeenergie, die das Objekt für diesen Datensatz enthält.

Definiert in Zeile 37 der Datei Analyzer.h.

#### 8.2.2.2 `vector<AnalyzerData_material> Analyzer::AnalyzerData_dataset::mat_data`

Die Analyseergebnisse für die Einzelnen Materialien.

Definiert in Zeile 38 der Datei Analyzer.h.

#### 8.2.2.3 `string Analyzer::AnalyzerData_dataset::name`

Der Name des Sensordatensatzes.

Definiert in Zeile 36 der Datei Analyzer.h.

Die Dokumentation für diese Struktur wurde erzeugt aufgrund der Datei:

- `/daten/Projekte/eclipse_workspace/simpleanalyzer-gui/src/processing/Analyzer.h`

### 8.3 `Analyzer::AnalyzerData_material` Strukturreferenz

Analyseergebnisse für ein Material.

```
#include <Analyzer.h>
```

#### Öffentliche Attribute

- `string name`  
*Der Name des Material.*
- `double volume`  
*Das Volumen, das dem Material zugeordnet ist.*
- `double heat_energy`  
*Die Wärmeenergie, die das dem Material zugeordnete Volumen enthält.*

#### 8.3.1 Ausführliche Beschreibung

Analyseergebnisse für ein Material.

Definiert in Zeile 26 der Datei Analyzer.h.

#### 8.3.2 Dokumentation der Datenelemente

##### 8.3.2.1 `double Analyzer::AnalyzerData_material::heat_energy`

Die Wärmeenergie, die das dem Material zugeordnete Volumen enthält.

<

Definiert in Zeile 29 der Datei Analyzer.h.

##### 8.3.2.2 `string Analyzer::AnalyzerData_material::name`

Der Name des Material.

<

Definiert in Zeile 27 der Datei Analyzer.h.

### 8.3.2.3 double Analyzer::AnalyzerData\_material::volume

Das Volumen, das dem Material zugeordnet ist.

<

Definiert in Zeile 28 der Datei Analyzer.h.

Die Dokumentation für diese Struktur wurde erzeugt aufgrund der Datei:

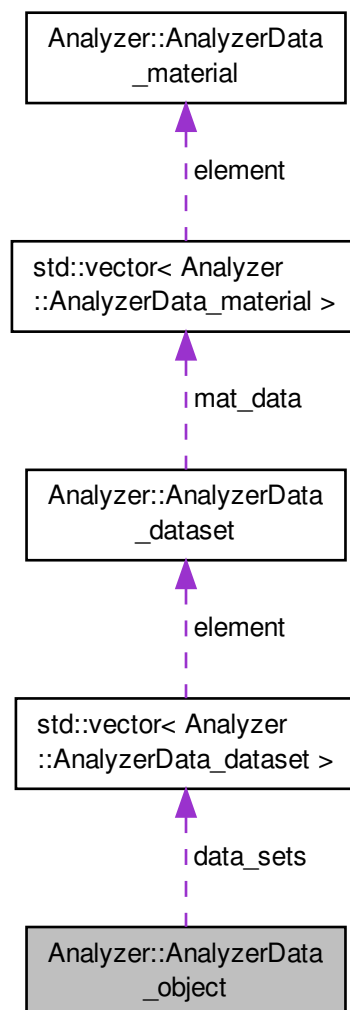
- /daten/Projekte/eclipse\_workspace/simpleanalyzer-gui/src/processing/[Analyzer.h](#)

## 8.4 Analyzer::AnalyzerData\_object Strukturreferenz

Analyseergebnisse für ein Objekt.

```
#include <Analyzer.h>
```

Zusammengehörigkeiten von Analyzer::AnalyzerData\_object:



## Öffentliche Attribute

- double [volume](#)  
*Das Volumen des Objekts.*
- [vector](#)< [AnalyzerData\\_dataset](#) > [data\\_sets](#)  
*Die Analyseergebnisse für die Sensordatensätze.*

### 8.4.1 Ausführliche Beschreibung

Analyseergebnisse für ein Objekt.

Definiert in Zeile 44 der Datei Analyzer.h.

### 8.4.2 Dokumentation der Datenelemente

#### 8.4.2.1 [vector](#)<[AnalyzerData\\_dataset](#)> [Analyzer::AnalyzerData\\_object::data\\_sets](#)

Die Analyseergebnisse für die Sensordatensätze.

<

Definiert in Zeile 46 der Datei Analyzer.h.

#### 8.4.2.2 double [Analyzer::AnalyzerData\\_object::volume](#)

Das Volumen des Objekts.

<

Definiert in Zeile 45 der Datei Analyzer.h.

Die Dokumentation für diese Struktur wurde erzeugt aufgrund der Datei:

- [/daten/Projekte/eclipse\\_workspace/simpleanalyzer-gui/src/processing/Analyzer.h](#)

## 8.5 [Analyzer::AnalyzerData\\_point](#) Strukturreferenz

Analyseergebnisse für einen Punkt.

```
#include <Analyzer.h>
```

## Öffentliche Attribute

- double [value](#)  
*Die Temperatur an diesem Punkt.*
- bool [extrapolated](#)  
*Ist der Punkt extrapoliert?*

### 8.5.1 Ausführliche Beschreibung

Analyseergebnisse für einen Punkt.

Definiert in Zeile 52 der Datei Analyzer.h.

## 8.5.2 Dokumentation der Datenelemente

### 8.5.2.1 bool Analyzer::AnalyzerData\_point::extrapolated

Ist der Punkt extrapoliert?

Definiert in Zeile 54 der Datei Analyzer.h.

### 8.5.2.2 double Analyzer::AnalyzerData\_point::value

Die Temperatur an diesem Punkt.

Definiert in Zeile 53 der Datei Analyzer.h.

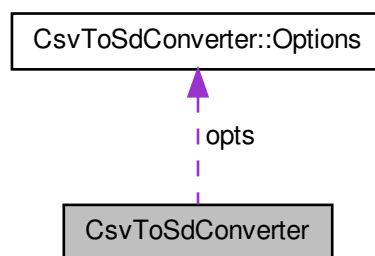
Die Dokumentation für diese Struktur wurde erzeugt aufgrund der Datei:

- /daten/Projekte/eclipse\_workspace/simpleanalyzer-gui/src/processing/[Analyzer.h](#)

## 8.6 CsvToSdConverter Klassenreferenz

Konverter von .csv zu .tsd.

Zusammengehörigkeiten von CsvToSdConverter:



### Klassen

- struct [Options](#)  
*Struktur für die Programmeinstellungen.*

### Öffentliche Methoden

- int [convert](#) (int argc, char \*argv[])  
*Wandelt die Daten der .csv-Datei in eine .tsd-Datei um.*

### Geschützte Methoden

- bool [contains](#) (std::vector< string > &Vec, const string &Element)  
*Testet, ob sich ein String in einer Liste von Strings befindet.*

- bool `contains` (`std::vector< int > &Vec`, `const int &Element`)  
*Testet, ob sich eine Ganzzahl in einer Liste von Ganzzahlen befindet.*
- string `getTextBlock` (string `data`, int `n`)  
*Gibt den n-ten durch Leerzeichen abgetrennten Block aus einem String zurück.*
- void `parseLine` (string `line`, `vector< string > &out`, `vector< string > *timestamps`, `vector< string > *names`, `vector< int > *valid_cols`)  
*Sammelt Daten aus einer Textzeile (string).*
- void `replaceAll` (string `&str`, `const string from`, `const string to`)  
*Ersetzt in einem String alle Vorkommen eines Teilstrings durch einen Anderen.*
- bool `readConfiguration` (string `binary_path`)  
*Liest und setzt die Programmkonfiguration aus der Konfigurationsdatei.*
- bool `readSensorDefinitions` (string `path`, `vector< string > *sensor_names`, `vector< string > *sensor_data`)  
*Liest die Daten aus der Sensordefinitionsdatei.*
- bool `parseArguments` (int `argc`, char `*argv[]`, string `&sdef_file`, string `&input_file`, string `&output_file`)  
*Wertet die Programmargumente aus.*
- bool `readInputFile` (string `path`, `vector< string > &sensor_names`, `vector< vector< string > > &values`, `vector< string > &timestamps`, `vector< string > &names`)  
*Liest die Daten aus der Eingabedatei.*
- bool `writeOutputFile` (string `path`, `vector< string > &sensor_names`, `vector< string > &sensor_data`, `vector< vector< string > > &values`, `vector< string > &timestamps`, `vector< string > &names`)  
*Schreibt die Ausgabedatei.*

## Geschützte Attribute

- string `configpaths` [`NUMBEROFPATHS`]  
*Suchpfade für die Konfigurationsdatei.*
- struct `CsvToSdConverter::Options` `opts`  
*Hält die verwendeten Programmeinstellungen.*

## Statische, geschützte Attribute

- static const int `NUMBEROFPATHS` = 3  
*Anzahl der Suchpfade für die Konfigurationsdatei.*

### 8.6.1 Ausführliche Beschreibung

Konverter von .csv zu .tsd.

Definiert in Zeile 19 der Datei main.cpp.

### 8.6.2 Dokumentation der Elementfunktionen

8.6.2.1 bool `CsvToSdConverter::contains` ( `std::vector< string > & Vec`, `const string & Element` ) [inline],  
[protected]

Testet, ob sich ein String in einer Liste von Strings befindet.

Parameter

---

<i>Vec</i>	Liste der Strings.
<i>Element</i>	Der zu suchende String.

**Rückgabe**

true, wenn das Element gefunden wurde, sonst false.

Definiert in Zeile 55 der Datei main.cpp.

**8.6.2.2** `bool CsvToSdConverter::contains ( std::vector< int > & Vec, const int & Element ) [inline], [protected]`

Testet, ob sich eine Ganzzahl in einer Liste von Ganzzahlen befindet.

**Parameter**

<i>Vec</i>	Liste der Ganzzahlen.
<i>Element</i>	Die zu suchende Ganzzahl.

**Rückgabe**

true, wenn das Element gefunden wurde, sonst false.

Definiert in Zeile 71 der Datei main.cpp.

**8.6.2.3** `int CsvToSdConverter::convert ( int argc, char * argv[] ) [inline]`

Wandelt die Daten der .csv-Datei in eine .tsd-Datei um.

Wird durch die Funktion `main()` von außerhalb des Namespaces aufgerufen.

**Parameter**

<i>argc</i>	Anzahl der Programmargumente.
<i>argv</i>	Die Programmargumente.

Definiert in Zeile 621 der Datei main.cpp.

Hier ist ein Graph der zeigt, wo diese Funktion aufgerufen wird:



**8.6.2.4** `string CsvToSdConverter::getTextBlock ( string data, int n ) [inline], [protected]`

Gibt den n-ten durch Leerzeichen abgetrennten Block aus einem String zurück.

## Parameter

<i>data</i>	Der Ausgangsstring.
<i>n</i>	Index des zu findenden Blocks.

## Rückgabe

Der n-te durch Leerzeichen getrennte Teilstring. "" Bei ungültigem Index.

Definiert in Zeile 87 der Datei main.cpp.

**8.6.2.5** `bool CsvToSdConverter::parseArguments ( int argc, char * argv[], string & sdef_file, string & input_file, string & output_file )` `[inline]`, `[protected]`

Wertet die Programmargumente aus.

## Parameter

<i>argc</i>	Anzahl der Programmargumente.
<i>argv</i>	Die Programmargumente.
<i>sdef_file</i>	Ausgabe für den Pfad zur Sensordefinitionsdatei.
<i>input_file</i>	Ausgabe für den Pfad zur Eingabedatei.
<i>output_file</i>	Ausgabe für den Pfad zur Ausgabedatei.

## Rückgabe

Soll das Programm weiter ablaufen?

Definiert in Zeile 346 der Datei main.cpp.

**8.6.2.6** `void CsvToSdConverter::parseLine ( string line, vector< string > & out, vector< string > * timestamps, vector< string > * names, vector< int > * valid_cols )` `[inline]`, `[protected]`

Sammelt Daten aus einer Textzeile (string).

## Parameter

<i>line</i>	Die zu untersuchende Textzeile.
<i>out</i>	Ausgabevariable für die Sensordaten der Zeile. Alle Spalten nach opts.start_col werden als Sensordatenspalten betrachtet.
<i>timestamps</i>	Wenn nicht NULL, Ausgabevariable für den Zeitstempel der Zeile (opts.timecol). Der Zeitstempel wird an die übergebene Liste angehängt.
<i>names</i>	Wenn nicht NULL, Ausgabevariable für den Namen der Zeile (opts.namecol). Der Name wird an die übergebene Liste angehängt.
<i>valid_cols</i>	Wenn nicht NULL, werden nur die Sensordaten-Spalten mit den Indices dieser Liste ausgewertet.

Definiert in Zeile 127 der Datei main.cpp.

**8.6.2.7** `bool CsvToSdConverter::readConfiguration ( string binary_path )` `[inline]`, `[protected]`

Liest und setzt die Programmkonfiguration aus der Konfigurationsdatei.



## Parameter

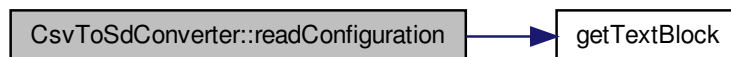
<i>binary_path</i>	Pfad zur Binärdatei.
--------------------	----------------------

## Rückgabe

War das Einlesen erfolgreich?

Definiert in Zeile 209 der Datei main.cpp.

Hier ist ein Graph, der zeigt, was diese Funktion aufruft:



**8.6.2.8** `bool CsvToSdConverter::readInputFile ( string path, vector< string > & sensor_names, vector< vector< string > > & values, vector< string > & timestamps, vector< string > & names )` [inline], [protected]

Liest die Daten aus der Eingabedatei.

## Parameter

<i>path</i>	Der Pfad zur Eingabedatei.
<i>sensor_names</i>	Liste der Namen der verwendeten Sensoren.
<i>values</i>	Liste für die extrahierten Sensorwerte.
<i>timestamps</i>	Liste für die Zeitstempel der Messwerte.
<i>names</i>	Liste für die Namen der Datensätze.

## Rückgabe

War das Einlesen erfolgreich?

Definiert in Zeile 454 der Datei main.cpp.

**8.6.2.9** `bool CsvToSdConverter::readSensorDefinitions ( string path, vector< string > * sensor_names, vector< string > * sensor_data )` [inline], [protected]

Liest die Daten aus der Sensordefinitionsdatei.

## Parameter

<i>path</i>	Pfad zur Binärdatei.
<i>sensor_names</i>	Liste für die Namen der Sensoren.
<i>sensor_data</i>	Liste für die Daten der Sensorden (Koordinaten).

## Rückgabe

War das Einlesen erfolgreich?

Definiert in Zeile 265 der Datei main.cpp.

8.6.2.10 `void CsvToSdConverter::replaceAll ( string & str, const string from, const string to )` `[inline]`,  
`[protected]`

Ersetzt in einem String alle Vorkommen eines Teilstrings durch einen Anderen.

## Parameter

<i>str</i>	Der zu durchsuchende String.
<i>from</i>	Der zu ersetzende Teilstring.
<i>to</i>	Der Teilstring, durch den ersetzt werden soll.

Definiert in Zeile 187 der Datei main.cpp.

**8.6.2.11** `bool CsvToSdConverter::writeOutputFile ( string path, vector< string > & sensor_names, vector< string > & sensor_data, vector< vector< string > > & values, vector< string > & timestamps, vector< string > & names ) [inline], [protected]`

Schreibt die Ausgabedatei.

## Parameter

<i>path</i>	Der Pfad zur Ausgabedatei.
<i>sensor_names</i>	Liste der Namen der verwendeten Sensoren.
<i>sensor_data</i>	Liste der Koordinaten der verwendeten Sensoren.
<i>values</i>	Liste für die extrahierten Sensorwerte.
<i>timestamps</i>	Liste für die Zeitstempel der Messwerte.
<i>names</i>	Liste für die Namen der Datensätze.

## Rückgabe

War das Schreiben erfolgreich?

Definiert in Zeile 572 der Datei main.cpp.

### 8.6.3 Dokumentation der Datenelemente

**8.6.3.1** `string CsvToSdConverter::configpaths[NUMBEROFPATHS] [protected]`

## Initialisierung:

```
{
    "/etc/simpleanalyzer/csvtosd.conf",
    "/usr/local/share/simpleanalyzer/csvtosd.conf",
    "/usr/share/simpleanalyzer/csvtosd.conf" }
```

Suchpfade für die Konfigurationsdatei.

Das Verzeichnis der ausführbaren Datei wird immer geprüft.

Definiert in Zeile 30 der Datei main.cpp.

**8.6.3.2** `const int CsvToSdConverter::NUMBEROFPATHS = 3 [static], [protected]`

Anzahl der Suchpfade für die Konfigurationsdatei.

Definiert in Zeile 24 der Datei main.cpp.

**8.6.3.3** `struct CsvToSdConverter::Options CsvToSdConverter::opts [protected]`

Hält die verwendeten Programmeinstellungen.

Die Dokumentation für diese Klasse wurde erzeugt aufgrund der Datei:

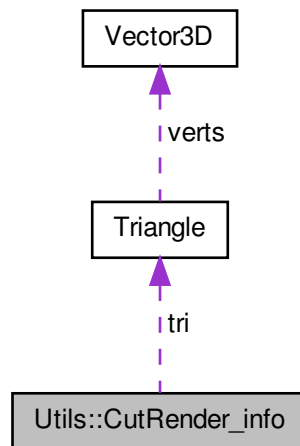
- /daten/Projekte/eclipse\_workspace/csvtosd/[main.cpp](#)

## 8.7 Utils::CutRender\_info Strukturreferenz

Daten zur Darstellung einer 2D-Temperaturverteilungs-Ebene.

```
#include <utils.h>
```

Zusammengehörigkeiten von Utils::CutRender\_info:



### Öffentliche Attribute

- [Triangle](#) \* [tri](#)  
*Das die Ebene beschreibende Dreieck.*
- float [mmperpixel](#)  
*Maßstab der Darstellung der Temperaturverteilung in  $\frac{mm}{Pixel}$ .*
- int [img\\_width](#)  
*Breite der Darstellung der Temperaturverteilung.*
- int [img\\_height](#)  
*Höhe der Darstellung der Temperaturverteilung.*
- [PIM\\_algorithm in\\_volume\\_algorithm](#)  
*Der zu verwendende Punkt-in-Volumen-Testalgorithmus.*

### 8.7.1 Ausführliche Beschreibung

Daten zur Darstellung einer 2D-Temperaturverteilungs-Ebene.

Definiert in Zeile 78 der Datei utils.h.

### 8.7.2 Dokumentation der Datenelemente

#### 8.7.2.1 int Utils::CutRender\_info::img\_height

Höhe der Darstellung der Temperaturverteilung.

Definiert in Zeile 82 der Datei utils.h.

8.7.2.2 `int Utils::CutRender_info::img_width`

Breite der Darstellung der Temperaturverteilung.

Definiert in Zeile 81 der Datei `utils.h`.

8.7.2.3 `PIM_algorithm Utils::CutRender_info::in_volume_algorithm`

Der zu verwendende Punkt-in-Volumen-Testalgorithmus.

Immer `ALGORITHM_TETRAHEDRONS`.

Definiert in Zeile 83 der Datei `utils.h`.

8.7.2.4 `float Utils::CutRender_info::mmperpixel`

Maßstab der Darstellung der Temperaturverteilung in  $\frac{mm}{Pixel}$ .

Definiert in Zeile 80 der Datei `utils.h`.

8.7.2.5 `Triangle* Utils::CutRender_info::tri`

Das die Ebene beschreibende Dreieck.

Der erste Punkt ist dabei das Zentrum der später ermittelten Temperaturverteilung.

Definiert in Zeile 79 der Datei `utils.h`.

Die Dokumentation für diese Struktur wurde erzeugt aufgrund der Datei:

- [/daten/Projekte/eclipse\\_workspace/simpleanalyzer-gui/src/processing/utils.h](#)

## 8.8 Exporter Klassenreferenz

Export der gewonnenen Daten.

```
#include <Exporter.h>
```

### Öffentliche Methoden

- [Exporter](#) ()  
*Der Konstruktor.*
- [ObjectData::ObjectDataStatus ExportLegacyVTK](#) (string filename, [ObjectData](#) \*data)  
*Exportiert die aktuell berechnete dreidimensionale Temperaturverteilung und das Modell als VTK-Datei.*
- [ObjectData::ObjectDataStatus ExportCutCSV](#) (string filename, float \*values, [CutRender\\_info](#) \*info)  
*Exportiert die zweidimensionale Temperaturverteilung (Schnitt durch das Modell) als csv-Datei.*
- virtual [~Exporter](#) ()  
*Der Destruktor.*

### Geschützte Attribute

- const char \* [CSV\\_SEPARATOR](#)  
*Das in der .csv-Datei verwendete Separatorzeichen.*

### 8.8.1 Ausführliche Beschreibung

Export der gewonnenen Daten.

Klasse zum Export der dreidimensionalen Temperaturverteilung als VTK-Datei und der zweidimensionalen Temperaturverteilung (Schnitt durch das Modell) als .csv-Datei.

Definiert in Zeile 22 der Datei Exporter.h.

### 8.8.2 Beschreibung der Konstruktoren und Destruktoren

#### 8.8.2.1 Exporter::Exporter ( )

Der Konstruktor.

Definiert in Zeile 15 der Datei Exporter.cpp.

#### 8.8.2.2 Exporter::~~Exporter ( ) [virtual]

Der Destruktor.

Definiert in Zeile 168 der Datei Exporter.cpp.

### 8.8.3 Dokumentation der Elementfunktionen

#### 8.8.3.1 ObjectData::ObjectDataStatus Exporter::ExportCutCSV ( string filename, float \* values, CutRender\_info \* info )

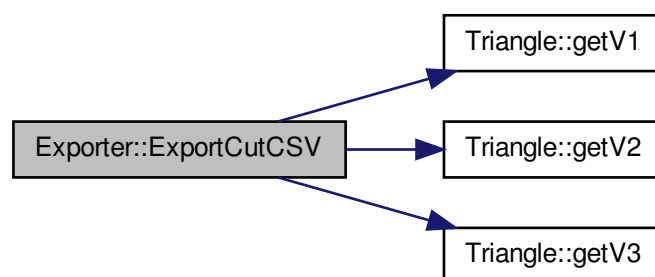
Exportiert die zweidimensionale Temperaturverteilung (Schnitt durch das Modell) als csv-Datei.

##### Rückgabe

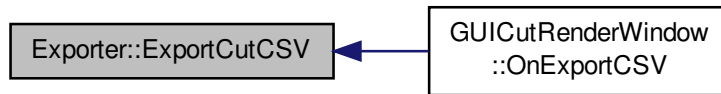
Der Fehlercode.

Definiert in Zeile 124 der Datei Exporter.cpp.

Hier ist ein Graph, der zeigt, was diese Funktion aufruft:



Hier ist ein Graph der zeigt, wo diese Funktion aufgerufen wird:



### 8.8.3.2 ObjectData::ObjectDataStatus Exporter::ExportLegacyVTK ( string filename, ObjectData \* data )

Exportiert die aktuell berechnete dreidimensionale Temperaturverteilung und das Modell als VTK-Datei.

#### Rückgabe

Der Fehlercode.

Definiert in Zeile 23 der Datei Exporter.cpp.

Hier ist ein Graph, der zeigt, was diese Funktion aufruft:



Hier ist ein Graph der zeigt, wo diese Funktion aufgerufen wird:



## 8.8.4 Dokumentation der Datenelemente

### 8.8.4.1 const char\* Exporter::CSV\_SEPARATOR [protected]

Das in der .csv-Datei verwendete Separatorzeichen.

Definiert in Zeile 50 der Datei Exporter.h.

Die Dokumentation für diese Klasse wurde erzeugt aufgrund der Dateien:

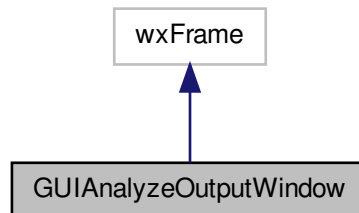
- [/daten/Projekte/eclipse\\_workspace/simpleanalyzer-gui/src/fileIO/Exporter.h](#)
- [/daten/Projekte/eclipse\\_workspace/simpleanalyzer-gui/src/fileIO/Exporter.cpp](#)

## 8.9 GUIAnalyzeOutputWindow Klassenreferenz

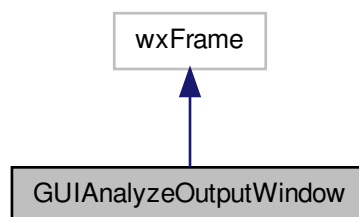
Übersichtsfenster über die Analysedaten.

```
#include <GUIAnalyzeOutputWindow.h>
```

Klassendiagramm für GUIAnalyzeOutputWindow:



Zusammengehörigkeiten von GUIAnalyzeOutputWindow:



### Öffentliche Methoden

- [GUIAnalyzeOutputWindow](#) (`wxWindow *parent`, `const wxChar *title`, `int xpos`, `int ypos`, `int width`, `int height`)  
*Der Konstruktor.*
- void [Update](#) ()  
*Methode zum aktualisieren des Fensters, alle Objekte werden erneut analysiert und die aktualisierten Ergebnisse angezeigt.*
- virtual [~GUIAnalyzeOutputWindow](#) ()  
*Der Destruktor.*

### Private Methoden

- void [OnKeyPress](#) (`wxKeyEvent &event`)  
*Event-Tabellendeklaration für wxWidgets.*
- void [SelectAll](#) ()



*Selektiert alle Zellen der Tabelle.*

- void [ToClipboard](#) ()

*Kopiert die Inhalte der Tabelle in die Zwischenablage.*

## Private Attribute

- wxGrid \* [table](#)

*Die Tabellenkomponente.*

### 8.9.1 Ausführliche Beschreibung

Übersichtsfenster über die Analysedaten.

Dieses Fenster zeigt eine Tabelle mit den zur Analyse markierten Zeitpunkten für alle Objekte und deren Datensätze und Materialien. Nicht-zeitabhängige Sensordaten werden immer angezeigt.

Definiert in Zeile 22 der Datei GUIAnalyzeOutputWindow.h.

### 8.9.2 Beschreibung der Konstruktoren und Destruktoren

**8.9.2.1** GUIAnalyzeOutputWindow::GUIAnalyzeOutputWindow ( wxWindow \* *parent*, const wxChar \* *title*, int *xpos*, int *ypos*, int *width*, int *height* )

Der Konstruktor.

Definiert in Zeile 23 der Datei GUIAnalyzeOutputWindow.cpp.

**8.9.2.2** GUIAnalyzeOutputWindow::~GUIAnalyzeOutputWindow ( ) [virtual]

Der Destruktor.

Definiert in Zeile 210 der Datei GUIAnalyzeOutputWindow.cpp.

Hier ist ein Graph, der zeigt, was diese Funktion aufruft:



### 8.9.3 Dokumentation der Elementfunktionen

**8.9.3.1** void GUIAnalyzeOutputWindow::OnKeyPress ( wxKeyEvent & *event* ) [private]

Event-Tabellendeklaration für wxWidgets.

Behandelt das Drücken von Strg+C und Strg+A.

Definiert in Zeile 197 der Datei GUIAnalyzeOutputWindow.cpp.

### 8.9.3.2 void GUIAnalyzeOutputWindow::SelectAll ( ) [private]

Selektiert alle Zellen der Tabelle.

Definiert in Zeile 189 der Datei GUIAnalyzeOutputWindow.cpp.

### 8.9.3.3 void GUIAnalyzeOutputWindow::ToClipboard ( ) [private]

Kopiert die Inhalte der Tabelle in die Zwischenablage.

Basierend auf <http://forums.wxwidgets.org/viewtopic.php?f=20&t=2200#p148731>.

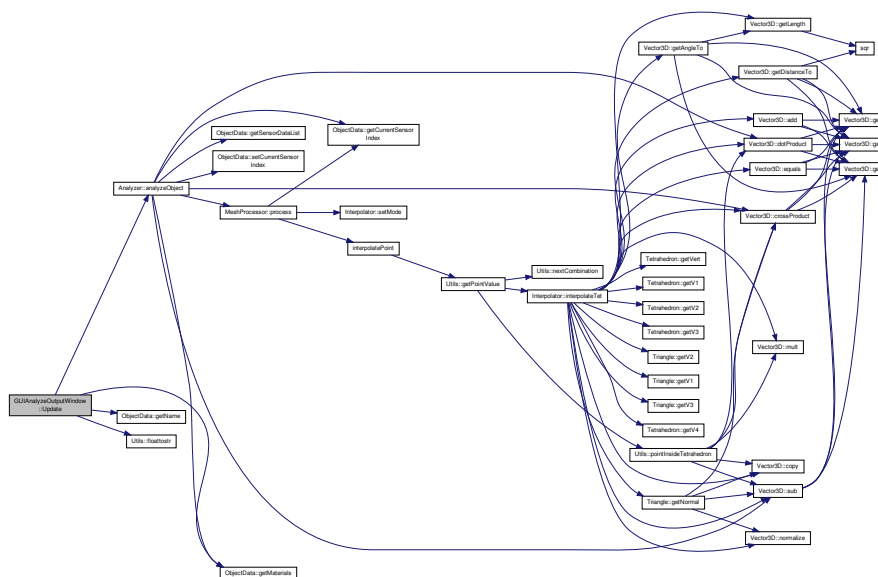
Definiert in Zeile 151 der Datei GUIAnalyzeOutputWindow.cpp.

### 8.9.3.4 void GUIAnalyzeOutputWindow::Update ( )

Methode zum aktualisieren des Fensters, alle Objekte werden erneut analysiert und die aktualisierten Ergebnisse angezeigt.

Definiert in Zeile 37 der Datei GUIAnalyzeOutputWindow.cpp.

Hier ist ein Graph, der zeigt, was diese Funktion aufruft:



## 8.9.4 Dokumentation der Datenelemente

### 8.9.4.1 wxGrid\* GUIAnalyzeOutputWindow::table [private]

Die Tabellenkomponente.

Definiert in Zeile 65 der Datei GUIAnalyzeOutputWindow.h.

Die Dokumentation für diese Klasse wurde erzeugt aufgrund der Dateien:

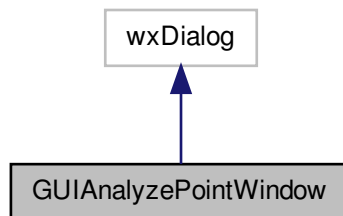
- [/daten/Projekte/eclipse\\_workspace/simpleanalyzer-gui/src/GUI/GUIAnalyzeOutputWindow.h](#)
- [/daten/Projekte/eclipse\\_workspace/simpleanalyzer-gui/src/GUI/GUIAnalyzeOutputWindow.cpp](#)

## 8.10 GUIAnalyzePointWindow Klassenreferenz

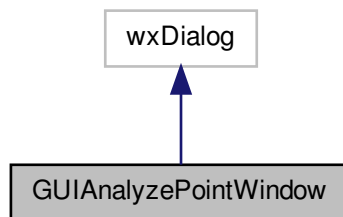
Analysefenster für einen Punkt.

```
#include <GUIAnalyzePointWindow.h>
```

Klassendiagramm für GUIAnalyzePointWindow:



Zusammengehörigkeiten von GUIAnalyzePointWindow:



### Öffentliche Methoden

- `GUIAnalyzePointWindow` (`wxWindow *parent`, `const wxChar *title`, `int xpos`, `int ypos`, `int width`, `int height`)  
*Der Konstruktor.*
- `virtual ~GUIAnalyzePointWindow` ()  
*Der Destruktor.*

### Private Methoden

- `void analyzePoint` (`wxCommandEvent &event`)  
*Event-Tabellendeklaration für wxWidgets.*

### Private Attribute

- `wxStaticText * label`

*Beschriftung der Fensterkomponenten.*

- wxTextCtrl \* [xedit](#)

*Eingabefeld für die X-Koordinate.*

- wxTextCtrl \* [yedit](#)

*Eingabefeld für die Y-Koordinate.*

- wxTextCtrl \* [zedit](#)

*Eingabefeld für die Z-Koordinate.*

- wxStaticText \* [interpolationModeLabel](#)

*Beschriftung für den Interpolationsmodus.*

- wxComboBox \* [interpolationModeList](#)

*Dropdown-Menü für den Interpolationsmodus.*

- wxButton \* [calcbt](#)

*Button zum Auslösen der Analyseprozedur.*

### 8.10.1 Ausführliche Beschreibung

Analysefenster für einen Punkt.

Definiert in Zeile 16 der Datei GUIAnalyzePointWindow.h.

### 8.10.2 Beschreibung der Konstruktoren und Destruktoren

#### 8.10.2.1 GUIAnalyzePointWindow::GUIAnalyzePointWindow ( wxWindow \* *parent*, const wxChar \* *title*, int *xpos*, int *ypos*, int *width*, int *height* )

Der Konstruktor.

Definiert in Zeile 22 der Datei GUIAnalyzePointWindow.cpp.

#### 8.10.2.2 GUIAnalyzePointWindow::~~GUIAnalyzePointWindow ( ) [virtual]

Der Destruktor.

Definiert in Zeile 93 der Datei GUIAnalyzePointWindow.cpp.

### 8.10.3 Dokumentation der Elementfunktionen

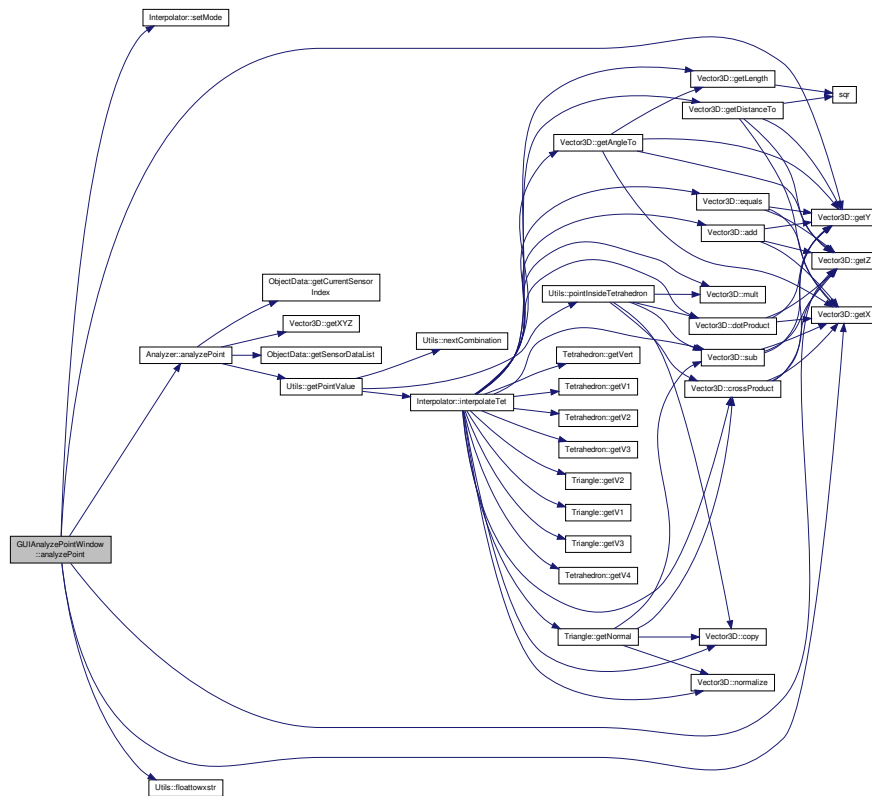
#### 8.10.3.1 void GUIAnalyzePointWindow::analyzePoint ( wxCommandEvent & *event* ) [private]

Event-Tabellendeklaration für wxWidgets.

Ermittelt Temperatur und Art des Punktes (Interpoliert/Extrapoliert). Wird durch Event ausgelöst.

Definiert in Zeile 56 der Datei GUIAnalyzePointWindow.cpp.

Hier ist ein Graph, der zeigt, was diese Funktion aufruft:



## 8.10.4 Dokumentation der Datenelemente

### 8.10.4.1 wxButton\* GUIAnalyzePointWindow::calcbtn [private]

Button zum Auslösen der Analyseprozedur.

Definiert in Zeile 72 der Datei GUIAnalyzePointWindow.h.

### 8.10.4.2 wxStaticText\* GUIAnalyzePointWindow::interpolationModelLabel [private]

Beschriftung für den Interpolationsmodus.

Definiert in Zeile 62 der Datei GUIAnalyzePointWindow.h.

### 8.10.4.3 wxComboBox\* GUIAnalyzePointWindow::interpolationModeList [private]

Dropdown-Menü für den Interpolationsmodus.

Definiert in Zeile 67 der Datei GUIAnalyzePointWindow.h.

### 8.10.4.4 wxStaticText\* GUIAnalyzePointWindow::label [private]

Beschriftung der Fensterkomponenten.

Definiert in Zeile 42 der Datei GUIAnalyzePointWindow.h.

#### 8.10.4.5 wxTextCtrl\* GUIAnalyzePointWindow::xedit [private]

Eingabefeld für die X-Koordinate.

Definiert in Zeile 47 der Datei GUIAnalyzePointWindow.h.

#### 8.10.4.6 wxTextCtrl\* GUIAnalyzePointWindow::yedit [private]

Eingabefeld für die Y-Koordinate.

Definiert in Zeile 52 der Datei GUIAnalyzePointWindow.h.

#### 8.10.4.7 wxTextCtrl\* GUIAnalyzePointWindow::zedit [private]

Eingabefeld für die Z-Koordinate.

Definiert in Zeile 57 der Datei GUIAnalyzePointWindow.h.

Die Dokumentation für diese Klasse wurde erzeugt aufgrund der Dateien:

- [/daten/Projekte/eclipse\\_workspace/simpleanalyzer-gui/src/GUI/GUIAnalyzePointWindow.h](#)
- [/daten/Projekte/eclipse\\_workspace/simpleanalyzer-gui/src/GUI/GUIAnalyzePointWindow.cpp](#)

## 8.11 GUIColorScalePanel Klassenreferenz

Farbige Temperaturskala für zweidimensionale Temperaturverteilung.

```
#include <GUIColorScalePanel.h>
```

### Öffentliche Typen

- enum [ScaleMode](#) { [SCM\\_NONE](#) = 0, [SCM\\_HORIZONTAL](#), [SCM\\_VERTICAL](#) }
- Modus der Skalendarstellung.*

### Öffentliche Methoden

- [GUIColorScalePanel](#) ()  
*Der Konstruktor.*
- void [refresh](#) (int img\_width, int img\_height)  
*Zeichnet die Temperaturskala neu.*
- void [paintTo](#) (wxDC &dc, float zoom, wxPoint &img\_coords)  
*Zeichnet die Temperaturskala mit einem bestimmten device context.*
- void [handleMouse](#) (wxMouseEvent &event, wxPoint &img\_coords, wxPoint &img\_dim, float zoom)  
*Behandelt die Mausektionen und verändert ggf.*
- void [getDisplayArea](#) (wxRect \*rect, float zoom)  
*Gibt die bei einem bestimmten Zoomfaktor eingenommene Fläche zurück.*
- void [fitBounds](#) (wxPoint &img\_dim, bool to\_scale)  
*Passt die Größe und Position der Skala an die Größe der Grafik an.*
- bool [mouseOnDisplayArea](#) (wxPoint &img\_coords, float zoom, wxPoint &mouse\_pos)  
*Gibt zurück, ob sich die Maus über der Fläche der Skala befindet.*
- int [getX](#) ()
- int [getY](#) ()
- int [getFontSize](#) () const

- void `setFontSize` (int fontSize)  
*Setzt die Schriftgröße der Skala.*
- `ScaleMode` `getMode` () const
- void `setMode` (`ScaleMode` mode)  
*Setzt den Modus der Skala.*
- const wxColour & `getTextColor` () const
- void `setTextColor` (const wxColour &textColor)  
*Setzt die Schriftfarbe der Skala.*
- int `getStepWidth` () const  
*Gibt die Schrittweite der Skalenbeschriftung.*
- void `setStepWidth` (int stepWidth)  
*Setzt die Schrittweite der Skalenbeschriftung.*
- wxImage \* `getImage` () const
- virtual `~GUIColorScalePanel` ()  
*Der Destruktor.*

### Private Attribute

- int `step_width`  
*Schrittweite der Beschriftung.*
- int `font_size`  
*Die Schriftgröße.*
- `ScaleMode` `mode`  
*Der Darstellungsmodus.*
- wxColour `text_color`  
*Die Schriftfarbe.*
- wxImage \* `image`  
*Bild, das die Skala ohne Steuerelemente enthält.*
- int `current_mx`  
*Zwischenspeicher für die Mausposition, zum behandeln von Mausinteraktionen.*
- int `current_my`  
*Zwischenspeicher für die Mausposition, zum behandeln von Mausinteraktionen.*
- float `x`  
*Position (X) der Skala.*
- float `y`  
*Position Y) der Skala.*
- float `width`  
*Breite der Skala.*
- float `height`  
*Höhe der Skala.*
- bool `scaling`  
*Wird gerade in der Größe verändert.*
- bool `transforming`  
*Wird gerade transformiert (Größe oder Position).*
- bool `prev_mouse_down`  
*zwischenspeicher für den Mausstatus.*

### 8.11.1 Ausführliche Beschreibung

Farbige Temperaturskala für zweidimensionale Temperaturverteilung.

Farbige Temperaturskala für zweidimensionale Temperaturverteilung. Wird für die Darstellung einer farbigen Temperaturskala im Anzeigefenster auf der als zweidimensionale Temperaturverteilung erzeugten Grafik verwendet.

Definiert in Zeile 21 der Datei GUIColorScalePanel.h.

### 8.11.2 Dokumentation der Aufzählungstypen

#### 8.11.2.1 enum GUIColorScalePanel::ScaleMode

Modus der Skalendarstellung.

Aufzählungswerte

**SCM\_NONE** Keine Skala.

**SCM\_HORIZONTAL** Eine horizontal ausgerichtete Skala.

**SCM\_VERTICAL** Eine vertikal ausgerichtete Skala.

Definiert in Zeile 26 der Datei GUIColorScalePanel.h.

### 8.11.3 Beschreibung der Konstruktoren und Destruktoren

#### 8.11.3.1 GUIColorScalePanel::GUIColorScalePanel ( )

Der Konstruktor.

Definiert in Zeile 21 der Datei GUIColorScalePanel.cpp.

#### 8.11.3.2 GUIColorScalePanel::~GUIColorScalePanel ( ) [virtual]

Der Destruktor.

Definiert in Zeile 457 der Datei GUIColorScalePanel.cpp.

### 8.11.4 Dokumentation der Elementfunktionen

#### 8.11.4.1 void GUIColorScalePanel::fitBounds ( wxPoint & img\_dim, bool to\_scale )

Passt die Größe und Position der Skala an die Größe der Grafik an.

Parameter

<i>img_dim</i>	Größe der Grafik.
<i>to_scale</i>	Größe statt der Position verändern.

Definiert in Zeile 296 der Datei GUIColorScalePanel.cpp.



Hier ist ein Graph der zeigt, wo diese Funktion aufgerufen wird:



#### 8.11.4.2 void GUIColorScalePanel::getDisplayArea ( wxRect \* rect, float zoom )

Gibt die bei einem bestimmten Zoomfaktor eingenommene Fläche zurück.

Definiert in Zeile 261 der Datei GUIColorScalePanel.cpp.

#### 8.11.4.3 int GUIColorScalePanel::getFontSize ( ) const

**Rückgabe**

Schriftgröße der Skala.

Definiert in Zeile 421 der Datei GUIColorScalePanel.cpp.

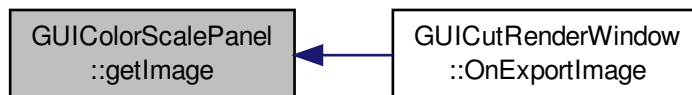
#### 8.11.4.4 wxImage \* GUIColorScalePanel::getImage ( ) const

**Rückgabe**

Skala als Grafik.

Definiert in Zeile 453 der Datei GUIColorScalePanel.cpp.

Hier ist ein Graph der zeigt, wo diese Funktion aufgerufen wird:



#### 8.11.4.5 GUIColorScalePanel::ScaleMode GUIColorScalePanel::getMode ( ) const

**Rückgabe**

Modus der Skala.

Definiert in Zeile 429 der Datei GUIColorScalePanel.cpp.

#### 8.11.4.6 `int GUIColorScalePanel::getStepWidth ( ) const`

Gibt die Schrittweite der Skalenbeschriftung.

Definiert in Zeile 445 der Datei GUIColorScalePanel.cpp.

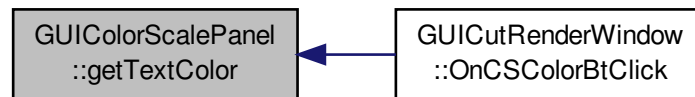
#### 8.11.4.7 `const wxColour & GUIColorScalePanel::getTextColor ( ) const`

##### Rückgabe

Schriftfarbe der Skala.

Definiert in Zeile 437 der Datei GUIColorScalePanel.cpp.

Hier ist ein Graph der zeigt, wo diese Funktion aufgerufen wird:



#### 8.11.4.8 `int GUIColorScalePanel::getX ( )`

##### Rückgabe

horizontale Position auf der Zeichenfläche.

Definiert in Zeile 288 der Datei GUIColorScalePanel.cpp.

#### 8.11.4.9 `int GUIColorScalePanel::getY ( )`

##### Rückgabe

vertikale Position auf der Zeichenfläche.

Definiert in Zeile 292 der Datei GUIColorScalePanel.cpp.

#### 8.11.4.10 `void GUIColorScalePanel::handleMouse ( wxMouseEvent & event, wxPoint & img_coords, wxPoint & img_dim, float zoom )`

Behandelt die Mauseaktionen und verändert ggf.

Größe oder Position des Skala.

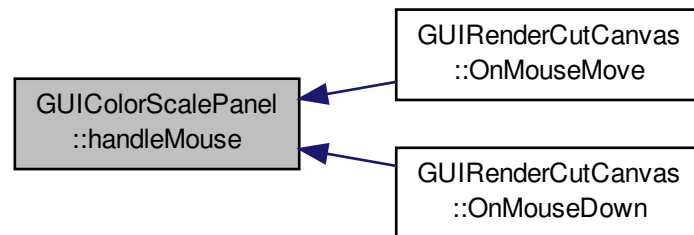
##### Parameter

<i>event</i>	Das zu behandelnde Maus-event.
--------------	--------------------------------

<i>img_coords</i>	Position der Grafik auf der Zeichenfläche.
<i>img_dim</i>	Größe der Grafik.
<i>zoom</i>	aktueller Vergrößerungsfaktor des Betrachtungsfensters.

Definiert in Zeile 360 der Datei GUIColorScalePanel.cpp.

Hier ist ein Graph der zeigt, wo diese Funktion aufgerufen wird:



#### 8.11.4.11 bool GUIColorScalePanel::mouseOnDisplayArea ( wxPoint & *img\_coords*, float *zoom*, wxPoint & *mouse\_pos* )

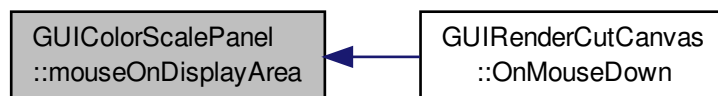
Gibt zurück, ob sich die Maus über der Fläche der Skala befindet.

Parameter

<i>img_coords</i>	Position der Grafik auf der Zeichenfläche.
<i>zoom</i>	aktueller Vergrößerungsfaktor des Betrachtungsfensters.
<i>mouse_pos</i>	Position der Maus auf der Zeichenfläche.

Definiert in Zeile 269 der Datei GUIColorScalePanel.cpp.

Hier ist ein Graph der zeigt, wo diese Funktion aufgerufen wird:



#### 8.11.4.12 void GUIColorScalePanel::paintTo ( wxDC & *dc*, float *zoom*, wxPoint & *img\_coords* )

Zeichnet die Temperaturskala mit einem bestimmten device context.

## Parameter

<i>dc</i>	Der zum Zeichnen zu verwendende device context.
<i>zoom</i>	Faktor zum Skalieren der Skala.
<i>img_coords</i>	Position der Grafik auf der Zeichenfläche.

Definiert in Zeile 42 der Datei GUIColorScalePanel.cpp.

Hier ist ein Graph der zeigt, wo diese Funktion aufgerufen wird:



#### 8.11.4.13 void GUIColorScalePanel::refresh ( int *img\_width*, int *img\_height* )

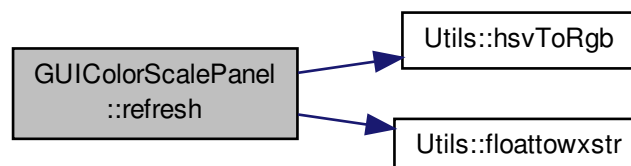
Zeichnet die Temperaturskala neu.

## Parameter

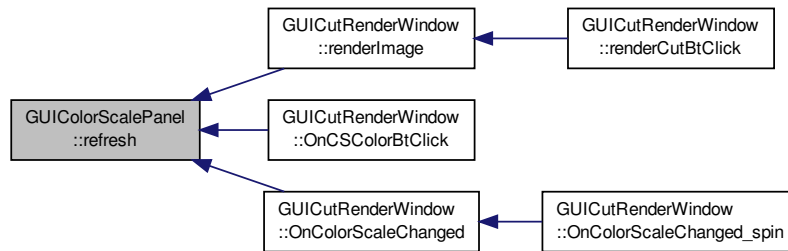
<i>img_width</i>	Breite des Bildes, für das die Skala gezeichnet wird.
<i>img_height</i>	Höhe des Bildes, für das die Skala gezeichnet wird.

Definiert in Zeile 85 der Datei GUIColorScalePanel.cpp.

Hier ist ein Graph, der zeigt, was diese Funktion aufruft:



Hier ist ein Graph der zeigt, wo diese Funktion aufgerufen wird:

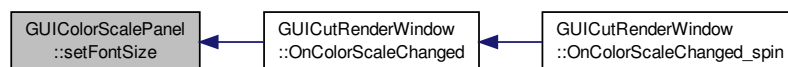


#### 8.11.4.14 void GUIColorScalePanel::setFontSize ( int *fontSize* )

Setzt die Schriftgröße der Skala.

Definiert in Zeile 425 der Datei `GUIColorScalePanel.cpp`.

Hier ist ein Graph der zeigt, wo diese Funktion aufgerufen wird:

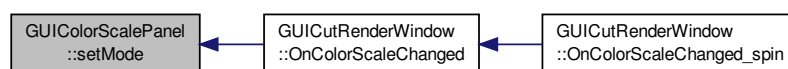


#### 8.11.4.15 void GUIColorScalePanel::setMode ( ScaleMode *mode* )

Setzt den Modus der Skala.

Definiert in Zeile 433 der Datei `GUIColorScalePanel.cpp`.

Hier ist ein Graph der zeigt, wo diese Funktion aufgerufen wird:

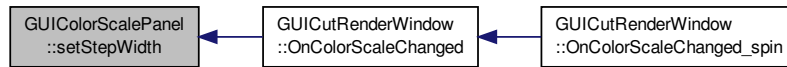


#### 8.11.4.16 void GUIColorScalePanel::setStepWidth ( int *stepWidth* )

Setzt die Schrittweite der Skalenbeschriftung.

Definiert in Zeile 449 der Datei `GUIColorScalePanel.cpp`.

Hier ist ein Graph der zeigt, wo diese Funktion aufgerufen wird:

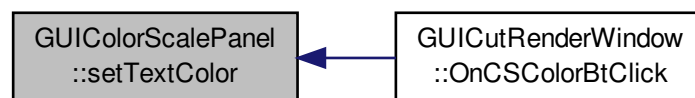


#### 8.11.4.17 void GUIColorScalePanel::setTextColor ( const wxColour & textColor )

Setzt die Schriftfarbe der Skala.

Definiert in Zeile 441 der Datei `GUIColorScalePanel.cpp`.

Hier ist ein Graph der zeigt, wo diese Funktion aufgerufen wird:



### 8.11.5 Dokumentation der Datenelemente

#### 8.11.5.1 int GUIColorScalePanel::current\_mx [private]

Zwischenspeicher für die Mausposition, zum behandeln von Mausinteraktionen.

Definiert in Zeile 170 der Datei `GUIColorScalePanel.h`.

#### 8.11.5.2 int GUIColorScalePanel::current\_my [private]

Zwischenspeicher für die Mausposition, zum behandeln von Mausinteraktionen.

Definiert in Zeile 175 der Datei `GUIColorScalePanel.h`.

#### 8.11.5.3 int GUIColorScalePanel::font\_size [private]

Die Schriftgröße.

Definiert in Zeile 150 der Datei `GUIColorScalePanel.h`.

#### 8.11.5.4 float GUIColorScalePanel::height [private]

Höhe der Skala.

Definiert in Zeile 195 der Datei `GUIColorScalePanel.h`.

**8.11.5.5 wxImage\* GUIColorScalePanel::image [private]**

Bild, das die Skala ohne Steuerelemente enthält.

Definiert in Zeile 165 der Datei GUIColorScalePanel.h.

**8.11.5.6 ScaleMode GUIColorScalePanel::mode [private]**

Der Darstellungsmodus.

Definiert in Zeile 155 der Datei GUIColorScalePanel.h.

**8.11.5.7 bool GUIColorScalePanel::prev\_mouse\_down [private]**

zwischenspeicher für den Mausstatus.

Definiert in Zeile 210 der Datei GUIColorScalePanel.h.

**8.11.5.8 bool GUIColorScalePanel::scaling [private]**

Wird gerade in der Größe verändert.

Definiert in Zeile 200 der Datei GUIColorScalePanel.h.

**8.11.5.9 int GUIColorScalePanel::step\_width [private]**

Schrittweite der Beschriftung.

Definiert in Zeile 145 der Datei GUIColorScalePanel.h.

**8.11.5.10 wxColour GUIColorScalePanel::text\_color [private]**

Die Schriftfarbe.

Definiert in Zeile 160 der Datei GUIColorScalePanel.h.

**8.11.5.11 bool GUIColorScalePanel::transforming [private]**

Wird gerade transformiert (Größe oder Position).

Definiert in Zeile 205 der Datei GUIColorScalePanel.h.

**8.11.5.12 float GUIColorScalePanel::width [private]**

Breite der Skala.

Definiert in Zeile 190 der Datei GUIColorScalePanel.h.

**8.11.5.13 float GUIColorScalePanel::x [private]**

Position (X) der Skala.

Definiert in Zeile 180 der Datei GUIColorScalePanel.h.

#### 8.11.5.14 float GUIColorScalePanel::y [private]

Position Y) der Skala.

Definiert in Zeile 185 der Datei GUIColorScalePanel.h.

Die Dokumentation für diese Klasse wurde erzeugt aufgrund der Dateien:

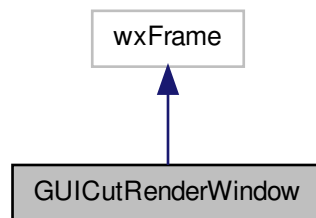
- [/daten/Projekte/eclipse\\_workspace/simpleanalyzer-gui/src/GUI/GUIColorScalePanel.h](#)
- [/daten/Projekte/eclipse\\_workspace/simpleanalyzer-gui/src/GUI/GUIColorScalePanel.cpp](#)

## 8.12 GUICutRenderWindow Klassenreferenz

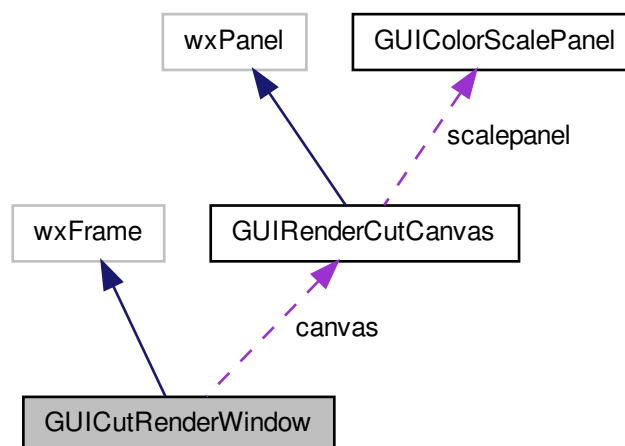
Fenster zum erstellen zweidimensionaler Temperaturverteilungen.

```
#include <GUICutRenderWindow.h>
```

Klassendiagramm für GUICutRenderWindow:



Zusammengehörigkeiten von GUICutRenderWindow:





## Öffentliche Methoden

- [GUICutRenderWindow](#) (wxWindow \*parent, const wxChar \*title, int xpos, int ypos, int width, int height)  
*Der Konstruktor.*
- virtual [~GUICutRenderWindow](#) ()  
*Der Destruktor.*

## Geschützte Methoden

- [DECLARE\\_EVENT\\_TABLE](#) ()  
*Event-Tabellendeklaration für wxWidgets.*

## Private Methoden

- [CutRender\\_info](#) \* [getCutRenderProperties](#) ()  
*Gibt die aktuell eingestellten Eigenschaften für die zweidimensionale Temperaturverteilung zurück, damit Sie später an den [Renderer](#) des 3D-Fensters zur Visualisierung übergeben werden können.*
- void [renderCutBtClick](#) (wxCommandEvent &event)  
*Behandelt das Drücken des Buttons zur Berechnung der zweidimensionalen Temperaturverteilung.*
- void [OnResize](#) (wxSizeEvent &event)  
*Behandelt Änderungen der Größe des Fensters.*
- void [OnCutPropsChanged](#) (wxCommandEvent &event)  
*Behandelt das Ändern von Parametern zur Berechnung der 2D-Temperaturverteilung.*
- void [refreshVisualisation](#) ()  
*Aktualisiert die Visualisierung der Schnittebene im Hauptfenster.*
- void [OnExportImage](#) (wxCommandEvent &event)  
*Fragt den Benutzer nach dem Pfad und Exportiert eine Grafik aus 2D-Temperaturverteilung und Temperaturskala.*
- void [OnExportCSV](#) (wxCommandEvent &event)  
*Fragt den Benutzer nach dem Pfad und Exportiert die 2D-Temperaturverteilung als .csv-Datei.*
- void [OnSCutPropsChanged\\_spin](#) (wxSpinEvent &event)  
*Behandelt das Ändern von Parametern zur Berechnung der 2D-Temperaturverteilung.*
- void [OnColorScaleChanged](#) (wxCommandEvent &event)  
*Behandelt das Ändern von Parametern zur darstellung der Temperaturskala.*
- void [OnColorScaleChanged\\_spin](#) (wxSpinEvent &event)  
*Behandelt das Ändern von Parametern zur darstellung der Temperaturskala.*
- void [OnCSColorBtClick](#) (wxCommandEvent &event)  
*Behandelt das Klicken auf den Button zur Wahl der Schriftfarbe auf der Skala.*
- void [renderImage](#) (wxImage \*image)  
*Berechnet die 2D-Temperaturverteilung als Grafik.*

## Private Attribute

- wxScrolledWindow \* [scroll\\_pane](#)  
*Scrollender Bereich, in den die anderen Komponenten außer der Zeichenfläche (canvas) eingebettet sind.*
- wxTextCtrl \* [p1xedit](#)  
*Textfeld zur Eingabe der Position des Dreiecks, dass die Schnittebene definiert.*
- wxTextCtrl \* [p1yedit](#)  
*Textfeld zur Eingabe der Position des Dreiecks, dass die Schnittebene definiert.*
- wxTextCtrl \* [p1zedit](#)  
*Textfeld zur Eingabe der Position des Dreiecks, dass die Schnittebene definiert.*

- wxTextCtrl \* [p2xedit](#)  
*Textfeld zur Eingabe der Position des Dreiecks, dass die Schnittebene definiert.*
- wxTextCtrl \* [p2yedit](#)  
*Textfeld zur Eingabe der Position des Dreiecks, dass die Schnittebene definiert.*
- wxTextCtrl \* [p2zedit](#)  
*Textfeld zur Eingabe der Position des Dreiecks, dass die Schnittebene definiert.*
- wxTextCtrl \* [p3xedit](#)  
*Textfeld zur Eingabe der Position des Dreiecks, dass die Schnittebene definiert.*
- wxTextCtrl \* [p3yedit](#)  
*Textfeld zur Eingabe der Position des Dreiecks, dass die Schnittebene definiert.*
- wxTextCtrl \* [p3zedit](#)  
*Textfeld zur Eingabe der Position des Dreiecks, dass die Schnittebene definiert.*
- wxSpinCtrl \* [imgWidthEdit](#)  
*Textfeld zur Eingabe der Breite des Bereichs, für den die 2D-Temperaturverteilung als Grafik berechnet wird.*
- wxSpinCtrl \* [imgHeightEdit](#)  
*Feld zur Eingabe der Höhe des Bereichs, für den die 2D-Temperaturverteilung als Grafik berechnet wird.*
- wxSpinCtrl \* [threadcountedit](#)  
*Feld zur Eingabe der zum Berechnen zu verwendenden Prozessorkerne.*
- wxTextCtrl \* [mmperpixeledit](#)  
*Feld zur Eingabe des Maßstabs in  $\frac{mm}{px}$ .*
- wxStaticText \* [p1label](#)  
*Beschriftung für den 1.*
- wxStaticText \* [p2label](#)  
*Beschriftung für den 2.*
- wxStaticText \* [p3label](#)  
*Beschriftung für den 3.*
- wxStaticText \* [mmperpixellabel](#)  
*Beschriftung für den Maßstab in  $\frac{mm}{px}$ .*
- wxStaticText \* [trilabel](#)  
*Beschriftung für das die Schnittebene definierende Dreieck.*
- wxStaticText \* [optionslbl](#)  
*Beschriftung für die die 2D-Temperaturverteilung betreffenden Parameter.*
- wxStaticText \* [widthHeightlbl](#)  
*Beschriftung für Breite und Höhe der Grafik.*
- wxStaticText \* [threadcountlbl](#)  
*Beschriftung für die Anzahl bei der Berechnung zu verwendender Prozessorkerne.*
- wxStaticText \* [scalelbl](#)  
*Beschriftung für die die Skala betreffenden Optionen.*
- wxStaticText \* [scalemodelbl](#)  
*Beschriftung für den Darstellungsmodus der Skala.*
- wxComboBox \* [scalemodecb](#)  
*Menübox zur Auswahl des Darstellungsmodus der Skala.*
- wxStaticText \* [scalefontpropslbl](#)  
*Beschriftung für die Schrifteigenschaften der Skala.*
- wxSpinCtrl \* [scalefontsizeedit](#)  
*Feld zur Eingabe der Schriftgröße der Skala.*
- wxButton \* [scalefontcolorbt](#)  
*Button zur Auswahl der Schriftfarbe.*
- wxSpinCtrl \* [scalestepedit](#)  
*Feld zur Eingabe der Schrittweite der Skala.*
- wxButton \* [calcbt](#)

*Button zum Starten der Berechnung der 2D-Temperaturverteilung.*

- wxButton \* [export\\_img\\_bt](#)

*Button zum Export der Grafik.*

- wxButton \* [export\\_csv\\_bt](#)

*Button zum Export der Temperaturverteilung als .csv-Datei.*

- [GUIRenderCutCanvas](#) \* [canvas](#)

*Die Zeichenfläche zur Darstellung der berechneten Grafik und der Skala.*

- wxImage \* [image](#)

*Die berechnete Temperaturverteilung als Grafik.*

- float \* [value\\_img](#)

*Die berechnete Temperaturverteilung als Temperaturwerte.*

- int [core\\_count](#)

*Die Anzahl der zu bei der Berechnung zu verwendender Prozessorkerne.*

### 8.12.1 Ausführliche Beschreibung

Fenster zum erstellen zweidimensionaler Temperaturverteilungen.

Das Fenster ermöglicht es, eine zweidimensionale Temperaturverteilung auf einer Schnittebene durch das dreidimensionale Modell zu berechnen. Diese Schnittebene wird im 3D-Fenster des Hauptfensters visualisiert.

Definiert in Zeile 24 der Datei GUICutRenderWindow.h.

### 8.12.2 Beschreibung der Konstruktoren und Destruktoren

#### 8.12.2.1 GUICutRenderWindow::GUICutRenderWindow ( wxWindow \* *parent*, const wxChar \* *title*, int *xpos*, int *ypos*, int *width*, int *height* )

Der Konstruktor.

Parameter

<i>parent</i>	Das Übergeordnete Fenster. Muss vom Typ <a href="#">GUIMainWindow</a> sein.
<i>title</i>	Titel des Fensters.
<i>xpos</i>	horizontale Position des Fensters.
<i>ypos</i>	vertikale Position des Fensters.
<i>width</i>	Breite des Fensters.
<i>height</i>	Höhe des Fenster

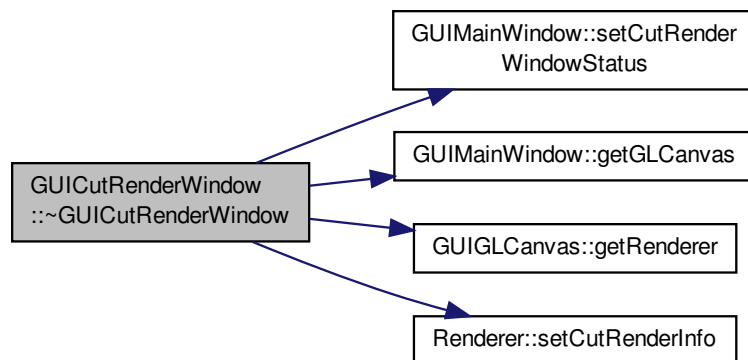
Definiert in Zeile 35 der Datei GUICutRenderWindow.cpp.

#### 8.12.2.2 GUICutRenderWindow::~GUICutRenderWindow ( ) [virtual]

Der Destruktor.

Definiert in Zeile 601 der Datei GUICutRenderWindow.cpp.

Hier ist ein Graph, der zeigt, was diese Funktion aufruft:



### 8.12.3 Dokumentation der Elementfunktionen

#### 8.12.3.1 `GUICutRenderWindow::DECLARE_EVENT_TABLE ( )` `[protected]`

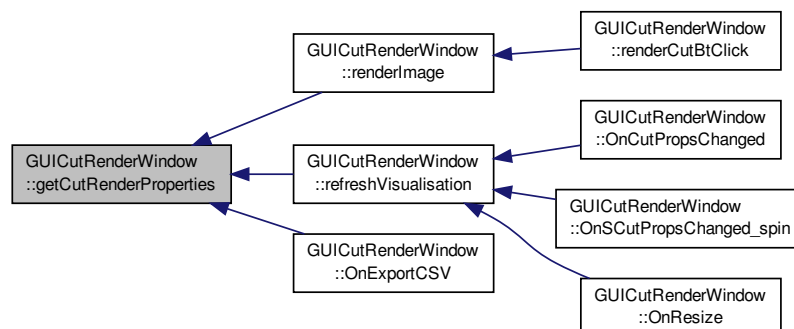
Event-Tabellendeklaration für wxWidgets.

#### 8.12.3.2 `CutRender_info * GUICutRenderWindow::getCutRenderProperties ( )` `[private]`

Gibt die aktuell eingestellten Eigenschaften für die zweidimensionale Temperaturverteilung zurück, damit Sie später an den [Renderer](#) des 3D-Fensters zur Visualisierung übergeben werden können.

Definiert in Zeile 546 der Datei `GUICutRenderWindow.cpp`.

Hier ist ein Graph der zeigt, wo diese Funktion aufgerufen wird:

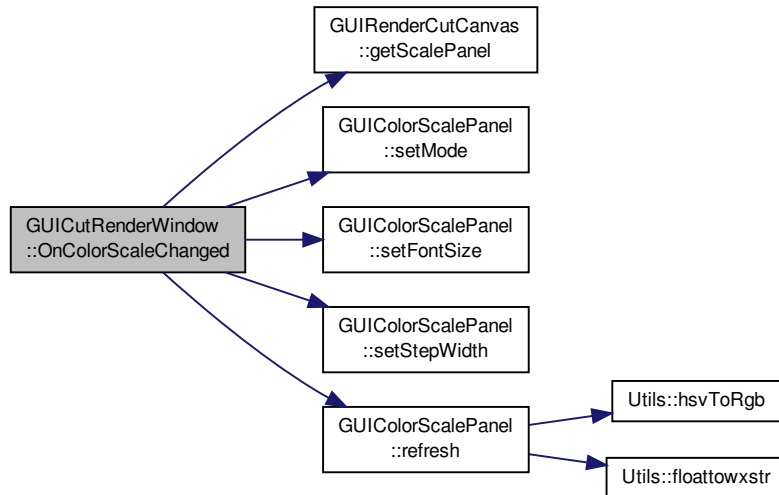


#### 8.12.3.3 `void GUICutRenderWindow::OnColorScaleChanged ( wxCommandEvent & event )` `[private]`

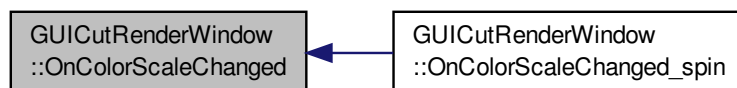
Behandelt das Ändern von Parametern zur Darstellung der Temperaturskala.

Definiert in Zeile 585 der Datei GUICutRenderWindow.cpp.

Hier ist ein Graph, der zeigt, was diese Funktion aufruft:



Hier ist ein Graph der zeigt, wo diese Funktion aufgerufen wird:

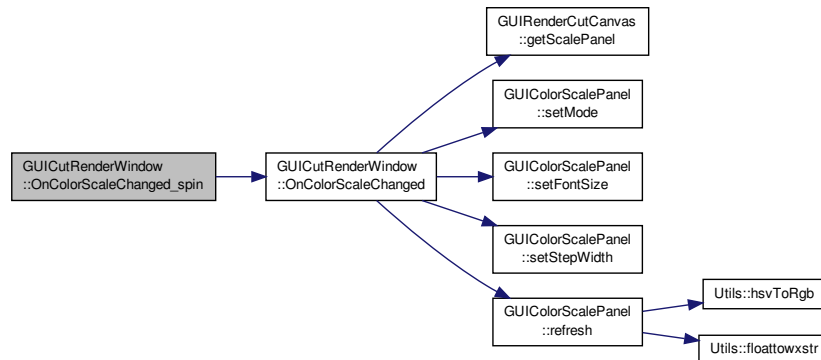


#### 8.12.3.4 void GUICutRenderWindow::OnColorScaleChanged\_spin ( wxSpinEvent & event ) [private]

Behandelt das Ändern von Parametern zur Darstellung der Temperaturskala.

Definiert in Zeile 567 der Datei GUICutRenderWindow.cpp.

Hier ist ein Graph, der zeigt, was diese Funktion aufruft:

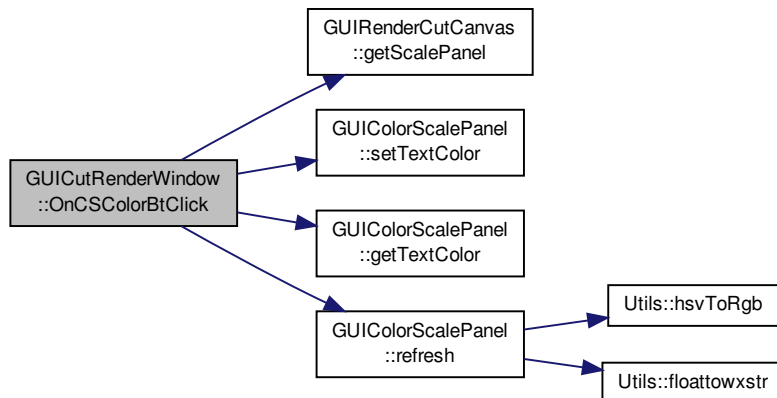


#### 8.12.3.5 void GUICutRenderWindow::OnCSColorBtClick ( wxCommandEvent & event ) [private]

Behandelt das Klicken auf den Button zur Wahl der Schriftfarbe auf der Skala.

Definiert in Zeile 572 der Datei `GUICutRenderWindow.cpp`.

Hier ist ein Graph, der zeigt, was diese Funktion aufruft:

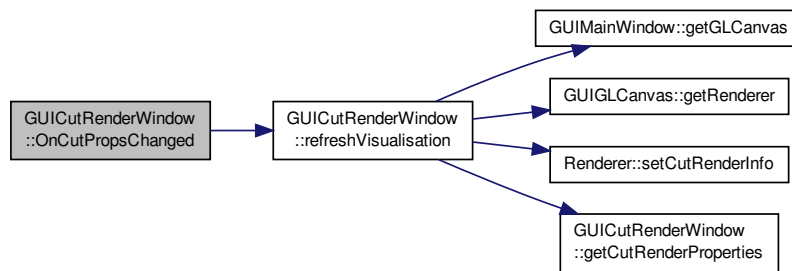


#### 8.12.3.6 void GUICutRenderWindow::OnCutPropsChanged ( wxCommandEvent & event ) [private]

Behandelt das Ändern von Parametern zur Berechnung der 2D-Temperaturverteilung.

Definiert in Zeile 427 der Datei `GUICutRenderWindow.cpp`.

Hier ist ein Graph, der zeigt, was diese Funktion aufruft:

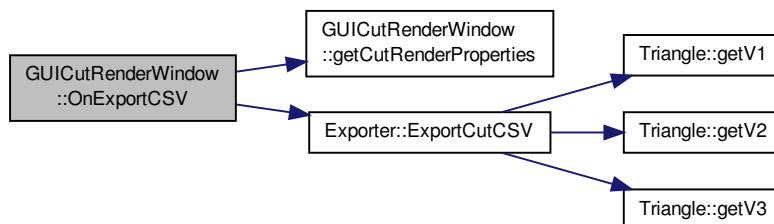


#### 8.12.3.7 void GUICutRenderWindow::OnExportCSV ( wxCommandEvent & event ) [private]

Fragt den Benutzer nach dem Pfad und Exportiert die 2D-Temperaturverteilung als .csv-Datei.

Definiert in Zeile 486 der Datei `GUICutRenderWindow.cpp`.

Hier ist ein Graph, der zeigt, was diese Funktion aufruft:

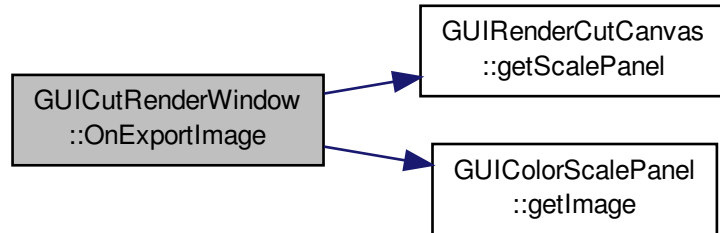


#### 8.12.3.8 void GUICutRenderWindow::OnExportImage ( wxCommandEvent & event ) [private]

Fragt den Benutzer nach dem Pfad und Exportiert eine Grafik aus 2D-Temperaturverteilung und Temperaturskala.

Definiert in Zeile 504 der Datei `GUICutRenderWindow.cpp`.

Hier ist ein Graph, der zeigt, was diese Funktion aufruft:

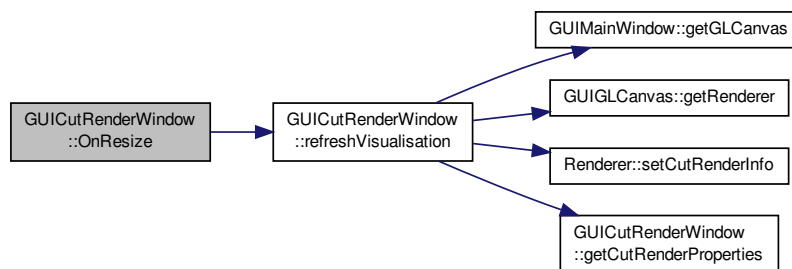


#### 8.12.3.9 void GUICutRenderWindow::OnResize ( wxSizeEvent & event ) [private]

Behandelt Änderungen der Größe des Fensters.

Definiert in Zeile 442 der Datei `GUICutRenderWindow.cpp`.

Hier ist ein Graph, der zeigt, was diese Funktion aufruft:



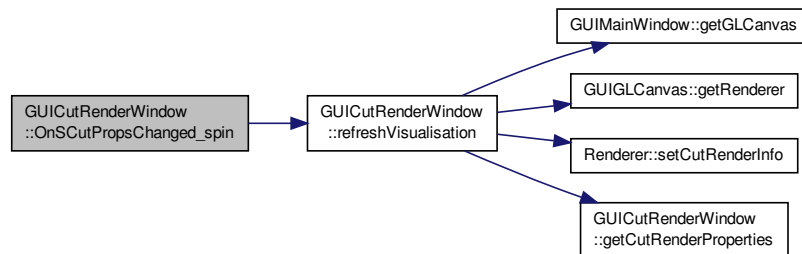
#### 8.12.3.10 void GUICutRenderWindow::OnSCutPropsChanged\_spin ( wxSpinEvent & event ) [private]

Behandelt das Ändern von Parametern zur Berechnung der 2D-Temperaturverteilung.

Definiert in Zeile 431 der Datei `GUICutRenderWindow.cpp`.



Hier ist ein Graph, der zeigt, was diese Funktion aufruft:

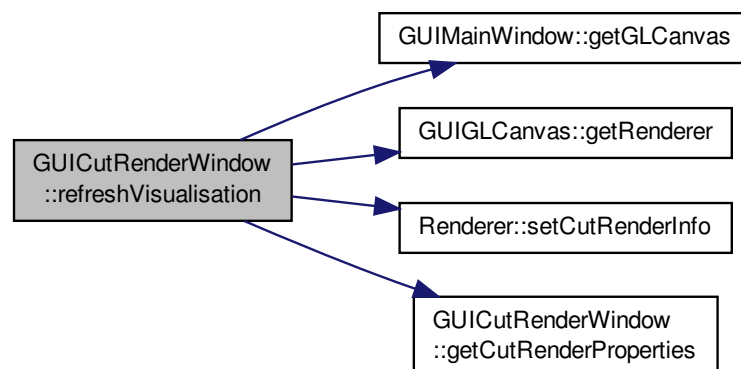


#### 8.12.3.11 void GUICutRenderWindow::refreshVisualisation ( ) [private]

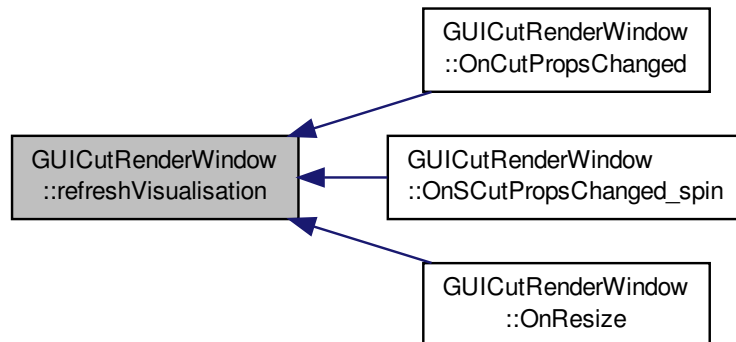
Aktualisiert die Visualisierung der Schnittebene im Hauptfenster.

Definiert in Zeile 435 der Datei `GUICutRenderWindow.cpp`.

Hier ist ein Graph, der zeigt, was diese Funktion aufruft:



Hier ist ein Graph der zeigt, wo diese Funktion aufgerufen wird:

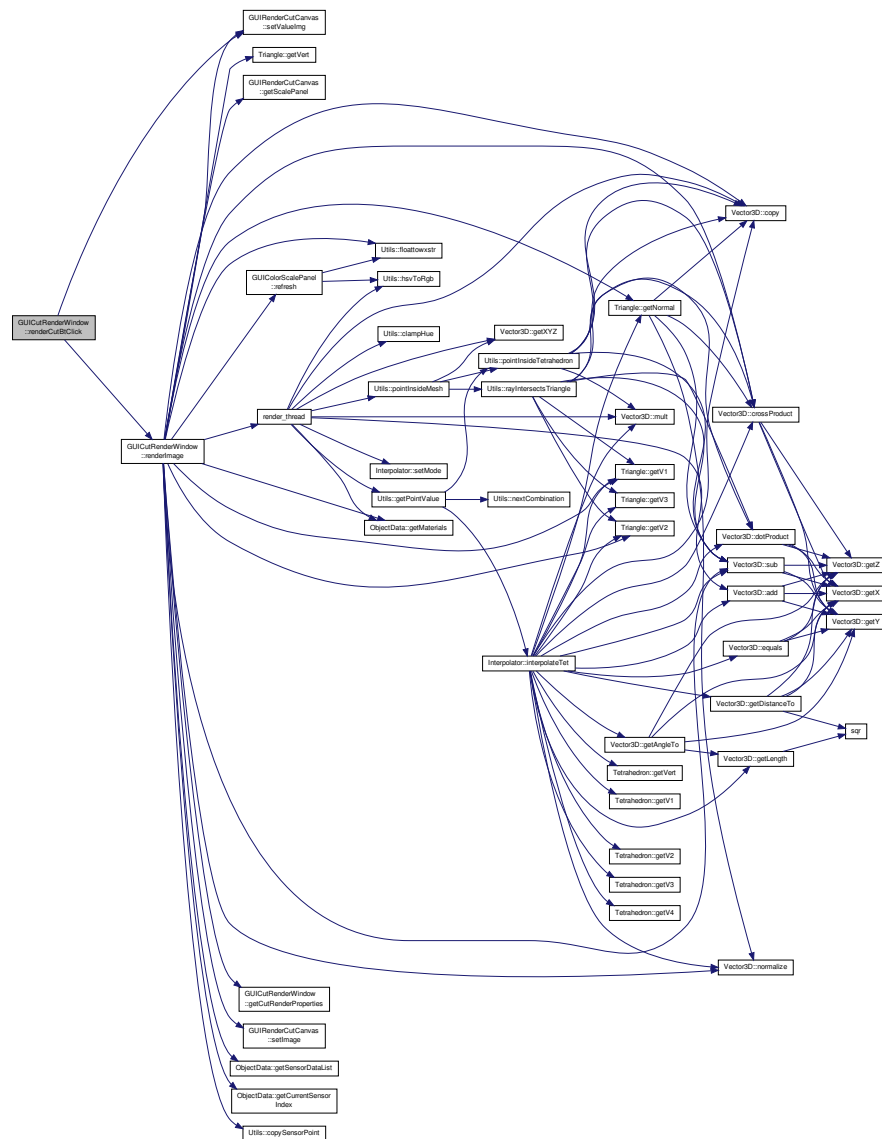


8.12.3.12 `void GUICutRenderWindow::renderCutBtClick ( wxCommandEvent & event )` [private]

Behandelt das Drücken des Buttons zur Berechnung der zweidimensionalen Temperaturverteilung.

Definiert in Zeile 595 der Datei `GUICutRenderWindow.cpp`.

Hier ist ein Graph, der zeigt, was diese Funktion aufruft:



### 8.12.3.13 void GUICutRenderWindow::renderImage ( wxImage \* image ) [private]

Berechnet die 2D-Temperaturverteilung als Grafik.

Definiert in Zeile 264 der Datei GUICutRenderWindow.cpp.



Definiert in Zeile 256 der Datei GUICutRenderWindow.h.

#### 8.12.4.2 `GUIRenderCutCanvas*` `GUICutRenderWindow::canvas` `[private]`

Die Zeichenfläche zur Darstellung der berechneten Grafik und der Skala.

Definiert in Zeile 271 der Datei GUICutRenderWindow.h.

#### 8.12.4.3 `int` `GUICutRenderWindow::core_count` `[private]`

Die Anzahl der zu bei der Berechnung zu verwendender Prozessorkerne.

Definiert in Zeile 286 der Datei GUICutRenderWindow.h.

#### 8.12.4.4 `wxButton*` `GUICutRenderWindow::export_csv_bt` `[private]`

Button zum Export der Temperaturverteilung als .csv-Datei.

Definiert in Zeile 266 der Datei GUICutRenderWindow.h.

#### 8.12.4.5 `wxButton*` `GUICutRenderWindow::export_img_bt` `[private]`

Button zum Export der Grafik.

Definiert in Zeile 261 der Datei GUICutRenderWindow.h.

#### 8.12.4.6 `wxImage*` `GUICutRenderWindow::image` `[private]`

Die berechnete Temperaturverteilung als Grafik.

Definiert in Zeile 276 der Datei GUICutRenderWindow.h.

#### 8.12.4.7 `wxSpinCtrl*` `GUICutRenderWindow::imgHeightEdit` `[private]`

Feld zur Eingabe der Höhe des Bereichs, für den die 2D-Temperaturverteilung als Grafik berechnet wird.

Definiert in Zeile 166 der Datei GUICutRenderWindow.h.

#### 8.12.4.8 `wxSpinCtrl*` `GUICutRenderWindow::imgWidthEdit` `[private]`

Textfeld zur Eingabe der Breite des Bereichs, für den die 2D-Temperaturverteilung als Grafik berechnet wird.

Definiert in Zeile 161 der Datei GUICutRenderWindow.h.

#### 8.12.4.9 `wxTextCtrl*` `GUICutRenderWindow::mmperpixedit` `[private]`

Feld zur Eingabe des Maßstabs in  $\frac{mm}{px}$ .

Definiert in Zeile 176 der Datei GUICutRenderWindow.h.

#### 8.12.4.10 `wxStaticText*` `GUICutRenderWindow::mmperpixellabel` `[private]`

Beschriftung für den Maßstab in  $\frac{mm}{px}$ .

Definiert in Zeile 196 der Datei GUICutRenderWindow.h.

**8.12.4.11 wxStaticText\* GUICutRenderWindow::optionslbl [private]**

Beschriftung für die die 2D-Temperaturverteilung betreffenden Parameter.

Definiert in Zeile 206 der Datei GUICutRenderWindow.h.

**8.12.4.12 wxStaticText\* GUICutRenderWindow::p1label [private]**

Beschriftung für den 1.

die Schnittebene definierenden Punkt.

Definiert in Zeile 181 der Datei GUICutRenderWindow.h.

**8.12.4.13 wxTextCtrl\* GUICutRenderWindow::p1xedit [private]**

Textfeld zur Eingabe der Position des Dreiecks, dass die Schnittebene definiert.

Definiert in Zeile 116 der Datei GUICutRenderWindow.h.

**8.12.4.14 wxTextCtrl\* GUICutRenderWindow::p1yedit [private]**

Textfeld zur Eingabe der Position des Dreiecks, dass die Schnittebene definiert.

Definiert in Zeile 121 der Datei GUICutRenderWindow.h.

**8.12.4.15 wxTextCtrl\* GUICutRenderWindow::p1zedit [private]**

Textfeld zur Eingabe der Position des Dreiecks, dass die Schnittebene definiert.

Definiert in Zeile 126 der Datei GUICutRenderWindow.h.

**8.12.4.16 wxStaticText\* GUICutRenderWindow::p2label [private]**

Beschriftung für den 2.

die Schnittebene definierenden Punkt.

Definiert in Zeile 186 der Datei GUICutRenderWindow.h.

**8.12.4.17 wxTextCtrl\* GUICutRenderWindow::p2xedit [private]**

Textfeld zur Eingabe der Position des Dreiecks, dass die Schnittebene definiert.

Definiert in Zeile 131 der Datei GUICutRenderWindow.h.

**8.12.4.18 wxTextCtrl\* GUICutRenderWindow::p2yedit [private]**

Textfeld zur Eingabe der Position des Dreiecks, dass die Schnittebene definiert.

Definiert in Zeile 136 der Datei GUICutRenderWindow.h.

**8.12.4.19 wxTextCtrl\* GUICutRenderWindow::p2zedit [private]**

Textfeld zur Eingabe der Position des Dreiecks, dass die Schnittebene definiert.

Definiert in Zeile 141 der Datei GUICutRenderWindow.h.

**8.12.4.20 wxStaticText\* GUICutRenderWindow::p3label [private]**

Beschriftung für den 3.

die Schnittebene definierenden Punkt.

Definiert in Zeile 191 der Datei GUICutRenderWindow.h.

**8.12.4.21 wxTextCtrl\* GUICutRenderWindow::p3xedit [private]**

Textfeld zur Eingabe der Position des Dreiecks, dass die Schnittebene definiert.

Definiert in Zeile 146 der Datei GUICutRenderWindow.h.

**8.12.4.22 wxTextCtrl\* GUICutRenderWindow::p3yedit [private]**

Textfeld zur Eingabe der Position des Dreiecks, dass die Schnittebene definiert.

Definiert in Zeile 151 der Datei GUICutRenderWindow.h.

**8.12.4.23 wxTextCtrl\* GUICutRenderWindow::p3zedit [private]**

Textfeld zur Eingabe der Position des Dreiecks, dass die Schnittebene definiert.

Definiert in Zeile 156 der Datei GUICutRenderWindow.h.

**8.12.4.24 wxButton\* GUICutRenderWindow::scalefontcolorbt [private]**

Button zur Auswahl der Schriftfarbe.

Definiert in Zeile 246 der Datei GUICutRenderWindow.h.

**8.12.4.25 wxStaticText\* GUICutRenderWindow::scalefontpropslbl [private]**

Beschriftung für die Schrifteigenschaften der Skala.

Definiert in Zeile 236 der Datei GUICutRenderWindow.h.

**8.12.4.26 wxSpinCtrl\* GUICutRenderWindow::scalefontsizeedit [private]**

Feld zur Eingabe der Schriftgröße der Skala.

Definiert in Zeile 241 der Datei GUICutRenderWindow.h.

**8.12.4.27 wxStaticText\* GUICutRenderWindow::scalelbl [private]**

Beschriftung für die die Skala betreffenden Optionen.

Definiert in Zeile 221 der Datei GUICutRenderWindow.h.

**8.12.4.28 wxComboBox\* GUICutRenderWindow::scalemodecb [private]**

Menübox zur Auswahl des Darstellungsmodus der Skala.

Definiert in Zeile 231 der Datei GUICutRenderWindow.h.

**8.12.4.29 wxStaticText\* GUICutRenderWindow::scalemodelbl** [private]

Beschriftung für den Darstellungsmodus der Skala.

Definiert in Zeile 226 der Datei GUICutRenderWindow.h.

**8.12.4.30 wxSpinCtrl\* GUICutRenderWindow::scalestepedit** [private]

Feld zur Eingabe der Schrittweite der Skala.

Definiert in Zeile 251 der Datei GUICutRenderWindow.h.

**8.12.4.31 wxScrolledWindow\* GUICutRenderWindow::scroll\_pane** [private]

Scrollender Bereich, in den die anderen Komponenten außer der Zeichenfläche (canvas) eingebettet sind.

Definiert in Zeile 111 der Datei GUICutRenderWindow.h.

**8.12.4.32 wxSpinCtrl\* GUICutRenderWindow::threadcountedit** [private]

Feld zur Eingabe der zum Berechnen zu verwendenden Prozessorkerne.

Definiert in Zeile 171 der Datei GUICutRenderWindow.h.

**8.12.4.33 wxStaticText\* GUICutRenderWindow::threadcountlbl** [private]

Beschriftung für die Anzahl bei der Berechnung zu verwendender Prozessorkerne.

Definiert in Zeile 216 der Datei GUICutRenderWindow.h.

**8.12.4.34 wxStaticText\* GUICutRenderWindow::trilabel** [private]

Beschriftung für das die Schnittebene definierende Dreieck.

Definiert in Zeile 201 der Datei GUICutRenderWindow.h.

**8.12.4.35 float\* GUICutRenderWindow::value\_img** [private]

Die berechnete Temperaturverteilung als Temperaturwerte.

Definiert in Zeile 281 der Datei GUICutRenderWindow.h.

**8.12.4.36 wxStaticText\* GUICutRenderWindow::widthHeightlbl** [private]

Beschriftung für Breite und Höhe der Grafik.

Definiert in Zeile 211 der Datei GUICutRenderWindow.h.

Die Dokumentation für diese Klasse wurde erzeugt aufgrund der Dateien:

- [/daten/Projekte/eclipse\\_workspace/simpleanalyzer-gui/src/GUI/GUICutRenderWindow.h](#)
- [/daten/Projekte/eclipse\\_workspace/simpleanalyzer-gui/src/GUI/GUICutRenderWindow.cpp](#)

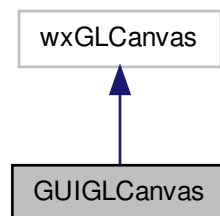


## 8.13 GUIGLCanvas Klassenreferenz

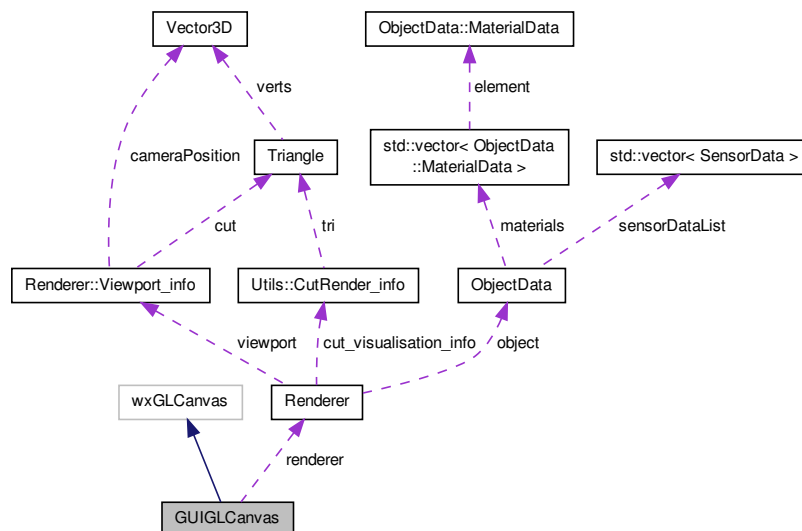
Zeichenfläche für das 3D-Fenster.

```
#include <GUIGLCanvas.h>
```

Klassendiagramm für GUIGLCanvas:



Zusammengehörigkeiten von GUIGLCanvas:



### Öffentliche Methoden

- **GUIGLCanvas** (wxFrame \*parent)  
*Der Konstruktor.*
- void **setRenderObject** (ObjectData \*obj)  
*Setzt das darzustellende Objekt.*
- **Renderer** \* **getRenderer** ()  
*Gibt den **Renderer** der Zeichenfläche zurück.*
- void **refresh** ()

- Zeichnet den Inhalt des 3D-Fensters neu und aktualisiert den [Renderer](#), z.B.*
- virtual [~GUIGLCanvas](#) ()  
*Der Destruktor.*

### Private Methoden

- void [OnPaint](#) (wxPaintEvent &event)  
*Event-Tabellendeklaration für wxWidgets.*
- void [OnMouseWheel](#) (wxMouseEvent &event)  
*Behandelt Mausradbewegungen (zoomen).*
- void [OnMouseMove](#) (wxMouseEvent &event)  
*Behandelt Mausbewegungen (verschieben und drehen der Ansicht).*
- void [OnResize](#) (wxSizeEvent &event)  
*Behandelt Größenänderungen der Zeichenfläche.*

### Private Attribute

- [Renderer](#) [renderer](#)  
*Der verwendete [Renderer](#).*
- bool [is\\_initialized](#)  
*Initialisierungsstatus des Objekts.*
- bool [do\\_refresh](#)  
*Statusvariable, gibt an ob beim Zeichnen auch der [Renderer](#) aktualisiert wird.*
- int [prev\\_mouse\\_x](#)  
*Zwischenspeicher für die vorherige Mausposition (X).*
- int [prev\\_mouse\\_y](#)  
*Zwischenspeicher für die vorherige Mausposition (Y).*

## 8.13.1 Ausführliche Beschreibung

Zeichenfläche für das 3D-Fenster.

Klasse zum Verwalten der im 3D-Fenster angezeigten Inhalte. Auch zuständig für Drehen, Verschieben und Zoomen der Ansicht.

Definiert in Zeile 22 der Datei GUIGLCanvas.h.

## 8.13.2 Beschreibung der Konstruktoren und Destruktoren

### 8.13.2.1 GUIGLCanvas::GUIGLCanvas ( wxFrame \* parent )

Der Konstruktor.

Parameter

<i>parent</i>	Das Fenster, auf dem sich die Zeichenfläche befindet.
---------------	---

Definiert in Zeile 22 der Datei GUIGLCanvas.cpp.

### 8.13.2.2 GUIGLCanvas::~GUIGLCanvas ( ) [virtual]

Der Destruktor.

Definiert in Zeile 150 der Datei GUIGLCanvas.cpp.

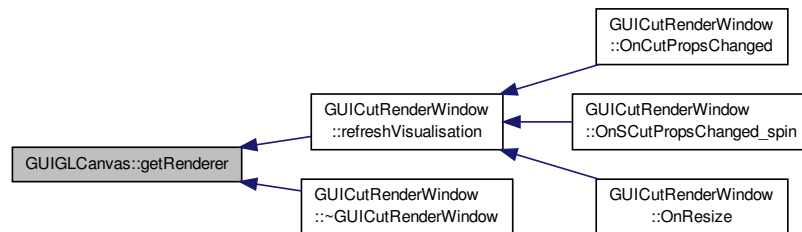
### 8.13.3 Dokumentation der Elementfunktionen

#### 8.13.3.1 `Renderer * GUIGLCanvas::getRenderer ( )`

Gibt den [Renderer](#) der Zeichenfläche zurück.

Definiert in Zeile 101 der Datei GUIGLCanvas.cpp.

Hier ist ein Graph der zeigt, wo diese Funktion aufgerufen wird:

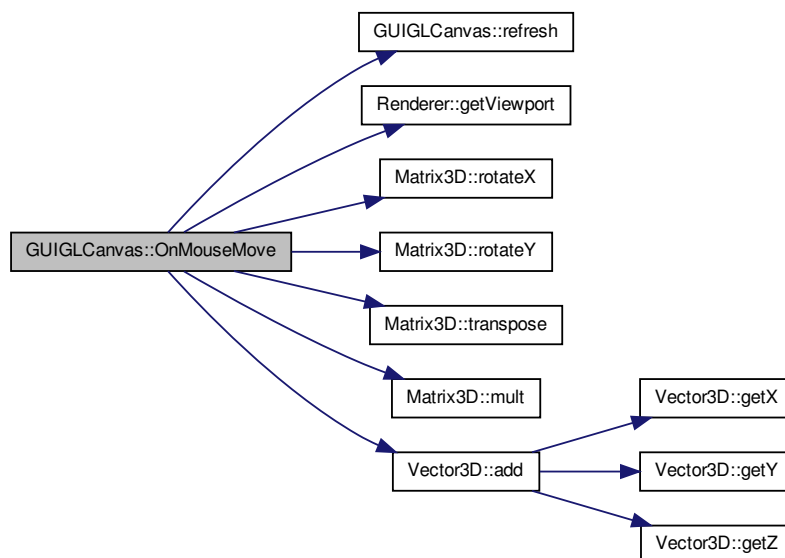


#### 8.13.3.2 `void GUIGLCanvas::OnMouseMove ( wxMouseEvent & event ) [private]`

Behandelt Mausbewegungen (verschieben und drehen der Ansicht).

Definiert in Zeile 105 der Datei GUIGLCanvas.cpp.

Hier ist ein Graph, der zeigt, was diese Funktion aufruft:

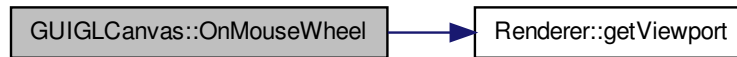


### 8.13.3.3 void GUIGLCanvas::OnMouseWheel ( wxMouseEvent & event ) [private]

Behandelt Mausradbewegungen (zoomen).

Definiert in Zeile 43 der Datei GUIGLCanvas.cpp.

Hier ist ein Graph, der zeigt, was diese Funktion aufruft:



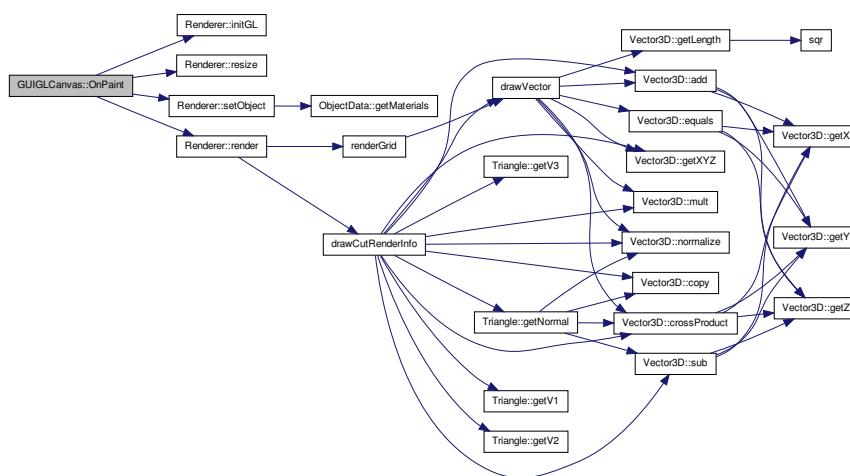
### 8.13.3.4 void GUIGLCanvas::OnPaint ( wxPaintEvent & event ) [private]

Event-Tabellendeklaration für wxWidgets.

Behandelt das Zeichenevent und zeichnet die Inhalte des 3D-Fensters.

Definiert in Zeile 56 der Datei GUIGLCanvas.cpp.

Hier ist ein Graph, der zeigt, was diese Funktion aufruft:



### 8.13.3.5 void GUIGLCanvas::OnResize ( wxSizeEvent & event ) [private]

Behandelt Größenänderungen der Zeichenfläche.

Definiert in Zeile 34 der Datei GUIGLCanvas.cpp.

### 8.13.3.6 void GUIGLCanvas::refresh ( )

Zeichnet den Inhalt des 3D-Fensters neu und aktualisiert den [Renderer](#), z.B.

bei geänderter Fenstergröße oder geänderten Eigenschaften des angezeigten Objekts.

Definiert in Zeile 95 der Datei GUIGLCanvas.cpp.

Hier ist ein Graph der zeigt, wo diese Funktion aufgerufen wird:

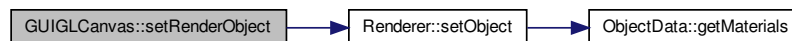


#### 8.13.3.7 void GUIGLCanvas::setRenderObject ( ObjectData \* obj )

Setzt das darzustellende Objekt.

Definiert in Zeile 83 der Datei GUIGLCanvas.cpp.

Hier ist ein Graph, der zeigt, was diese Funktion aufruft:



### 8.13.4 Dokumentation der Datenelemente

#### 8.13.4.1 bool GUIGLCanvas::do\_refresh [private]

Statusvariable, gibt an ob beim Zeichnen auch der [Renderer](#) aktualisiert wird.

Dies tritt beispielsweise bei Größenänderungen oder Änderungen am Objekt ein, da die Daten teilweise neu an den [Renderer](#) übermittelt werden müssen.

Definiert in Zeile 91 der Datei GUIGLCanvas.h.

#### 8.13.4.2 bool GUIGLCanvas::is\_initialized [private]

Initialisierungsstatus des Objekts.

Definiert in Zeile 84 der Datei GUIGLCanvas.h.

#### 8.13.4.3 int GUIGLCanvas::prev\_mouse\_x [private]

Zwischenspeicher für die vorherige Mausposition (X).

Definiert in Zeile 96 der Datei GUIGLCanvas.h.

#### 8.13.4.4 int GUIGLCanvas::prev\_mouse\_y [private]

Zwischenspeicher für die vorherige Mausposition (Y).

Definiert in Zeile 101 der Datei GUIGLCanvas.h.

#### 8.13.4.5 Renderer GUIGLCanvas::render() [private]

Der verwendete [Renderer](#).

Definiert in Zeile 79 der Datei GUIGLCanvas.h.

Die Dokumentation für diese Klasse wurde erzeugt aufgrund der Dateien:

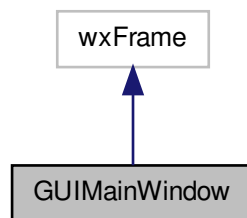
- /daten/Projekte/eclipse\_workspace/simpleanalyzer-gui/src/GUI/GLCanvas.h
- /daten/Projekte/eclipse\_workspace/simpleanalyzer-gui/src/GUI/GLCanvas.cpp

## 8.14 GUIMainWindow Klassenreferenz

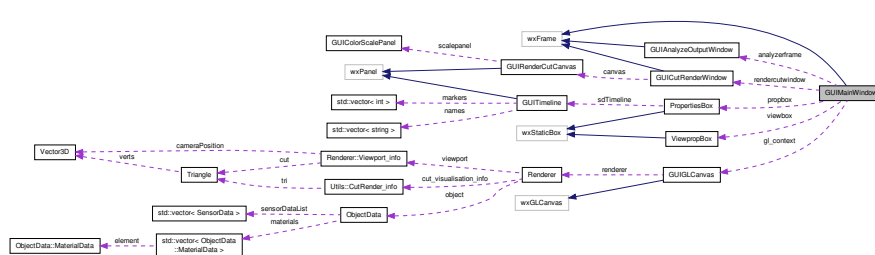
Hauptfenster mit Hauptmenü und Zugriff auf die einzelnen Programmfunktionen.

```
#include <GUIMainWindow.h>
```

Klassendiagramm für GUIMainWindow:



Zusammengehörigkeiten von GUIMainWindow:



## Öffentliche Methoden

- [GUIMainWindow](#) (const wxChar \*title, int xpos, int ypos, int width, int height)  
*Der Konstruktor.*
- void [setAnalyzeWindowStatus](#) (bool isValid)  
*Setzt den Status des Übersichtsfensters über die Analysedaten.*
- void [setCutRenderWindowStatus](#) (bool isValid)  
*Setzt den Status des Übersichtsfensters über die Analysedaten.*
- [GUIGLCanvas](#) \* [getGLCanvas](#) ()

Gibt die Zeichenfläche des 3D-Fensters zurück.

- virtual [~GUIMainWindow](#) ()

Der Destruktor.

## Geschützte Attribute

- string [configpaths](#) [[NUMBEROFPATHS](#)]

Suchpfade für die Anwendungsdaten.

## Statische, geschützte Attribute

- static const int [NUMBEROFPATHS](#) = 2

Anzahl der Suchpfade für die Anwendungsdaten (z.B.

## Private Methoden

- void [OnMenuImportObj](#) (wxCommandEvent &event)  
*Event-Tabellendeklaration für wxWidgets.*
- void [OnMenuImportSD](#) (wxCommandEvent &event)  
*Öffnet den Dialog zum Importieren einfacher Sensordaten.*
- void [OnMenuImportTSD](#) (wxCommandEvent &event)  
*Öffnet den Dialog zum Importieren zeitbezogener Sensordaten.*
- void [OnMenuFileQuit](#) (wxCommandEvent &event)  
*Beendet das Programm.*
- void [OnMenuHelpAbout](#) (wxCommandEvent &event)  
*Öffnet ein Fenster mit Informationen über das Programm.*
- void [OnMenuOpenManual](#) (wxCommandEvent &event)  
*Öffnet das Handbuch mit dem PDF-Viewer des Systems.*
- void [OnRecalcBtClick](#) (wxCommandEvent &event)  
*Berechnet die 3D-Temperaturverteilung neu.*
- void [OnResize](#) (wxSizeEvent &event)  
*Behandelt Größenänderungen des Fensters.*
- void [OnMaterialSelect](#) (wxCommandEvent &event)  
*Aktualisiert die Oberfläche nach dem Auswählen eines anderen Materials im Objekteigenschaften-Fenster.*
- void [OnAnalyze](#) (wxCommandEvent &event)  
*Öffnet das Analysedaten-Übersichtsfenster.*
- void [OnImmediateUpdatePropChange](#) (wxCommandEvent &event)  
*Behandelt das Aktualisieren der Oberfläche nach einer Änderung an Objekteigenschaften, bei denen ein sofortiges Update der Oberfläche möglich ist, durch den Nutzer.*
- void [OnGeneralPropChange](#) (wxCommandEvent &event)  
*Behandelt das Aktualisieren der Oberfläche nach einer Änderung an Objekteigenschaften durch den Nutzer.*
- void [OnViewPropChange](#) (wxCommandEvent &event)  
*Behandelt das Aktualisieren der Oberfläche nach einer Änderung an Visualisierungsoptionen durch den Nutzer.*
- void [OnViewPropSpinChange](#) (wxSpinEvent &event)  
*Behandelt das Aktualisieren der Oberfläche nach einer Änderung an Visualisierungsoptionen durch den Nutzer.*
- void [OnSensorDataChange](#) (wxCommandEvent &event)  
*Behandelt das Auswählen eines anderen Sensordatensatzes.*
- void [OnSDTimelineChange](#) (wxCommandEvent &event)  
*Behandelt Änderungen an der Sensordaten-Zeitleiste (bei zeitbezogenen Datensätzen).*
- void [OnSDTLMarkerClear](#) (wxCommandEvent &event)

- Löscht alle Markierungen auf der Sensordaten-Zeitleiste (bei zeitbezogenen Datensätzen).*
- void [OnSDTLNextMarker](#) (wxCommandEvent &event)  
*Setzen des auf der Sensordaten-Zeitleiste ausgewählten Zeitpunktes auf den nächsten markierten Zeitpunkt.*
- void [OnSDTLPrevMarker](#) (wxCommandEvent &event)  
*Setzen des auf der Sensordaten-Zeitleiste ausgewählten Zeitpunktes auf den vorherigen markierten Zeitpunkt.*
- void [OnAnalyzeMarkerChange](#) (wxCommandEvent &event)  
*Behandelt das Markieren eines Zeitpunktes auf der Sensordaten-Zeitleiste.*
- void [OnActiveObjectChangePopup](#) (wxCommandEvent &event)  
*Setzt das aktive Objekt nach dem Auswählen im Popup-Menü.*
- void [OnActiveObjectChange](#) (wxCommandEvent &event)  
*Öffnet das Popup-Menü zum auswählen des aktiven Objekts.*
- void [OnActiveObjectDelete](#) (wxCommandEvent &event)  
*Löscht das aktive Objekt, sofern es nicht das einzige geladene Objekt ist.*
- void [OnAnalyzePoint](#) (wxCommandEvent &event)  
*Öffnet das Fenster zur Analyse eines Punktes ([GUIAnalyzePointWindow](#)).*
- void [OnRenderCut](#) (wxCommandEvent &event)  
*Öffnet das Fenster zur Berechnung einer zweidimensionalen Temperaturverteilung.*
- void [addObject](#) ([ObjectData](#) \*obj)  
*Registriert ein neues (Versuchs-) Objekt im Programm.*
- void [setActiveObject](#) (int index)  
*Setzt das aktive Objekt.*
- void [OnExportViewportImage](#) (wxCommandEvent &event)  
*Öffnet ein Fenster zum Exportieren der Ansicht des 3D-Fensters.*
- void [OnExportVTK](#) (wxCommandEvent &event)  
*Öffnet ein Fenster zum Exportieren der Temperaturverteilung und des Objekts im VTK-Format.*
- void [OnFindMaxTSD](#) (wxCommandEvent &event)  
*Sucht den Zeitpunkt zwischen zwei markierten Stellen auf der Sensordaten-Zeitleiste, für den der Wärmeenergiegehalt maximal wird.*
- void [OnAutoUpdateChange](#) (wxCommandEvent &event)  
*Behandelt das aktivieren/deaktivieren der Option zum automatischen Neuberechnen der Temperaturverteilung eines Objekts, sobald Änderungen an dessen Eigenschaften vorgenommen werden.*
- void [assignCurrentObjectProps](#) ()  
*Überträgt die in der GUI eingetragenen Objekteigenschaften in das aktive Objekt.*
- void [updateObjectPropGUI](#) ()  
*Überträgt die Eigenschaften des aktiven Objekts in die GUI.*
- void [assignViewProps](#) ()  
*Speichert die Visualisierungsoptionen aus der GUI.*
- void [updateViewPropGUI](#) ()  
*Lädt die Visualisierungsoptionen in die GUI.*

## Private Attribute

- [GUIGLCanvas](#) \* [gl\\_context](#)  
*Die Zeichenfläche für das 3D-Fenster.*
- wxToolBar \* [toolbar](#)  
*Die Tollbarkomponente.*
- wxMenuBar \* [mwMenuBar](#)  
*Die Hauptmenükomponente.*
- wxMenu \* [mwFileMenu](#)  
*Das "Datei"-Untermenü.*
- wxMenu \* [mwHelpMenu](#)



- Das "Hilfe"-Untermenü.*
- wxMenu \* [mwImportMenu](#)
- Das "Import"-Untermenü.*
- wxMenu \* [mwExportMenu](#)
- Das "Export"-Untermenü.*
- wxMenu \* [mwAnalyzeMenu](#)
- Das "Analysieren"-Untermenü.*
- wxMenu \* [mwEditMenu](#)
- Das "Bearbeiten"-Untermenü*
- [PropertiesBox](#) \* [propbox](#)
- Die Unterkomponente, die die Objekteigenschaften-Oberfläche enthält.*
- [ViewpropBox](#) \* [viewbox](#)
- Die Unterkomponente, die die Visualisierungsoptionen-Oberfläche enthält.*
- [GUIAnalyzeOutputWindow](#) \* [analyzerframe](#)
- Das Analysedaten-Übersichtsfenster.*
- [GUICutRenderWindow](#) \* [rendercutwindow](#)
- Das Fenster zur Berechnung einer zweidimensionalen Temperaturverteilung.*
- wxScrolledWindow \* [prop\\_scroll\\_win](#)
- Scrollender Bereich, in den die Objekteigenschaften-Oberfläche eingebettet ist.*
- wxScrolledWindow \* [view\\_scroll\\_win](#)
- Scrollender Bereich, in den die Visualisierungsoptionen-Oberfläche eingebettet ist.*
- bool [updating](#)
- Die Oberfläche wird gerade vom Programm verändert.*
- bool [analyze\\_window\\_valid](#)
- Das Analysedaten-Übersichtsfenster ist gerade geöffnet.*
- bool [render\\_cut\\_window\\_valid](#)
- Das Fenster zur Berechnung einer zweidimensionalen Temperaturverteilung ist gerade geöffnet.*
- string [data\\_directory](#)
- Der Pfad zum Verzeichnis, das die von der Anwendung verwendeten Daten (z.B.*

### 8.14.1 Ausführliche Beschreibung

Hauptfenster mit Hauptmenü und Zugriff auf die einzelnen Programmfunktionen.

Das Hauptfenster bietet über das Hauptmenü und die Oberfläche Zugriff auf die Funktionen des Programms. Dazu kann das aktuelle Objekt gewählt werden, welches dann im eingebetteten 3D-Fenster angezeigt wird. Eigenschaften der Visualisierung und des Objekts können ebenfalls über die Oberfläche des Hauptfensters festgelegt werden.

Definiert in Zeile 28 der Datei GUIMainWindow.h.

### 8.14.2 Beschreibung der Konstruktoren und Destruktoren

#### 8.14.2.1 GUIMainWindow::GUIMainWindow ( const wxChar \* title, int xpos, int ypos, int width, int height )

Der Konstruktor.

Parameter

<i>title</i>	Der Titel des Programmfensters.
--------------	---------------------------------

<i>xpos</i>	horizontale Position des Fensters.
<i>ypos</i>	vertikale Position des Fensters.
<i>width</i>	Breite des Fensters.
<i>height</i>	Höhe des Fensters.

Erstellen und initialisieren der Fensterkomponenten

Definiert in Zeile 75 der Datei GUIMainWindow.cpp.

#### 8.14.2.2 GUIMainWindow::~~GUIMainWindow ( ) [virtual]

Der Destruktor.

Definiert in Zeile 984 der Datei GUIMainWindow.cpp.

### 8.14.3 Dokumentation der Elementfunktionen

#### 8.14.3.1 void GUIMainWindow::addObject ( ObjectData \* obj ) [private]

Registriert ein neues (Versuchs-) Objekt im Programm.

Parameter

<i>obj</i>	Das zu registrierende Objekt.
------------	-------------------------------

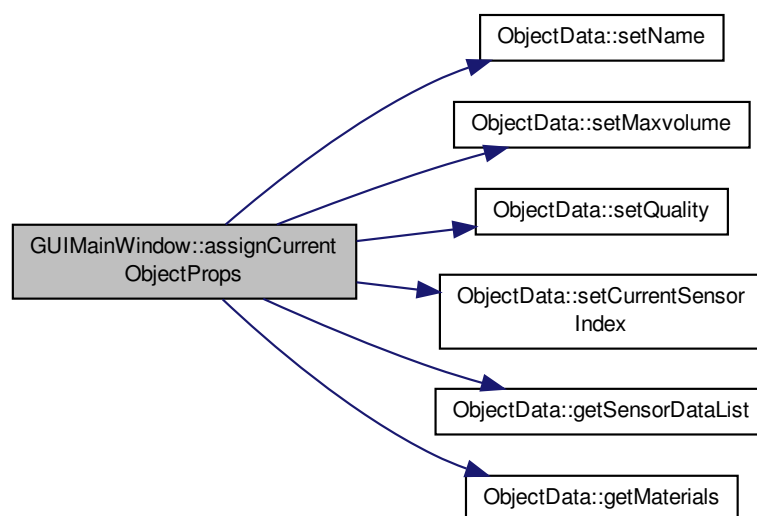
Definiert in Zeile 290 der Datei GUIMainWindow.cpp.

#### 8.14.3.2 void GUIMainWindow::assignCurrentObjectProps ( ) [private]

Überträgt die in der GUI eingetragenen Objekteigenschaften in das aktive Objekt.

Definiert in Zeile 482 der Datei GUIMainWindow.cpp.

Hier ist ein Graph, der zeigt, was diese Funktion aufruft:



#### 8.14.3.3 void GUIMainWindow::assignViewProps ( ) [private]

Speichert die Visualisierungsoptionen aus der GUI.

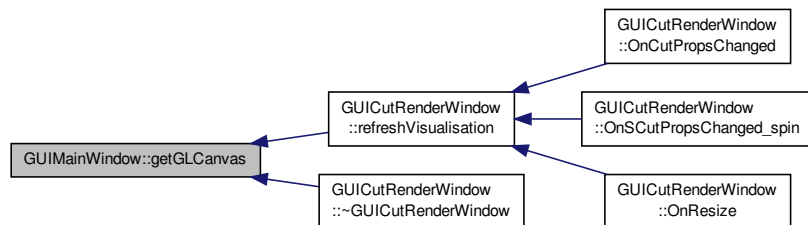
Definiert in Zeile 688 der Datei GUIMainWindow.cpp.

#### 8.14.3.4 GUIGLCanvas \* GUIMainWindow::getGLCanvas ( )

Gibt die Zeichenfläche des 3D-Fensters zurück.

Definiert in Zeile 286 der Datei GUIMainWindow.cpp.

Hier ist ein Graph der zeigt, wo diese Funktion aufgerufen wird:



#### 8.14.3.5 void GUIMainWindow::OnActiveObjectChange ( wxCommandEvent & event ) [private]

Öffnet das Popup-Menü zum auswählen des aktiven Objekts.

Definiert in Zeile 936 der Datei GUIMainWindow.cpp.

Hier ist ein Graph, der zeigt, was diese Funktion aufruft:

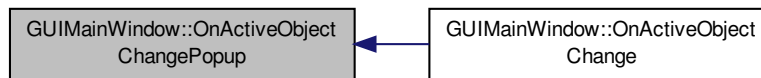


#### 8.14.3.6 void GUIMainWindow::OnActiveObjectChangePopup ( wxCommandEvent & event ) [private]

Setzt das aktive Objekt nach dem Auswählen im Popup-Menü.

Definiert in Zeile 929 der Datei GUIMainWindow.cpp.

Hier ist ein Graph der zeigt, wo diese Funktion aufgerufen wird:



**8.14.3.7** `void GUIMainWindow::OnActiveObjectDelete ( wxCommandEvent & event ) [private]`

Löscht das aktive Objekt, sofern es nicht das einzige geladene Objekt ist.

Definiert in Zeile 308 der Datei GUIMainWindow.cpp.

**8.14.3.8** `void GUIMainWindow::OnAnalyze ( wxCommandEvent & event ) [private]`

Öffnet das Analysedaten-Übersichtsfenster.

Definiert in Zeile 615 der Datei GUIMainWindow.cpp.

**8.14.3.9** `void GUIMainWindow::OnAnalyzeMarkerChange ( wxCommandEvent & event ) [private]`

Behandelt das Markieren eines Zeitpunktes auf der Sensordaten-Zeitleiste.

Definiert in Zeile 380 der Datei GUIMainWindow.cpp.

**8.14.3.10** `void GUIMainWindow::OnAnalyzePoint ( wxCommandEvent & event ) [private]`

Öffnet das Fenster zur Analyse eines Punktes ([GUIAnalyzePointWindow](#)).

Definiert in Zeile 631 der Datei GUIMainWindow.cpp.

**8.14.3.11** `void GUIMainWindow::OnAutoUpdateChange ( wxCommandEvent & event ) [private]`

Behandelt das aktivieren/deaktivieren der Option zum automatischen Neuberechnen der Temperaturverteilung eines Objekts, sobald Änderungen an dessen Eigenschaften vorgenommen werden.

Definiert in Zeile 476 der Datei GUIMainWindow.cpp.

**8.14.3.12** `void GUIMainWindow::OnExportViewportImage ( wxCommandEvent & event ) [private]`

Öffnet ein Fenster zum Exportieren der Ansicht des 3D-Fensters.

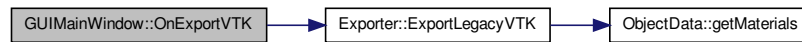
Definiert in Zeile 875 der Datei GUIMainWindow.cpp.

**8.14.3.13** `void GUIMainWindow::OnExportVTK ( wxCommandEvent & event ) [private]`

Öffnet ein Fenster zum Exportieren der Temperaturverteilung und des Objekts im VTK-Format.

Definiert in Zeile 904 der Datei GUIMainWindow.cpp.

Hier ist ein Graph, der zeigt, was diese Funktion aufruft:



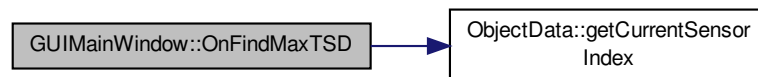
#### 8.14.3.14 void GUIMainWindow::OnFindMaxTSD ( wxCommandEvent & event ) [private]

Sucht den Zeitpunkt zwischen zwei markierten Stellen auf der Sensordaten-Zeitleiste, für den der Wärmeenergiegehalt maximal wird.

Dabei wird der Bereich zwischen den beiden markierten Stellen ausgewählt, zwischen denen sich der aktuell ausgewählte Zeitpunkt befindet.

Definiert in Zeile 390 der Datei `GUIMainWindow.cpp`.

Hier ist ein Graph, der zeigt, was diese Funktion aufruft:



#### 8.14.3.15 void GUIMainWindow::OnGeneralPropChange ( wxCommandEvent & event ) [private]

Behandelt das Aktualisieren der Oberfläche nach einer Änderung an Objekteigenschaften durch den Nutzer.

Definiert in Zeile 674 der Datei `GUIMainWindow.cpp`.

#### 8.14.3.16 void GUIMainWindow::OnImmediateUpdatePropChange ( wxCommandEvent & event ) [private]

Behandelt das Aktualisieren der Oberfläche nach einer Änderung an Objekteigenschaften, bei denen ein sofortiges Update der Oberfläche möglich ist, durch den Nutzer.

Definiert in Zeile 662 der Datei `GUIMainWindow.cpp`.

#### 8.14.3.17 void GUIMainWindow::OnMaterialSelect ( wxCommandEvent & event ) [private]

Aktualisiert die Oberfläche nach dem Auswählen eines anderen Materials im Objekteigenschaften-Fenster.

Definiert in Zeile 582 der Datei `GUIMainWindow.cpp`.

#### 8.14.3.18 void GUIMainWindow::OnMenuFileQuit ( wxCommandEvent & event ) [private]

Beendet das Programm.

Definiert in Zeile 960 der Datei `GUIMainWindow.cpp`.

#### 8.14.3.19 void GUIMainWindow::OnMenuHelpAbout ( wxCommandEvent & event ) [private]

Öffnet ein Fenster mit Informationen über das Programm.

Definiert in Zeile 980 der Datei GUIMainWindow.cpp.

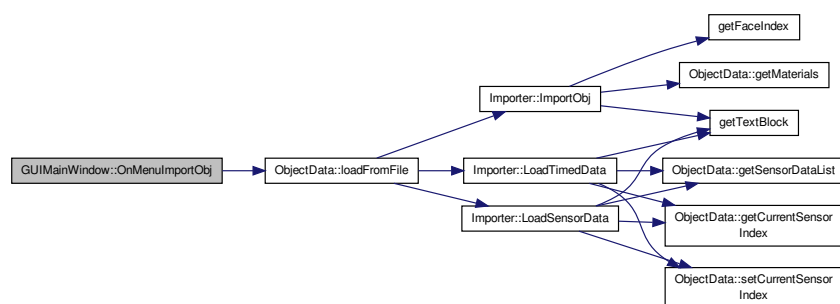
#### 8.14.3.20 void GUIMainWindow::OnMenuImportObj ( wxCommandEvent & event ) [private]

Event-Tabellendeklaration für wxWidgets.

Öffnet den Dialog zum Importieren eines Objekts.

Definiert in Zeile 781 der Datei GUIMainWindow.cpp.

Hier ist ein Graph, der zeigt, was diese Funktion aufruft:

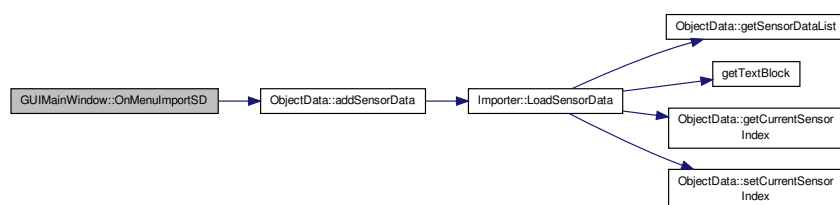


#### 8.14.3.21 void GUIMainWindow::OnMenuImportSD ( wxCommandEvent & event ) [private]

Öffnet den Dialog zum Importieren einfacher Sensordaten.

Definiert in Zeile 833 der Datei GUIMainWindow.cpp.

Hier ist ein Graph, der zeigt, was diese Funktion aufruft:

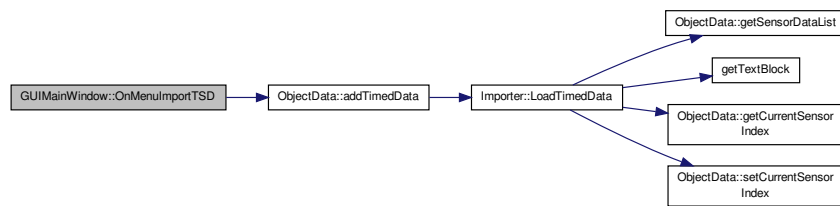


#### 8.14.3.22 void GUIMainWindow::OnMenuImportTSD ( wxCommandEvent & event ) [private]

Öffnet den Dialog zum Importieren zeitbezogener Sensordaten.

Definiert in Zeile 854 der Datei GUIMainWindow.cpp.

Hier ist ein Graph, der zeigt, was diese Funktion aufruft:



#### 8.14.3.23 void GUI\_MainWindow::OnMenuOpenManual ( wxCommandEvent & event ) [private]

Öffnet das Handbuch mit dem PDF-Viewer des Systems.

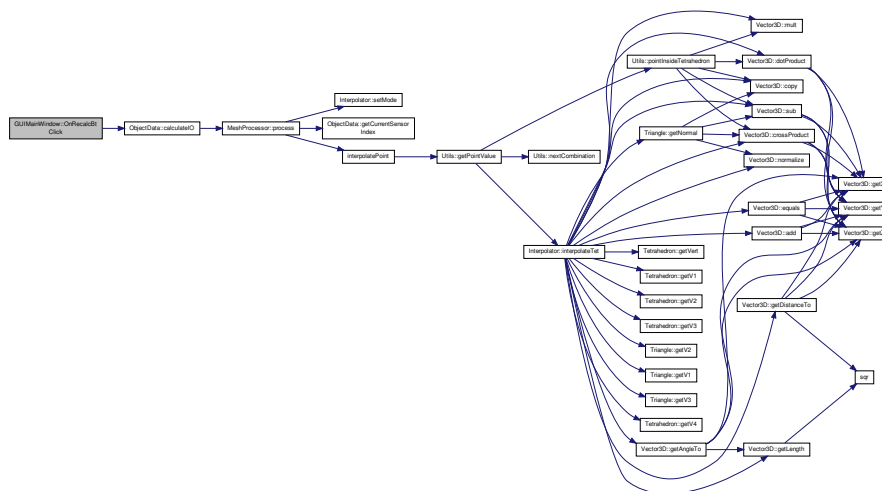
Definiert in Zeile 965 der Datei GUI\_MainWindow.cpp.

#### 8.14.3.24 void GUI\_MainWindow::OnRecalcBtClick ( wxCommandEvent & event ) [private]

Berechnet die 3D-Temperaturverteilung neu.

Definiert in Zeile 592 der Datei GUI\_MainWindow.cpp.

Hier ist ein Graph, der zeigt, was diese Funktion aufruft:



#### 8.14.3.25 void GUI\_MainWindow::OnRenderCut ( wxCommandEvent & event ) [private]

Öffnet das Fenster zur Berechnung einer zweidimensionalen Temperaturverteilung.

Definiert in Zeile 644 der Datei GUI\_MainWindow.cpp.

#### 8.14.3.26 void GUI\_MainWindow::OnResize ( wxSizeEvent & event ) [private]

Behandelt Größenänderungen des Fensters.

Definiert in Zeile 323 der Datei GUI\_MainWindow.cpp.

**8.14.3.27** void GUIMainWindow::OnSDTimelineChange ( wxCommandEvent & *event* ) [private]

Behandelt Änderungen an der Sensordaten-Zeitleiste (bei zeitbezogenen Datensätzen).

Definiert in Zeile 348 der Datei GUIMainWindow.cpp.

**8.14.3.28** void GUIMainWindow::OnSDTLMarkerClear ( wxCommandEvent & *event* ) [private]

Löscht alle Markierungen auf der Sensordaten-Zeitleiste (bei zeitbezogenen Datensätzen).

Definiert in Zeile 370 der Datei GUIMainWindow.cpp.

**8.14.3.29** void GUIMainWindow::OnSDTLNextMarker ( wxCommandEvent & *event* ) [private]

Setzen des auf der Sensordaten-Zeitleiste ausgewählten Zeitpunktes auf den nächsten markierten Zeitpunkt.

Definiert in Zeile 420 der Datei GUIMainWindow.cpp.

**8.14.3.30** void GUIMainWindow::OnSDTLPrevMarker ( wxCommandEvent & *event* ) [private]

Setzen des auf der Sensordaten-Zeitleiste ausgewählten Zeitpunktes auf den vorherigen markierten Zeitpunkt.

Definiert in Zeile 440 der Datei GUIMainWindow.cpp.

**8.14.3.31** void GUIMainWindow::OnSensorDataChange ( wxCommandEvent & *event* ) [private]

Behandelt das Auswählen eines anderen Sensordatensatzes.

Definiert in Zeile 360 der Datei GUIMainWindow.cpp.

**8.14.3.32** void GUIMainWindow::OnViewPropChange ( wxCommandEvent & *event* ) [private]

Behandelt das Aktualisieren der Oberfläche nach einer Änderung an Visualisierungsoptionen durch den Nutzer.

Definiert in Zeile 767 der Datei GUIMainWindow.cpp.

**8.14.3.33** void GUIMainWindow::OnViewPropSpinChange ( wxSpinEvent & *event* ) [private]

Behandelt das Aktualisieren der Oberfläche nach einer Änderung an Visualisierungsoptionen durch den Nutzer.

Definiert in Zeile 774 der Datei GUIMainWindow.cpp.

**8.14.3.34** void GUIMainWindow::setActiveObject ( int *index* ) [private]

Setzt das aktive Objekt.

Parameter

<i>index</i>	Index des als aktives Objekt zu verwendeten Objekts.
--------------	--

Definiert in Zeile 297 der Datei GUIMainWindow.cpp.

**8.14.3.35** void GUIMainWindow::setAnalyzeWindowStatus ( bool *isValid* )

Setzt den Status des Übersichtsfensters über die Analysedaten.

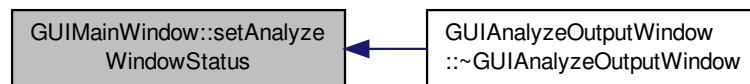


## Parameter

<i>isValid</i>	Ob das Fenster ein gültiges Objekt oder ob der Speicher bereits freigegeben ist.
----------------	--

Definiert in Zeile 278 der Datei GUIMainWindow.cpp.

Hier ist ein Graph der zeigt, wo diese Funktion aufgerufen wird:



#### 8.14.3.36 void GUIMainWindow::setCutRenderWindowStatus ( bool isValid )

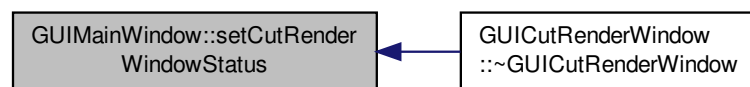
Setzt den Status des Übersichtsfensters über die Analysedaten.

## Parameter

<i>isValid</i>	Ob das Fenster ein gültiges Objekt oder ob der Speicher bereits freigegeben ist.
----------------	--

Definiert in Zeile 282 der Datei GUIMainWindow.cpp.

Hier ist ein Graph der zeigt, wo diese Funktion aufgerufen wird:

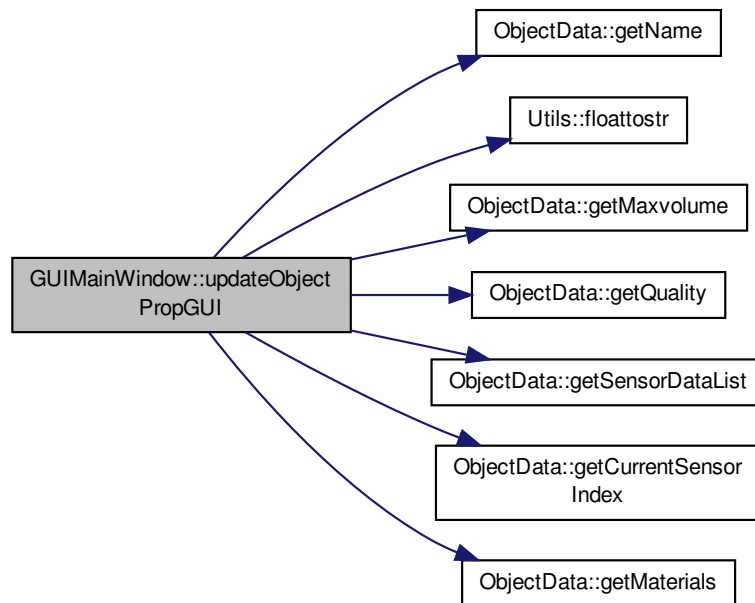


#### 8.14.3.37 void GUIMainWindow::updateObjectPropGUI ( ) [private]

Überträgt die Eigenschaften des aktiven Objekts in die GUI.

Definiert in Zeile 514 der Datei GUIMainWindow.cpp.

Hier ist ein Graph, der zeigt, was diese Funktion aufruft:

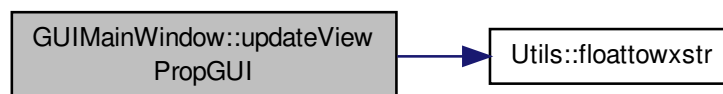


#### 8.14.3.38 void GUIMainWindow::updateViewPropGUI ( ) [private]

Lädt die Visualisierungsoptionen in die GUI.

Definiert in Zeile 725 der Datei GUIMainWindow.cpp.

Hier ist ein Graph, der zeigt, was diese Funktion aufruft:



### 8.14.4 Dokumentation der Datenelemente

#### 8.14.4.1 bool GUIMainWindow::analyze\_window\_valid [private]

Das Analysedaten-Übersichtsfenster ist gerade geöffnet.

Definiert in Zeile 348 der Datei GUIMainWindow.h.

**8.14.4.2 GUIAnalyzeOutputWindow\* GUIMainWindow::analyzerframe** [private]

Das Analysedaten-Übersichtsfenster.

Der Zeiger ist ungültig, wenn das Analysedaten-Übersichtsfenster nicht geöffnet ist. (siehe analyze\_window\_valid)

Definiert in Zeile 322 der Datei GUIMainWindow.h.

**8.14.4.3 string GUIMainWindow::configpaths[NUMBEROFPATHS]** [protected]**Initialisierung:**

```
{  
    "/usr/local/share/simpleanalyzer/",  
    "/usr/share/simpleanalyzer/",  
}
```

Suchpfade für die Anwendungsdaten.

Das Verzeichnis der ausführbaren Datei wird immer und zuerst geprüft.

Definiert in Zeile 72 der Datei GUIMainWindow.h.

**8.14.4.4 string GUIMainWindow::data\_directory** [private]

Der Pfad zum Verzeichnis, das die von der Anwendung verwendeten Daten (z.B. Icons) enthält. Wird im Konstruktor bestimmt.

Definiert in Zeile 358 der Datei GUIMainWindow.h.

**8.14.4.5 GUIGLCanvas\* GUIMainWindow::gl\_context** [private]

Die Zeichenfläche für das 3D-Fenster.

Definiert in Zeile 266 der Datei GUIMainWindow.h.

**8.14.4.6 wxMenu\* GUIMainWindow::mwAnalyzeMenu** [private]

Das "Analysieren"-Untermenü.

Definiert in Zeile 301 der Datei GUIMainWindow.h.

**8.14.4.7 wxMenu\* GUIMainWindow::mwEditMenu** [private]

Das "Bearbeiten"-Untermenü

Definiert in Zeile 306 der Datei GUIMainWindow.h.

**8.14.4.8 wxMenu\* GUIMainWindow::mwExportMenu** [private]

Das "Export"-Untermenü.

Definiert in Zeile 296 der Datei GUIMainWindow.h.

**8.14.4.9 wxMenu\* GUIMainWindow::mwFileMenu** [private]

Das "Datei"-Untermenü.

Definiert in Zeile 281 der Datei GUIMainWindow.h.

**8.14.4.10** `wxMenu* GUIMainWindow::mwHelpMenu` `[private]`

Das "Hilfe"-Untermenü.

Definiert in Zeile 286 der Datei GUIMainWindow.h.

**8.14.4.11** `wxMenu* GUIMainWindow::mwImportMenu` `[private]`

Das "Import"-Untermenü.

Definiert in Zeile 291 der Datei GUIMainWindow.h.

**8.14.4.12** `wxMenuBar* GUIMainWindow::mwMenuBar` `[private]`

Die Hauptmenükomponente.

Definiert in Zeile 276 der Datei GUIMainWindow.h.

**8.14.4.13** `const int GUIMainWindow::NUMBEROFPATHS = 2` `[static], [protected]`

Anzahl der Suchpfade für die Anwendungsdaten (z.B. Icons).

Definiert in Zeile 65 der Datei GUIMainWindow.h.

**8.14.4.14** `wxScrolledWindow* GUIMainWindow::prop_scroll_win` `[private]`

Scrollender Bereich, in den die Objekteigenschaften-Oberfläche eingebettet ist.

Definiert in Zeile 333 der Datei GUIMainWindow.h.

**8.14.4.15** `PropertiesBox* GUIMainWindow::propbox` `[private]`

Die Unterkomponente, die die Objekteigenschaften-Oberfläche enthält.

Definiert in Zeile 311 der Datei GUIMainWindow.h.

**8.14.4.16** `bool GUIMainWindow::render_cut_window_valid` `[private]`

Das Fenster zur Berechnung einer zweidimensionalen Temperaturverteilung ist gerade geöffnet.

Definiert in Zeile 353 der Datei GUIMainWindow.h.

**8.14.4.17** `GUICutRenderWindow* GUIMainWindow::rendercutwindow` `[private]`

Das Fenster zur Berechnung einer zweidimensionalen Temperaturverteilung.

Der Zeiger ist ungültig, wenn das 2D-Fenster nicht geöffnet ist. (siehe `render_cut_window_valid`)

Definiert in Zeile 328 der Datei GUIMainWindow.h.

**8.14.4.18** `wxToolBar* GUIMainWindow::toolbar` `[private]`

Die Tollbarkomponente.

Definiert in Zeile 271 der Datei GUIMainWindow.h.

#### 8.14.4.19 `bool GUIMainWindow::updating` [private]

Die Oberfläche wird gerade vom Programm verändert.

Signalisiert, dass die Eingabe nicht durch den Nutzer erfolgt ist.

Definiert in Zeile 343 der Datei GUIMainWindow.h.

#### 8.14.4.20 `wxScrolledWindow* GUIMainWindow::view_scroll_win` [private]

Scrollender Bereich, in den die Visualisierungsoptionen-Oberfläche eingebettet ist.

Definiert in Zeile 338 der Datei GUIMainWindow.h.

#### 8.14.4.21 `ViewpropBox* GUIMainWindow::viewbox` [private]

Die Unterkomponente, die die Visualisierungsoptionen-Oberfläche enthält.

Definiert in Zeile 316 der Datei GUIMainWindow.h.

Die Dokumentation für diese Klasse wurde erzeugt aufgrund der Dateien:

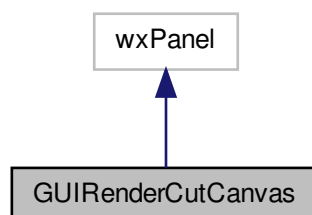
- [/daten/Projekte/eclipse\\_workspace/simpleanalyzer-gui/src/GUI/GUIMainWindow.h](#)
- [/daten/Projekte/eclipse\\_workspace/simpleanalyzer-gui/src/GUI/GUIMainWindow.cpp](#)

## 8.15 GUIRenderCutCanvas Klassenreferenz

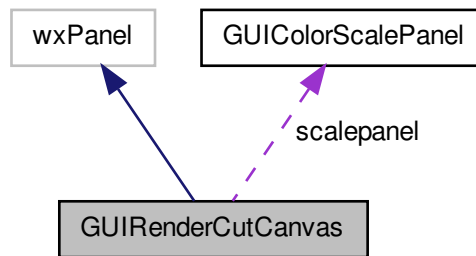
Zeichenfläche für die 2D-Temperaturverteilung.

```
#include <GUIRenderCutCanvas.h>
```

Klassendiagramm für GUIRenderCutCanvas:



Zusammengehörigkeiten von GUIRenderCutCanvas:



## Öffentliche Methoden

- **GUIRenderCutCanvas** (wxWindow \*parent)  
*Der Konstruktor.*
- void **setImage** (wxImage \*img)  
*Setzt die aktuell angezeigte Grafik.*
- void **setValueImg** (float \*img)  
*Setzt die zum anzeigen von werten verwendete Temperaturverteilung.*
- **GUIColorScalePanel** \* **getScalePanel** ()  
*Gibt das Temperaturskala-Objekt zurück.*
- virtual **~GUIRenderCutCanvas** ()  
*Der Destruktor.*

## Private Methoden

- void **onCanvasPaint** (wxPaintEvent &event)  
*Event-Tabellendeklaration für wxWidgets.*
- void **OnMouseWheel** (wxMouseEvent &event)  
*Behandelt das Zoomen in der Grafik.*
- void **OnMouseMove** (wxMouseEvent &event)  
*Behandelt das verschieben der Ansicht und speichert die Mauszeigerposition zur Ermittlung des Wertes an dieser Stelle in **onCanvasPaint()**.*
- void **OnResize** (wxSizeEvent &event)  
*Behandelt Größenänderungen der Zeichenfläche.*
- void **OnMouseDown** (wxMouseEvent &event)  
*Behandelt klicken mit der Maus, deren Status zum verschieben der Ansicht benötigt wird.*

## Private Attribute

- float **zoom**  
*Der aktuelle Zoomfaktor für die Zeichenfläche.*
- float **deltaX**  
*horizontale Verschiebung der Ansicht.*
- float **deltaY**

*vertikale Verschiebung der Ansicht.*

- int [current\\_mx](#)

*Zwischenspeicher für die horizontale Mausposition.*

- int [current\\_my](#)

*Zwischenspeicher für die vertikale Mausposition.*

- bool [mouse\\_to\\_scalepanel](#)

*Müssen die Mauseaktionen zur Skala weitergeleitet werden? (Wird diese gerade Transformiert?)*

- wxImage \* [image](#)

*Die aktuelle dargestellte Temperaturverteilung als Grafik.*

- float \* [value\\_img](#)

*Die aktuelle dargestellte Temperaturverteilung.*

- [GUIColorScalePanel](#) \* [scalepanel](#)

*Die Temperaturskala.*

### 8.15.1 Ausführliche Beschreibung

Zeichenfläche für die 2D-Temperaturverteilung.

Zeichenfläche für das Fenster zur Berechnung einer 2D-Temperaturverteilung. Zeigt die berechnete Grafik, Skala und eine Statusleiste an. Verwaltet auch Mauseingaben zum Verschieben und Zoomen der Ansicht.

Definiert in Zeile 20 der Datei GUIRenderCutCanvas.h.

### 8.15.2 Beschreibung der Konstruktoren und Destruktoren

#### 8.15.2.1 GUIRenderCutCanvas::GUIRenderCutCanvas ( wxWindow \* *parent* )

Der Konstruktor.

Parameter

<i>parent</i>	Das Fenster, auf dem die Zeichenfläche liegen soll.
---------------	---

Definiert in Zeile 29 der Datei GUIRenderCutCanvas.cpp.

#### 8.15.2.2 GUIRenderCutCanvas::~~GUIRenderCutCanvas ( ) [virtual]

Der Destruktor.

Definiert in Zeile 230 der Datei GUIRenderCutCanvas.cpp.

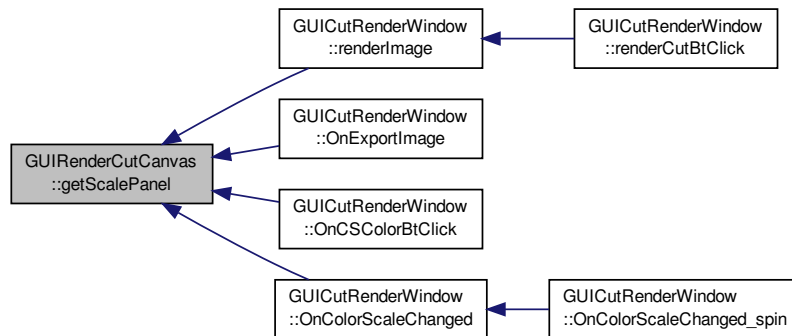
### 8.15.3 Dokumentation der Elementfunktionen

#### 8.15.3.1 GUIColorScalePanel \* GUIRenderCutCanvas::getScalePanel ( )

Gibt das Temperaturskala-Objekt zurück.

Definiert in Zeile 226 der Datei GUIRenderCutCanvas.cpp.

Hier ist ein Graph der zeigt, wo diese Funktion aufgerufen wird:



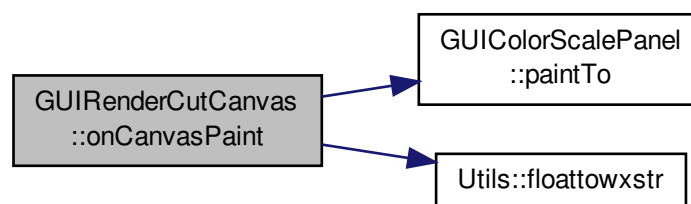
#### 8.15.3.2 void GUIRenderCutCanvas::onCanvasPaint ( wxPaintEvent & event ) [private]

Event-Tabellendeklaration für wxWidgets.

Zeichnet die Temperaturverteilung und die Anzeigeelemente (Informationsleiste, Skala).

Definiert in Zeile 130 der Datei `GUIRenderCutCanvas.cpp`.

Hier ist ein Graph, der zeigt, was diese Funktion aufruft:



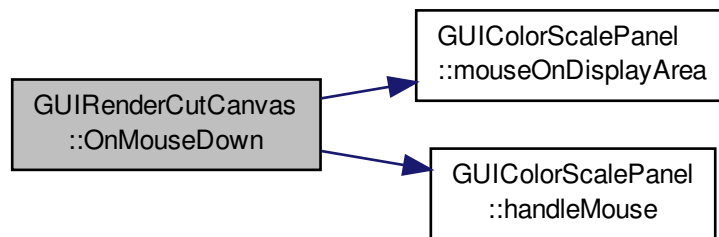
#### 8.15.3.3 void GUIRenderCutCanvas::OnMouseDown ( wxMouseEvent & event ) [private]

Behandelt klicken mit der Maus, deren Status zum verschieben der Ansicht benötigt wird.

Definiert in Zeile 108 der Datei `GUIRenderCutCanvas.cpp`.



Hier ist ein Graph, der zeigt, was diese Funktion aufruft:

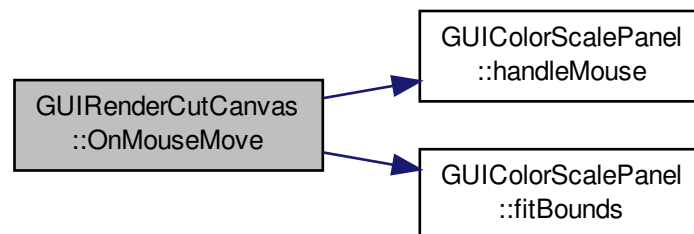


#### 8.15.3.4 `void GUIRenderCutCanvas::OnMouseMove ( wxMouseEvent & event ) [private]`

Behandelt das verschieben der Ansicht und speichert die Mauszeigerposition zur Ermittlung des Wertes an dieser Stelle in [onCanvasPaint\(\)](#).

Definiert in Zeile 65 der Datei `GUIRenderCutCanvas.cpp`.

Hier ist ein Graph, der zeigt, was diese Funktion aufruft:



#### 8.15.3.5 `void GUIRenderCutCanvas::OnMouseWheel ( wxMouseEvent & event ) [private]`

Behandelt das Zoomen in der Grafik.

Definiert in Zeile 55 der Datei `GUIRenderCutCanvas.cpp`.

#### 8.15.3.6 `void GUIRenderCutCanvas::OnResize ( wxSizeEvent & event ) [private]`

Behandelt Größenänderungen der Zeichenfläche.

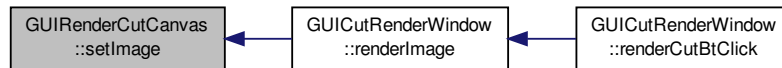
Definiert in Zeile 104 der Datei `GUIRenderCutCanvas.cpp`.

#### 8.15.3.7 void GUIRenderCutCanvas::setImage ( wxImage \* img )

Setzt die aktuell angezeigte Grafik.

Definiert in Zeile 47 der Datei GUIRenderCutCanvas.cpp.

Hier ist ein Graph der zeigt, wo diese Funktion aufgerufen wird:

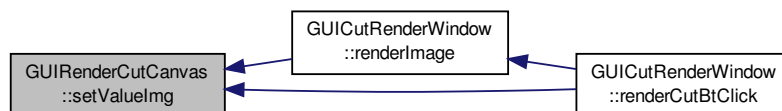


#### 8.15.3.8 void GUIRenderCutCanvas::setValueImg ( float \* img )

Setzt die zum anzeigen von werten verwendete Temperaturverteilung.

Definiert in Zeile 51 der Datei GUIRenderCutCanvas.cpp.

Hier ist ein Graph der zeigt, wo diese Funktion aufgerufen wird:



### 8.15.4 Dokumentation der Datenelemente

#### 8.15.4.1 int GUIRenderCutCanvas::current\_mx [private]

Zwischenspeicher für die horizontale Mausposition.

Definiert in Zeile 97 der Datei GUIRenderCutCanvas.h.

#### 8.15.4.2 int GUIRenderCutCanvas::current\_my [private]

Zwischenspeicher für die vertikale Mausposition.

Definiert in Zeile 102 der Datei GUIRenderCutCanvas.h.

#### 8.15.4.3 float GUIRenderCutCanvas::deltaX [private]

horizontale Verschiebung der Ansicht.

Definiert in Zeile 87 der Datei GUIRenderCutCanvas.h.

#### 8.15.4.4 float GUIRenderCutCanvas::deltaY [private]

vertikale Verschiebung der Ansicht.

Definiert in Zeile 92 der Datei GUIRenderCutCanvas.h.

#### 8.15.4.5 wxImage\* GUIRenderCutCanvas::image [private]

Die aktuelle dargestellte Temperaturverteilung als Grafik.

Definiert in Zeile 112 der Datei GUIRenderCutCanvas.h.

#### 8.15.4.6 bool GUIRenderCutCanvas::mouse\_to\_scalepanel [private]

Müssen die Mauseaktionen zur Skala weitergeleitet werden? (Wird diese gerade Transformiert?)

Definiert in Zeile 107 der Datei GUIRenderCutCanvas.h.

#### 8.15.4.7 GUIColorScalePanel\* GUIRenderCutCanvas::scalepanel [private]

Die Temperaturskala.

Definiert in Zeile 122 der Datei GUIRenderCutCanvas.h.

#### 8.15.4.8 float\* GUIRenderCutCanvas::value\_img [private]

Die aktuelle dargestellte Temperaturverteilung.

Definiert in Zeile 117 der Datei GUIRenderCutCanvas.h.

#### 8.15.4.9 float GUIRenderCutCanvas::zoom [private]

Der aktuelle Zoomfaktor für die Zeichenfläche.

Definiert in Zeile 82 der Datei GUIRenderCutCanvas.h.

Die Dokumentation für diese Klasse wurde erzeugt aufgrund der Dateien:

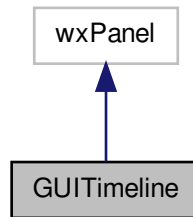
- [/daten/Projekte/eclipse\\_workspace/simpleanalyzer-gui/src/GUI/GUIRenderCutCanvas.h](#)
- [/daten/Projekte/eclipse\\_workspace/simpleanalyzer-gui/src/GUI/GUIRenderCutCanvas.cpp](#)

## 8.16 GUITimeline Klassenreferenz

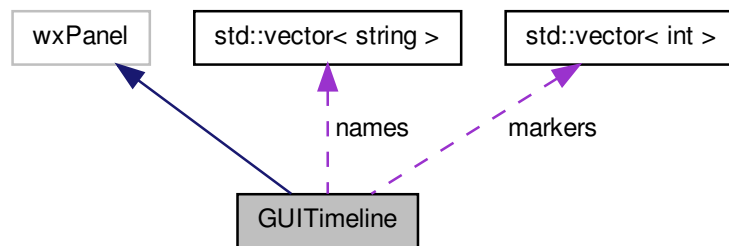
Eine Zeitleistenkomponente.

```
#include <GUITimeline.h>
```

Klassendiagramm für GUITimeline:



Zusammengehörigkeiten von GUITimeline:



## Öffentliche Typen

- enum `GUI_TIMELINE_STYLE` { `GTL_DEFAULT` = 0 }  
*Darstellungsstil der Zeitleiste.*

## Öffentliche Methoden

- `GUITimeline` (`wxWindow *parent`, `wxWindowID id`, `const wxPoint &pos=wxDefaultPosition`, `const wxSize &size=wxDefaultSize`, `long style=GTL_DEFAULT`, `const wxString &name=wxT("Timeline")`)  
*Der Konstruktor.*
- void `findMaxValue` (`ObjectData *obj`, `bool fast`)  
*Sucht den Zeitpunkt zwischen zwei markierten Stellen auf der Sensordaten-Zeitleiste, für den der Wäremeenergiegehalt maximal wird.*
- int `getValue` ()  
*gibt den Index des aktuell ausgewählten Zeitpunkts zurück.*
- int `getMaxValue` ()  
*Gibt den maximal auswählbaren Index zurück.*
- int `getMinValue` ()  
*Gibt den minimal auswählbaren Index zurück.*

- void `setValue` (int val)  
*Setzt den Index des aktuell ausgewählten Zeitpunkts.*
- void `setMaxValue` (int val)  
*Setzt den maximal auswählbaren Index.*
- void `setMinValue` (int val)  
*Setzt den minimal auswählbaren Index.*
- void `setNameList` (vector< string > \*namelist)  
*Setzt die Liste der Namen für die jeweiligen Indices der Zeitpunkte.*
- void `setMarked` (int pos, bool state)  
*Markiert/De-markiert einen bestimmten Zeitpunkt.*
- bool `isMarked` (int pos)  
*Gibt zurück, ob ein Zeitpunkt markiert ist.*
- void `clearMarkers` ()  
*Entfernt alle Markierungen.*
- void `setMarkerList` (vector< int > \*mlist)  
*Setzt die Liste der markierten Stellen.*
- vector< int > \* `getMarkers` ()  
*Gibt die Liste der markierten Stellen zurück.*
- void `setMarkers` (vector< int > \*mlist)  
*Markiert eine Liste von Indices.*
- virtual `~GUITimeline` ()  
*Der Destruktor.*

### Private Methoden

- void `OnPaint` (wxPaintEvent &)  
*Event-Tabellendeklaration für wxWidgets.*
- void `OnMouseWheel` (wxMouseEvent &event)  
*Behandelt Scrolleingaben (zoomen).*
- void `OnMouseMove` (wxMouseEvent &event)  
*Behandelt Mausbewegungen (verschieben der Ansicht).*
- void `OnResize` (wxSizeEvent &event)  
*Behandelt Größenänderungen der Zeitleiste.*
- void `OnMouseDown` (wxMouseEvent &event)  
*Behandelt klicken (verschieben der Ansicht, setzen des aktuellen Zeitpunkts).*
- void `OnKeyDown` (wxKeyEvent &event)  
*Behandelt Tastendruck (setzen des aktuellen Zeitpunkts).*
- void `posToVal` (int mouse\_x)  
*Setzt den aktuellen Zeitpunkt anhand der Mausposition.*
- void `sendTimelineEvent` ()  
*Löst ein wxEVT\_TIMELINE\_CHANGE-Event aus.*
- int `calcStepWidth` ()  
*Berechnet die für die aktuelle Darstellung günstigste Schrittweite für die Beschriftung.*

## Private Attribute

- int `value`  
*Der Index des aktuell ausgewählten Zeitpunkts.*
- int `maxvalue`  
*Der größte anzuzeigende Zeitpunkt.*
- int `minvalue`  
*Der kleinste anzuzeigende Zeitpunkt.*
- int `maxdigits`  
*Maximale Anzahl an anzuzeigenden Nachkommastellen.*
- float `zoom`  
*Aktueller Zoomfaktor.*
- float `delta_v_view`  
*Verschiebung der Ansicht.*
- int `prev_mouse_x`  
*Zwischenspeicher für die vorherige horizontale Mausposition.*
- `vector< string > * names`  
*Liste der Zeitpunktnamen.*
- `vector< int > * markers`  
*Liste der markierten Zeitpunkte.*

### 8.16.1 Ausführliche Beschreibung

Eine Zeitleistenkomponente.

Die Komponente kann Zeitpunkte als Zeitleiste darstellen, wobei die Zeitpunkte anhand von Indices ausgewählt werden können. Zusätzlich kann eine Liste von Namen für die Zeitpunkte festgelegt werden, wodurch auch der Name des gewählten Zeitpunkts angezeigt wird. Weiterhin können Zeitpunkte markiert werden.

Definiert in Zeile 32 der Datei GUITimeline.h.

### 8.16.2 Dokumentation der Aufzählungstypen

#### 8.16.2.1 enum GUITimeline::GUI\_TIMELINE\_STYLE

Darstellungsstil der Zeitleiste.

Aufzählungswerte

***GTL\_DEFAULT***

Definiert in Zeile 37 der Datei GUITimeline.h.

### 8.16.3 Beschreibung der Konstruktoren und Destruktoren

**8.16.3.1** `GUITimeline::GUITimeline ( wxWindow * parent, wxWindowID id, const wxPoint & pos = wxDefaultPosition, const wxSize & size = wxDefaultSize, long style = GTL_DEFAULT, const wxString & name = wxT("Timeline") )`

Der Konstruktor.

## Parameter

<i>parent</i>	Die übergeordnete Komponente.
<i>id</i>	Die ID des Objekts.
<i>pos</i>	Die Position der Zeitleiste.
<i>size</i>	Die Größe der Zeitleiste.
<i>style</i>	Darstellungsstil der Zeitleiste.
<i>name</i>	Name der Zeitleiste (Komponentenname, nicht sichtbar).

Definiert in Zeile 36 der Datei GUITimeline.cpp.

### 8.16.3.2 GUITimeline::~GUITimeline ( ) [virtual]

Der Destruktor.

Definiert in Zeile 543 der Datei GUITimeline.cpp.

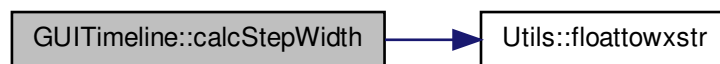
## 8.16.4 Dokumentation der Elementfunktionen

### 8.16.4.1 int GUITimeline::calcStepWidth ( ) [private]

Berechnet die für die aktuelle Darstellung günstigste Schrittweite für die Beschriftung.

Definiert in Zeile 211 der Datei GUITimeline.cpp.

Hier ist ein Graph, der zeigt, was diese Funktion aufruft:



Hier ist ein Graph der zeigt, wo diese Funktion aufgerufen wird:



### 8.16.4.2 void GUITimeline::clearMarkers ( )

Entfernt alle Markierungen.

Definiert in Zeile 512 der Datei GUITimeline.cpp.

#### 8.16.4.3 void GUITimeline::findMaxValue ( ObjectData \* obj, bool fast )

Sucht den Zeitpunkt zwischen zwei markierten Stellen auf der Sensordaten-Zeitleiste, für den der Wärmeenergiegehalt maximal wird.

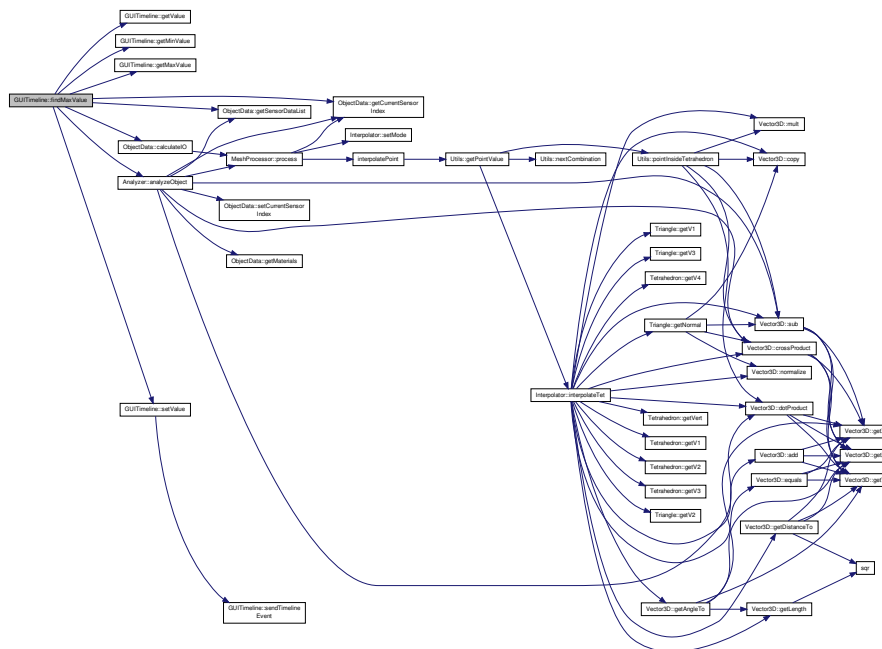
Dabei wird der Bereich zwischen den beiden markierten Stellen ausgewählt, zwischen denen sich der aktuell ausgewählte Zeitpunkt befindet.

Parameter

<i>obj</i>	Das zu untersuchende Objekt.
<i>fast</i>	Schnelle Methode verwenden. D.h., es wird statt der Temperaturverteilung nur die Durchschnittstemperatur verglichen.

Definiert in Zeile 112 der Datei GUITimeline.cpp.

Hier ist ein Graph, der zeigt, was diese Funktion aufruft:



#### 8.16.4.4 vector< int > \* GUITimeline::getMarkers ( )

Gibt die Liste der markierten Stellen zurück.

Definiert in Zeile 517 der Datei GUITimeline.cpp.

#### 8.16.4.5 int GUITimeline::getMaxValue ( )

Gibt den maximal auswählbaren Index zurück.

Definiert in Zeile 431 der Datei GUITimeline.cpp.



Hier ist ein Graph der zeigt, wo diese Funktion aufgerufen wird:



#### 8.16.4.6 `int GUITimeline::getMinValue ( )`

Gibt den minimal auswählbaren Index zurück.

Definiert in Zeile 435 der Datei `GUITimeline.cpp`.

Hier ist ein Graph der zeigt, wo diese Funktion aufgerufen wird:



#### 8.16.4.7 `int GUITimeline::getValue ( )`

gibt den Index des aktuell ausgewählten Zeitpunkts zurück.

Definiert in Zeile 427 der Datei `GUITimeline.cpp`.

Hier ist ein Graph der zeigt, wo diese Funktion aufgerufen wird:



#### 8.16.4.8 `bool GUITimeline::isMarked ( int pos )`

Gibt zurück, ob ein Zeitpunkt markiert ist.

## Parameter

<i>pos</i>	Index des Zeitpunkts.
------------	-----------------------

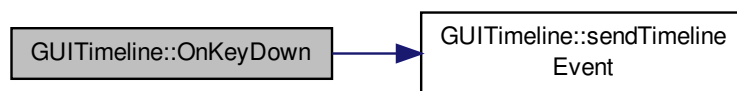
Definiert in Zeile 501 der Datei GUITimeline.cpp.

#### 8.16.4.9 void GUITimeline::OnKeyDown ( wxKeyEvent & event ) [private]

Behandelt Tastendruck (setzen des aktuellen Zeitpunkts).

Definiert in Zeile 85 der Datei GUITimeline.cpp.

Hier ist ein Graph, der zeigt, was diese Funktion aufruft:

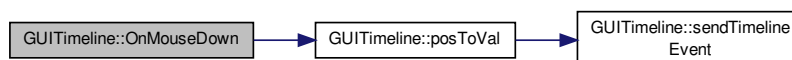


#### 8.16.4.10 void GUITimeline::OnMouseDown ( wxMouseEvent & event ) [private]

Behandelt klicken (verschieben der Ansicht, setzen des aktuellen Zeitpunkts).

Definiert in Zeile 79 der Datei GUITimeline.cpp.

Hier ist ein Graph, der zeigt, was diese Funktion aufruft:

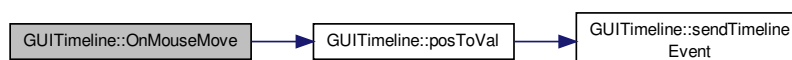


#### 8.16.4.11 void GUITimeline::OnMouseMove ( wxMouseEvent & event ) [private]

Behandelt Mausbewegungen (verschieben der Ansicht).

Definiert in Zeile 278 der Datei GUITimeline.cpp.

Hier ist ein Graph, der zeigt, was diese Funktion aufruft:



**8.16.4.12 void GUITimeline::OnMouseWheel ( wxMouseEvent & event ) [private]**

Behandelt Scrolleingaben (zoomen).

Definiert in Zeile 59 der Datei GUITimeline.cpp.

**8.16.4.13 void GUITimeline::OnPaint ( wxPaintEvent & ) [private]**

Event-Tabellendeklaration für wxWidgets.

Zeichnet die Zeitleiste.

Definiert in Zeile 307 der Datei GUITimeline.cpp.

Hier ist ein Graph, der zeigt, was diese Funktion aufruft:

**8.16.4.14 void GUITimeline::OnResize ( wxSizeEvent & event ) [private]**

Behandelt Größenänderungen der Zeitleiste.

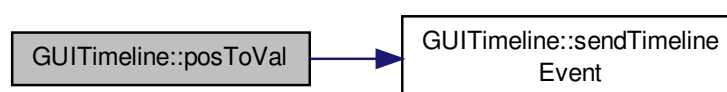
Definiert in Zeile 303 der Datei GUITimeline.cpp.

**8.16.4.15 void GUITimeline::posToVal ( int mouse\_x ) [private]**

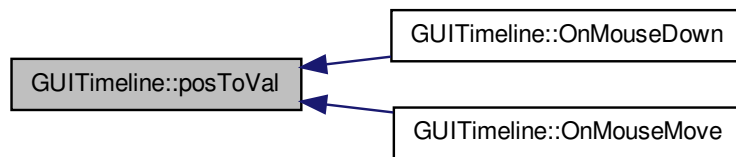
Setzt den aktuellen Zeitpunkt anhand der Mausposition.

Definiert in Zeile 254 der Datei GUITimeline.cpp.

Hier ist ein Graph, der zeigt, was diese Funktion aufruft:



Hier ist ein Graph der zeigt, wo diese Funktion aufgerufen wird:

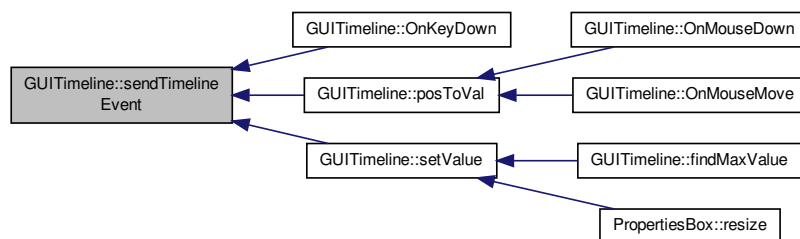


#### 8.16.4.16 void GUITimeline::sendTimelineEvent ( ) [private]

Löst ein `wxEVT_TIMELINE_CHANGE`-Event aus.

Definiert in Zeile 52 der Datei `GUITimeline.cpp`.

Hier ist ein Graph der zeigt, wo diese Funktion aufgerufen wird:



#### 8.16.4.17 void GUITimeline::setMarked ( int pos, bool state )

Markiert/De-markiert einen bestimmten Zeitpunkt.

Parameter

<i>pos</i>	Index des zu setzenden Zeitpunkts.
<i>state</i>	Status des Punktes (markiert - true, nicht markiert - false).

Definiert in Zeile 476 der Datei `GUITimeline.cpp`.

#### 8.16.4.18 void GUITimeline::setMarkerList ( vector< int > \* mlist )

Setzt die Liste der markierten Stellen.

Parameter

<i>mlist</i>	Liste mit den Indices der markierten stellen.
--------------	---

Definiert in Zeile 471 der Datei GUITimeline.cpp.

Hier ist ein Graph der zeigt, wo diese Funktion aufgerufen wird:



#### 8.16.4.19 void GUITimeline::setMarkers ( vector< int > \* *mlist* )

Markiert eine Liste von Indices.

Parameter

<i>mlist</i>	Liste aller zu markierenden Indices.
--------------	--------------------------------------

Definiert in Zeile 521 der Datei GUITimeline.cpp.

#### 8.16.4.20 void GUITimeline::setMaxValue ( int *val* )

Setzt den maximal auswählbaren Index.

Definiert in Zeile 458 der Datei GUITimeline.cpp.

Hier ist ein Graph der zeigt, wo diese Funktion aufgerufen wird:

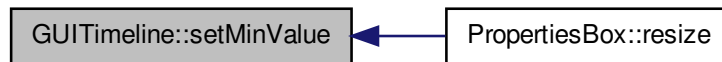


#### 8.16.4.21 void GUITimeline::setMinValue ( int *val* )

Setzt den minimal auswählbaren Index.

Definiert in Zeile 462 der Datei GUITimeline.cpp.

Hier ist ein Graph der zeigt, wo diese Funktion aufgerufen wird:



#### 8.16.4.22 void GUITimeline::setNameList ( vector< string > \* *namelist* )

Setzt die Liste der Namen für die jeweiligen Indices der Zeitpunkte.

Parameter

<i>namelist</i>	Liste mit einem Namen für jeden Index.
-----------------	--

Definiert in Zeile 466 der Datei GUITimeline.cpp.

Hier ist ein Graph der zeigt, wo diese Funktion aufgerufen wird:

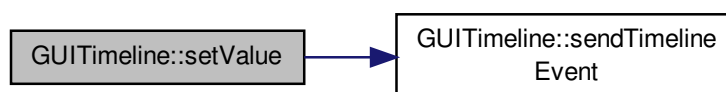


#### 8.16.4.23 void GUITimeline::setValue ( int *val* )

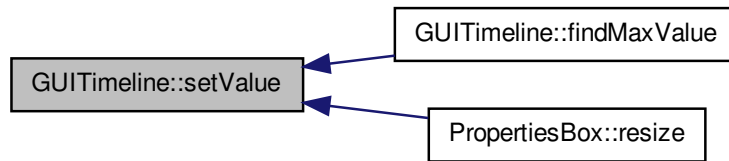
Setzt den Index des aktuell ausgewählten Zeitpunkts.

Definiert in Zeile 439 der Datei GUITimeline.cpp.

Hier ist ein Graph, der zeigt, was diese Funktion aufruft:



Hier ist ein Graph der zeigt, wo diese Funktion aufgerufen wird:



### 8.16.5 Dokumentation der Datenelemente

#### 8.16.5.1 `float GUITimeline::delta_v_view` [private]

Verschiebung der Ansicht.

Definiert in Zeile 217 der Datei `GUITimeline.h`.

#### 8.16.5.2 `vector<int>* GUITimeline::markers` [private]

Liste der markierten Zeitpunkte.

Definiert in Zeile 232 der Datei `GUITimeline.h`.

#### 8.16.5.3 `int GUITimeline::maxdigits` [private]

Maximale Anzahl an anzuzeigenden Nachkommastellen.

Definiert in Zeile 207 der Datei `GUITimeline.h`.

#### 8.16.5.4 `int GUITimeline::maxvalue` [private]

Der größte anzuzeigende Zeitpunkt.

Definiert in Zeile 197 der Datei `GUITimeline.h`.

#### 8.16.5.5 `int GUITimeline::minvalue` [private]

Der kleinste anzuzeigende Zeitpunkt.

Definiert in Zeile 202 der Datei `GUITimeline.h`.

#### 8.16.5.6 `vector<string>* GUITimeline::names` [private]

Liste der Zeitpunktnamen.

Definiert in Zeile 227 der Datei `GUITimeline.h`.

#### 8.16.5.7 `int GUITimeline::prev_mouse_x` [private]

Zwischenspeicher für die vorherige horizontale Mausposition.

Definiert in Zeile 222 der Datei GUITimeline.h.

**8.16.5.8** `int GUITimeline::value` `[private]`

Der Index des aktuell ausgewählten Zeitpunkts.

Definiert in Zeile 192 der Datei GUITimeline.h.

**8.16.5.9** `float GUITimeline::zoom` `[private]`

Aktueller Zoomfaktor.

Definiert in Zeile 212 der Datei GUITimeline.h.

Die Dokumentation für diese Klasse wurde erzeugt aufgrund der Dateien:

- [/daten/Projekte/eclipse\\_workspace/simpleanalyzer-gui/src/GUI/GUITimeline.h](#)
- [/daten/Projekte/eclipse\\_workspace/simpleanalyzer-gui/src/GUI/GUITimeline.cpp](#)

## 8.17 Importer Klassenreferenz

Importieren von 3D-Modell (.obj) und Sensordaten (.tsd oder .sd).

```
#include <Importer.h>
```

### Öffentliche Methoden

- [Importer](#) ()  
*Der Konstruktor.*
- [ObjectData::ObjectDataStatus LoadSensorData](#) (string filename, [ObjectData](#) \*data)  
*Lädt von einfache Sensordaten (ohne Zeit) und Verknüpft diese mit dem Objekt.*
- [ObjectData::ObjectDataStatus LoadTimedData](#) (string filename, [ObjectData](#) \*data)  
*Lädt zeitgesteuerte Sensordaten und Verknüpft diese mit dem Objekt.*
- [ObjectData::ObjectDataStatus ImportObj](#) (string filename, [ObjectData](#) \*data)  
*Lädt Objektdaten aus einer .obj-Datei.*
- `virtual ~Importer` ()  
*Der Destruktor.*

### 8.17.1 Ausführliche Beschreibung

Importieren von 3D-Modell (.obj) und Sensordaten (.tsd oder .sd).

Definiert in Zeile 18 der Datei Importer.h.

### 8.17.2 Beschreibung der Konstruktoren und Destrukturen

#### 8.17.2.1 `Importer::Importer ( )`

Der Konstruktor.

Definiert in Zeile 24 der Datei Importer.cpp.



### 8.17.2.2 Importer::~~Importer ( ) [virtual]

Der Destruktor.

Definiert in Zeile 504 der Datei Importer.cpp.

## 8.17.3 Dokumentation der Elementfunktionen

### 8.17.3.1 ObjectData::ObjectDataStatus Importer::ImportObj ( string filename, ObjectData \* data )

Lädt Objektdaten aus einer .obj-Datei.

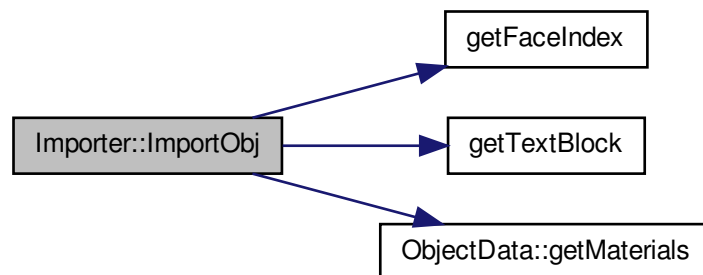
Das Objekt ist zwar schon im Speicher erstellt, wird aber erst durch diese Methode mit Daten gefüllt.

#### Rückgabe

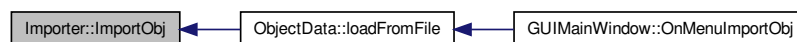
Der Fehlercode.

Definiert in Zeile 83 der Datei Importer.cpp.

Hier ist ein Graph, der zeigt, was diese Funktion aufruft:



Hier ist ein Graph, der zeigt, wo diese Funktion aufgerufen wird:



### 8.17.3.2 ObjectData::ObjectDataStatus Importer::LoadSensorData ( string filename, ObjectData \* data )

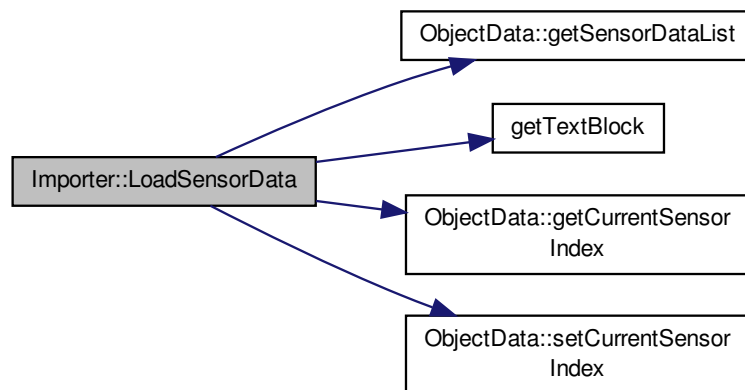
Lädt von einfache Sensordaten (ohne Zeit) und Verknüpft diese mit dem Objekt.

**Rückgabe**

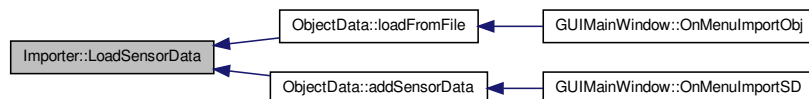
Der Fehlercode.

Definiert in Zeile 342 der Datei Importer.cpp.

Hier ist ein Graph, der zeigt, was diese Funktion aufruft:



Hier ist ein Graph der zeigt, wo diese Funktion aufgerufen wird:



### 8.17.3.3 `ObjectData::ObjectDataStatus` `Importer::LoadTimedData ( string filename, ObjectData * data )`

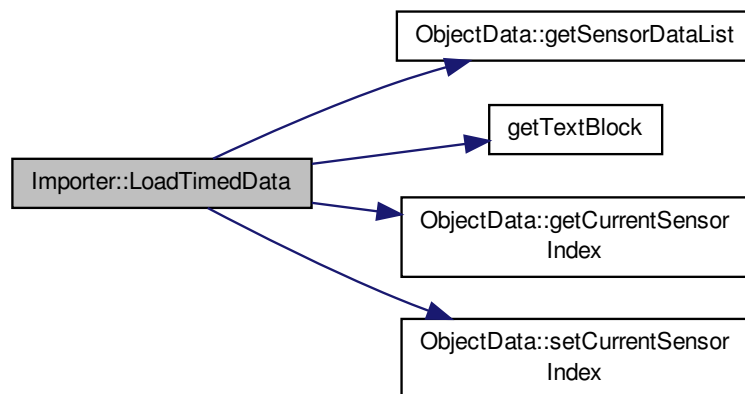
Lädt zeitgesteuerte Sensordaten und Verknüpft diese mit dem Objekt.

**Rückgabe**

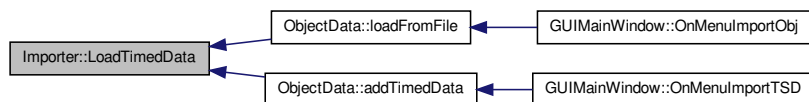
Der Fehlercode.

Definiert in Zeile 416 der Datei Importer.cpp.

Hier ist ein Graph, der zeigt, was diese Funktion aufruft:



Hier ist ein Graph der zeigt, wo diese Funktion aufgerufen wird:



Die Dokumentation für diese Klasse wurde erzeugt aufgrund der Dateien:

- [/daten/Projekte/eclipse\\_workspace/simpleanalyzer-gui/src/fileIO/Importer.h](#)
- [/daten/Projekte/eclipse\\_workspace/simpleanalyzer-gui/src/fileIO/Importer.cpp](#)

## 8.18 Interpolator Klassenreferenz

2- und 3-dimensionale Inter-/Extrapolation

```
#include <Interpolator.h>
```

**Öffentliche Typen**

- enum [InterpolationMode](#) { [LINEAR](#), [LOGARITHMIC](#) }

*Der Typ der verwendeten Interpolationsfunktion.*

## Öffentliche Methoden

- [Interpolator](#) ()  
*Der Konstruktor.*
- double [interpolateTri](#) ([Triangle](#) \*tri, [Vector3D](#) \*pos, double \*values)  
*Ermittelt den Wert für einen beliebigen Punkt in einer Ebene.*
- double [interpolateTet](#) ([Tetrahedron](#) \*tet, [Vector3D](#) \*pos, double \*values)  
*Ermittelt den Wert für einen beliebigen Punkt im Raum.*
- void [setMode](#) ([InterpolationMode](#) mode)  
*Setzt den verwendeten Interpolationsmodus (die Interpolationsfunktion).*
- virtual [~Interpolator](#) ()  
*Der Destruktor.*

## Private Attribute

- [InterpolationMode](#) mode  
*Der verwendete Interpolationsmodus bzw.*

### 8.18.1 Ausführliche Beschreibung

2- und 3-dimensionale Inter-/Extrapolation

Klasse zur Bi- und Trilinearen Inter-/Extrapolation, wobei die Interpolationsfunktion zwischen zwei Werten entweder linear oder logarithmisch sein kann ([InterpolationMode](#)).

Definiert in Zeile 20 der Datei [Interpolator.h](#).

### 8.18.2 Dokumentation der Aufzählungstypen

#### 8.18.2.1 `enum Interpolator::InterpolationMode`

Der Typ der verwendeten Interpolationsfunktion.

Aufzählungswerte

***LINEAR***

***LOGARITHMIC***

Definiert in Zeile 25 der Datei [Interpolator.h](#).

### 8.18.3 Beschreibung der Konstruktoren und Destrukturen

#### 8.18.3.1 `Interpolator::Interpolator ( )`

Der Konstruktor.

Definiert in Zeile 12 der Datei [Interpolator.cpp](#).

#### 8.18.3.2 `Interpolator::~~Interpolator ( )` [virtual]

Der Destruktor.

Definiert in Zeile 315 der Datei [Interpolator.cpp](#).

### 8.18.4 Dokumentation der Elementfunktionen

#### 8.18.4.1 `double Interpolator::interpolateTet ( Tetrahedron * tet, Vector3D * pos, double * values )`

Ermittelt den Wert für einen beliebigen Punkt im Raum.

Dabei wird wie bei der bilinearen Interpolation ([http://en.wikipedia.org/wiki/Trilinear\\_interpolation](http://en.wikipedia.org/wiki/Trilinear_interpolation)) vorgegangen, es kann jedoch auch eine logarithmische Interpolationsfunktion verwendet werden.

##### Parameter

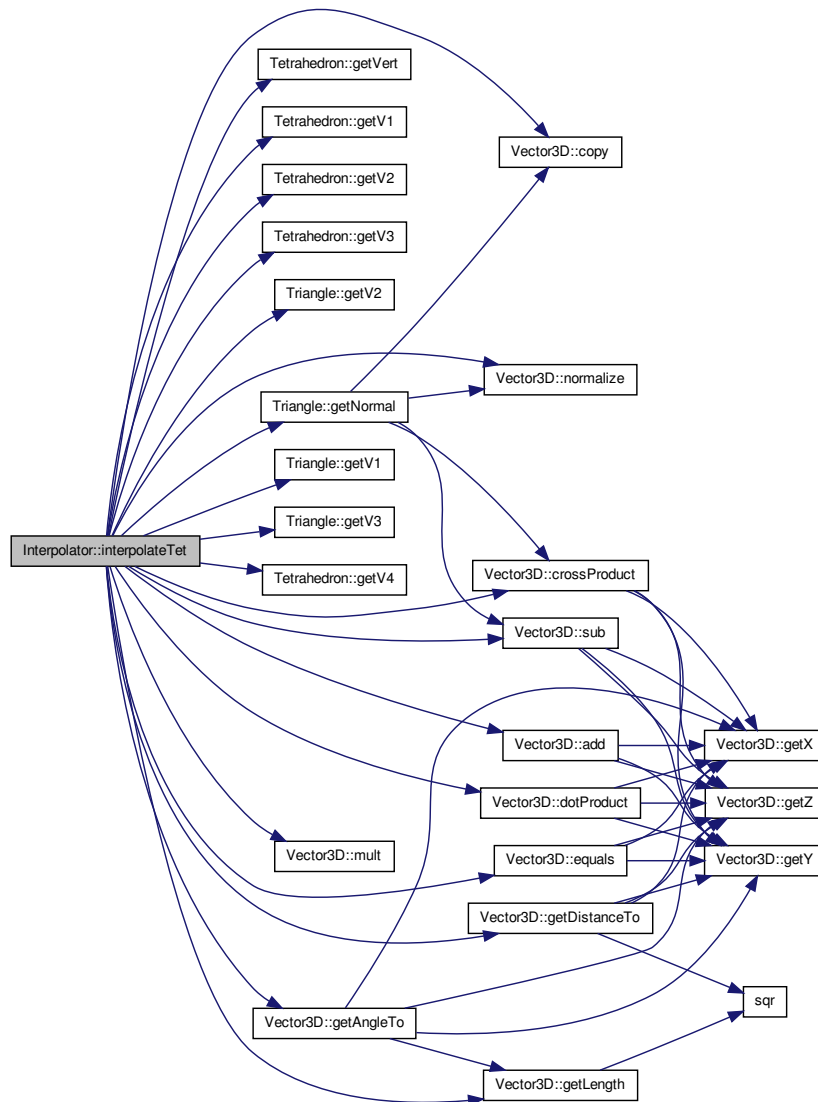
<i>tet</i>	Tetraeder, durch den die Punkte für die gegebenen Werte gegeben sind.
<i>pos</i>	Position des Punktes, für den der Wert ermittelt werden soll.
<i>values</i>	Die Werte, die den Punkten des Tetraeders entsprechen. Dabei ist values[0] der Wert des ersten Punktes des Tetraeders, values[1] der Zweite usw.

##### Rückgabe

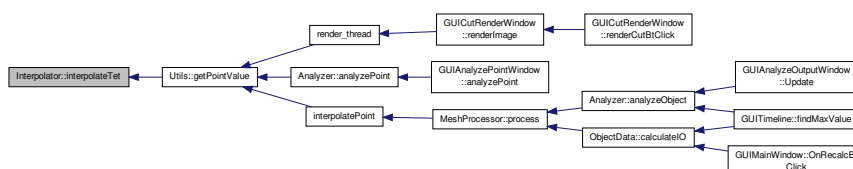
Der Wert für den angegebenen Punkt (*pos*).

Definiert in Zeile 149 der Datei Interpolator.cpp.

Hier ist ein Graph, der zeigt, was diese Funktion aufruft:



Hier ist ein Graph der zeigt, wo diese Funktion aufgerufen wird:



#### 8.18.4.2 double Interpolator::interpolateTri ( Triangle \* tri, Vector3D \* pos, double \* values )

Ermittelt den Wert für einen beliebigen Punkt in einer Ebene.

Dabei wird wie bei der bilinearen Interpolation ([http://en.wikipedia.org/wiki/Bilinear\\_interpolation](http://en.wikipedia.org/wiki/Bilinear_interpolation)) vorgegangen, es kann jedoch auch eine logarithmische Interpolationsfunktion verwendet werden.

#### Parameter

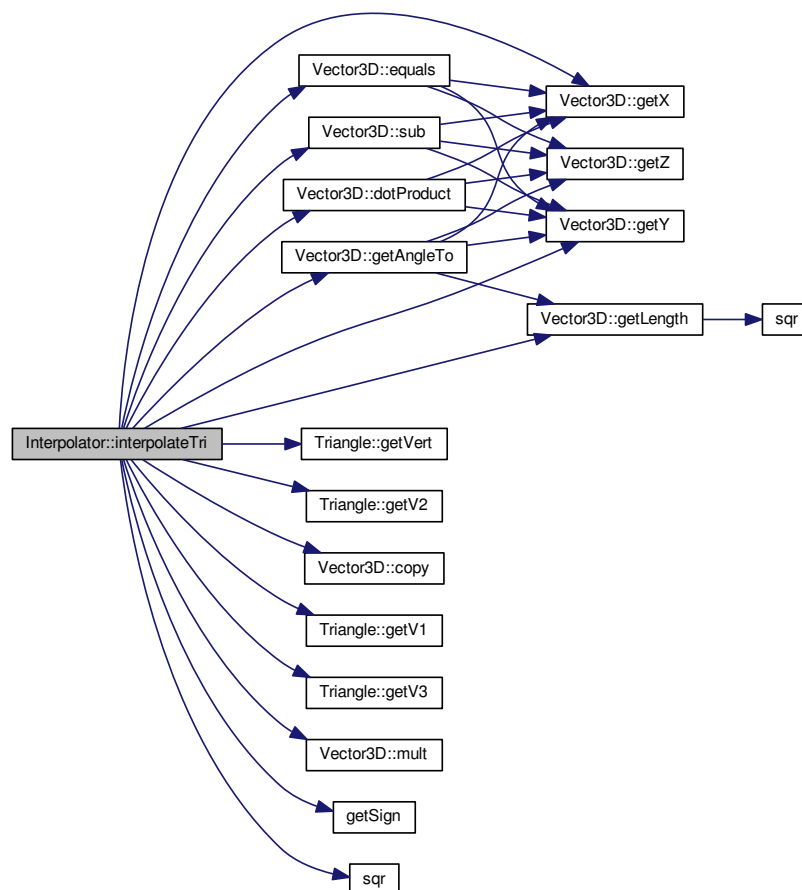
<i>tri</i>	Dreieck, durch das die Ebene beschrieben wird.
<i>pos</i>	Position des Punktes, für den der Wert ermittelt werden soll.
<i>values</i>	Die Werte, die den Punkten des Dreiecks entsprechen. Dabei ist values[0] der Wert des ersten Punktes des Dreiecks, values[0] der Zweite usw.

#### Rückgabe

Der Wert für den angegebenen Punkt (pos).

Definiert in Zeile 24 der Datei Interpolator.cpp.

Hier ist ein Graph, der zeigt, was diese Funktion aufruft:

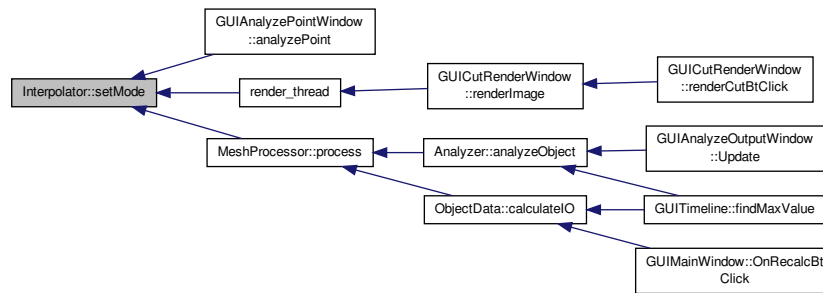


#### 8.18.4.3 void Interpolator::setMode ( InterpolationMode mode )

Setzt den verwendeten Interpolationsmodus (die Interpolationsfunktion).

Definiert in Zeile 16 der Datei Interpolator.cpp.

Hier ist ein Graph der zeigt, wo diese Funktion aufgerufen wird:



## 8.18.5 Dokumentation der Datenelemente

### 8.18.5.1 InterpolationMode Interpolator::mode [private]

Der verwendete Interpolationsmodus bzw.

die Interpolationsfunktion.

Definiert in Zeile 69 der Datei Interpolator.h.

Die Dokumentation für diese Klasse wurde erzeugt aufgrund der Dateien:

- /daten/Projekte/eclipse\_workspace/simpleanalyzer-gui/src/libraries/interpolate/Interpolator.h
- /daten/Projekte/eclipse\_workspace/simpleanalyzer-gui/src/libraries/interpolate/Interpolator.cpp

## 8.19 ObjectData::MaterialData Strukturreferenz

Die Daten eines Materials.

```
#include <ObjectData.h>
```

### Öffentliche Attribute

- string **name**  
*Der Name des Materials.*
- Interpolator::InterpolationMode **interpolation\_mode**  
*Der zu verwendende Interpolationsmodus.*
- tetgenio \* **tetgeninput**  
*Originalgeometrie im Tetgen-Format (s.*
- tetgenio \* **tetgenoutput**  
*Durch Zerlegung erstellte Geometrie im Tetgen-Format (s.*
- bool \* **extrapolated**  
*Liste, die für jeden Punkt in der aktuellen Geometrie angibt, ob er extra- (true) oder interpoliert (false) ist.*
- float **color** [3]  
*Die Farbe des Materials im RGB-Format.*
- double **density**  
*Die Dichte in  $\frac{kg}{m^3}$ .*
- double **specificheatcapacity**



- *Spezifische Wärmekapazität in  $\frac{kJ}{kg \cdot K}$ .*  
 • bool **visible**  
*Soll das Material angezeigt werden?*

### 8.19.1 Ausführliche Beschreibung

Die Daten eines Materials.

Definiert in Zeile 32 der Datei ObjectData.h.

### 8.19.2 Dokumentation der Datenelemente

#### 8.19.2.1 float ObjectData::MaterialData::color[3]

Die Farbe des Materials im RGB-Format.

Definiert in Zeile 38 der Datei ObjectData.h.

#### 8.19.2.2 double ObjectData::MaterialData::density

Die Dichte in  $\frac{kg}{m^3}$ .

Definiert in Zeile 39 der Datei ObjectData.h.

#### 8.19.2.3 bool\* ObjectData::MaterialData::extrapolated

Liste, die für jeden Punkt in der aktuellen Geometrie angibt, ob er extra- (true) oder interpoliert (false) ist.

Definiert in Zeile 37 der Datei ObjectData.h.

#### 8.19.2.4 Interpolator::InterpolationMode ObjectData::MaterialData::interpolation\_mode

Der zu verwendende Interpolationsmodus.

Definiert in Zeile 34 der Datei ObjectData.h.

#### 8.19.2.5 string ObjectData::MaterialData::name

Der Name des Materials.

Definiert in Zeile 33 der Datei ObjectData.h.

#### 8.19.2.6 double ObjectData::MaterialData::specificheatcapacity

Spezifische Wärmekapazität in  $\frac{kJ}{kg \cdot K}$ .

Definiert in Zeile 40 der Datei ObjectData.h.

#### 8.19.2.7 tetgenio\* ObjectData::MaterialData::tetgeninput

Originalgeometrie im Tetgen-Format (s.

Tetgen Dokumentation)

Definiert in Zeile 35 der Datei ObjectData.h.

#### 8.19.2.8 tetgenio\* ObjectData::MaterialData::tetgenoutput

Durch Zerlegung erstellte Geometrie im Tetgen-Format (s. Tetgen Dokumentation)

Definiert in Zeile 36 der Datei ObjectData.h.

#### 8.19.2.9 bool ObjectData::MaterialData::visible

Soll das Material angezeigt werden?

Definiert in Zeile 41 der Datei ObjectData.h.

Die Dokumentation für diese Struktur wurde erzeugt aufgrund der Datei:

- /daten/Projekte/eclipse\_workspace/simpleanalyzer-gui/src/processing/[ObjectData.h](#)

## 8.20 Matrix3D Klassenreferenz

3x3-Matrixklasse mit Operationen.

```
#include <GeometryClasses.h>
```

### Öffentliche Methoden

- [Matrix3D](#) ()  
*Der Standardkonstruktor.*
- [Matrix3D](#) (double x1, double y1, double z1, double x2, double y2, double z2, double x3, double y3, double z3)  
*Erzeugt eine Matrix mit den gegebenen Elementen.*
- void [mult](#) ([Matrix3D](#) \*other)  
*Multipliziert die Matrix mit einer anderen Matrix.*
- [Vector3D](#) \* [mult](#) ([Vector3D](#) \*other)  
*Multipliziert die Matrix mit einem Vektor.*
- void [rotateX](#) (double angle)  
*Rotiert die Matrix um einen bestimmten Winkel auf der X-Achse.*
- void [rotateY](#) (double angle)  
*Rotiert die Matrix um einen bestimmten Winkel auf der Y-Achse.*
- void [rotateZ](#) (double angle)  
*Rotiert die Matrix um einen bestimmten Winkel auf der Z-Achse.*
- void [transpose](#) ()  
*Transponiert die Matrix.*
- void [print](#) ()  
*Gibt die Matrix auf dem cout-Stream aus.*

### Private Attribute

- double [elements](#) [9]  
*Die Elemente der Matrix.*

#### 8.20.1 Ausführliche Beschreibung

3x3-Matrixklasse mit Operationen.

Definiert in Zeile 151 der Datei GeometryClasses.h.

## 8.20.2 Beschreibung der Konstruktoren und Destruktoren

### 8.20.2.1 Matrix3D::Matrix3D ( )

Der Standardkonstruktor.

Erzeugt eine Standardmatrix:

$$\begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

Definiert in Zeile 141 der Datei GeometryClasses.cpp.

### 8.20.2.2 Matrix3D::Matrix3D ( double x1, double y1, double z1, double x2, double y2, double z2, double x3, double y3, double z3 )

Erzeugt eine Matrix mit den gegebenen Elementen.

$$\begin{pmatrix} x1 & y1 & z1 \\ x2 & y2 & z2 \\ x3 & y3 & z3 \end{pmatrix}$$

Definiert in Zeile 153 der Datei GeometryClasses.cpp.

## 8.20.3 Dokumentation der Elementfunktionen

### 8.20.3.1 void Matrix3D::mult ( Matrix3D \* other )

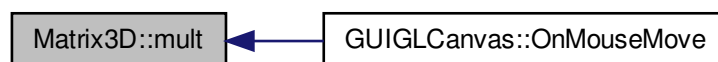
Multipliziert die Matrix mit einer anderen Matrix.

Parameter

<i>other</i>	Die Matrix, mit der multipliziert werden soll.
--------------	--

Definiert in Zeile 168 der Datei GeometryClasses.cpp.

Hier ist ein Graph der zeigt, wo diese Funktion aufgerufen wird:



#### 8.20.3.2 `Vector3D * Matrix3D::mult ( Vector3D * other )`

Multipliziert die Matrix mit einem Vektor.

## Parameter

<i>other</i>	Der Vektor, mit dem multipliziert werden soll.
--------------	--

## Rückgabe

Der durch die Multiplikation entstandene Vektor. Der zurückgegebene Vektor muss manuell mit delete Freigeben werden!

Definiert in Zeile 186 der Datei GeometryClasses.cpp.

Hier ist ein Graph, der zeigt, was diese Funktion aufruft:



## 8.20.3.3 void Matrix3D::print ( )

Gibt die Matrix auf dem cout-Stream aus.

Definiert in Zeile 230 der Datei GeometryClasses.cpp.

8.20.3.4 void Matrix3D::rotateX ( double *angle* )

Rotiert die Matrix um einen bestimmten Winkel auf der X-Achse.

## Parameter

<i>angle</i>	Der Winkel, um den rotiert werden soll in RAD.
--------------	--

Definiert in Zeile 197 der Datei GeometryClasses.cpp.

Hier ist ein Graph der zeigt, wo diese Funktion aufgerufen wird:

8.20.3.5 void Matrix3D::rotateY ( double *angle* )

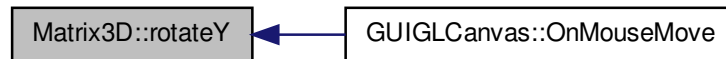
Rotiert die Matrix um einen bestimmten Winkel auf der Y-Achse.

## Parameter

<i>angle</i>	Der Winkel, um den rotiert werden soll in RAD.
--------------	--

Definiert in Zeile 204 der Datei GeometryClasses.cpp.

Hier ist ein Graph der zeigt, wo diese Funktion aufgerufen wird:



#### 8.20.3.6 void Matrix3D::rotateZ ( double *angle* )

Rotiert die Matrix um einen bestimmten Winkel auf der Z-Achse.

## Parameter

<i>angle</i>	Der Winkel, um den rotiert werden soll in RAD.
--------------	--

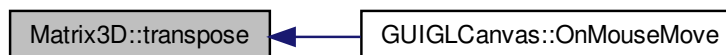
Definiert in Zeile 211 der Datei GeometryClasses.cpp.

#### 8.20.3.7 void Matrix3D::transpose ( )

Transponiert die Matrix.

Definiert in Zeile 218 der Datei GeometryClasses.cpp.

Hier ist ein Graph der zeigt, wo diese Funktion aufgerufen wird:



## 8.20.4 Dokumentation der Datenelemente

### 8.20.4.1 double Matrix3D::elements[9] [private]

Die Elemente der Matrix.

Definiert in Zeile 221 der Datei GeometryClasses.h.

Die Dokumentation für diese Klasse wurde erzeugt aufgrund der Dateien:

- [/daten/Projekte/eclipse\\_workspace/simpleanalyzer-gui/src/libraries/interpolate/GeometryClasses.h](#)
- [/daten/Projekte/eclipse\\_workspace/simpleanalyzer-gui/src/libraries/interpolate/GeometryClasses.cpp](#)

## 8.21 MeshProcessor Klassenreferenz

Errechnet die Temperaturverteilung für ein Objekt.

```
#include <MeshProcessor.h>
```

### Öffentliche Methoden

- [MeshProcessor](#) ()

*Der Konstruktor.*

- void [process](#) ([ObjectData](#) \*object)

*Berechnet die Temperaturverteilung für ein Objekt.*

- virtual [~MeshProcessor](#) ()

*Der Destruktor.*

### 8.21.1 Ausführliche Beschreibung

Errechnet die Temperaturverteilung für ein Objekt.

Definiert in Zeile 16 der Datei MeshProcessor.h.

### 8.21.2 Beschreibung der Konstruktoren und Destrukturen

#### 8.21.2.1 MeshProcessor::MeshProcessor ( )

Der Konstruktor.

Definiert in Zeile 17 der Datei MeshProcessor.cpp.

#### 8.21.2.2 MeshProcessor::~~MeshProcessor ( ) [virtual]

Der Destruktor.

Definiert in Zeile 85 der Datei MeshProcessor.cpp.

### 8.21.3 Dokumentation der Elementfunktionen

#### 8.21.3.1 void MeshProcessor::process ( ObjectData \* object )

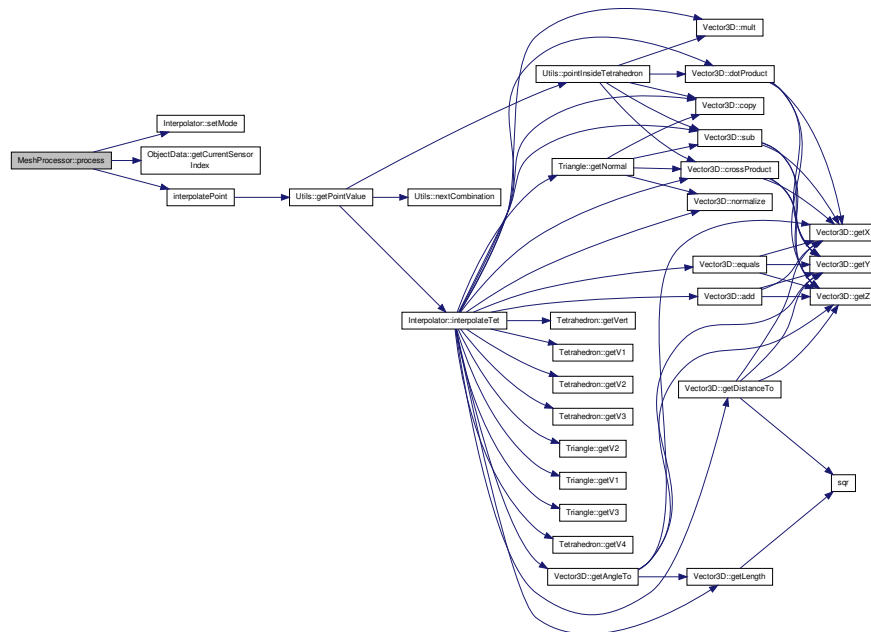
Berechnet die Temperaturverteilung für ein Objekt.

Parameter

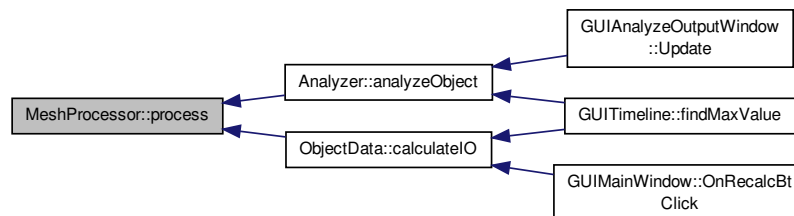
<i>object</i>	Das Objekt, für das die Temperaturverteilung ermittelt werden soll.
---------------	---

Definiert in Zeile 35 der Datei MeshProcessor.cpp.

Hier ist ein Graph, der zeigt, was diese Funktion aufruft:



Hier ist ein Graph der zeigt, wo diese Funktion aufgerufen wird:



Die Dokumentation für diese Klasse wurde erzeugt aufgrund der Dateien:

- /daten/Projekte/eclipse\_workspace/simpleanalyzer-gui/src/processing/[MeshProcessor.h](#)
- /daten/Projekte/eclipse\_workspace/simpleanalyzer-gui/src/processing/[MeshProcessor.cpp](#)

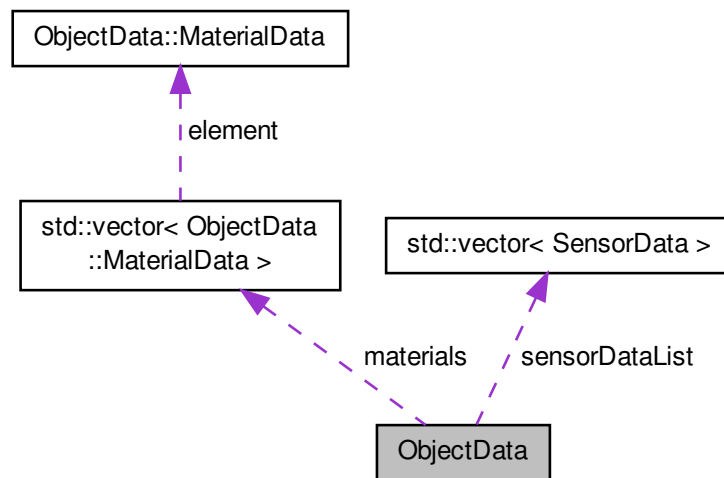
## 8.22 ObjectData Klassenreferenz

Die Daten eines Versuchsobjekts.

```
#include <ObjectData.h>
```



Zusammengehörigkeiten von ObjectData:



## Klassen

- struct `MaterialData`  
*Die Daten eines Materials.*

## Öffentliche Typen

- enum `ObjectDataStatus` {  
`OD_SUCCESS = 1, OD_FAILURE, OD_LOAD_ALREADY_LOADED, OD_LOAD_INVALID_FILE,`  
`OD_LOAD_INVALID_SENSOR_FILE` }  
*Status einer die Objektdaten betreffenden Aktion.*

## Öffentliche Methoden

- `ObjectData ()`  
*Der Konstruktor.*
- `int loadFromFile (wxString &path)`  
*Lädt ein Objekt und erste Sensordaten.*
- `int addSensorData (wxString &path)`  
*Lädt einfache Sensordaten und verknüpft sie mit dem Objekt.*
- `int addTimedData (wxString &path)`  
*Lädt zeitbezogene Sensordaten und verknüpft sie mit dem Objekt.*
- `int calculateIO ()`  
*Zerlegt das Objekt in Tetraeder (Schnittstelle zur Tetgen-Bibliothek) und Berechnet die Temperaturverteilung für die aktuell ausgewählten Sensordaten (und den aktuelle ausgewählten Zeitpunkt).*
- `vector< MaterialData > * getMaterials ()`  
*Gibt eine Referenz auf die Liste der Materialien (mit Materialdaten) des Objekts zurück.*
- `double getMaxvolume ()`

- Gibt das maximale Tetraedervolumen für der Zerlegung zurück.*
- void `setMaxvolume` (double `maxvolume`)  
*Setzt das maximale Tetraedervolumen für der Zerlegung.*
- string `getName` ()  
*Gibt den Namen des Objekts zurück.*
- void `setName` (string `name`)  
*Setzt den Namen des Objekts.*
- double `getQuality` ()  
*Gibt die Qualitätseinstellung für die Tetraeder bei der Zerlegung (s.*
- void `setQuality` (double `quality`)  
*Setzt die Qualitätseinstellung für die Tetraeder bei der Zerlegung (s.*
- `vector< SensorData > * getSensorDataList` ()  
*Gibt eine Referenz auf die Sensordaten des Objekts zurück.*
- int `getCurrentSensorIndex` ()  
*Gibt den Index des aktuell verwendeten Sensordatensatzes zurück.*
- void `setCurrentSensorIndex` (int `currentSensorIndex`)  
*Setzt den Index des aktuell verwendeten Sensordatensatzes.*
- virtual `~ObjectData` ()  
*Der Destruktor.*

## Private Attribute

- int `current_sensor_index`  
*Index des aktuell verwendeten Sensordatensatzes.*
- string `name`  
*Name des Objekts.*
- double `maxvolume`  
*Maximales Volumen für Tetraeder bei der Zerlegung.*
- double `quality`  
*Qualität der Tetraeder bei der Zerlegung (s.*
- `vector< MaterialData > materials`  
*Liste der Materialien des Objekts.*
- `vector< SensorData > sensorDataList`  
*Liste der Sensordaten des Objekts.*

### 8.22.1 Ausführliche Beschreibung

Die Daten eines Versuchsobjekts.

Diese Klasse hält Objekteigenschaften, Materialien und Sensordaten eines untersuchten Objekts. Desweiteren stellt sie die Schnittstelle zur Tetgen-Bibliothek (<http://wias-berlin.de/software/tetgen/>) zum zerlegen des Objekts dar.

Definiert in Zeile 27 der Datei ObjectData.h.

### 8.22.2 Dokumentation der Aufzählungstypen

#### 8.22.2.1 enum `ObjectData::ObjectDataStatus`

Status einer die Objektdaten betreffenden Aktion.

## Aufzählungswerte

***OD\_SUCCESS******OD\_FAILURE******OD\_LOAD\_ALREADY\_LOADED******OD\_LOAD\_INVALID\_FILE******OD\_LOAD\_INVALID\_SENSOR\_FILE***

Definiert in Zeile 47 der Datei ObjectData.h.

**8.22.3 Beschreibung der Konstruktoren und Destruktoren****8.22.3.1 ObjectData::ObjectData ( )**

Der Konstruktor.

Definiert in Zeile 23 der Datei ObjectData.cpp.

**8.22.3.2 ObjectData::~~ObjectData ( ) [virtual]**

Der Destruktor.

Definiert in Zeile 200 der Datei ObjectData.cpp.

**8.22.4 Dokumentation der Elementfunktionen****8.22.4.1 int ObjectData::addSensorData ( wxString & path )**

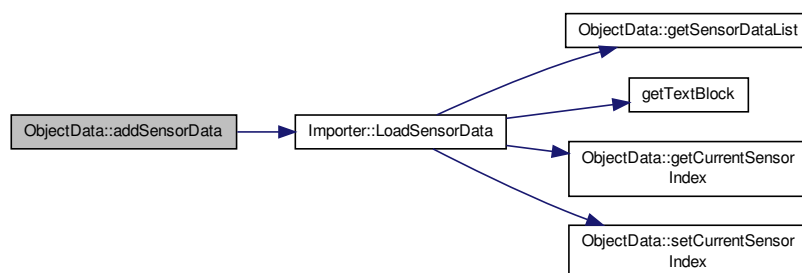
Lädt einfache Sensordaten und verknüpft sie mit dem Objekt.

**Parameter**

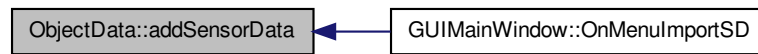
<i>path</i>	Pfad zur .sd-Datei.
-------------	---------------------

Definiert in Zeile 108 der Datei ObjectData.cpp.

Hier ist ein Graph, der zeigt, was diese Funktion aufruft:



Hier ist ein Graph der zeigt, wo diese Funktion aufgerufen wird:



#### 8.22.4.2 `int ObjectData::addTimedData ( wxString & path )`

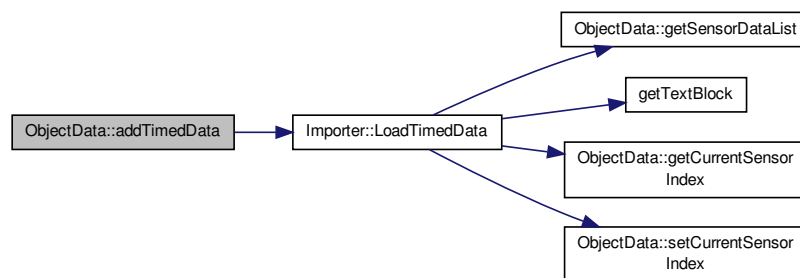
Lädt zeitbezogene Sensordaten und verknüpft sie mit dem Objekt.

Parameter

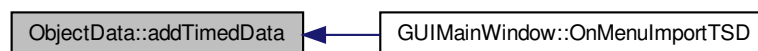
<i>path</i>	Pfad zur .tsd-Datei.
-------------	----------------------

Definiert in Zeile 114 der Datei `ObjectData.cpp`.

Hier ist ein Graph, der zeigt, was diese Funktion aufruft:



Hier ist ein Graph der zeigt, wo diese Funktion aufgerufen wird:

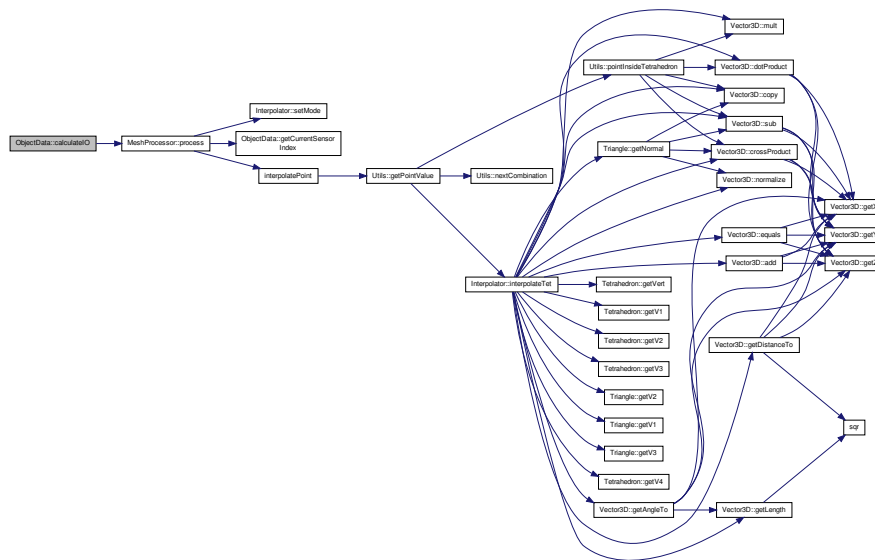


#### 8.22.4.3 `int ObjectData::calculateIO ( )`

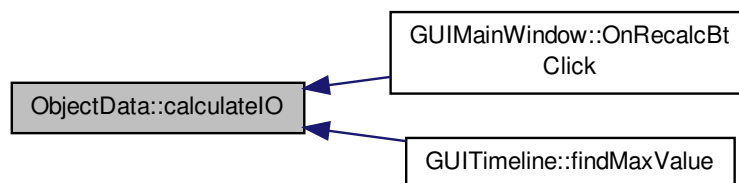
Zerlegt das Objekt in Tetraeder (Schnittstelle zur Tetgen-Bibliothek) und Berechnet die Temperaturverteilung für die aktuell ausgewählten Sensordaten (und den aktuelle ausgewählten Zeitpunkt).

Definiert in Zeile 120 der Datei `ObjectData.cpp`.

Hier ist ein Graph, der zeigt, was diese Funktion aufruft:



Hier ist ein Graph der zeigt, wo diese Funktion aufgerufen wird:

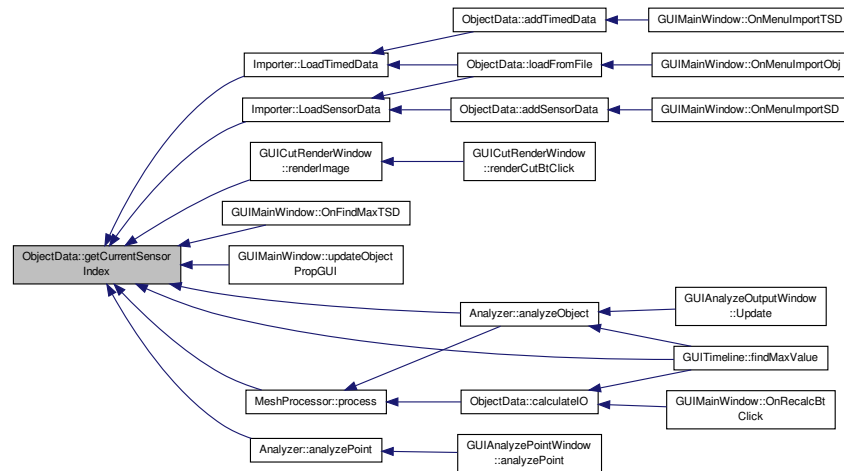


#### 8.22.4.4 int ObjectData::getCurrentSensorIndex ( )

Gibt den Index des aktuell verwendeten Sensordatensatzes zurück.

Definiert in Zeile 192 der Datei ObjectData.cpp.

Hier ist ein Graph der zeigt, wo diese Funktion aufgerufen wird:

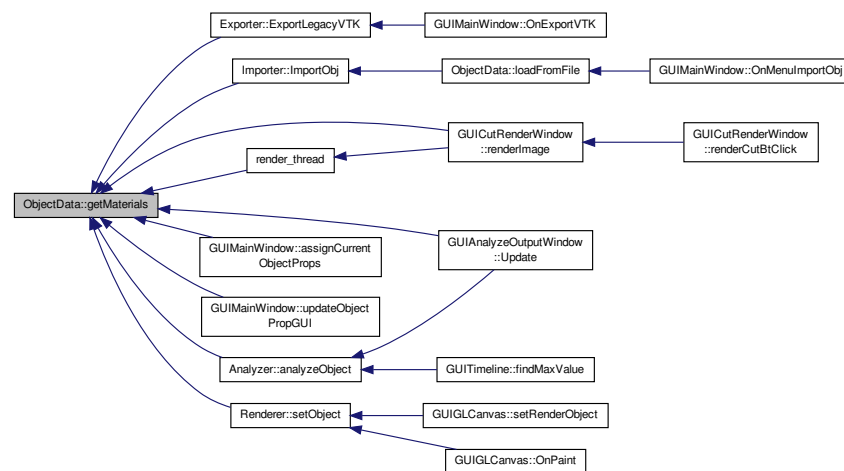


#### 8.22.4.5 `vector< ObjectData::MaterialData > * ObjectData::getMaterials ( )`

Gibt eine Referenz auf die Liste der Materialien (mit Materialdaten) des Objekts zurück.

Definiert in Zeile 160 der Datei ObjectData.cpp.

Hier ist ein Graph der zeigt, wo diese Funktion aufgerufen wird:

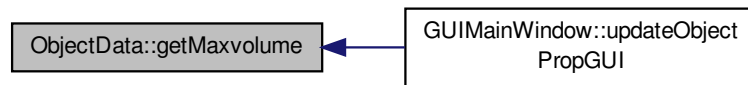


#### 8.22.4.6 `double ObjectData::getMaxvolume ( )`

Gibt das maximale Tetraedervolumen für der Zerlegung zurück.

Definiert in Zeile 164 der Datei ObjectData.cpp.

Hier ist ein Graph der zeigt, wo diese Funktion aufgerufen wird:

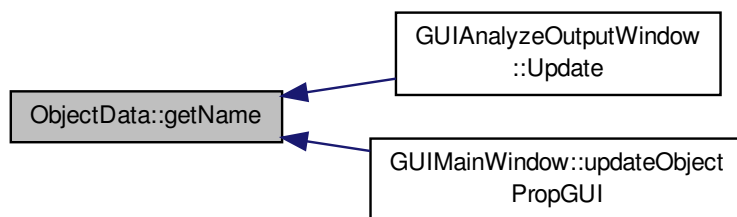


#### 8.22.4.7 `string ObjectData::getName ( )`

Gibt den Namen des Objekts zurück.

Definiert in Zeile 172 der Datei `ObjectData.cpp`.

Hier ist ein Graph der zeigt, wo diese Funktion aufgerufen wird:



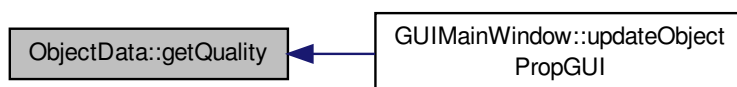
#### 8.22.4.8 `double ObjectData::getQuality ( )`

Gibt die Qualisätseinstellung für die Tetraeder bei der Zerlegung (s.

Tetgen Dokumentation) zurück.

Definiert in Zeile 180 der Datei `ObjectData.cpp`.

Hier ist ein Graph der zeigt, wo diese Funktion aufgerufen wird:

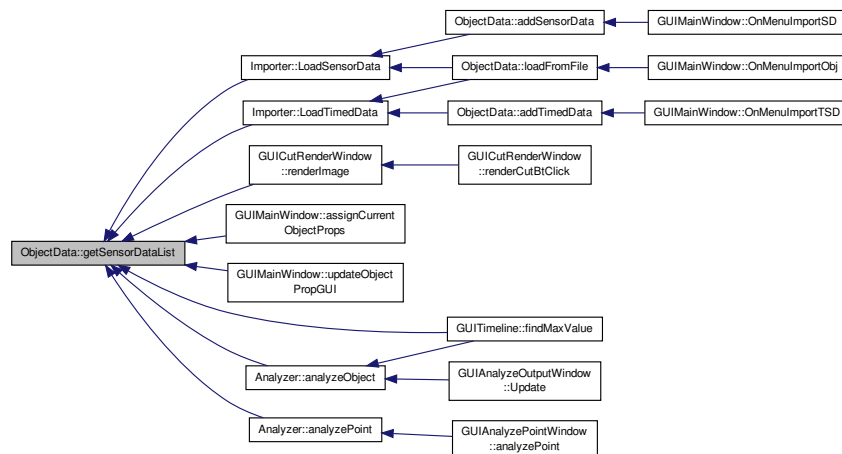


#### 8.22.4.9 `vector< SensorData > * ObjectData::getSensorDataList ( )`

Gibt eine Referenz auf die Sensordaten des Objekts zurück.

Definiert in Zeile 188 der Datei ObjectData.cpp.

Hier ist ein Graph der zeigt, wo diese Funktion aufgerufen wird:



#### 8.22.4.10 `int ObjectData::loadFromFile ( wxString & path )`

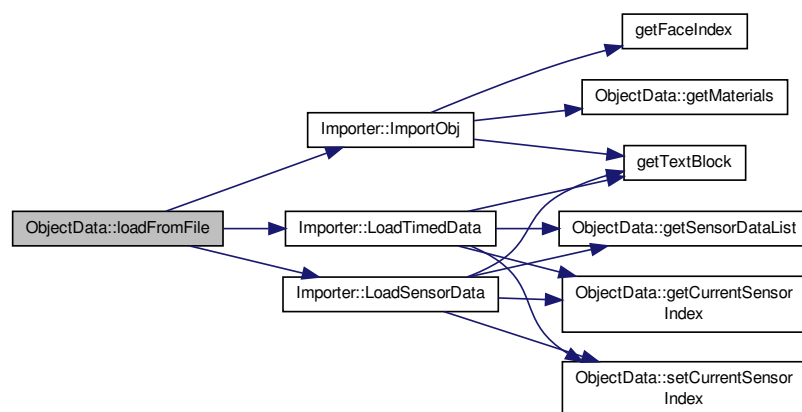
Lädt ein Objekt und erste Sensordaten.

Parameter

<i>path</i>	Pfad zur 3D-Modell(.obj)-Datei.
-------------	---------------------------------

Definiert in Zeile 33 der Datei ObjectData.cpp.

Hier ist ein Graph, der zeigt, was diese Funktion aufruft:





Hier ist ein Graph der zeigt, wo diese Funktion aufgerufen wird:

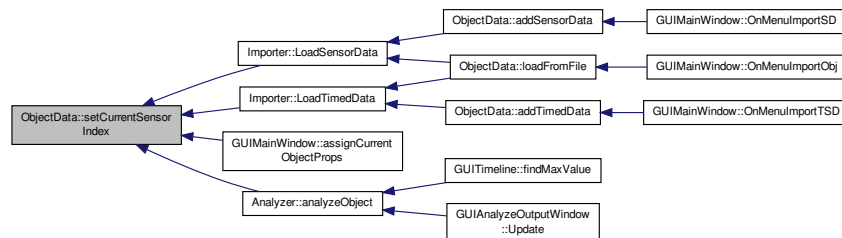


#### 8.22.4.11 void ObjectData::setCurrentSensorIndex ( int *currentSensorIndex* )

Setzt den Index des aktuell verwendeten Sensordatensatzes.

Definiert in Zeile 196 der Datei ObjectData.cpp.

Hier ist ein Graph der zeigt, wo diese Funktion aufgerufen wird:



#### 8.22.4.12 void ObjectData::setMaxvolume ( double *maxvolume* )

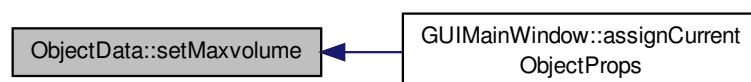
Setzt das maximale Tetraedervolumen für der Zerlegung.

Parameter

<i>maxvolume</i>	Maximales Tetraedervolumen.
------------------	-----------------------------

Definiert in Zeile 168 der Datei ObjectData.cpp.

Hier ist ein Graph der zeigt, wo diese Funktion aufgerufen wird:



#### 8.22.4.13 void ObjectData::setName ( string *name* )

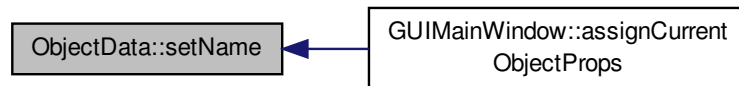
Setzt den Namen des Objekts.

## Parameter

<i>name</i>	Der neue Name des Objekts.
-------------	----------------------------

Definiert in Zeile 176 der Datei ObjectData.cpp.

Hier ist ein Graph der zeigt, wo diese Funktion aufgerufen wird:

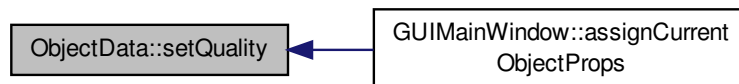


#### 8.22.4.14 void ObjectData::setQuality ( double *quality* )

Setzt die Qualitätseinstellung für die Tetraeder bei der Zerlegung (s. Tetgen Dokumentation).

Definiert in Zeile 184 der Datei ObjectData.cpp.

Hier ist ein Graph der zeigt, wo diese Funktion aufgerufen wird:



### 8.22.5 Dokumentation der Datenelemente

#### 8.22.5.1 int ObjectData::current\_sensor\_index [private]

Index des aktuell verwendeten Sensordatensatzes.

Definiert in Zeile 144 der Datei ObjectData.h.

#### 8.22.5.2 vector<MaterialData> ObjectData::materials [private]

Liste der Materialien des Objekts.

Definiert in Zeile 164 der Datei ObjectData.h.

#### 8.22.5.3 double ObjectData::maxvolume [private]

Maximales Volumen für Tetraeder bei der Zerlegung.

Definiert in Zeile 154 der Datei ObjectData.h.

8.22.5.4 `string ObjectData::name` [private]

Name des Objekts.

Definiert in Zeile 149 der Datei ObjectData.h.

8.22.5.5 `double ObjectData::quality` [private]

Qualität der Tetraeder bei der Zerlegung (s.

Tetgen Dokumentation).

Definiert in Zeile 159 der Datei ObjectData.h.

8.22.5.6 `vector<SensorData> ObjectData::sensorDataList` [private]

Liste der Sensordaten des Objekts.

Definiert in Zeile 169 der Datei ObjectData.h.

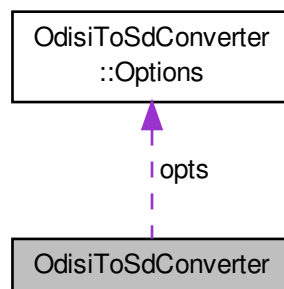
Die Dokumentation für diese Klasse wurde erzeugt aufgrund der Dateien:

- [/daten/Projekte/eclipse\\_workspace/simpleanalyzer-gui/src/processing/ObjectData.h](#)
- [/daten/Projekte/eclipse\\_workspace/simpleanalyzer-gui/src/processing/ObjectData.cpp](#)

## 8.23 OdisiToSdConverter Klassenreferenz

Konverter von ODiSI zu .tsd.

Zusammengehörigkeiten von OdisiToSdConverter:



### Klassen

- struct [Options](#)  
Struktur für die Programmeinstellungen.

### Öffentliche Methoden

- int [convert](#) (int argc, char \*argv[])

Liest die Programmargumente, die Konfiguration, die Sensordefinitionsdatei und die ODiSI-Datei um eine .tsd-Datei zu generieren, bzw.

## Geschützte Methoden

- bool `contains` (`std::vector`< string > &Vec, const string &Element)  
*Testet, ob sich ein String in einer Liste von Strings befindet.*
- bool `contains` (`std::vector`< int > &Vec, const int &Element)  
*Testet, ob sich eine Ganzzahl in einer Liste von Ganzzahlen befindet.*
- void `replaceAll` (string &str, const string from, const string to)  
*Ersetzt in einem String alle Vorkommen eines Teilstrings durch einen Anderen.*
- string `floattostr` (float val)  
*Wandelt eine Zeichenkette (String) um.*
- string `getTextBlock` (string data, int n)  
*Gibt den n-ten durch Leerzeichen abgetrennten Block aus einem String zurück.*
- void `parseLine` (string line, `vector`< float > \*out, `vector`< float > \*times, `vector`< int > \*debug\_positions, size\_t row\_count)  
*Sammelt Daten aus einer Textzeile (string).*
- bool `readConfiguration` (string binary\_path)  
*Liest und setzt die Programmkonfiguration aus der Konfigurationsdatei.*
- bool `parseArguments` (int argc, char \*argv[], string &def\_filename, string &data\_filename, string &out\_filename, string &err\_filename)  
*Wertet die Programmargumente aus.*
- bool `readSensorDefinitions` (string path, `vector`< float > &inlist, `vector`< float > &outlist, `vector`< float > &in\_x, `vector`< float > &out\_x, `vector`< bool > &dirlist)  
*Liest die Daten aus der Sensordefinitionsdatei.*
- bool `readInputFile` (string path, `vector`< `vector`< float > > &values, `vector`< `vector`< int > > &debug\_positions, `vector`< float > &times, `vector`< float > &lin\_positions)  
*Liest die Daten aus der Eingabedatei.*
- bool `writeOutputFile` (string path, string logpath, `vector`< `vector`< float > > &values, `vector`< `vector`< int > > &debug\_positions, `vector`< float > &times, `vector`< float > &lin\_positions, `vector`< float > &inlist, `vector`< float > &outlist, `vector`< float > &in\_x, `vector`< float > &out\_x, `vector`< bool > &dirlist)  
*Schreibt die Ausgabedatei.*

## Geschützte Attribute

- string `configpaths` [NUMBEROFPATHS]  
*Suchpfade für die Konfigurationsdatei.*
- struct `OdisiToSdConverter::Options` opts  
*Hält die verwendeten Programmeinstellungen.*

## Statische, geschützte Attribute

- static const int `NUMBEROFPATHS` = 3  
*Anzahl der Suchpfade für die Konfigurationsdatei.*

### 8.23.1 Ausführliche Beschreibung

Konverter von ODiSI zu .tsd.

Zusätzlich können Ausreißerwerte erkannt und eliminiert werden.

Definiert in Zeile 22 der Datei main.cpp.

### 8.23.2 Dokumentation der Elementfunktionen

**8.23.2.1** `bool OdisiToSdConverter::contains ( std::vector< string > & Vec, const string & Element )` `[inline]`,  
`[protected]`

Testet, ob sich ein String in einer Liste von Strings befindet.

Parameter

<i>Vec</i>	Liste der Strings.
<i>Element</i>	Der zu suchende String.

Rückgabe

true, wenn das Element gefunden wurde, sonst false.

Definiert in Zeile 65 der Datei main.cpp.

**8.23.2.2** `bool OdisiToSdConverter::contains ( std::vector< int > & Vec, const int & Element )` `[inline]`,  
`[protected]`

Testet, ob sich eine Ganzzahl in einer Liste von Ganzzahlen befindet.

Parameter

<i>Vec</i>	Liste der Ganzzahlen.
<i>Element</i>	Die zu suchende Ganzzahl.

Rückgabe

true, wenn das Element gefunden wurde, sonst false.

Definiert in Zeile 81 der Datei main.cpp.

**8.23.2.3** `int OdisiToSdConverter::convert ( int argc, char * argv[] )` `[inline]`

Liest die Programmargumente, die Konfiguration, die Sensordefinitionsdatei und die ODiSI-Datei um eine .tsd-Datei zu generieren, bzw.

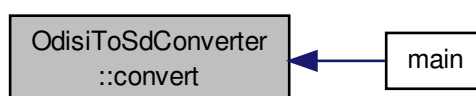
die Daten der ODiSI-Datei in eine .tsd-Datei umzuwandeln. Wird durch die Funktion [main\(\)](#) von außerhalb des Namespaces aufgerufen.

Parameter

<i>argc</i>	Anzahl der Programmargumente.
<i>argv</i>	Die Programmargumente.

Definiert in Zeile 908 der Datei main.cpp.

Hier ist ein Graph der zeigt, wo diese Funktion aufgerufen wird:



#### 8.23.2.4 `string OdisiToSdConverter::floattostr ( float val )` `[inline]`, `[protected]`

Wandelt eine Zeichenkette (String) um.

Parameter

<i>val</i>	Die umzuwandelnde Zahl.
------------	-------------------------

Rückgabe

Die Entsprechung der Zahl als String.

Definiert in Zeile 119 der Datei main.cpp.

#### 8.23.2.5 `string OdisiToSdConverter::getTextBlock ( string data, int n )` `[inline]`, `[protected]`

Gibt den n-ten durch Leerzeichen abgetrennten Block aus einem String zurück.

Parameter

<i>data</i>	Der Ausgangsstring.
<i>n</i>	Index des zu findenden Blocks.

Rückgabe

Der n-te durch Leerzeichen getrennte Teilstring. "" Bei ungültigem Index.

Definiert in Zeile 132 der Datei main.cpp.

#### 8.23.2.6 `bool OdisiToSdConverter::parseArguments ( int argc, char * argv[], string & def_filename, string & data_filename, string & out_filename, string & err_filename )` `[inline]`, `[protected]`

Wertet die Programmargumente aus.

Parameter

<i>argc</i>	Anzahl der Programmargumente.
<i>argv</i>	Die Programmargumente.
<i>def_filename</i>	Ausgabe für den Pfad zur Sensordefinitionsdatei.
<i>data_filename</i>	Ausgabe für den Pfad zur Eingabedatei.
<i>out_filename</i>	Ausgabe für den Pfad zur Ausgabedatei.
<i>err_filename</i>	Ausgabe für den Pfad zur Logdatei.

Rückgabe

Soll das Programm weiter ablaufen?

Definiert in Zeile 339 der Datei main.cpp.

#### 8.23.2.7 `void OdisiToSdConverter::parseLine ( string line, vector< float > * out, vector< float > * times, vector< int > * debug_positions, size_t row_count )` `[inline]`, `[protected]`

Sammelt Daten aus einer Textzeile (string).

## Parameter

<i>line</i>	Die zu untersuchende Textzeile.
<i>out</i>	Ausgabevariable für die Sensordaten der Zeile. Alle Spalten nach <code>opts.start_col</code> werden als Sensordatenspalten betrachtet. Wenn <code>row_count == opts.startrow</code> ist, werden statt der Sensordaten die Faserpositionen eingelesen!
<i>times</i>	Wenn nicht NULL, Ausgabevariable für den Zeitstempel der Zeile ( <code>opts.timecol</code> ). Der Zeitstempel wird an die übergebene Liste angehängt.
<i>debug_positions</i>	Wenn nicht NULL, Ausgabevariable für die Position der einzelnen Messwerte in der Datei. Diese Positionen werden in der Logdatei zur Fehlerkorrektur angegeben, um ein Wiederfinden der Werte für den Nutzer einfacher zu machen.
<i>row_count</i>	Nummer der aktuellen Zeile (Index+1).

Komma mit Punkt als Dezimaltrennzeichen ersetzen?

Definiert in Zeile 174 der Datei `main.cpp`.

#### 8.23.2.8 `bool OdisiToSdConverter::readConfiguration ( string binary_path ) [inline], [protected]`

Liest und setzt die Programmkonfiguration aus der Konfigurationsdatei.

## Parameter

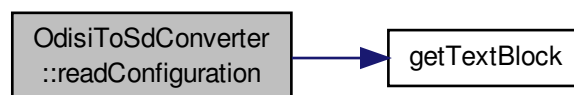
<i>binary_path</i>	Pfad zur Binärdatei.
--------------------	----------------------

## Rückgabe

War das Einlesen erfolgreich?

Definiert in Zeile 266 der Datei `main.cpp`.

Hier ist ein Graph, der zeigt, was diese Funktion aufruft:



#### 8.23.2.9 `bool OdisiToSdConverter::readInputFile ( string path, vector< vector< float > > & values, vector< vector< int > > & debug_positions, vector< float > & times, vector< float > & lin_positions ) [inline], [protected]`

Liest die Daten aus der Eingabedatei.

## Parameter

<i>path</i>	Der Pfad zur Eingabedatei.
<i>values</i>	Liste für die extrahierten Sensorwerte.

<i>times</i>	Liste für die Zeitstempel der Messwerte.
<i>debug_positions</i>	Liste für die Positionen der Messwerte in der Datei.
<i>lin_positions</i>	Liste für die Positionen auf der Faser.

#### Rückgabe

War das Einlesen erfolgreich?

Definiert in Zeile 587 der Datei main.cpp.

**8.23.2.10** `bool OdisiToSdConverter::readSensorDefinitions ( string path, vector< float > & inlist, vector< float > & outlist, vector< float > & in_x, vector< float > & out_x, vector< bool > & dirlist )` [inline], [protected]

Liest die Daten aus der Sensordefinitionsdatei.

#### Parameter

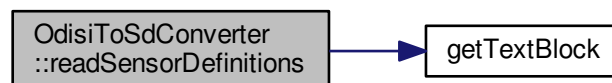
<i>path</i>	Pfad zur Binärdatei.
<i>inlist</i>	Liste für die Positionen der Fasereingänge auf der Faser.
<i>outlist</i>	Liste für die Positionen der Faserausgänge auf der Faser.
<i>in_x</i>	Liste für die X-Positionen der Fasereingänge.
<i>out_x</i>	Liste für die X-Positionen der Faserausgänge.
<i>dirlist</i>	Liste für die Richtungen der Faser zwischen Ein-und Ausgang bezüglich der Z-Richtung.

#### Rückgabe

War das Einlesen erfolgreich?

Definiert in Zeile 481 der Datei main.cpp.

Hier ist ein Graph, der zeigt, was diese Funktion aufruft:



**8.23.2.11** `void OdisiToSdConverter::replaceAll ( string & str, const string from, const string to )` [inline], [protected]

Ersetzt in einem String alle Vorkommen eines Teilstrings durch einen Anderen.

#### Parameter

<i>str</i>	Der zu durchsuchende String.
<i>from</i>	Der zu ersetzende Teilstring.



<i>to</i>	Der Teilstring, durch den ersetzt werden soll.
-----------	--

Definiert in Zeile 97 der Datei main.cpp.

**8.23.2.12** `bool OdisiToSdConverter::writeOutputFile ( string path, string logpath, vector< vector< float > > & values, vector< vector< int > > & debug_positions, vector< float > & times, vector< float > & lin_positions, vector< float > & inlist, vector< float > & outlist, vector< float > & in_x, vector< float > & out_x, vector< bool > & dirlist ) [inline], [protected]`

Schreibt die Ausgabedatei.

Parameter

<i>path</i>	Der Pfad zur Ausgabedatei.
<i>logpath</i>	Der Pfad zur Logdatei. Bei "" wird keine Logdatei angelegt.
<i>debug_positions</i>	Liste der Positionen der Sensorwerte in der Eingabedatei.
<i>values</i>	Die extrahierten Sensorwerte.
<i>times</i>	Zeitstempel der Datensätze.
<i>lin_positions</i>	Die Position der Messstellen auf der Faser.
<i>inlist</i>	Positionen der Fasereingänge auf der Faser.
<i>outlist</i>	Positionen der Faserausgänge auf der Faser.
<i>in_x</i>	X-Positionen der Fasereingänge.
<i>out_x</i>	X-Positionen der Faserausgänge.
<i>dirlist</i>	Liste für die Richtungen der Faser zwischen Ein-und Ausgang bezüglich der Z-Richtung.

Rückgabe

War das Schreiben erfolgreich?

Definiert in Zeile 687 der Datei main.cpp.

### 8.23.3 Dokumentation der Datenelemente

**8.23.3.1** `string OdisiToSdConverter::configpaths[NUMBEROFPATHS] [protected]`

Initialisierung:

```
{
    "/etc/simpleanalyzer/odisitosd.conf",
    "/usr/local/share/simpleanalyzer/odisitosd.conf",
    "/usr/share/simpleanalyzer/odisitosd.conf" }
```

Suchpfade für die Konfigurationsdatei.

Das Verzeichnis der ausführbaren Datei wird immer geprüft.

Definiert in Zeile 33 der Datei main.cpp.

**8.23.3.2** `const int OdisiToSdConverter::NUMBEROFPATHS = 3 [static], [protected]`

Anzahl der Suchpfade für die Konfigurationsdatei.

Definiert in Zeile 27 der Datei main.cpp.

**8.23.3.3** `struct OdisiToSdConverter::Options OdisiToSdConverter::opts [protected]`

Hält die verwendeten Programmeinstellungen.

Die Dokumentation für diese Klasse wurde erzeugt aufgrund der Datei:

- [/daten/Projekte/eclipse\\_workspace/odisitosd/main.cpp](#)

## 8.24 CsvToSdConverter::Options Strukturreferenz

Strunktur für die Programmeinstellungen.

### Öffentliche Attribute

- `size_t` [start\\_col](#)  
*Index der Spalte, in der die ersten Sensordaten stehen.*
- `char` [separator](#)  
*Das verwendete Separatorzeichen.*
- `bool` [replace\\_comma\\_with\\_point](#)  
*Sollen Kommata durch Punkte ersetzt werdden?*
- `size_t` [timecol](#)  
*Index der Spalte, die die Zeitstempel enthält.*
- `size_t` [namecol](#)  
*Index der Spalte, die die Namen für die Datensätze enthält.*
- `int` [time\\_step\\_delta](#)  
*Schrittweite beim Auslesen der Sensordaten (nur jeder time\_step\_delta Zeitpunkt wird verwendet).*
- `long` [max\\_time](#)  
*Nur bis maximal zu diesem Zeitstempel auslesen.*
- `long` [min\\_time](#)  
*Ab diesem Zeitstempel auslesen.*

### 8.24.1 Ausführliche Beschreibung

Strunktur für die Programmeinstellungen.

Definiert in Zeile 38 der Datei main.cpp.

### 8.24.2 Dokumentation der Datenelemente

#### 8.24.2.1 `long` CsvToSdConverter::Options::max\_time

Nur bis maximal zu diesem Zeitstempel auslesen.

Definiert in Zeile 45 der Datei main.cpp.

#### 8.24.2.2 `long` CsvToSdConverter::Options::min\_time

Ab diesem Zeitstempel auslesen.

Definiert in Zeile 46 der Datei main.cpp.

#### 8.24.2.3 `size_t` CsvToSdConverter::Options::namecol

Index der Spalte, die die Namen für die Datensätze enthält.

Definiert in Zeile 43 der Datei main.cpp.

#### 8.24.2.4 bool CsvToSdConverter::Options::replace\_comma\_with\_point

Sollen Kommata durch Punkte ersetzt werden?

Definiert in Zeile 41 der Datei main.cpp.

#### 8.24.2.5 char CsvToSdConverter::Options::separator

Das verwendete Separatorzeichen.

Definiert in Zeile 40 der Datei main.cpp.

#### 8.24.2.6 size\_t CsvToSdConverter::Options::start\_col

Index der Spalte, in der die ersten Sensordaten stehen.

Definiert in Zeile 39 der Datei main.cpp.

#### 8.24.2.7 int CsvToSdConverter::Options::time\_step\_delta

Schrittweite beim Auslesen der Sensordaten (nur jeder time\_step\_delta Zeitpunkt wird verwendet).

Definiert in Zeile 44 der Datei main.cpp.

#### 8.24.2.8 size\_t CsvToSdConverter::Options::timecol

Index der Spalte, die die Zeitstempel enthält.

Definiert in Zeile 42 der Datei main.cpp.

Die Dokumentation für diese Struktur wurde erzeugt aufgrund der Datei:

- [/daten/Projekte/eclipse\\_workspace/csvtosd/main.cpp](#)

## 8.25 TsdMerger::Options Strukturreferenz

Struktur für die Programmeinstellungen.

### Öffentliche Attribute

- int [offset](#)  
*Ein zusätzlicher Versatz, der zu den Zeitstempeln der zweiten Datei addiert wird.*
- unsigned int [max\\_dt](#)  
*Maximale Zeitdifferenz zwischen den Zeitstempeln um die Datensätze zusammenführen zu können.*
- long int [delta](#)  
*Ein Versatz, der zu den Zeitstempeln der zweiten Datei addiert wird.*
- bool [auto\\_delta](#)  
*Delta automatisch aus der Differenz der jeweils ersten Zeitstempel der Eingabedateien ermitteln.*

### 8.25.1 Ausführliche Beschreibung

Struktur für die Programmeinstellungen.

Definiert in Zeile 27 der Datei mergetsd.cpp.

## 8.25.2 Dokumentation der Datenelemente

### 8.25.2.1 `bool TsdMerger::Options::auto_delta`

Delta automatisch aus der Differenz der jeweils ersten Zeitstempel der Eingabedateien ermitteln.

Definiert in Zeile 31 der Datei `mergetsd.cpp`.

### 8.25.2.2 `long int TsdMerger::Options::delta`

Ein Versatz, der zu den Zeitstempeln der zweiten Datei addiert wird.

Definiert in Zeile 30 der Datei `mergetsd.cpp`.

### 8.25.2.3 `unsigned int TsdMerger::Options::max_dt`

Maximale Zeitdifferenz zwischen den Zeitstempeln um die Datensätze zusammenführen zu können.

Definiert in Zeile 29 der Datei `mergetsd.cpp`.

### 8.25.2.4 `int TsdMerger::Options::offset`

Ein zusätzlicher Versatz, der zu den Zeitstempeln der zweiten Datei addiert wird.

Definiert in Zeile 28 der Datei `mergetsd.cpp`.

Die Dokumentation für diese Struktur wurde erzeugt aufgrund der Datei:

- [/daten/Projekte/eclipse\\_workspace/mergetsd/src/mergetsd.cpp](#)

## 8.26 `OdisiToSdConverter::Options` Strukturreferenz

Strunktur für die Programmeinstellungen.

### Öffentliche Attribute

- `size_t startrow`  
*Index der ersten Zeile in der Odisi-Datei, die Sensordaten enthält.*
- `char separator`  
*Das verwendete Separatorzeichen.*
- `bool replace_comma_with_point`  
*Sollen Kommata durch Punkte ersetzt werdden?*
- `size_t timecol`  
*Index der Spalte, die die Zeitstempel enthält.*
- `float error_threshold`  
*Maximal zulässige Differenz zum Vorgängerwert für einen gültigen Messwert bei der Fehlerkorrektur.*
- `int maxfwcount`  
*Maximale Schrittzahl zum finden eines gültigen Messwertes bei der Fehlerkorrektur.*
- `int tab_space_count`  
*Anzahl der Leerzeichen für TAB (Für die Positionsangabe in der Log-Datei).*
- `float height`  
*Höhe der Faserebene in m.*
- `float basetemp`

*Temperatur zu Beginn des Versuches (Die Odisi-Daten sind Differenzen zu dieser Anfangstemperatur).*

- float `objwidth`

*Position der Messwerte auf der X-Achse um diesen Wert verschieben.*

- bool `flipobj`

*Position auf der X-Achse spiegeln?*

- int `fiber_step_delta`

*Schrittweite beim Auslesen der Sensordaten (nur jeder fiber\_step\_delta Messpunkt auf der Faser wird verwendet).*

- int `time_step_delta`

*Schrittweite beim Auslesen der Sensordaten (nur jeder time\_step\_delta Zeitpunkt wird verwendet).*

- double `max_time`

*Nur bis maximal zu diesem Zeitstempel auslesen.*

- double `min_time`

*Ab diesem Zeitstempel auslesen.*

### 8.26.1 Ausführliche Beschreibung

Struktur für die Programmeinstellungen.

Definiert in Zeile 41 der Datei main.cpp.

### 8.26.2 Dokumentation der Datenelemente

#### 8.26.2.1 float OdisiToSdConverter::Options::basetemp

Temperatur zu Beginn des Versuches (Die Odisi-Daten sind Differenzen zu dieser Anfangstemperatur).

Definiert in Zeile 50 der Datei main.cpp.

#### 8.26.2.2 float OdisiToSdConverter::Options::error\_threshold

Maximal zulässige Differenz zum Vorgängerwert für einen gültigen Messwert bei der Fehlerkorrektur.

Definiert in Zeile 46 der Datei main.cpp.

#### 8.26.2.3 int OdisiToSdConverter::Options::fiber\_step\_delta

Schrittweite beim Auslesen der Sensordaten (nur jeder fiber\_step\_delta Messpunkt auf der Faser wird verwendet).

Definiert in Zeile 53 der Datei main.cpp.

#### 8.26.2.4 bool OdisiToSdConverter::Options::flipobj

Position auf der X-Achse spiegeln?

Definiert in Zeile 52 der Datei main.cpp.

#### 8.26.2.5 float OdisiToSdConverter::Options::height

Höhe der Faserebene in *m*.

Definiert in Zeile 49 der Datei main.cpp.

#### 8.26.2.6 double OdisiToSdConverter::Options::max\_time

Nur bis maximal zu diesem Zeitstempel auslesen.

Definiert in Zeile 55 der Datei main.cpp.

#### 8.26.2.7 int OdisiToSdConverter::Options::maxfwcount

Maximale Schrittzahl zum finden eines gültigen Messwertes bei der Fehlerkorrektur.

Definiert in Zeile 47 der Datei main.cpp.

#### 8.26.2.8 double OdisiToSdConverter::Options::min\_time

Ab diesem Zeitstempel auslesen.

Definiert in Zeile 56 der Datei main.cpp.

#### 8.26.2.9 float OdisiToSdConverter::Options::objwidth

Position der Messwerte auf der X-Achse um diesen Wert verschieben.

Definiert in Zeile 51 der Datei main.cpp.

#### 8.26.2.10 bool OdisiToSdConverter::Options::replace\_comma\_with\_point

Sollen Kommata durch Punkte ersetzt werden?

Definiert in Zeile 44 der Datei main.cpp.

#### 8.26.2.11 char OdisiToSdConverter::Options::separator

Das verwendete Separatorzeichen.

(Hier Leerzeichen)

Definiert in Zeile 43 der Datei main.cpp.

#### 8.26.2.12 size\_t OdisiToSdConverter::Options::startrow

Index der ersten Zeile in der Odisi-Datei, die Sensordaten enthält.

Definiert in Zeile 42 der Datei main.cpp.

#### 8.26.2.13 int OdisiToSdConverter::Options::tab\_space\_count

Anzahl der Leerzeichen für TAB (Für die Positionsangabe in der Log-Datei).

Definiert in Zeile 48 der Datei main.cpp.

#### 8.26.2.14 int OdisiToSdConverter::Options::time\_step\_delta

Schrittweite beim Auslesen der Sensordaten (nur jeder time\_step\_delta Zeitpunkt wird verwendet).

Definiert in Zeile 54 der Datei main.cpp.

## 8.26.2.15 size\_t OdisiToSdConverter::Options::timecol

Index der Spalte, die die Zeitstempel enthält.

Definiert in Zeile 45 der Datei main.cpp.

Die Dokumentation für diese Struktur wurde erzeugt aufgrund der Datei:

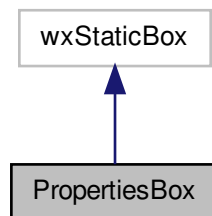
- /daten/Projekte/eclipse\_workspace/odisitosd/main.cpp

## 8.27 PropertiesBox Klassenreferenz

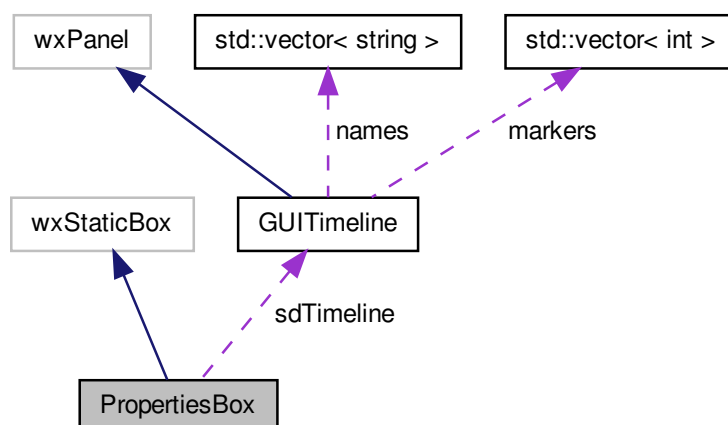
Oberfläche zum Verändern/Anzeigen der Eigenschaften eines Objekts.

```
#include <PropertiesBox.h>
```

Klassendiagramm für PropertiesBox:



Zusammengehörigkeiten von PropertiesBox:



## Öffentliche Methoden

- [PropertiesBox](#) (wxWindow \*parent)  
*Der Konstruktor.*
- void [resize](#) ()  
*Behandelt Größenänderungen und passt die Positionen der Komponenten an.*
- wxCheckBox \* [getAnalyzeMarkerCheckBox](#) ()  
*Gibt die Checkbox zum markieren des auf der Zeitleiste ausgewählten Zeitpunkts zurück.*
- wxCheckBox \* [getAutoUpdateCeckBox](#) ()  
*Gibt die Checkbox zum automatischen Neuberechnen der Temperaturverteilung nach einer Änderung an den Objekteigenschaften zurück.*
- wxButton \* [getClearAnalyzeMarkerBt](#) ()  
*Gibt den Button zum Löschen aller Markierungen (s.*
- wxTextCtrl \* [getSpecificHeatCapEdit](#) ()  
*Gibt das Eingabefeld für die spezifische Wärmekapazität zurück.*
- int [getCurrentMaterial](#) ()  
*Gibt den Index des aktuell ausgewählten Materials zurück.*
- void [setCurrentMaterial](#) (int index)  
*Setzt den Index des aktuell ausgewählten Materials.*
- wxTextCtrl \* [getDensityEdit](#) ()  
*Gibt das Eingabefeld für die Dichte des Materials zurück.*
- wxButton \* [getFindMaxBt](#) ()  
*Gibt den Button zum Suchen des maximums zwischen zwei markierten Zeitpunkten (s.*
- wxComboBox \* [getInterpolationModeList](#) ()  
*Gibt das Auswahlfeld für den zu verwendenden Interpolationsmodus zurück.*
- wxListBox \* [getMatListBox](#) ()  
*Gibt die Auswahlbox für das Material, dessen Eigenschaften angezeigt werden sollen, zurück.*
- wxTextCtrl \* [getMatNameEdit](#) ()  
*Gibt das Eingabefeld für den Materialnamen zurück.*
- wxStaticBox \* [getMatPropBox](#) ()  
*Gibt den Bereich, der die Materialeigenschaften enthält zurück.*
- wxTextCtrl \* [getMaxVolumeEdit](#) ()  
*Gibt das Eingabefeld für das maximale Tetraedervolumen zurück.*
- wxButton \* [getNextMarkerBt](#) ()  
*Gibt den Button zum Auswählen der nächsten Markierung (s.*
- wxTextCtrl \* [getObjNameEdit](#) ()  
*Gibt das Eingabefeld für den Objektnamen.*
- wxButton \* [getPrevMarkerBt](#) ()  
*Gibt den Button zum Auswählen der vorherigen Markierung (s.*
- wxTextCtrl \* [getQualityEdit](#) ()  
*Gibt das Eingabefeld für die Zerlegungsqualität des Modells (s.*
- wxButton \* [getRecalcButton](#) ()  
*Gibt den Button zum Neuberechnen der Temperaturverteilung zurück.*
- [GUITimeline](#) \* [getSdTimeline](#) ()  
*Gibt die Die Zeitleiste für zeitbezogene Sensordaten zurück.*
- wxComboBox \* [getSensorDataList](#) ()  
*Gibt das Auswahlfeld für den zu verwendenden Sensordatensatz zurück.*
- wxStaticText \* [getUpToDateLbl](#) ()  
*Gibt die Beschriftungskomponente für die Warnung bei geänderten Objekteigenschaften zurück.*
- virtual [~PropertiesBox](#) ()  
*Der Destruktor.*



## Private Attribute

- wxButton \* [recalcButton](#)  
*Button zum Neuberechnen der Temperaturverteilung.*
- wxStaticText \* [objNameLbl](#)  
*Beschriftung für das Objektnamen-Eingabefeld.*
- wxTextCtrl \* [objNameEdit](#)  
*Eingabefeld für den Objektnamen.*
- wxStaticText \* [matNameLbl](#)  
*Beschriftung für das Materialnamen-Eingabefeld.*
- wxTextCtrl \* [matNameEdit](#)  
*Eingabefeld für den Materialnamen.*
- wxStaticText \* [upToDateLbl](#)  
*Beschriftung für die Warnung bei geänderten Objekteigenschaften.*
- wxStaticText \* [maxVolumeLbl](#)  
*Beschriftung für das max.*
- wxTextCtrl \* [maxVolumeEdit](#)  
*Eingabefeld für das maximale Tetraedervolumen.*
- wxStaticText \* [qualityLbl](#)  
*Beschriftung für das Zerlegungsqualität-Eingabefeld.*
- wxTextCtrl \* [qualityEdit](#)  
*Eingabefeld für die Zerlegungsqualität des Modells (s.*
- wxStaticText \* [sensorDataLbl](#)  
*Beschriftung für das Sensordatensatz-Auswahlfeld.*
- wxComboBox \* [sensorDataList](#)  
*Auswahlfeld für den zu verwendenden Sensordatensatz.*
- wxListBox \* [matListBox](#)  
*Auswahlbox für das Material, dessen Eigenschaften angezeigt werden sollen.*
- wxStaticText \* [matListBoxLbl](#)  
*Beschriftung für die Materialauswahl-Box.*
- wxStaticBox \* [matPropBox](#)  
*Bereich, der die Materialeigenschaften enthält.*
- wxComboBox \* [interpolationModeList](#)  
*Auswahlfeld für den zu verwendenden Interpolationsmodus.*
- wxStaticText \* [interpolationModeLbl](#)  
*Beschriftung für das Interpolationsmodus-Auswahlfeld.*
- wxTextCtrl \* [densityEdit](#)  
*Eingabefeld für die Dichte des Materials.*
- wxStaticText \* [densityLbl](#)  
*Beschriftung für das Dichte-Eingabefeld.*
- wxTextCtrl \* [specificHeatCapEdit](#)  
*Eingabefeld für die spezifische Wärmekapazität.*
- wxStaticText \* [specificHeatCapLbl](#)  
*Beschriftung für das Wärmekapazitäts-Eingabefeld.*
- [GUITimeline](#) \* [sdTimeline](#)  
*Die Zeitleiste für zeitbezogene Sensordaten.*
- wxCheckBox \* [analyzeMarkerCheckBox](#)  
*Checkbox zum markieren des auf der Zeitleiste ausgewählten Zeitpunkts.*
- wxButton \* [findMaxBt](#)  
*Button zum Suchen des maximums zwischen zwei markierten Zeitpunkten (s.*
- wxButton \* [clearAnalyzeMarkerBt](#)

*Button zum Löschen aller Markierungen (s.*

- wxButton \* [nextMarkerBt](#)

*Button zum Auswählen der nächsten Markierung (s.*

- wxButton \* [prevMarkerBt](#)

*Button zum Auswählen der vorherigen Markierung (s.*

- wxCheckBox \* [autoUpdateCeckBox](#)

*Checkbox zum automatischen Neuberechnen der Temperaturverteilung nach einer Änderung an den Objekteigenschaften.*

- int [current\\_material](#)

*Index des aktuell ausgewählten Materials.*

### 8.27.1 Ausführliche Beschreibung

Oberfläche zum Verändern/Anzeigen der Eigenschaften eines Objekts.

Diese Klasse verwaltet nur das Layout des Objekteigenschaften-Bereichs. Die Funktionalität wird in [GUIMain-Window](#) behandelt.

Definiert in Zeile 19 der Datei PropertiesBox.h.

### 8.27.2 Beschreibung der Konstruktoren und Destruktoren

#### 8.27.2.1 PropertiesBox::PropertiesBox ( wxWindow \* *parent* )

Der Konstruktor.

Parameter

<i>parent</i>	Die übergeordnete Komponente.
---------------	-------------------------------

Definiert in Zeile 19 der Datei PropertiesBox.cpp.

#### 8.27.2.2 PropertiesBox::~~PropertiesBox ( ) [virtual]

Der Destruktor.

Definiert in Zeile 224 der Datei PropertiesBox.cpp.

### 8.27.3 Dokumentation der Elementfunktionen

#### 8.27.3.1 wxCheckBox \* PropertiesBox::getAnalyzeMarkerCheckBox ( )

Gibt die Checkbox zum markieren des auf der Zeitleiste ausgewählten Zeitpunkts zurück.

Definiert in Zeile 141 der Datei PropertiesBox.cpp.

#### 8.27.3.2 wxCheckBox \* PropertiesBox::getAutoUpdateCeckBox ( )

Gibt die Checkbox zum automatischen Neuberechnen der Temperaturverteilung nach einer Änderung an den Objekteigenschaften zurück.

Definiert in Zeile 145 der Datei PropertiesBox.cpp.

**8.27.3.3 wxButton \* PropertiesBox::getClearAnalyzeMarkerBt ( )**

Gibt den Button zum Löschen aller Markierungen (s. [GUITimeline](#)) zurück.

Definiert in Zeile 149 der Datei PropertiesBox.cpp.

**8.27.3.4 int PropertiesBox::getCurrentMaterial ( )**

Gibt den Index des aktuell ausgewählten Materials zurück.

Definiert in Zeile 157 der Datei PropertiesBox.cpp.

**8.27.3.5 wxTextCtrl \* PropertiesBox::getDensityEdit ( )**

Gibt das Eingabefeld für die Dichte des Materials zurück.

Definiert in Zeile 164 der Datei PropertiesBox.cpp.

**8.27.3.6 wxButton \* PropertiesBox::getFindMaxBt ( )**

Gibt den Button zum Suchen des maximums zwischen zwei markierten Zeitpunkten (s. [GUITimeline](#)) zurück.

Definiert in Zeile 168 der Datei PropertiesBox.cpp.

**8.27.3.7 wxComboBox \* PropertiesBox::getInterpolationModeList ( )**

Gibt das Auswahlfeld für den zu verwendenden Interpolationsmodus zurück.

Definiert in Zeile 172 der Datei PropertiesBox.cpp.

**8.27.3.8 wxListBox \* PropertiesBox::getMatListBox ( )**

Gibt die Auswahlbox für das Material, dessen Eigenschaften angezeigt werden sollen, zurück.

Definiert in Zeile 176 der Datei PropertiesBox.cpp.

**8.27.3.9 wxTextCtrl \* PropertiesBox::getMatNameEdit ( )**

Gibt das Eingabefeld für den Materialnamen zurück.

Definiert in Zeile 180 der Datei PropertiesBox.cpp.

**8.27.3.10 wxStaticBox \* PropertiesBox::getMatPropBox ( )**

Gibt den Bereich, der die Materialeigenschaften enthält zurück.

Definiert in Zeile 184 der Datei PropertiesBox.cpp.

**8.27.3.11 wxTextCtrl \* PropertiesBox::getMaxVolumeEdit ( )**

Gibt das Eingabefeld für das maximale Tetraedervolumen zurück.

Definiert in Zeile 188 der Datei PropertiesBox.cpp.

**8.27.3.12 wxButton \* PropertiesBox::getNextMarkerBt ( )**

Gibt den Button zum Auswählen der nächsten Markierung (s. [GUITimeline](#)) zurück.

Definiert in Zeile 192 der Datei PropertiesBox.cpp.

**8.27.3.13 wxTextCtrl \* PropertiesBox::getObjNameEdit ( )**

Gibt das Eingabefeld für den Objektnamen.  
zurück.

Definiert in Zeile 196 der Datei PropertiesBox.cpp.

**8.27.3.14 wxButton \* PropertiesBox::getPrevMarkerBt ( )**

Gibt den Button zum Auswählen der vorherigen Markierung (s. [GUITimeline](#)) zurück.

Definiert in Zeile 200 der Datei PropertiesBox.cpp.

**8.27.3.15 wxTextCtrl \* PropertiesBox::getQualityEdit ( )**

Gibt das Eingabefeld für die Zerlegungsqualität des Modells (s. Tetgen-dokumentation für weitere Informationen) zurück.

Definiert in Zeile 204 der Datei PropertiesBox.cpp.

**8.27.3.16 wxButton \* PropertiesBox::getRecalcButton ( )**

Gibt den Button zum Neuberechnen der Temperaturverteilung zurück.

Definiert in Zeile 208 der Datei PropertiesBox.cpp.

**8.27.3.17 GUITimeline \* PropertiesBox::getSdTimeline ( )**

Gibt die Die Zeitleiste für zeitbezogene Sensordaten zurück.

Definiert in Zeile 212 der Datei PropertiesBox.cpp.

**8.27.3.18 wxComboBox \* PropertiesBox::getSensorDataList ( )**

Gibt das Auswahlfeld für den zu verwendenden Sensordatensatz zurück.

Definiert in Zeile 216 der Datei PropertiesBox.cpp.

**8.27.3.19 wxTextCtrl \* PropertiesBox::getSpecificHeatCapEdit ( )**

Gibt das Eingabefeld für die spezifische Wärmekapazität zurück.

Definiert in Zeile 153 der Datei PropertiesBox.cpp.

### 8.27.3.20 wxStaticText \* PropertiesBox::getUpToDateLbl ( )

Gibt die Beschriftungskomponente für die Warnung bei geänderten Objekteigenschaften zurück.

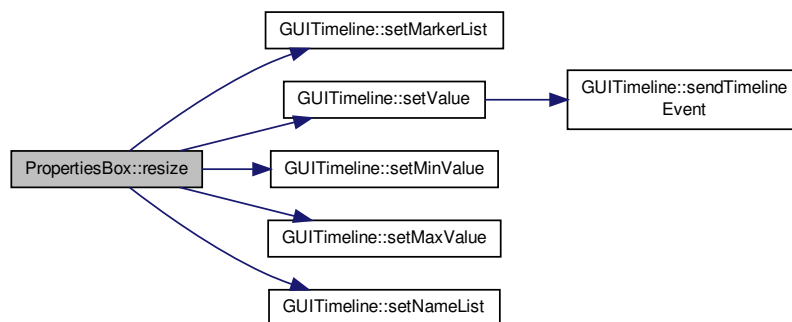
Definiert in Zeile 220 der Datei PropertiesBox.cpp.

### 8.27.3.21 void PropertiesBox::resize ( )

Behandelt Größenänderungen und passt die Positionen der Komponenten an.

Definiert in Zeile 72 der Datei PropertiesBox.cpp.

Hier ist ein Graph, der zeigt, was diese Funktion aufruft:



### 8.27.3.22 void PropertiesBox::setCurrentMaterial ( int index )

Setzt den Index des aktuell ausgewählten Materials.

Parameter

<i>index</i>	Index des auszuwählenden Materials.
--------------	-------------------------------------

Definiert in Zeile 160 der Datei PropertiesBox.cpp.

## 8.27.4 Dokumentation der Datenelemente

### 8.27.4.1 wxCheckBox\* PropertiesBox::analyzeMarkerCheckBox [private]

Checkbox zum markieren des auf der Zeitleiste ausgewählten Zeitpunkts.

Dieser Zeitpunkt wird dann im Analysedaten-Übersichtsfenster ([GUIAnalyzeOutputWindow](#)) angezeigt.

Definiert in Zeile 258 der Datei PropertiesBox.h.

### 8.27.4.2 wxCheckBox\* PropertiesBox::autoUpdateCeckBox [private]

Checkbox zum automatischen Neuberechnen der Temperaturverteilung nach einer Änderung an den Objekteigenschaften.

Definiert in Zeile 283 der Datei PropertiesBox.h.

#### 8.27.4.3 `wxButton* PropertiesBox::clearAnalyzeMarkerBt` [private]

Button zum Löschen aller Markierungen (s. [GUITimeline](#)).

Definiert in Zeile 268 der Datei PropertiesBox.h.

#### 8.27.4.4 `int PropertiesBox::current_material` [private]

Index des aktuell ausgewählten Materials.

Definiert in Zeile 288 der Datei PropertiesBox.h.

#### 8.27.4.5 `wxTextCtrl* PropertiesBox::densityEdit` [private]

Eingabefeld für die Dichte des Materials.

Definiert in Zeile 232 der Datei PropertiesBox.h.

#### 8.27.4.6 `wxStaticText* PropertiesBox::densityLbl` [private]

Beschriftung für das Dichte-Eingabefeld.

Definiert in Zeile 237 der Datei PropertiesBox.h.

#### 8.27.4.7 `wxButton* PropertiesBox::findMaxBt` [private]

Button zum Suchen des maximums zwischen zwei markierten Zeitpunkten (s. [GUITimeline](#)).

Definiert in Zeile 263 der Datei PropertiesBox.h.

#### 8.27.4.8 `wxStaticText* PropertiesBox::interpolationModeLbl` [private]

Beschriftung für das Interpolationsmodus-Auswahlfeld.

Definiert in Zeile 227 der Datei PropertiesBox.h.

#### 8.27.4.9 `wxComboBox* PropertiesBox::interpolationModeList` [private]

Auswahlfeld für den zu verwendenden Interpolationsmodus.

Definiert in Zeile 222 der Datei PropertiesBox.h.

#### 8.27.4.10 `wxListBox* PropertiesBox::matListBox` [private]

Auswahlbox für das Material, dessen Eigenschaften angezeigt werden sollen.

Definiert in Zeile 207 der Datei PropertiesBox.h.

#### 8.27.4.11 `wxStaticText* PropertiesBox::matListBoxLbl` [private]

Beschriftung für die Materialauswahl-Box.

Definiert in Zeile 212 der Datei PropertiesBox.h.

**8.27.4.12 wxTextCtrl\* PropertiesBox::matNameEdit** [private]

Eingabefeld für den Materialnamen.

Definiert in Zeile 167 der Datei PropertiesBox.h.

**8.27.4.13 wxStaticText\* PropertiesBox::matNameLbl** [private]

Beschriftung für das Materialnamen-Eingabefeld.

Definiert in Zeile 162 der Datei PropertiesBox.h.

**8.27.4.14 wxStaticBox\* PropertiesBox::matPropBox** [private]

Bereich, der die Materialeigenschaften enthält.

Definiert in Zeile 217 der Datei PropertiesBox.h.

**8.27.4.15 wxTextCtrl\* PropertiesBox::maxVolumeEdit** [private]

Eingabefeld für das maximale Tetraedervolumen.

Definiert in Zeile 182 der Datei PropertiesBox.h.

**8.27.4.16 wxStaticText\* PropertiesBox::maxVolumeLbl** [private]

Beschriftung für das max.

Tetraedervolumen-Eingabefeld.

Definiert in Zeile 177 der Datei PropertiesBox.h.

**8.27.4.17 wxButton\* PropertiesBox::nextMarkerBt** [private]

Button zum Auswählen der nächsten Markierung (s. [GUITimeline](#)).

Definiert in Zeile 273 der Datei PropertiesBox.h.

**8.27.4.18 wxTextCtrl\* PropertiesBox::objNameEdit** [private]

Eingabefeld für den Objektnamen.

Definiert in Zeile 157 der Datei PropertiesBox.h.

**8.27.4.19 wxStaticText\* PropertiesBox::objNameLbl** [private]

Beschriftung für das Objektnamen-Eingabefeld.

Definiert in Zeile 152 der Datei PropertiesBox.h.

**8.27.4.20 wxButton\* PropertiesBox::prevMarkerBt** [private]

Button zum Auswählen der vorherigen Markierung (s. [GUITimeline](#)).

Definiert in Zeile 278 der Datei PropertiesBox.h.

**8.27.4.21 wxTextCtrl\* PropertiesBox::qualityEdit [private]**

Eingabefeld für die Zerlegungsqualität des Modells (s. Tetgen-dokumentation für weitere Informationen).  
Definiert in Zeile 192 der Datei PropertiesBox.h.

**8.27.4.22 wxStaticText\* PropertiesBox::qualityLbl [private]**

Beschriftung für das Zerlegungsqualität-Eingabefeld.  
Definiert in Zeile 187 der Datei PropertiesBox.h.

**8.27.4.23 wxButton\* PropertiesBox::recalcButton [private]**

Button zum Neuberechnen der Temperaturverteilung.  
Definiert in Zeile 147 der Datei PropertiesBox.h.

**8.27.4.24 GUITimeline\* PropertiesBox::sdTimeline [private]**

Die Zeitleiste für zeitbezogene Sensordaten.  
Definiert in Zeile 252 der Datei PropertiesBox.h.

**8.27.4.25 wxStaticText\* PropertiesBox::sensorDataLbl [private]**

Beschriftung für das Sensordatensatz-Auswahlfeld.  
Definiert in Zeile 197 der Datei PropertiesBox.h.

**8.27.4.26 wxComboBox\* PropertiesBox::sensorDataList [private]**

Auswahlfeld für den zu verwendenden Sensordatensatz.  
Definiert in Zeile 202 der Datei PropertiesBox.h.

**8.27.4.27 wxTextCtrl\* PropertiesBox::specificHeatCapEdit [private]**

Eingabefeld für die spezifische Wärmekapazität.  
Definiert in Zeile 242 der Datei PropertiesBox.h.

**8.27.4.28 wxStaticText\* PropertiesBox::specificHeatCapLbl [private]**

Beschriftung für das Wärmekapazitäts-Eingabefeld.  
Definiert in Zeile 247 der Datei PropertiesBox.h.

**8.27.4.29 wxStaticText\* PropertiesBox::upToDateLbl [private]**

Beschriftung für die Warnung bei geänderten Objekteigenschaften.  
Definiert in Zeile 172 der Datei PropertiesBox.h.

Die Dokumentation für diese Klasse wurde erzeugt aufgrund der Dateien:



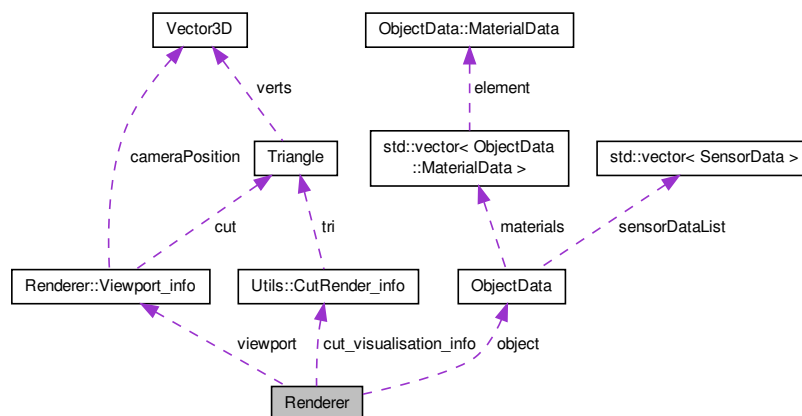
- [/daten/Projekte/eclipse\\_workspace/simpleanalyzer-gui/src/GUI/PropertiesBox.h](#)
- [/daten/Projekte/eclipse\\_workspace/simpleanalyzer-gui/src/GUI/PropertiesBox.cpp](#)

## 8.28 Renderer Klassenreferenz

Zeichnet den Inhalt der 3D-Fensters.

```
#include <Renderer.h>
```

Zusammengehörigkeiten von Renderer:



### Klassen

- struct [Viewport\\_info](#)  
*Informationen über die Ansicht des Modells (Virtuelle Kamera) und welche Elemente dargestellt werden.*

### Öffentliche Typen

- enum [RenderMode](#) { [RM\\_NONE](#) = 0, [RM\\_MATERIALCOLOR](#), [RM\\_VALUECOLOR](#) }  
*Darstellungsmodus für Elemente /Punkte, Kanten, Flächen) des 3D-Objekts.*

### Öffentliche Methoden

- [Renderer](#) ()  
*Der Konstruktor.*
- void [initGL](#) (int width, int height)  
*Initialisiert die OpenGL-Bibliothek.*
- void [resize](#) (int width, int height)  
*Verändert die Größe des Anzeigebereichs.*
- void [render](#) ()  
*Zeichnet das Objekt (Attribut object).*
- void [setObject](#) ([ObjectData](#) \*obj)  
*Setzt das zu zeichnende Objekt.*
- void [setCutRenderInfo](#) ([CutRender\\_info](#) \*info)

Setzt die Eigenschaften einer 2D-Temperaturverteilung, welche teilweise zur Visualisierung der Ebene der 2D-Temperaturverteilung benötigt werden.

- wxImage \* [getViewportImage](#) ()  
Gibt den Inhalt der Zeichenfläche als Bild zurück.
- [Viewport\\_info](#) \* [getViewport](#) ()  
Gibt eine Referenz auf die verwendeten Anzeigeeigenschaften zurück.
- virtual ~[Renderer](#) ()  
Der Destruktor.

## Private Methoden

- void [renderMaterial](#) ([ObjectData::MaterialData](#) \*mat)  
Zeichnet die Elemente eines Materials des Objekts.
- void [renderTetrahedra](#) ([ObjectData::MaterialData](#) \*mat, [RenderMode](#) rendermode)  
Zeichnet die Tetraeder eines Materials des Objekts.
- void [renderSensorData](#) ([vector](#)< [SensorPoint](#) > \*data)  
Zeichnet Sensordaten als Punkte.

## Private Attribute

- [Viewport\\_info](#) viewport  
Informationen über die Darstellung des zu zeichnenden Inhalts.
- [ObjectData](#) \* object  
Das darzustellende Objekt.
- [CutRender\\_info](#) \* [cut\\_visualisation\\_info](#)  
Eigenschaften einer 2D-Temperaturverteilung, welche teilweise zur Visualisierung der Ebene der 2D-Temperaturverteilung benötigt werden.
- int [displayList](#)  
Adresse der OpenGL-Displaylist, die die Geometriedaten auf der Grafikkarte vorhält.

## 8.28.1 Ausführliche Beschreibung

Zeichnet den Inhalt der 3D-Fensters.

Zeichnet das 3D-Objekt, Sensordaten und Koordinatensystem je nach Visualisierungsoptionen mithilfe der OpenGL-Bibliothek.

Definiert in Zeile 24 der Datei `Renderer.h`.

## 8.28.2 Dokumentation der Aufzählungstypen

### 8.28.2.1 enum `Renderer::RenderMode`

Darstellungsmodus für Elemente /Punkte, Kanten, Flächen) des 3D-Objekts.

Aufzählungswerte

**`RM_NONE`**

**`RM_MATERIALCOLOR`**

**`RM_VALUECOLOR`**

Definiert in Zeile 29 der Datei `Renderer.h`.

### 8.28.3 Beschreibung der Konstruktoren und Destruktoren

#### 8.28.3.1 `Renderer::Renderer ( )`

Der Konstruktor.

Definiert in Zeile 19 der Datei `Renderer.cpp`.

#### 8.28.3.2 `Renderer::~~Renderer ( ) [virtual]`

Der Destruktor.

Definiert in Zeile 700 der Datei `Renderer.cpp`.

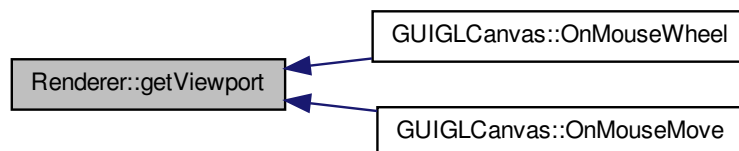
### 8.28.4 Dokumentation der Elementfunktionen

#### 8.28.4.1 `Renderer::Viewport_info * Renderer::getViewport ( )`

Gibt eine Referenz auf die verwendeten Anzeigeeigenschaften zurück.

Definiert in Zeile 42 der Datei `Renderer.cpp`.

Hier ist ein Graph der zeigt, wo diese Funktion aufgerufen wird:



#### 8.28.4.2 `wxImage * Renderer::getViewportImage ( )`

Gibt den Inhalt der Zeichenfläche als Bild zurück.

Definiert in Zeile 532 der Datei `Renderer.cpp`.

#### 8.28.4.3 `void Renderer::initGL ( int width, int height )`

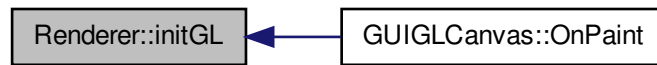
Initialisiert die OpenGL-Bibliothek.

Parameter

<i>width</i>	Breite des Anzeigebereichs.
<i>height</i>	Höhe des Anzeigebereichs.

Definiert in Zeile 346 der Datei `Renderer.cpp`.

Hier ist ein Graph der zeigt, wo diese Funktion aufgerufen wird:

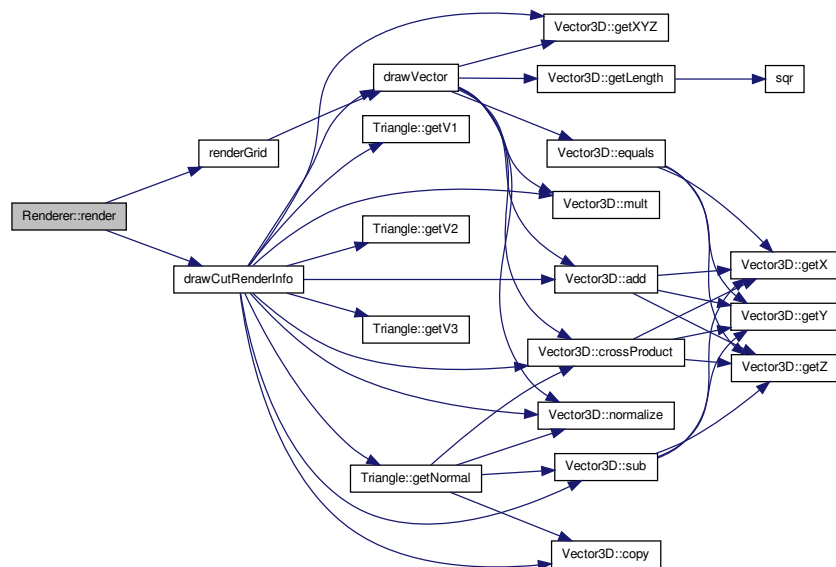


#### 8.28.4.4 void Renderer::render ( )

Zeichnet das Objekt (Attribut object).

Definiert in Zeile 657 der Datei `Renderer.cpp`.

Hier ist ein Graph, der zeigt, was diese Funktion aufruft:



Hier ist ein Graph der zeigt, wo diese Funktion aufgerufen wird:



8.28.4.5 void `Renderer::renderMaterial` ( `ObjectData::MaterialData * mat` ) [private]

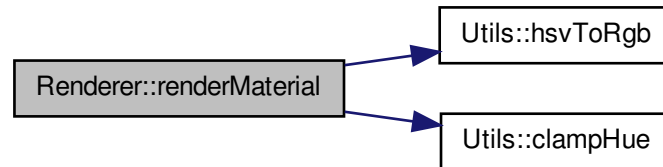
Zeichnet die Elemente eines Materials des Objekts.

## Parameter

<i>mat</i>	Das zu zeichnende Material.
------------	-----------------------------

Definiert in Zeile 274 der Datei `Renderer.cpp`.

Hier ist ein Graph, der zeigt, was diese Funktion aufruft:



#### 8.28.4.6 `void Renderer::renderSensorData ( vector< SensorPoint > * data ) [private]`

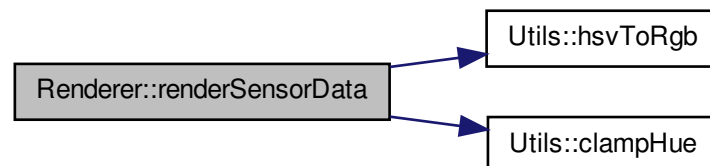
Zeichnet Sensordaten als Punkte.

## Parameter

<i>data</i>	Sensordaten als Liste von Punkten.
-------------	------------------------------------

Definiert in Zeile 236 der Datei `Renderer.cpp`.

Hier ist ein Graph, der zeigt, was diese Funktion aufruft:



#### 8.28.4.7 `void Renderer::renderTetrahedra ( ObjectData::MaterialData * mat, RenderMode rendermode ) [private]`

Zeichnet die Tetraeder eines Materials des Objekts.

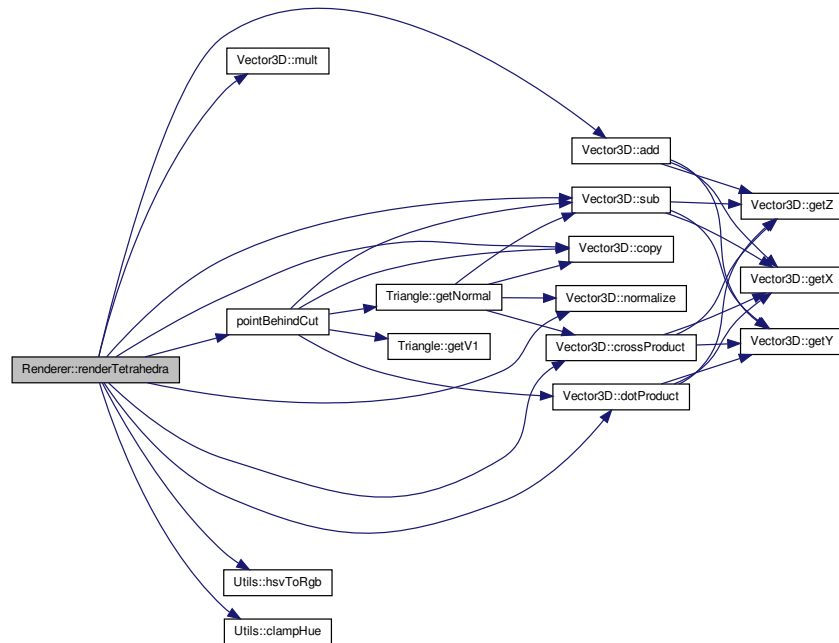
## Parameter

---

<i>mat</i>	Das zu zeichnende Material.
<i>rendermode</i>	Der zu verwendende Zeichenmodus.

Definiert in Zeile 94 der Datei `Renderer.cpp`.

Hier ist ein Graph, der zeigt, was diese Funktion aufruft:



#### 8.28.4.8 void Renderer::resize ( int width, int height )

Verändert die Größe des Anzeigebereichs.

Parameter

<i>width</i>	Neue Breite des Anzeigebereichs.
<i>height</i>	Neue Höhe des Anzeigebereichs.

Definiert in Zeile 353 der Datei `Renderer.cpp`.

Hier ist ein Graph der zeigt, wo diese Funktion aufgerufen wird:



#### 8.28.4.9 void Renderer::setCutRenderInfo ( CutRender\_info \* info )

Setzt die Eigenschaften einer 2D-Temperaturverteilung, welche teilweise zur Visualisierung der Ebene der 2D-Temperaturverteilung benötigt werden.

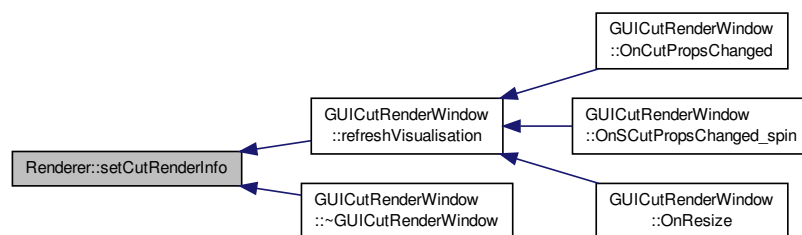
NULL bedeutet keine Visualisierung.

##### Parameter

<i>info</i>	Die Eigenschaften der 2D-Temperaturverteilung.
-------------	--

Definiert in Zeile 489 der Datei Renderer.cpp.

Hier ist ein Graph der zeigt, wo diese Funktion aufgerufen wird:



#### 8.28.4.10 void Renderer::setObject ( ObjectData \* obj )

Setzt das zu zeichnende Objekt.

##### Parameter

<i>obj</i>	Das zu zeichnende Objekt.
------------	---------------------------

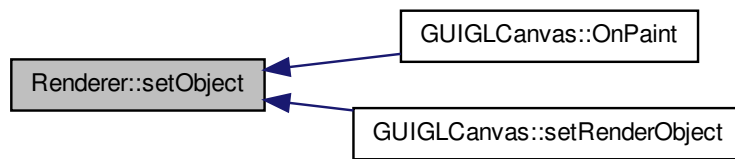
Definiert in Zeile 503 der Datei Renderer.cpp.

Hier ist ein Graph, der zeigt, was diese Funktion aufruft:





Hier ist ein Graph der zeigt, wo diese Funktion aufgerufen wird:



## 8.28.5 Dokumentation der Datenelemente

### 8.28.5.1 `CutRender_info* Renderer::cut_visualisation_info` [private]

Eigenschaften einer 2D-Temperaturverteilung, welche teilweise zur Visualisierung der Ebene der 2D-Temperaturverteilung benötigt werden.

NULL bedeutet keine Visualisierung.

Definiert in Zeile 142 der Datei `Renderer.h`.

### 8.28.5.2 `int Renderer::displayList` [private]

Adresse der OpenGL-Displaylist, die die Geometriedaten auf der Grafikkarte vorhält.

Definiert in Zeile 147 der Datei `Renderer.h`.

### 8.28.5.3 `ObjectData* Renderer::object` [private]

Das darzustellende Objekt.

Definiert in Zeile 135 der Datei `Renderer.h`.

### 8.28.5.4 `Viewport_info Renderer::viewport` [private]

Informationen über die Darstellung des zu zeichnenden Inhalts.

Definiert in Zeile 130 der Datei `Renderer.h`.

Die Dokumentation für diese Klasse wurde erzeugt aufgrund der Dateien:

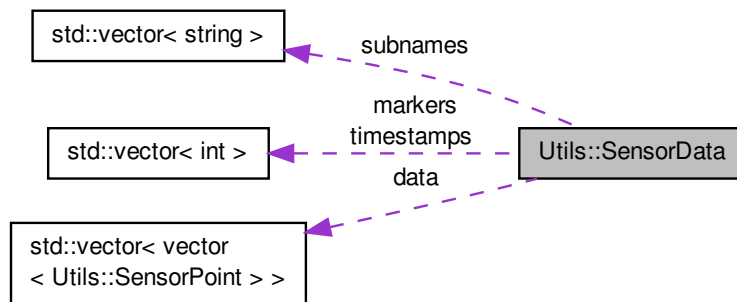
- `/daten/Projekte/eclipse_workspace/simpleanalyzer-gui/src/GUI/Renderer.h`
- `/daten/Projekte/eclipse_workspace/simpleanalyzer-gui/src/GUI/Renderer.cpp`

## 8.29 Utils::SensorData Strukturreferenz

Ein Sensordatensatz.

```
#include <utils.h>
```

Zusammengehörigkeiten von Utils::SensorData:



## Öffentliche Attribute

- `vector< vector< SensorPoint > > data`  
*Daten des Datensatzes (Sensorpunkte zu versch.*
- `vector< string > subnames`  
*Namen der einzelnen Zeitpunkte.*
- `vector< int > timestamps`  
*Zeitstempel der einzelnen Zeitpunkte.*
- `vector< int > markers`  
*Markierte Zeitpunkte.*
- `bool timed`  
*Sind die Sensordaten zeitbezogen? Wenn nein, ist die Länge von data 1.*
- `int current_time_index`  
*Index des aktuell ausgewählten Zeitpunkts.*
- `string name`  
*Name des Sensordatensatzes.*

### 8.29.1 Ausführliche Beschreibung

Ein Sensordatensatz.

Definiert in Zeile 89 der Datei utils.h.

### 8.29.2 Dokumentation der Datenelemente

#### 8.29.2.1 `int Utils::SensorData::current_time_index`

Index des aktuell ausgewählten Zeitpunkts.

Definiert in Zeile 95 der Datei utils.h.

**8.29.2.2   vector<vector<SensorPoint> > Utils::SensorData::data**

Daten des Datensatzes (Sensorpunkte zu versch. Zeitpunkten).

Definiert in Zeile 90 der Datei utils.h.

**8.29.2.3   vector<int> Utils::SensorData::markers**

Markierte Zeitpunkte.

Definiert in Zeile 93 der Datei utils.h.

**8.29.2.4   string Utils::SensorData::name**

Name des Sensordatensatzes.

Definiert in Zeile 96 der Datei utils.h.

**8.29.2.5   vector<string> Utils::SensorData::subnames**

Namen der einzelnen Zeitpunkte.

Definiert in Zeile 91 der Datei utils.h.

**8.29.2.6   bool Utils::SensorData::timed**

Sind die Sensordaten zeitbezogen? Wenn nein, ist die Länge von data 1.

Definiert in Zeile 94 der Datei utils.h.

**8.29.2.7   vector<int> Utils::SensorData::timestamps**

Zeitstempel der einzelnen Zeitpunkte.

Definiert in Zeile 92 der Datei utils.h.

Die Dokumentation für diese Struktur wurde erzeugt aufgrund der Datei:

- [/daten/Projekte/eclipse\\_workspace/simpleanalyzer-gui/src/processing/utils.h](#)

## 8.30   Utils::SensorPoint Strukturreferenz

Daten eines Sensordatenpunktes.

```
#include <utils.h>
```

### Öffentliche Attribute

- double [coords](#) [3]  
*Koordinaten des Punktes.*
- double [temperature](#)  
*Temperatur des Punktes.*

### 8.30.1 Ausführliche Beschreibung

Daten eines Sensordatenpunktes.

Definiert in Zeile 70 der Datei `utils.h`.

### 8.30.2 Dokumentation der Datenelemente

#### 8.30.2.1 `double Utils::SensorPoint::coords[3]`

Koordinaten des Punktes.

Definiert in Zeile 71 der Datei `utils.h`.

#### 8.30.2.2 `double Utils::SensorPoint::temperature`

Temperatur des Punktes.

Definiert in Zeile 72 der Datei `utils.h`.

Die Dokumentation für diese Struktur wurde erzeugt aufgrund der Datei:

- `/daten/Projekte/eclipse_workspace/simpleanalyzer-gui/src/processing/`[utils.h](#)

## 8.31 `Utils::SensorPointComparator` Strukturreferenz

Hilfsstruktur zum Vergleichen des Abstands von Messpunkten.

```
#include <utils.h>
```

### Öffentliche Methoden

- `double` [getDistance\\_d](#) (`double *p1`, `double *p2`)  
*Berechnet den Abstand zwischen zwei Punkten mithilfe des Satzes des Pythagoras.*
- `bool` [operator\(\)](#) ([SensorPoint](#) p1, [SensorPoint](#) p2)  
*Vergleichsoperator für den Abstand zum Punkt meshpoint.*

### Öffentliche Attribute

- `double` [meshpoint](#) [3]  
*Punkt, zu dem der Abstand ermittelt werden soll.*

### 8.31.1 Ausführliche Beschreibung

Hilfsstruktur zum Vergleichen des Abstands von Messpunkten.

Der Punkt, zu dem der Abstand berechnet werden soll ist in der Struktur gespeichert. Zum Vergleich zweier Messpunkte wird dann für beide der Abstand berechnet.

Wird für Sortieralgorithmen der Standardbibliothek benötigt.

Definiert in Zeile 107 der Datei `utils.h`.

## 8.31.2 Dokumentation der Elementfunktionen

8.31.2.1 `double Utils::SensorPointComparator::getDistance_d ( double * p1, double * p2 )` `[inline]`

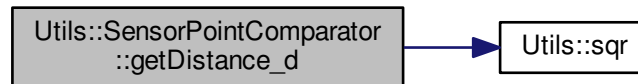
Berechnet den Abstand zwischen zwei Punkten mithilfe des Satzes des Pythagoras.

## Parameter

<i>p1</i>	Koordinaten des ersten Punktes als Liste dreier Koordinaten.
<i>p2</i>	Koordinaten des zweiten Punktes als Liste dreier Koordinaten.

Definiert in Zeile 113 der Datei utils.h.

Hier ist ein Graph, der zeigt, was diese Funktion aufruft:



**8.31.2.2** `bool Utils::SensorPointComparator::operator() ( SensorPoint p1, SensorPoint p2 )` `[inline]`

Vergleichsoperator für den Abstand zum Punkt meshpoint.

Definiert in Zeile 120 der Datei utils.h.

### 8.31.3 Dokumentation der Datenelemente

**8.31.3.1** `double Utils::SensorPointComparator::meshpoint[3]`

Punkt, zu dem der Abstand ermittelt werden soll.

Definiert in Zeile 116 der Datei utils.h.

Die Dokumentation für diese Struktur wurde erzeugt aufgrund der Datei:

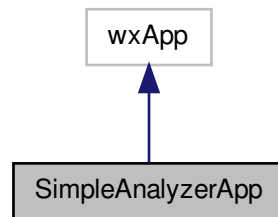
- `/daten/Projekte/eclipse_workspace/simpleanalyzer-gui/src/processing/utils.h`

## 8.32 SimpleAnalyzerApp Klassenreferenz

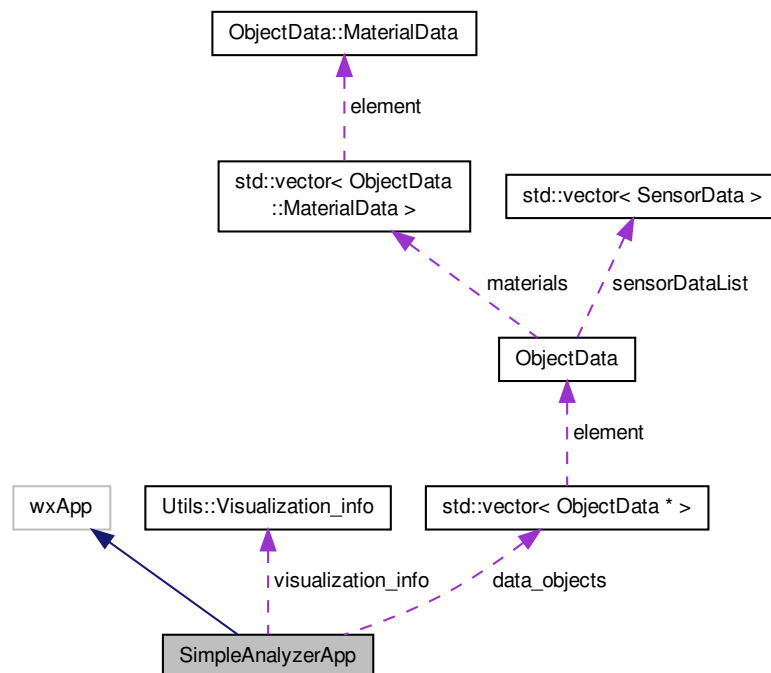
Regelt den allgemeinen Ablauf des Programms.

```
#include <SimpleAnalyzerApp.h>
```

Klassendiagramm für SimpleAnalyzerApp:



Zusammengehörigkeiten von SimpleAnalyzerApp:



## Öffentliche Methoden

- int [getCurrentDataObjectIndex](#) ()  
*Gibt den Index des aktiven Objekts zurück.*
- void [setCurrentDataObjectIndex](#) (int currentDataObjectIndex)  
*Setzt den Index des aktiven Objekts.*
- [vector< ObjectData \\* > \\* getDataObjects](#) ()  
*Gibt einen Verweis auf die Liste der geladenen Objekte zurück.*
- [Utils::Visualization\\_info \\* getVisualizationInfo](#) ()

- *Gibt einen Verweis auf verwendeten Visualisierungsoptionen zurück.*  
• `ObjectData * getActiveObject ()`  
*Gibt einen Verweis auf das aktuell aktive Objekt zurück.*
- `void addObject (ObjectData *obj)`  
*Fügt ein Objekt zur Objektliste hinzu.*
- `void removeCurrentObject ()`  
*Löscht das aktuelle Objekt aus der Objektliste.*
- `virtual ~SimpleAnalyzerApp ()`  
*Der Destruktor.*

## Private Methoden

- `virtual bool OnInit ()`  
*Wird beim Start der Anwendung ausgeführt und öffnet das Hauptfenster.*

## Private Attribute

- `vector< ObjectData * > data_objects`  
*Liste aller geladenen Objekte.*
- `int current_data_object_index = -1`  
*Index des aktuellen Objekts.*
- `Utils::Visualization_info visualization_info`  
*Die allgemein verwendeten Visualisierungsoptionen.*

### 8.32.1 Ausführliche Beschreibung

Regelt den allgemeinen Ablauf des Programms.

Eine eigene Anwendungsklasse wird von wxWidgets gefordert. Das zugrunde liegende System organisiert über diese Klasse den Programmablauf (MainLoop) und Events.

Definiert in Zeile 23 der Datei SimpleAnalyzerApp.h.

### 8.32.2 Beschreibung der Konstruktoren und Destruktoren

#### 8.32.2.1 SimpleAnalyzerApp::~SimpleAnalyzerApp ( ) [virtual]

Der Destruktor.

Definiert in Zeile 88 der Datei SimpleAnalyzerApp.cpp.

### 8.32.3 Dokumentation der Elementfunktionen

#### 8.32.3.1 void SimpleAnalyzerApp::addObject ( ObjectData \* obj )

Fügt ein Objekt zur Objektliste hinzu.

Parameter

<i>obj</i>	Das hinzuzufügende Objekt.
------------	----------------------------

Definiert in Zeile 53 der Datei SimpleAnalyzerApp.cpp.



**8.32.3.2   `ObjectData *` SimpleAnalyzerApp::getActiveObject (   )**

Gibt einen Verweis auf das aktuell aktive Objekt zurück.

**Rückgabe**

Pointer auf das aktuell aktive Objekt.

Definiert in Zeile 49 der Datei SimpleAnalyzerApp.cpp.

**8.32.3.3   `int` SimpleAnalyzerApp::getCurrentDataObjectIndex (   )**

Gibt den Index des aktiven Objekts zurück.

**Rückgabe**

Der Index des aktiven Objekts.

Definiert in Zeile 33 der Datei SimpleAnalyzerApp.cpp.

**8.32.3.4   `vector< ObjectData * > *` SimpleAnalyzerApp::getDataObjects (   )**

Gibt einen Verweis auf die Liste der geladenen Objekte zurück.

**Rückgabe**

Pointer zur Liste der geladenen Objekte.

Definiert in Zeile 41 der Datei SimpleAnalyzerApp.cpp.

**8.32.3.5   `Utils::Visualization_info *` SimpleAnalyzerApp::getVisualizationInfo (   )**

Gibt einen Verweis auf verwendeten Visualisierungsoptionen zurück.

**Rückgabe**

Pointer zu den verwendeten Visualisierungsoptionen.

Definiert in Zeile 45 der Datei SimpleAnalyzerApp.cpp.

**8.32.3.6   `bool` SimpleAnalyzerApp::OnInit (   )   `[private],[virtual]`**

Wird beim Start der Anwendung ausgeführt und öffnet das Hauptfenster.

Definiert in Zeile 72 der Datei SimpleAnalyzerApp.cpp.

**8.32.3.7   `void` SimpleAnalyzerApp::removeCurrentObject (   )**

Löscht das aktuelle Objekt aus der Objektliste.

Definiert in Zeile 57 der Datei SimpleAnalyzerApp.cpp.

**8.32.3.8   `void` SimpleAnalyzerApp::setCurrentDataObjectIndex (   `int currentDataObjectIndex`   )**

Setzt den Index des aktiven Objekts.

## Parameter

<i>currentData-ObjectIndex</i>	Index des auszuwählenden Objekts.
--------------------------------	-----------------------------------

Definiert in Zeile 37 der Datei SimpleAnalyzerApp.cpp.

### 8.32.4 Dokumentation der Datenelemente

8.32.4.1 `int SimpleAnalyzerApp::current_data_object_index = -1` [private]

Index des aktuellen Objekts.

Definiert in Zeile 79 der Datei SimpleAnalyzerApp.h.

8.32.4.2 `vector<ObjectData*> SimpleAnalyzerApp::data_objects` [private]

Liste aller geladenen Objekte.

Definiert in Zeile 74 der Datei SimpleAnalyzerApp.h.

8.32.4.3 `Utils::Visualization_info SimpleAnalyzerApp::visualization_info` [private]

Die allgemein verwendeten Visualisierungsoptionen.

Definiert in Zeile 84 der Datei SimpleAnalyzerApp.h.

Die Dokumentation für diese Klasse wurde erzeugt aufgrund der Dateien:

- [/daten/Projekte/eclipse\\_workspace/simpleanalyzer-gui/src/SimpleAnalyzerApp.h](#)
- [/daten/Projekte/eclipse\\_workspace/simpleanalyzer-gui/src/SimpleAnalyzerApp.cpp](#)

## 8.33 Utils::SortStruct Strukturreferenz

Hilfsstruktur zum Sortieren von Punkten nach dem Abstand zu einem anderen Punkt.

```
#include <utils.h>
```

### Öffentliche Attribute

- `double distance`  
*Abstand des Punktes.*
- `int pointIndex`  
*Index des entsprechenden Sensordatenpunktes.*

#### 8.33.1 Ausführliche Beschreibung

Hilfsstruktur zum Sortieren von Punkten nach dem Abstand zu einem anderen Punkt.

Definiert in Zeile 55 der Datei utils.h.

### 8.33.2 Dokumentation der Datenelemente

#### 8.33.2.1 `double Utils::SortStruct::distance`

Abstand des Punktes.

Definiert in Zeile 56 der Datei `utils.h`.

#### 8.33.2.2 `int Utils::SortStruct::pointIndex`

Index des entsprechenden Sensordatenpunktes.

Definiert in Zeile 57 der Datei `utils.h`.

Die Dokumentation für diese Struktur wurde erzeugt aufgrund der Datei:

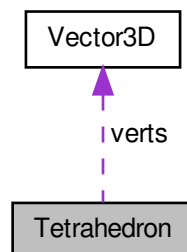
- `/daten/Projekte/eclipse_workspace/simpleanalyzer-gui/src/processing/`[utils.h](#)

## 8.34 Tetrahedron Klassenreferenz

Ein durch 4 Ortsvektoren beschriebener Tetraeder.

```
#include <GeometryClasses.h>
```

Zusammengehörigkeiten von Tetrahedron:



### Öffentliche Methoden

- `Tetrahedron (Vector3D *v1, Vector3D *v2, Vector3D *v3, Vector3D *v4)`  
*Der Konstruktor.*
- `Vector3D * getV1 ()`  
*Gibt eine Referenz auf den Ortsvektor zum 1.*
- `Vector3D * getV2 ()`  
*Gibt eine Referenz auf den Ortsvektor zum 2.*
- `Vector3D * getV3 ()`  
*Gibt eine Referenz auf den Ortsvektor zum 3.*
- `Vector3D * getV4 ()`  
*Gibt eine Referenz auf den Ortsvektor zum 3.*
- `Vector3D * getVert (int index)`  
*Gibt eine Referenz auf den Ortsvektor zum index+1 Punkt des Tetraeders zurück.*

## Private Attribute

- **Vector3D** \* **verts** [4]

Die Referenzen auf die Ortsvektoren zu den Eckpunkten des Tetraeders.

### 8.34.1 Ausführliche Beschreibung

Ein durch 4 Ortsvektoren beschriebener Tetraeder.

Definiert in Zeile 285 der Datei GeometryClasses.h.

### 8.34.2 Beschreibung der Konstruktoren und Destruktoren

#### 8.34.2.1 Tetrahedron::Tetrahedron ( Vector3D \* v1, Vector3D \* v2, Vector3D \* v3, Vector3D \* v4 )

Der Konstruktor.

Die übergebenen Vektorobjekte werden als Element der Klasse gespeichert (nicht kopiert).

Parameter

v1	Ortsvektor zum 1. Punkt des Tetraeders.
v2	Ortsvektor zum 2. Punkt des Tetraeders.
v3	Ortsvektor zum 3. Punkt des Tetraeders.
v4	Ortsvektor zum 4. Punkt des Tetraeders.

Definiert in Zeile 287 der Datei GeometryClasses.cpp.

### 8.34.3 Dokumentation der Elementfunktionen

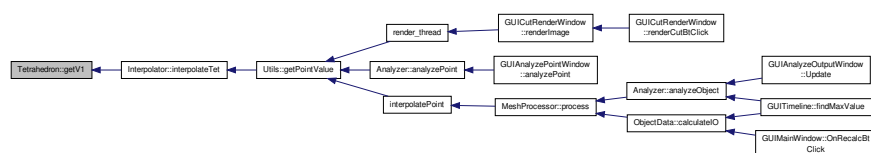
#### 8.34.3.1 Vector3D \* Tetrahedron::getV1 ( )

Gibt eine Referenz auf den Ortsvektor zum 1.

Punkt des Tetraeders zurück.

Definiert in Zeile 295 der Datei GeometryClasses.cpp.

Hier ist ein Graph der zeigt, wo diese Funktion aufgerufen wird:



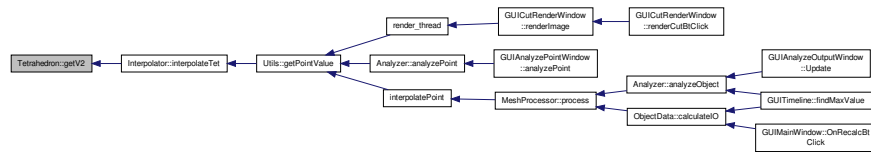
#### 8.34.3.2 Vector3D \* Tetrahedron::getV2 ( )

Gibt eine Referenz auf den Ortsvektor zum 2.

Punkt des Tetraeders zurück.

Definiert in Zeile 299 der Datei GeometryClasses.cpp.

Hier ist ein Graph der zeigt, wo diese Funktion aufgerufen wird:



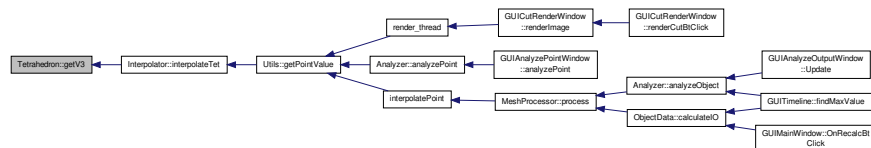
### 8.34.3.3 Vector3D \* Tetrahedron::getV3 ( )

Gibt eine Referenz auf den Ortsvektor zum 3.

Punkt des Tetraeders zurück.

Definiert in Zeile 303 der Datei GeometryClasses.cpp.

Hier ist ein Graph der zeigt, wo diese Funktion aufgerufen wird:



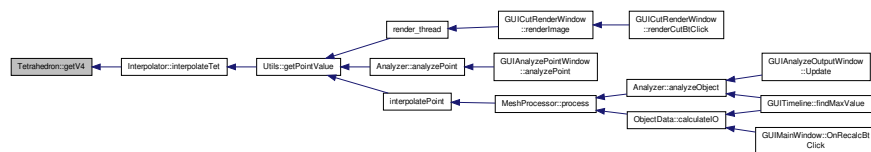
#### 8.34.3.4 Vector3D \* Tetrahedron::getV4 ( )

Gibt eine Referenz auf den Ortsvektor zum 3.

Punkt des Tetraeders zurück.

Definiert in Zeile 307 der Datei GeometryClasses.cpp.

Hier ist ein Graph der zeigt, wo diese Funktion aufgerufen wird:



### 8.34.3.5 Vector3D \* Tetrahedron::getVert ( int *index* )

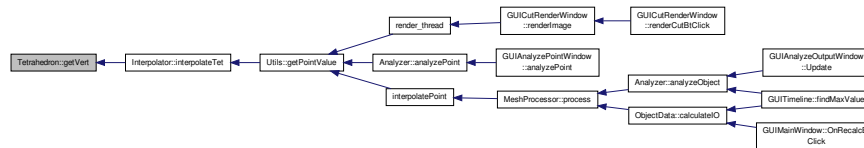
Gibt eine Referenz auf den Ortsvektor zum index+1 Punkt des Tetraeders zurück.

## Parameter

<i>index</i>	Der Index des gesuchten Punktes (0..3).
--------------	---

Definiert in Zeile 311 der Datei GeometryClasses.cpp.

Hier ist ein Graph der zeigt, wo diese Funktion aufgerufen wird:



## 8.34.4 Dokumentation der Datenelemente

### 8.34.4.1 Vector3D\* Tetrahedron::verts[4] [private]

Die Referenzen auf die Ortsvektoren zu den Eckpunkten des Tetraeders.

Definiert in Zeile 326 der Datei GeometryClasses.h.

Die Dokumentation für diese Klasse wurde erzeugt aufgrund der Dateien:

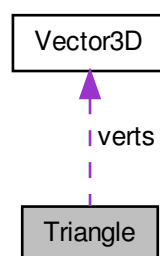
- /daten/Projekte/eclipse\_workspace/simpleanalyzer-gui/src/libraries/interpolate/GeometryClasses.h
- /daten/Projekte/eclipse\_workspace/simpleanalyzer-gui/src/libraries/interpolate/GeometryClasses.cpp

## 8.35 Triangle Klassenreferenz

Ein durch 3 Ortsvektoren beschriebenes Dreieck.

```
#include <GeometryClasses.h>
```

Zusammengehörigkeiten von Triangle:



## Öffentliche Methoden

- [Triangle](#) ([Vector3D](#) \*v1, [Vector3D](#) \*v2, [Vector3D](#) \*v3)
- Der Konstruktor.

- `Vector3D * getV1 ()`  
*Gibt eine Referenz auf den Ortsvektor zum ersten Punkt des Dreiecks zurück.*
- `Vector3D * getV2 ()`  
*Gibt eine Referenz auf den Ortsvektor zum zweiten Punkt des Dreiecks zurück.*
- `Vector3D * getV3 ()`  
*Gibt eine Referenz auf den Ortsvektor zum dritten Punkt des Dreiecks zurück.*
- `Vector3D * getVert (int index)`  
*Gibt eine Referenz auf den Ortsvektor zum index+1 Punkt des Dreiecks zurück.*
- `void print ()`  
*Gibt die Punkte des Dreiecks auf dem cout-Stream aus.*
- `Vector3D * getNormal ()`  
*Gibt die Normale des Dreiecks zurück.*
- `~Triangle ()`  
*Der Destruktor.*

### Private Attribute

- `Vector3D * verts [3]`  
*Die Referenzen auf die Ortsvektoren zu den Eckpunkten des Dreiecks.*

### 8.35.1 Ausführliche Beschreibung

Ein durch 3 Ortsvektoren beschriebenes Dreieck.

Definiert in Zeile 228 der Datei GeometryClasses.h.

### 8.35.2 Beschreibung der Konstruktoren und Destrukturen

#### 8.35.2.1 Triangle::Triangle ( Vector3D \* v1, Vector3D \* v2, Vector3D \* v3 )

Der Konstruktor.

Die übergebenen Vektorobjekte werden als Element der Klasse gespeichert (nicht kopiert).

Parameter

v1	Ortsvektor zum 1. Punkt des Dreiecks.
v2	Ortsvektor zum 2. Punkt des Dreiecks.
v3	Ortsvektor zum 3. Punkt des Dreiecks.

Definiert in Zeile 240 der Datei GeometryClasses.cpp.

#### 8.35.2.2 Triangle::~~Triangle ( )

Der Destruktor.

Definiert in Zeile 283 der Datei GeometryClasses.cpp.

### 8.35.3 Dokumentation der Elementfunktionen

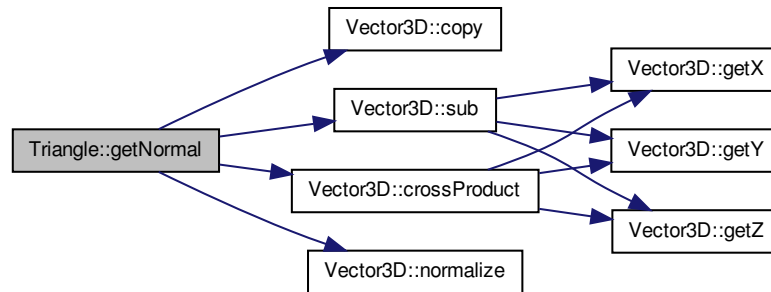
#### 8.35.3.1 Vector3D \* Triangle::getNormal ( )

Gibt die Normale des Dreiecks zurück.

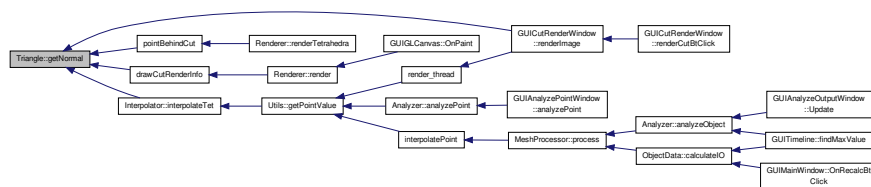
Der zurückgegebene Vektor muss manuell mit delete freigegeben werden!

Definiert in Zeile 262 der Datei GeometryClasses.cpp.

Hier ist ein Graph, der zeigt, was diese Funktion aufruft:



Hier ist ein Graph der zeigt, wo diese Funktion aufgerufen wird:

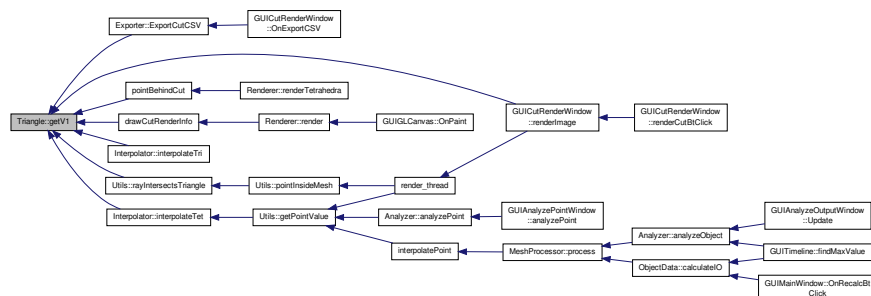


### 8.35.3.2 Vector3D \* Triangle::getV1 ( )

Gibt eine Referenz auf den Ortsvektor zum ersten Punkt des Dreiecks zurück.

Definiert in Zeile 246 der Datei GeometryClasses.cpp.

Hier ist ein Graph der zeigt, wo diese Funktion aufgerufen wird:



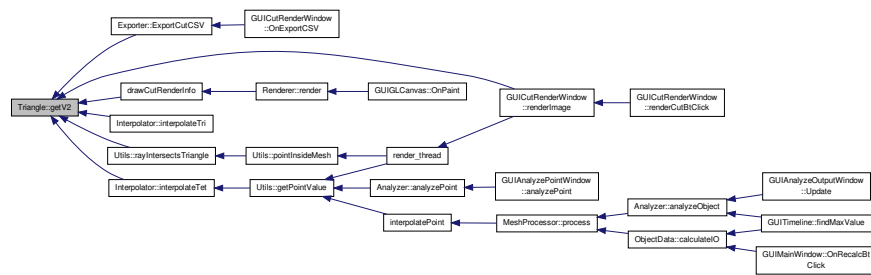
### 8.35.3.3 Vector3D \* Triangle::getV2 ( )

Gibt eine Referenz auf den Ortsvektor zum zweiten Punkt des Dreiecks zurück.



Definiert in Zeile 250 der Datei GeometryClasses.cpp.

Hier ist ein Graph der zeigt, wo diese Funktion aufgerufen wird:

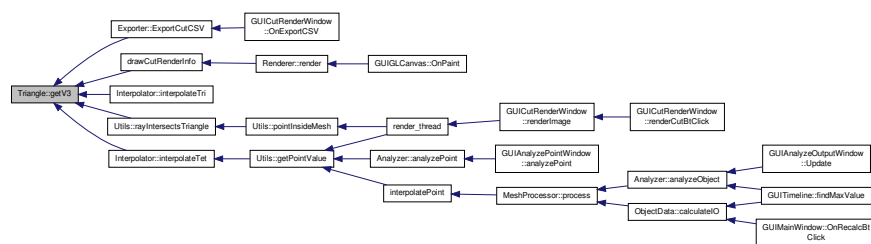


#### 8.35.3.4 Vector3D \* Triangle::getV3 ( )

Gibt eine Referenz auf den Ortsvektor zum dritten Punkt des Dreiecks zurück.

Definiert in Zeile 254 der Datei GeometryClasses.cpp.

Hier ist ein Graph der zeigt, wo diese Funktion aufgerufen wird:



#### 8.35.3.5 Vector3D \* Triangle::getVert ( int index )

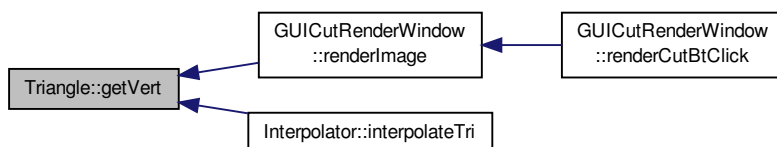
Gibt eine Referenz auf den Ortsvektor zum index+1 Punkt des Dreiecks zurück.

Parameter

<i>index</i>	Der Index des gesuchten Punktes (0..2).
--------------	---

Definiert in Zeile 258 der Datei GeometryClasses.cpp.

Hier ist ein Graph der zeigt, wo diese Funktion aufgerufen wird:



#### 8.35.3.6 void Triangle::print ( )

Gibt die Punkte des Dreiecks auf dem cout-Stream aus.

Definiert in Zeile 274 der Datei GeometryClasses.cpp.

### 8.35.4 Dokumentation der Datenelemente

#### 8.35.4.1 Vector3D\* Triangle::verts[3] [private]

Die Referenzen auf die Ortsvektoren zu den Eckpunkten des Dreiecks.

Definiert in Zeile 278 der Datei GeometryClasses.h.

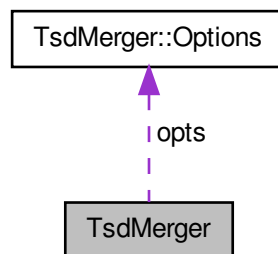
Die Dokumentation für diese Klasse wurde erzeugt aufgrund der Dateien:

- /daten/Projekte/eclipse\_workspace/simpleanalyzer-gui/src/libraries/interpolate/GeometryClasses.h
- /daten/Projekte/eclipse\_workspace/simpleanalyzer-gui/src/libraries/interpolate/GeometryClasses.cpp

## 8.36 TsdMerger Klassenreferenz

Zusammenführen zweier .tsd-Dateien.

Zusammengehörigkeiten von TsdMerger:



### Klassen

- struct [Options](#)

*Strunktur für die Programmeinstellungen.*

### Öffentliche Methoden

- int [merge](#) (int argc, char \*argv[])

*Liest die Programmargumente um die Eingabedateien anhand der Zeitstempel in eine .tsd-Datei zusammen zu führen.*

## Geschützte Methoden

- string `getTextBlock` (string data, int n)  
*Gibt den n-ten durch Leerzeichen abgetrennten Block aus einem String zurück.*
- int `parseFile` (string filename, vector< long > &timestamps, vector< string > &names, vector< string > &data)  
*Sammelt Daten aus einer .tsd-Datei.*
- bool `parseArguments` (int argc, char \*argv[], string &input1, string &input2, string &output\_file)  
*Wertet die Programmargumente aus.*
- bool `writeOutputFile` (string path, vector< long > &timestamps1, vector< string > &names1, vector< string > &data1, vector< long > &timestamps2, vector< string > &names2, vector< string > &data2)  
*Schreibt die Ausgabedatei.*

## Geschützte Attribute

- struct `TsdMerger::Options` `opts`  
*Hält die verwendeten Programmeinstellungen.*

### 8.36.1 Ausführliche Beschreibung

Zusammenführen zweier .tsd-Dateien.

Definiert in Zeile 21 der Datei mergetsd.cpp.

### 8.36.2 Dokumentation der Elementfunktionen

#### 8.36.2.1 string TsdMerger::getTextBlock ( string data, int n ) [inline], [protected]

Gibt den n-ten durch Leerzeichen abgetrennten Block aus einem String zurück.

Parameter

<i>data</i>	Der Ausgangsstring.
<i>n</i>	Index des zu findenden Blocks.

Rückgabe

Der n-te durch Leerzeichen getrennte Teilstring. "" Bei ungültigem Index.

Definiert in Zeile 40 der Datei mergetsd.cpp.

#### 8.36.2.2 int TsdMerger::merge ( int argc, char \* argv[] ) [inline]

Liest die Programmargumente um die Eingabedateien anhand der Zeitstempel in eine .tsd-Datei zusammen zu führen.

Wird durch die Funktion `main()` von außerhalb des Namespaces aufgerufen.

Parameter

<i>argc</i>	Anzahl der Programmargumente.
-------------	-------------------------------

<i>argv</i>	Die Programmargumente.
-------------	------------------------

Definiert in Zeile 313 der Datei mergetsd.cpp.

Hier ist ein Graph der zeigt, wo diese Funktion aufgerufen wird:



**8.36.2.3** `bool TsdMerger::parseArguments ( int argc, char * argv[], string & input1, string & input2, string & output_file )`  
`[inline], [protected]`

Wertet die Programmargumente aus.

#### Parameter

<i>argc</i>	Anzahl der Programmargumente.
<i>argv</i>	Die Programmargumente.
<i>input1</i>	Ausgabe für den Pfad zur Eingabedatei 1.
<i>input2</i>	Ausgabe für den Pfad zur Eingabedatei 2.
<i>output_file</i>	Ausgabe für den Pfad zur Ausgabedatei.

#### Rückgabe

Soll das Programm weiter ablaufen?

Definiert in Zeile 138 der Datei mergetsd.cpp.

**8.36.2.4** `int TsdMerger::parseFile ( string filename, vector< long > & timestamps, vector< string > & names, vector< string > & data )`  
`[inline], [protected]`

Sammelt Daten aus einer .tsd-Datei.

#### Parameter

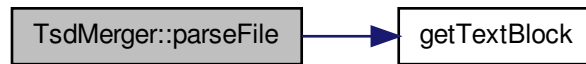
<i>filename</i>	Der Pfad zur .tsd-Datei.
<i>timestamps</i>	Ausgabevariable für die Zeitstempel der Datensätze.
<i>names</i>	Ausgabevariable für den Namen der Datensätze.
<i>data</i>	Ausgabevariable für die Sensordaten der Datensätze.

**Rückgabe**

Gibt 0 bei Erfolg zurück, 1, wenn die Datei nicht gefunden werden konnte.

Definiert in Zeile 80 der Datei mergetsd.cpp.

Hier ist ein Graph, der zeigt, was diese Funktion aufruft:



**8.36.2.5** `bool TsdMerger::writeOutputFile ( string path, vector< long > & timestamps1, vector< string > & names1, vector< string > & data1, vector< long > & timestamps2, vector< string > & names2, vector< string > & data2 ) [inline], [protected]`

Schreibt die Ausgabedatei.

**Parameter**

<i>path</i>	Der Pfad zur Ausgabedatei.
<i>timestamps1</i>	Zeitstempel der Datensätze der ersten Datei.
<i>timestamps2</i>	Zeitstempel der Datensätze der zweiten Datei.
<i>names1</i>	Namen der Datensätze der ersten Datei.
<i>names2</i>	Namen der Datensätze der zweiten Datei.
<i>data1</i>	Daten der Datensätze der ersten Datei.
<i>data2</i>	Daten der Datensätze der zweiten Datei.

**Rückgabe**

War das Schreiben erfolgreich?

Definiert in Zeile 256 der Datei mergetsd.cpp.

**8.36.3 Dokumentation der Datenelemente**

**8.36.3.1** `struct TsdMerger::Options TsdMerger::opts [protected]`

Hält die verwendeten Programmeinstellungen.

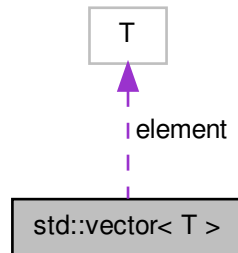
Die Dokumentation für diese Klasse wurde erzeugt aufgrund der Datei:

- [/daten/Projekte/eclipse\\_workspace/mergetsd/src/mergetsd.cpp](#)

**8.37 std::vector< T > Template-Klassenreferenz**

```
#include <doxygen_dep_dummy.h>
```

Zusammengehörigkeiten von `std::vector< T >`:



## Öffentliche Attribute

- [T element](#)

### 8.37.1 Ausführliche Beschreibung

```
template<class T>class std::vector< T >
```

STL vector class

Definiert in Zeile 3 der Datei `doxygen_dep_dummy.h`.

### 8.37.2 Dokumentation der Datenelemente

8.37.2.1 `template<class T> T std::vector< T >::element`

Definiert in Zeile 3 der Datei `doxygen_dep_dummy.h`.

Die Dokumentation für diese Klasse wurde erzeugt aufgrund der Datei:

- [doxygen\\_dep\\_dummy.h](#)

## 8.38 Vector3D Klassenreferenz

3D-Vektorklasse mit nützlichen Operationen.

```
#include <GeometryClasses.h>
```

## Öffentliche Methoden

- [Vector3D](#) (double x, double y, double z)  
*Konstruktor für Konstruktion aus einzelnen Koordinaten.*
- [Vector3D](#) (const double \*values)  
*Konstruktor für Konstruktion aus einer Koordinatenliste.*
- [Vector3D](#) ([Vector3D](#) \*other)

*Konstruktor für die Konstruktion aus einem anderen Vektor.*

- `Vector3D * copy ()`  
*Gibt eine Kopie des Vektors zurück.*
- `double getX ()`  
*Gibt das X-Element des Vektors zurück.*
- `double getY ()`  
*Gibt das Y-Element des Vektors zurück.*
- `double getZ ()`  
*Gibt das Z-Element des Vektors zurück.*
- `double getLength ()`  
*Gibt die Länge des Vektors zurück.*
- `double getAngleTo (Vector3D *other)`  
*Gibt den Winkel zu einem anderen Vektor in RAD zurück.*
- `double dotProduct (Vector3D *other)`  
*Gibt das Skalarprodukt mit einem anderen Vektor.*
- `Vector3D * crossProduct (Vector3D *other)`  
*Gibt das Kreuzprodukt mit einem anderen Vektor zurück.*
- `void add (Vector3D *other)`  
*Addiert einen Vektor zu diesem Vektor.*
- `void sub (Vector3D *other)`  
*Subtrahiert einen Vektor von diesem Vektor.*
- `void mult (double scalar)`  
*Multipliziert den Vektor mit einem Skalar.*
- `void normalize ()`  
*Normalisiert den Vektor.*
- `bool equals (Vector3D *other)`  
*Testet, ob zwei Vektoren identisch sind.*
- `double getDistanceTo (Vector3D *other)`  
*Gibt den Abstand zu einem anderen Vektor zurück.*
- `double * getXYZ ()`  
*Gibt eine Referenz auf die Vektorelemente zurück (Vor allem zur Übergabe an OpenGL verwendet).*
- `void print ()`  
*Gibt den Vektor auf dem cout-Stream aus.*
- `void printTo (std::ostream &stream) const`  
*Gibt den Vektor auf dem gegebenen Stream aus.*
- `virtual ~Vector3D ()`  
*Der Destruktor.*

## Private Attribute

- `double coords [3]`  
*Die Elemente des Vektors.*

## Freundbeziehungen

- `std::ostream & operator<< (std::ostream &out, const Vector3D &vec)`  
*Definition des <<-Operators für die Ausgabe eines Vektors.*

### 8.38.1 Ausführliche Beschreibung

3D-Vektorklasse mit nützlichen Operationen.

Definiert in Zeile 13 der Datei GeometryClasses.h.

### 8.38.2 Beschreibung der Konstruktoren und Destruktoren

#### 8.38.2.1 `Vector3D::Vector3D ( double x, double y, double z )`

Konstruktor für Konstruktion aus einzelnen Koordinaten.

Parameter

<i>x</i>	X-Element des Vektors.
<i>y</i>	Y-Element des Vektors.
<i>z</i>	Z-Element des Vektors.

Definiert in Zeile 19 der Datei GeometryClasses.cpp.

#### 8.38.2.2 `Vector3D::Vector3D ( const double * values )`

Konstruktor für Konstruktion aus einer Koordinatenliste.

Parameter

<i>values</i>	Liste der Koordinaten (x,y und z-Wert).
---------------	---

Definiert in Zeile 25 der Datei GeometryClasses.cpp.

#### 8.38.2.3 `Vector3D::Vector3D ( Vector3D * other )`

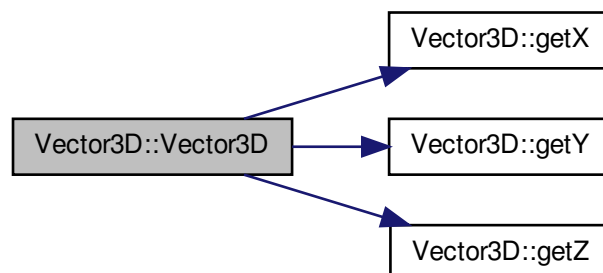
Konstruktor für die Konstruktion aus einem anderen Vektor.

Parameter

<i>other</i>	Der Vektor, dessen Eigenschaften übernommen werden sollen.
--------------	--

Definiert in Zeile 31 der Datei GeometryClasses.cpp.

Hier ist ein Graph, der zeigt, was diese Funktion aufruft:









8.38.3.4 `double Vector3D::dotProduct ( Vector3D * other )`

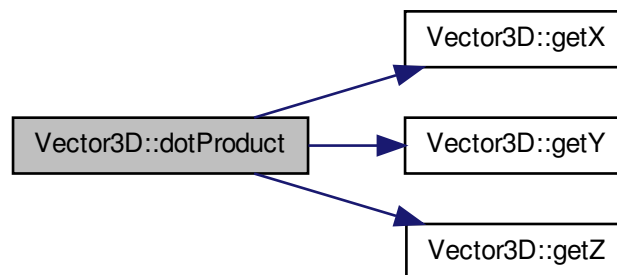
Gibt das Skalarprodukt mit einem anderen Vektor.

Parameter

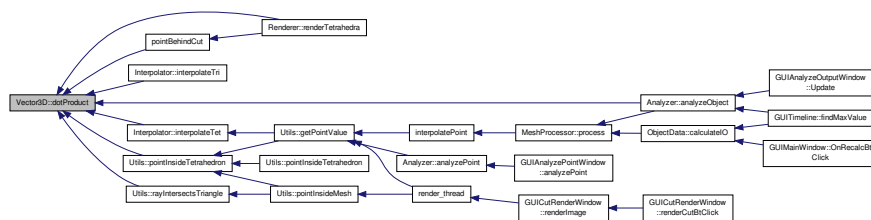
<i>other</i>	Der Vektor, mit dem das Skalarprodukt gebildet werden soll.
--------------	---

Definiert in Zeile 76 der Datei GeometryClasses.cpp.

Hier ist ein Graph, der zeigt, was diese Funktion aufruft:



Hier ist ein Graph der zeigt, wo diese Funktion aufgerufen wird:

8.38.3.5 `bool Vector3D::equals ( Vector3D * other )`

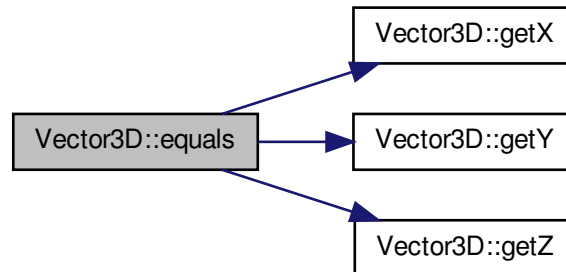
Testet, ob zwei Vektoren identisch sind.

Parameter

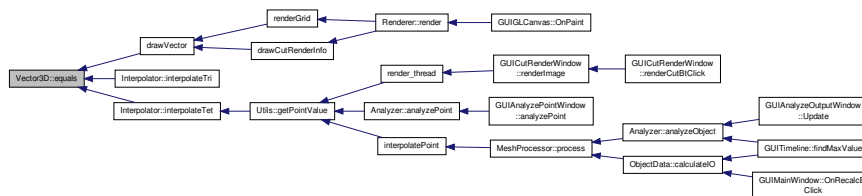
<i>other</i>	Der Vektor, mit dem verglichen werden soll.
--------------	---

Definiert in Zeile 42 der Datei GeometryClasses.cpp.

Hier ist ein Graph, der zeigt, was diese Funktion aufruft:



Hier ist ein Graph der zeigt, wo diese Funktion aufgerufen wird:



### 8.38.3.6 double Vector3D::getAngleTo ( Vector3D \* other )

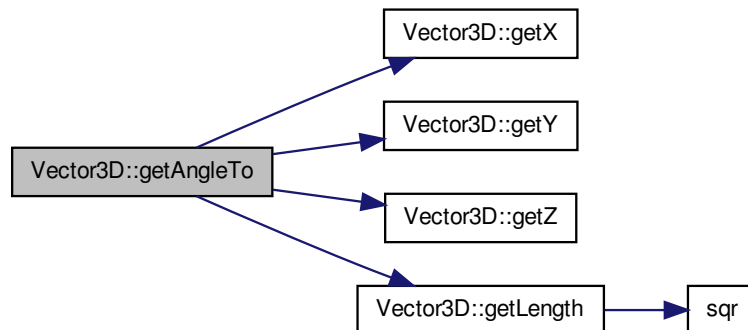
Gibt den Winkel zu einem anderen Vektor in RAD zurück.

Parameter

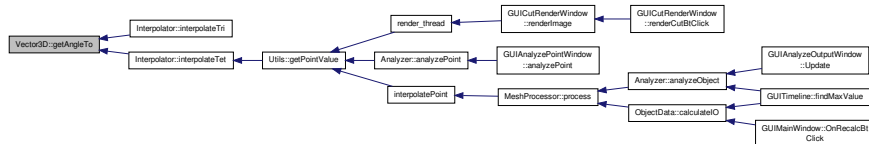
<i>other</i>	Der Vektor, zu dem der Winkel ermittelt werden soll.
--------------	--

Definiert in Zeile 64 der Datei GeometryClasses.cpp.

Hier ist ein Graph, der zeigt, was diese Funktion aufruft:



Hier ist ein Graph der zeigt, wo diese Funktion aufgerufen wird:



#### 8.38.3.7 double Vector3D::getDistanceTo ( Vector3D \* other )

Gibt den Abstand zu einem anderen Vektor zurück.

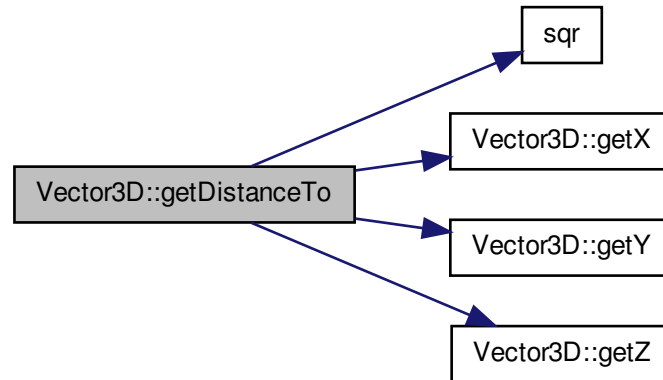
Dabei werden beide Vektoren als Ortsvektoren betrachtet.

Parameter

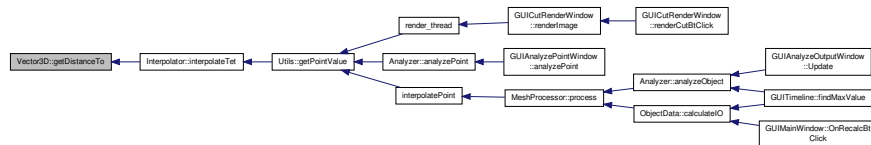
<i>other</i>	Der Vektor, zu dem der Abstand ermittelt werden soll.
--------------	---

Definiert in Zeile 117 der Datei GeometryClasses.cpp.

Hier ist ein Graph, der zeigt, was diese Funktion aufruft:



Hier ist ein Graph der zeigt, wo diese Funktion aufgerufen wird:

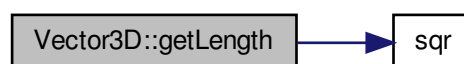


#### 8.38.3.8 double Vector3D::getLength ( )

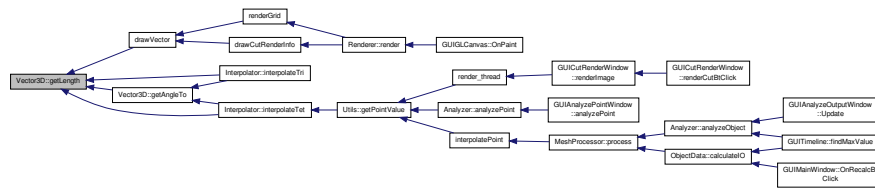
Gibt die Länge des Vektors zurück.

Definiert in Zeile 60 der Datei GeometryClasses.cpp.

Hier ist ein Graph, der zeigt, was diese Funktion aufruft:



Hier ist ein Graph der zeigt, wo diese Funktion aufgerufen wird:

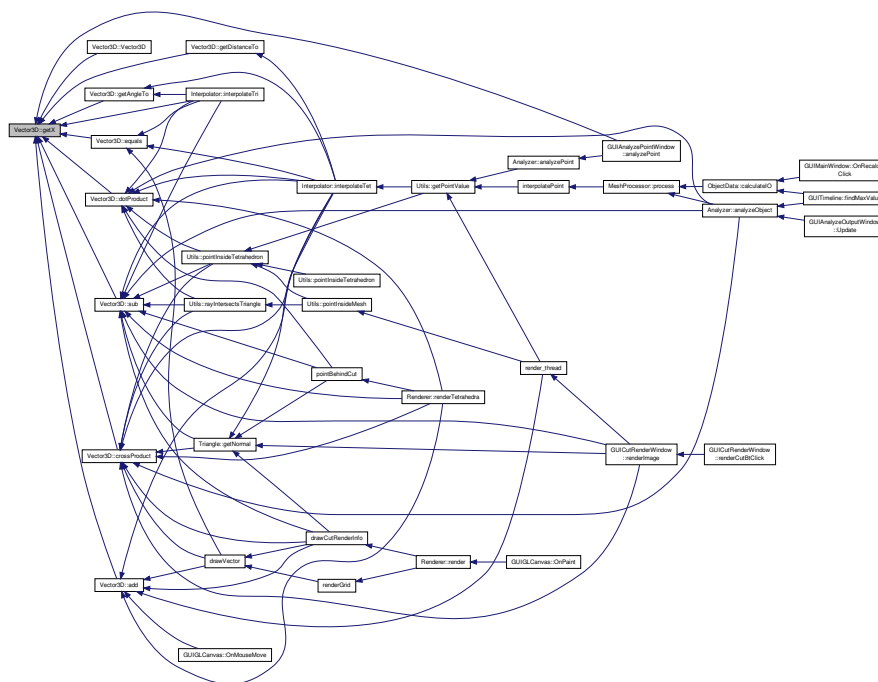


### 8.38.3.9 double Vector3D::getX ( )

Gibt das X-Element des Vektors zurück.

Definiert in Zeile 48 der Datei GeometryClasses.cpp.

Hier ist ein Graph der zeigt, wo diese Funktion aufgerufen wird:

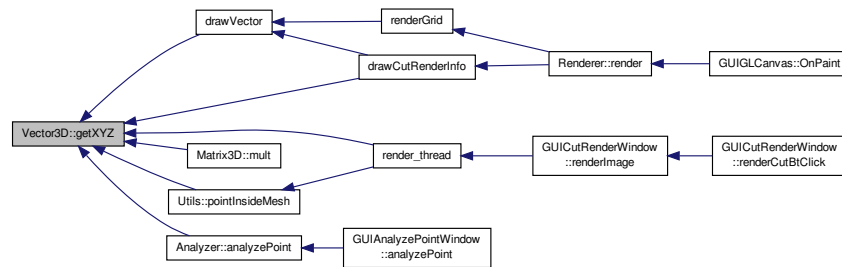


### 8.38.3.10 double \* Vector3D::getXYZ ( )

Gibt eine Referenz auf die Vektorelemente zurück (Vor allem zur Übergabe an OpenGL verwendet).

Definiert in Zeile 113 der Datei GeometryClasses.cpp.

Hier ist ein Graph der zeigt, wo diese Funktion aufgerufen wird:

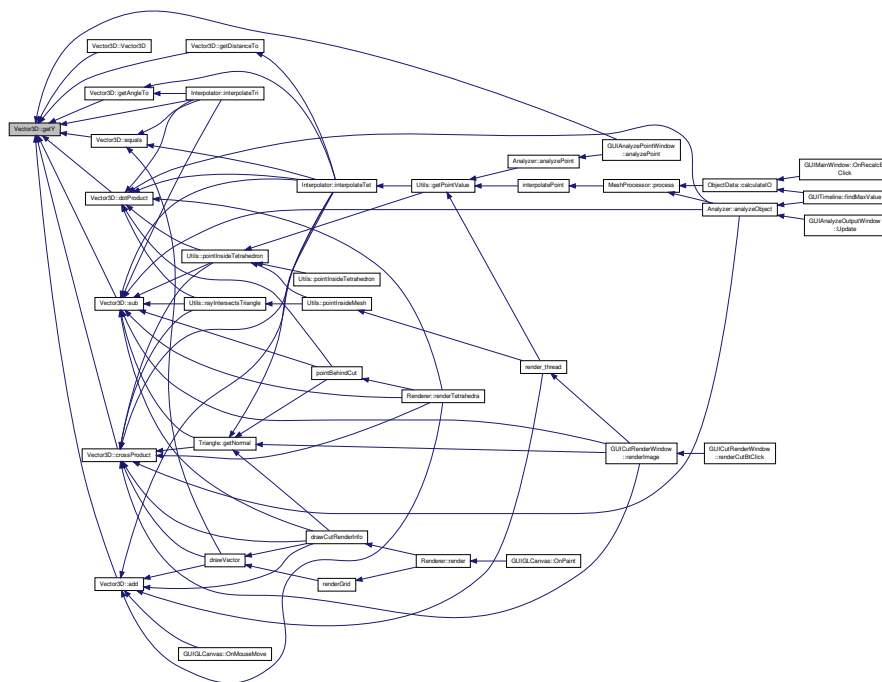


### 8.38.3.11 double Vector3D::getY ( )

Gibt das Y-Element des Vektors zurück.

Definiert in Zeile 52 der Datei GeometryClasses.cpp.

Hier ist ein Graph der zeigt, wo diese Funktion aufgerufen wird:



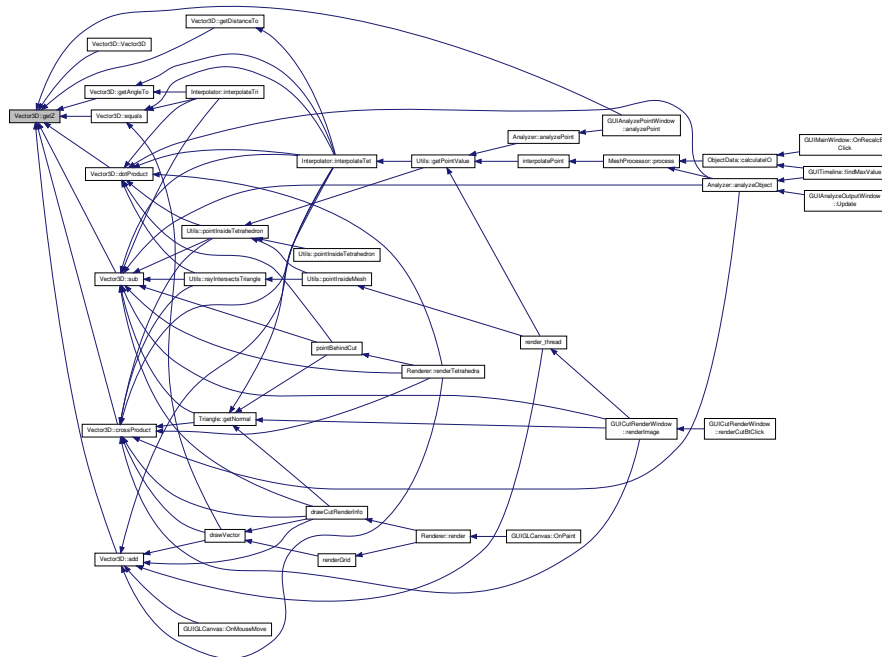
### 8.38.3.12 double Vector3D::getZ ( )

Gibt das Z-Element des Vektors zurück.

Definiert in Zeile 56 der Datei GeometryClasses.cpp.



Hier ist ein Graph der zeigt, wo diese Funktion aufgerufen wird:



**8.38.3.13** `void Vector3D::mult ( double scalar )`

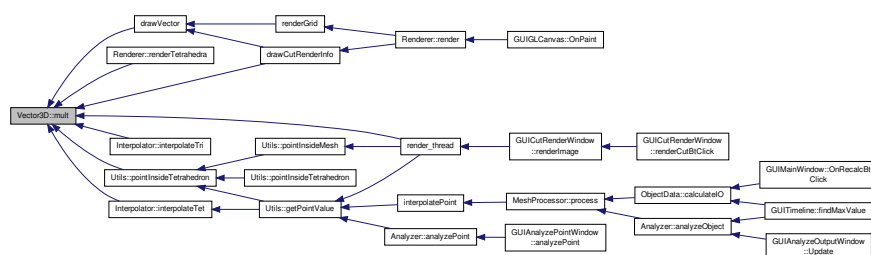
Multipliziert den Vektor mit einem Skalar.

### Parameter

<i>scalar</i>	Der Skalar.
---------------	-------------

Definiert in Zeile 100 der Datei GeometryClasses.cpp.

Hier ist ein Graph der zeigt, wo diese Funktion aufgerufen wird:



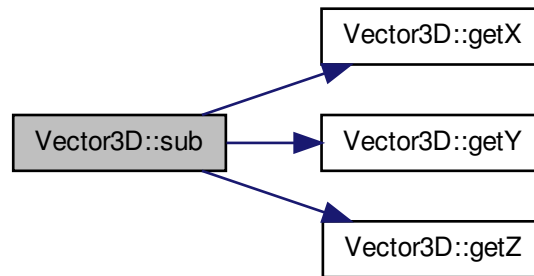
### 8.38.3.14 void Vector3D::normalize ( )

Normalisiert den Vektor.

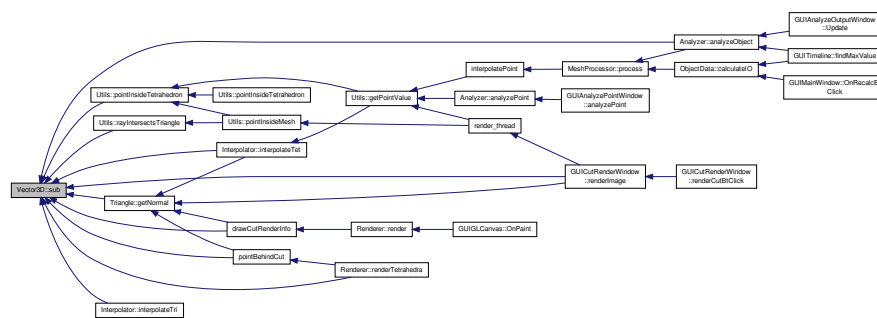
Definiert in Zeile 106 der Datei GeometryClasses.cpp.



Hier ist ein Graph, der zeigt, was diese Funktion aufruft:



Hier ist ein Graph der zeigt, wo diese Funktion aufgerufen wird:



#### 8.38.4 Freundbeziehungen und Funktionsdokumentation

```
8.38.4.1 std::ostream& operator<< ( std::ostream & out, const Vector3D & vec ) [friend]
```

Definition des  $\ll$ -Operators für die Ausgabe eines Vektors.

Definiert in Zeile 132 der Datei GeometryClasses.cpp.

### 8.38.5 Dokumentation der Datenelemente

### 8.38.5.1 double Vector3D::coords[3] [private]

Die Elemente des Vektors.

Definiert in Zeile 139 der Datei GeometryClasses.h.

Die Dokumentation für diese Klasse wurde erzeugt aufgrund der Dateien:

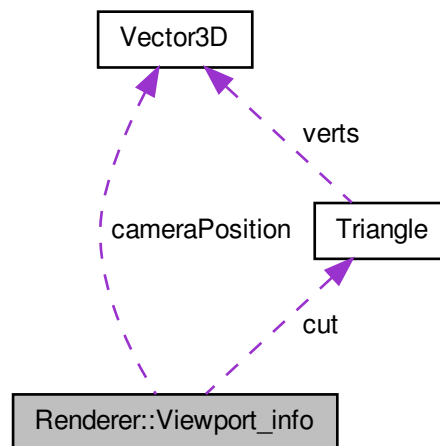
- /daten/Projekte/eclipse\_workspace/simpleanalyzer-gui/src/libraries/interpolate/GeometryClasses.h
- /daten/Projekte/eclipse\_workspace/simpleanalyzer-gui/src/libraries/interpolate/GeometryClasses.cpp

### 8.39 `Renderer::Viewport_info` Strukturreferenz

Informationen über die Ansicht des Modells (Virtuelle Kamera) und welche Elemente dargestellt werden.

```
#include <Renderer.h>
```

Zusammengehörigkeiten von `Renderer::Viewport_info`:



#### Öffentliche Attribute

- float `zoom`  
*Aktueller Zoomfaktor.*
- float `rotationY`  
*Rotation der Ansicht um die Y-Achse.*
- float `rotationX`  
*Rotation der Ansicht um die (Kameralokale) X-Achse.*
- `Vector3D` \* `cameraPosition`  
*position der Virtuellen Kamera*
- `Triangle` \* `cut`  
*Dreieck der Schnittebene, wenn nicht NULL, werden nur Elemente oberhalb der Dreiecksebene dargestellt (Momentan nicht verwendet).*
- bool `invertcut`  
*Nur Elemente unterhalb der durch cut definierte Ebene darstellen (Momentan nicht verwendet).*
- `RenderMode` `showPoints`  
*Modus der Darstellung von Punkten des 3D-Objekts.*
- `RenderMode` `showEdges`  
*Modus der Darstellung von Kanten des 3D-Objekts.*
- `RenderMode` `showFaces`  
*Modus der Darstellung von Flächen des 3D-Objekts.*
- bool `show_extrapolated`  
*Extrapolierte Elemente anzeigen.*
- bool `show_sensordata`  
*Sensordaten als Punkte anzeigen.*

- int `width`  
*Breite des dargestellten Bereichs.*
- int `height`  
*Höhe des dargestellten Bereichs.*
- float `scale`  
*Skalierungsfaktor für das 3D-Objekt.*

### 8.39.1 Ausführliche Beschreibung

Informationen über die Ansicht des Modells (Virtuelle Kamera) und welche Elemente dargestellt werden.

Definiert in Zeile 38 der Datei `Renderer.h`.

### 8.39.2 Dokumentation der Datenelemente

#### 8.39.2.1 `Vector3D*` `Renderer::Viewport_info::cameraPosition`

position der Virtuellen Kamera

Definiert in Zeile 42 der Datei `Renderer.h`.

#### 8.39.2.2 `Triangle*` `Renderer::Viewport_info::cut`

Dreieck der Schnittebene, wenn nicht NULL, werden nur Elemente oberhalb der Dreiecksebene dargestellt (Momentan nicht verwendet).

Definiert in Zeile 43 der Datei `Renderer.h`.

#### 8.39.2.3 `int` `Renderer::Viewport_info::height`

Höhe des dargestellten Bereichs.

Definiert in Zeile 51 der Datei `Renderer.h`.

#### 8.39.2.4 `bool` `Renderer::Viewport_info::invertcut`

Nur Elemente unterhalb der durch `cut` definierte Ebene darstellen (Momentan nicht verwendet).

Definiert in Zeile 44 der Datei `Renderer.h`.

#### 8.39.2.5 `float` `Renderer::Viewport_info::rotationX`

Rotation der Ansicht um die (Kameralokale) X-Achse.

Definiert in Zeile 41 der Datei `Renderer.h`.

#### 8.39.2.6 `float` `Renderer::Viewport_info::rotationY`

Rotation der Ansicht um die Y-Achse.

Definiert in Zeile 40 der Datei `Renderer.h`.

#### 8.39.2.7 float `Renderer::Viewport_info::scale`

Skalierungsfaktor für das 3D-Objekt.

Definiert in Zeile 52 der Datei `Renderer.h`.

#### 8.39.2.8 bool `Renderer::Viewport_info::show_extrapolated`

Extrapolierte Elemente anzeigen.

Definiert in Zeile 48 der Datei `Renderer.h`.

#### 8.39.2.9 bool `Renderer::Viewport_info::show_sensordata`

Sensordaten als Punkte anzeigen.

Definiert in Zeile 49 der Datei `Renderer.h`.

#### 8.39.2.10 `RenderMode` `Renderer::Viewport_info::showEdges`

Modus der Darstellung von Kanten des 3D-Objekts.

Definiert in Zeile 46 der Datei `Renderer.h`.

#### 8.39.2.11 `RenderMode` `Renderer::Viewport_info::showFaces`

Modus der Darstellung von Flächen des 3D-Objekts.

Definiert in Zeile 47 der Datei `Renderer.h`.

#### 8.39.2.12 `RenderMode` `Renderer::Viewport_info::showPoints`

Modus der Darstellung von Punkten des 3D-Objekts.

Definiert in Zeile 45 der Datei `Renderer.h`.

#### 8.39.2.13 int `Renderer::Viewport_info::width`

Breite des dargestellten Bereichs.

Definiert in Zeile 50 der Datei `Renderer.h`.

#### 8.39.2.14 float `Renderer::Viewport_info::zoom`

Aktueller Zoomfaktor.

Definiert in Zeile 39 der Datei `Renderer.h`.

Die Dokumentation für diese Struktur wurde erzeugt aufgrund der Datei:

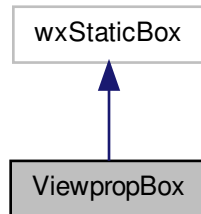
- `/daten/Projekte/eclipse_workspace/simpleanalyzer-gui/src/GUI/Renderer.h`

## 8.40 ViewpropBox Klassenreferenz

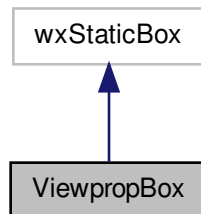
Oberfläche zum Verändern/Anzeigen der Visualisierungsoptionen.

```
#include <ViewpropBox.h>
```

Klassendiagramm für ViewpropBox:



Zusammengehörigkeiten von ViewpropBox:



## Öffentliche Methoden

- [ViewpropBox](#) (wxWindow \*parent)  
*Der Konstruktor.*
- void [resize](#) ()  
*Behandelt Größenänderungen und passt die Positionen der Komponenten an.*
- wxSpinCtrl \* [getColorRangeMaxEdit](#) ()  
*Gibt das Eingabefeld für die maximal Visualisierte Temperatur (entspricht der Farbe Rot) zurück.*
- wxSpinCtrl \* [getColorRangeMinEdit](#) ()  
*Gibt das Eingabefeld für die minimal Visualisierte Temperatur (entspricht der Farbe Blau) zurück.*
- wxRadioBox \* [getEdgesCheckBox](#) ()  
*Gibt das Auswahlfeld für den Darstellungsmodus von Kanten zurück.*
- wxRadioBox \* [getFacesCheckBox](#) ()  
*Gibt das Auswahlfeld für den Darstellungsmodus von Flächen zurück.*
- wxCheckListBox \* [getMatVisibilityListBox](#) ()  
*Gibt das Auswahlfeld für die Sichtbarkeit von Materialien zurück.*
- wxRadioBox \* [getPointsCheckBox](#) ()  
*Gibt das Auswahlfeld für den Darstellungsmodus von Punkten zurück.*

- wxCheckBox \* [getShowExtrapolatedCheckBox](#) ()  
*Gibt die Checkbox zum Anzeigen Extrapolierter Elemente zurück.*
- wxCheckBox \* [getShowShowSensorData](#) ()  
*Gibt die Checkbox zum Anzeigen der Sensordaten als Punkte zurück.*
- wxTextCtrl \* [getViewScaleEdit](#) ()  
*Gibt das Eingabefeld für einen Skalierungsfaktor für das 3D-Objekt zurück.*
- virtual [~ViewpropBox](#) ()  
*Der Destruktor.*

## Private Attribute

- wxRadioBox \* [pointsCheckBox](#)  
*Auswahlfeld für den Darstellungsmodus von Punkten.*
- wxRadioBox \* [edgesCheckBox](#)  
*Auswahlfeld für den Darstellungsmodus von Kanten.*
- wxRadioBox \* [facesCheckBox](#)  
*Auswahlfeld für den Darstellungsmodus von Flächen.*
- wxStaticText \* [matVisualizationLbl](#)  
*Beschriftung für das Auswahlfeld für die Sichtbarkeit von Materialien.*
- wxCheckListBox \* [matVisibilityListBox](#)  
*Auswahlfeld für die Sichtbarkeit von Materialien.*
- wxCheckBox \* [showExtrapolatedCheckBox](#)  
*Checkbox zum Anzeigen Extrapolierter Elemente.*
- wxCheckBox \* [showShowSensorData](#)  
*Checkbox zum Anzeigen der Sensordaten als Punkte.*
- wxStaticText \* [colorRangeLbl](#)  
*Beschriftung für die Eingabefelder des zur Visualisierung verwendeten Temperaturbereichs.*
- wxSpinCtrl \* [colorRangeMinEdit](#)  
*Eingabefeld für die minimal Visualisierte Temperatur (entspricht der Farbe Blau).*
- wxSpinCtrl \* [colorRangeMaxEdit](#)  
*Eingabefeld für die maximal Visualisierte Temperatur (entspricht der Farbe Rot).*
- wxStaticText \* [viewScaleLbl](#)  
*Beschriftung für das Eingabefeld eines Skalierungsfaktors für das 3D-Objekt.*
- wxTextCtrl \* [viewScaleEdit](#)  
*Eingabefeld für einen Skalierungsfaktor für das 3D-Objekt.*

### 8.40.1 Ausführliche Beschreibung

Oberfläche zum Verändern/Anzeigen der Visualisierungsoptionen.

Diese Klasse verwaltet nur das Layout des Visualisierungsoptionen-Bereichs. Die Funktionalität wird in [GUIMain-Window](#) behandelt.

Definiert in Zeile 19 der Datei ViewpropBox.h.

### 8.40.2 Beschreibung der Konstruktoren und Destruktoren

#### 8.40.2.1 ViewpropBox::ViewpropBox ( wxWindow \* parent )

Der Konstruktor.



## Parameter

<i>parent</i>	Die übergeordnete Komponente.
---------------	-------------------------------

Definiert in Zeile 17 der Datei ViewpropBox.cpp.

#### 8.40.2.2 ViewpropBox::~~ViewpropBox ( ) [virtual]

Der Destruktor.

Definiert in Zeile 105 der Datei ViewpropBox.cpp.

### 8.40.3 Dokumentation der Elementfunktionen

#### 8.40.3.1 wxSpinCtrl \* ViewpropBox::getColorRangeMaxEdit ( )

Gibt das Eingabefeld für die maximal Visualisierte Temperatur (entspricht der Farbe Rot) zurück.

Definiert in Zeile 69 der Datei ViewpropBox.cpp.

#### 8.40.3.2 wxSpinCtrl \* ViewpropBox::getColorRangeMinEdit ( )

Gibt das Eingabefeld für die minimal Visualisierte Temperatur (entspricht der Farbe Blau) zurück.

Definiert in Zeile 73 der Datei ViewpropBox.cpp.

#### 8.40.3.3 wxRadioBox \* ViewpropBox::getEdgesCheckBox ( )

Gibt das Auswahlfeld für den Darstellungsmodus von Kanten zurück.

Definiert in Zeile 77 der Datei ViewpropBox.cpp.

#### 8.40.3.4 wxRadioBox \* ViewpropBox::getFacesCheckBox ( )

Gibt das Auswahlfeld für den Darstellungsmodus von Flächen zurück.

Definiert in Zeile 81 der Datei ViewpropBox.cpp.

#### 8.40.3.5 wxCheckListBox \* ViewpropBox::getMatVisibilityListBox ( )

Gibt das Auswahlfeld für die Sichtbarkeit von Materialien zurück.

Definiert in Zeile 85 der Datei ViewpropBox.cpp.

#### 8.40.3.6 wxRadioBox \* ViewpropBox::getPointsCheckBox ( )

Gibt das Auswahlfeld für den Darstellungsmodus von Punkten zurück.

Definiert in Zeile 89 der Datei ViewpropBox.cpp.

#### 8.40.3.7 wxCheckBox \* ViewpropBox::getShowExtrapolatedCheckBox ( )

Gibt die Checkbox zum Anzeigen Extrapolierter Elemente zurück.

Definiert in Zeile 93 der Datei ViewpropBox.cpp.

#### 8.40.3.8 wxCheckBox \* ViewpropBox::getShowShowSensorData ( )

Gibt die Checkbox zum Anzeigen der Sensordaten als Punkte zurück.

Definiert in Zeile 97 der Datei ViewpropBox.cpp.

#### 8.40.3.9 wxTextCtrl \* ViewpropBox::getViewScaleEdit ( )

Gibt das Eingabefeld für einen Skalierungsfaktor für das 3D-Objekt zurück.

Diese Skalierung ist rein optisch.

Definiert in Zeile 101 der Datei ViewpropBox.cpp.

#### 8.40.3.10 void ViewpropBox::resize ( )

Behandelt Größenänderungen und passt die Positionen der Komponenten an.

Definiert in Zeile 48 der Datei ViewpropBox.cpp.

### 8.40.4 Dokumentation der Datenelemente

#### 8.40.4.1 wxStaticText\* ViewpropBox::colorRangeLbl [private]

Beschriftung für die Eingabefelder des zur Visualisierung verwendeten Temperaturbereichs.

Definiert in Zeile 120 der Datei ViewpropBox.h.

#### 8.40.4.2 wxSpinCtrl\* ViewpropBox::colorRangeMaxEdit [private]

Eingabefeld für die maximal Visualisierte Temperatur (entspricht der Farbe Rot).

Definiert in Zeile 130 der Datei ViewpropBox.h.

#### 8.40.4.3 wxSpinCtrl\* ViewpropBox::colorRangeMinEdit [private]

Eingabefeld für die minimal Visualisierte Temperatur (entspricht der Farbe Blau).

Definiert in Zeile 125 der Datei ViewpropBox.h.

#### 8.40.4.4 wxRadioBox\* ViewpropBox::edgesCheckBox [private]

Auswahlfeld für den Darstellungsmodus von Kanten.

Definiert in Zeile 90 der Datei ViewpropBox.h.

#### 8.40.4.5 wxRadioBox\* ViewpropBox::facesCheckBox [private]

Auswahlfeld für den Darstellungsmodus von Flächen.

Definiert in Zeile 95 der Datei ViewpropBox.h.

#### 8.40.4.6 wxCheckListBox\* ViewpropBox::matVisibilityListBox [private]

Auswahlfeld für die Sichtbarkeit von Materialien.

Definiert in Zeile 105 der Datei ViewpropBox.h.

**8.40.4.7 wxStaticText\* ViewpropBox::matVisualizationLbl [private]**

Beschriftung für das Auswahlfeld für die Sichtbarkeit von Materialien.

Definiert in Zeile 100 der Datei ViewpropBox.h.

**8.40.4.8 wxRadioBox\* ViewpropBox::pointsCheckBox [private]**

Auswahlfeld für den Darstellungsmodus von Punkten.

Definiert in Zeile 85 der Datei ViewpropBox.h.

**8.40.4.9 wxCheckBox\* ViewpropBox::showExtrapolatedCheckBox [private]**

Checkbox zum Anzeigen Extrapolierter Elemente.

Definiert in Zeile 110 der Datei ViewpropBox.h.

**8.40.4.10 wxCheckBox\* ViewpropBox::showShowSensorData [private]**

Checkbox zum Anzeigen der Sensordaten als Punkte.

Definiert in Zeile 115 der Datei ViewpropBox.h.

**8.40.4.11 wxTextCtrl\* ViewpropBox::viewScaleEdit [private]**

Eingabefeld für einen Skalierungsfaktor für das 3D-Objekt.

Diese Skalierung ist rein optisch.

Definiert in Zeile 140 der Datei ViewpropBox.h.

**8.40.4.12 wxStaticText\* ViewpropBox::viewScaleLbl [private]**

Beschriftung für das Eingabefeld eines Skalierungsfaktors für das 3D-Objekt.

Definiert in Zeile 135 der Datei ViewpropBox.h.

Die Dokumentation für diese Klasse wurde erzeugt aufgrund der Dateien:

- [/daten/Projekte/eclipse\\_workspace/simpleanalyzer-gui/src/GUI/ViewpropBox.h](#)
- [/daten/Projekte/eclipse\\_workspace/simpleanalyzer-gui/src/GUI/ViewpropBox.cpp](#)

## 8.41 Utils::Visualization\_info Strukturreferenz

Informationen über die Farbgebung bei der Visualisierung.

```
#include <utils.h>
```

### Öffentliche Attribute

- int [max\\_visualisation\\_temp](#) = 100  
*maximal Visualisierte Temperatur (entspricht der Farbe Rot).*
- int [min\\_visualisation\\_temp](#) = 0  
*minimal Visualisierte Temperatur (entspricht der Farbe Blau).*

### 8.41.1 Ausführliche Beschreibung

Informationen über die Farbgebung bei der Visualisierung.

Definiert in Zeile 47 der Datei `utils.h`.

### 8.41.2 Dokumentation der Datenelemente

#### 8.41.2.1 `int Utils::Visualization_info::max_visualisation_temp = 100`

maximal Visualisierte Temperatur (entspricht der Farbe Rot).

Definiert in Zeile 48 der Datei `utils.h`.

#### 8.41.2.2 `int Utils::Visualization_info::min_visualisation_temp = 0`

minimal Visualisierte Temperatur (entspricht der Farbe Blau).

Definiert in Zeile 49 der Datei `utils.h`.

Die Dokumentation für diese Struktur wurde erzeugt aufgrund der Datei:

- `/daten/Projekte/eclipse_workspace/simpleanalyzer-gui/src/processing/`[utils.h](#)

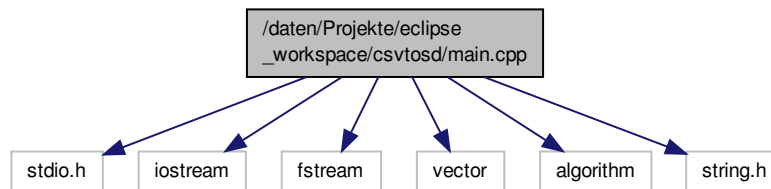
## Kapitel 9

# Datei-Dokumentation

### 9.1 /daten/Projekte/eclipse\_workspace/csvtosd/main.cpp-Dateireferenz

```
#include <stdio.h>
#include <iostream>
#include <fstream>
#include <vector>
#include <algorithm>
#include <string.h>
```

Include-Abhängigkeitsdiagramm für main.cpp:



### Klassen

- class [CsvToSdConverter](#)  
*Konverter von .csv zu .tsd.*
- struct [CsvToSdConverter::Options](#)  
*Struktur für die Programmeinstellungen.*

### Funktionen

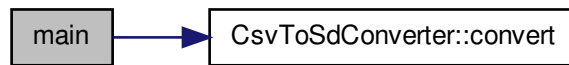
- int [main](#) (int argc, char \*argv[])

#### 9.1.1 Dokumentation der Funktionen

##### 9.1.1.1 int main ( int argc, char \* argv[] )

Definiert in Zeile 694 der Datei main.cpp.

Hier ist ein Graph, der zeigt, was diese Funktion aufruft:



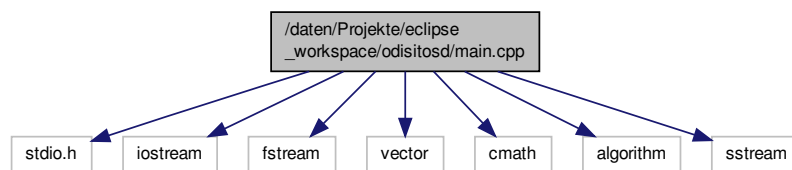
## 9.2 /daten/Projekte/eclipse\_workspace/odisitosd/main.cpp-Dateireferenz

```

#include <stdio.h>
#include <iostream>
#include <fstream>
#include <vector>
#include <cmath>
#include <algorithm>
#include <sstream>

```

Include-Abhängigkeitsdiagramm für main.cpp:



### Klassen

- class [OdisiToSdConverter](#)  
*Konverter von ODISI zu .tsd.*
- struct [OdisiToSdConverter::Options](#)  
*Struktur für die Programmeinstellungen.*

### Funktionen

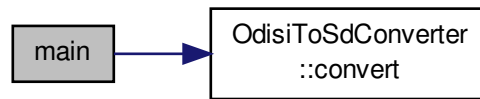
- int [main](#) (int argc, char \*argv[])

#### 9.2.1 Dokumentation der Funktionen

##### 9.2.1.1 int main ( int argc, char \* argv[] )

Definiert in Zeile 996 der Datei main.cpp.

Hier ist ein Graph, der zeigt, was diese Funktion aufruft:



### 9.3 doxygen\_dep\_dummy.h-Dateireferenz

#### Klassen

- class `std::vector< T >`

#### Namensbereiche

- `std`

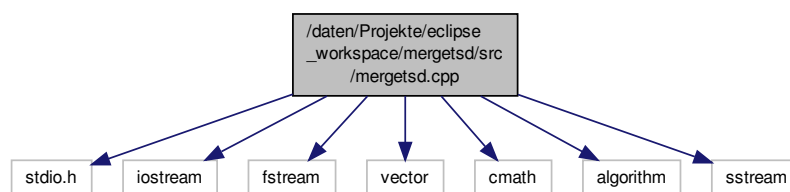
### 9.4 /daten/Projekte/eclipse\_workspace/mergetsd/src/mergetsd.cpp-Dateireferenz

```

#include <stdio.h>
#include <iostream>
#include <fstream>
#include <vector>
#include <cmath>
#include <algorithm>
#include <sstream>

```

Include-Abhängigkeitsdiagramm für mergetsd.cpp:



#### Klassen

- class `TsdMerger`  
*Zusammenführen zweier .tsd-Dateien.*
- struct `TsdMerger::Options`  
*Struktur für die Programmeinstellungen.*

## Funktionen

- int [main](#) (int argc, char \*argv[])

### 9.4.1 Dokumentation der Funktionen

#### 9.4.1.1 int main ( int argc, char \* argv[] )

Definiert in Zeile 366 der Datei mergetsd.cpp.

Hier ist ein Graph, der zeigt, was diese Funktion aufruft:



## 9.5 /daten/Projekte/eclipse\_workspace/README.md-Dateireferenz

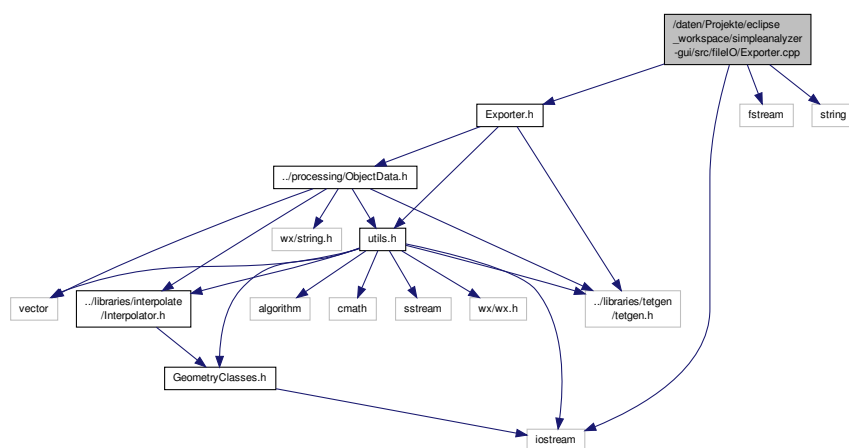
## 9.6 /daten/Projekte/eclipse\_workspace/simpleanalyzer-gui/src/fileIO/Exporter.cpp-Dateireferenz

```

#include "Exporter.h"
#include <iostream>
#include <fstream>
#include <string>

```

Include-Abhängigkeitsdiagramm für Exporter.cpp:



## Variablen

- const int [tetface\\_indices](#) [4][3]



## 9.6.1 Variablen-Dokumentation

### 9.6.1.1 `const int tetface_indices[4][3]`

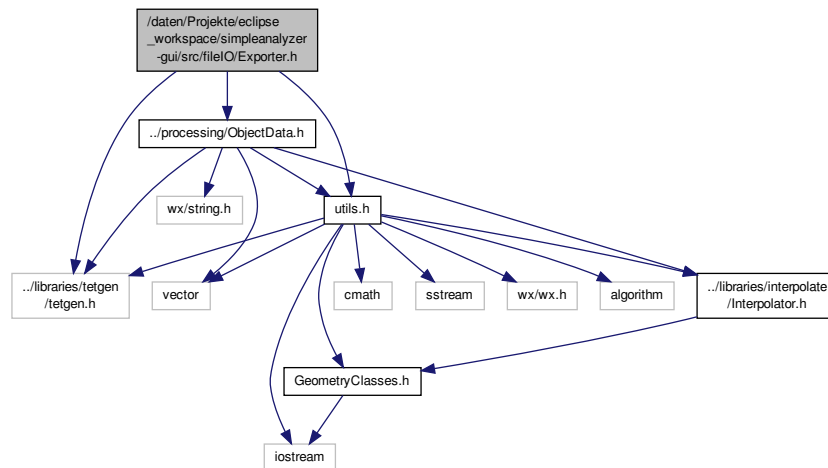
#### Initialisierung:

```
= { { 0, 1, 2 }, { 0, 1, 3 }, { 0, 2, 3 }, { 1,
    2, 3 } }
```

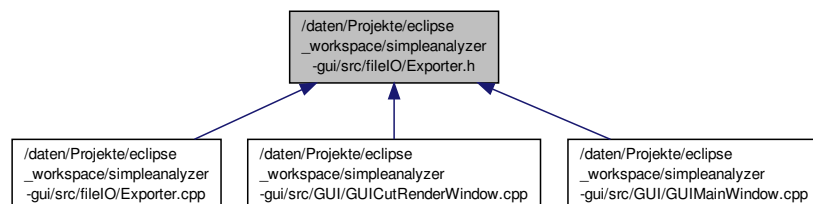
Definiert in Zeile 20 der Datei Exporter.cpp.

## 9.7 /daten/Projekte/eclipse\_workspace/simpleanalyzer-gui/src/fileIO/Exporter.h-Dateireferenz

```
#include "../libraries/tetgen/tetgen.h"
#include "../processing/ObjectData.h"
#include "../processing/Utils.h"
Include-Abhängigkeitsdiagramm für Exporter.h:
```



Dieser Graph zeigt, welche Datei direkt oder indirekt diese Datei enthält:

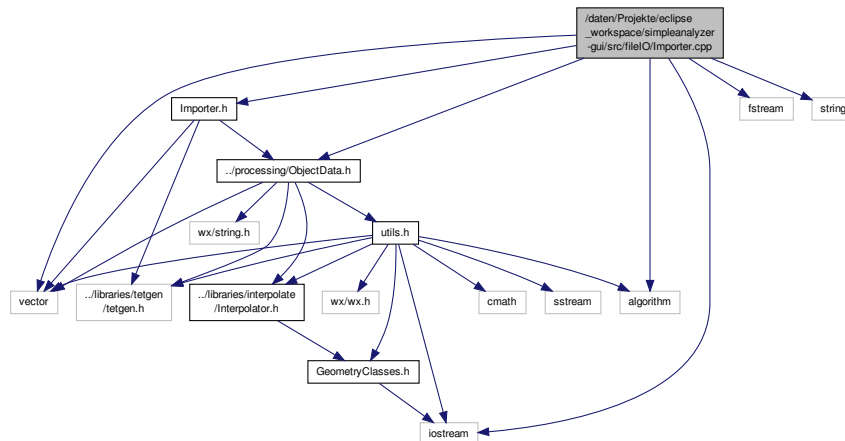


## Klassen

- class [Exporter](#)  
*Export der gewonnenen Daten.*

## 9.8 /daten/Projekte/eclipse\_workspace/simpleanalyzer-gui/src/fileO/Importer.cpp-Dateireferenz

```
#include "Importer.h"
#include <iostream>
#include <fstream>
#include <string>
#include <vector>
#include <algorithm>
#include "../processing/ObjectData.h"
Include-Abhängigkeitsdiagramm für Importer.cpp:
```



### Makrodefinitionen

- `#define PATH_SEPARATOR '/'`

### Funktionen

- `int getFaceIndex (string data, bool withUV)`  
Extrahiert den Index einer Fläche aus einem Textblock einer Zeile der .obj-Datei.
- `string getTextBlock (string data, int n)`  
Gibt den n-ten durch Leerzeichen abgetrennten Block aus einem String zurück.

#### 9.8.1 Makro-Dokumentation

##### 9.8.1.1 #define PATH\_SEPARATOR '/'

Definiert in Zeile 21 der Datei Importer.cpp.

#### 9.8.2 Dokumentation der Funktionen

##### 9.8.2.1 int getFaceIndex ( string data, bool withUV )

Extrahiert den Index einer Fläche aus einem Textblock einer Zeile der .obj-Datei.

**Parameter**

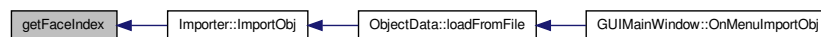
<i>data</i>	Der zu untersuchende Block.
<i>withUV</i>	Enthält die obj-Datei auch Texturdatenindices?

**Rückgabe**

Der Flächenindex.

Definiert in Zeile 34 der Datei Importer.cpp.

Hier ist ein Graph der zeigt, wo diese Funktion aufgerufen wird:

**9.8.2.2 string getTextBlock ( string data, int n )**

Gibt den n-ten durch Leerzeichen abgetrennten Block aus einem String zurück.

**Parameter**

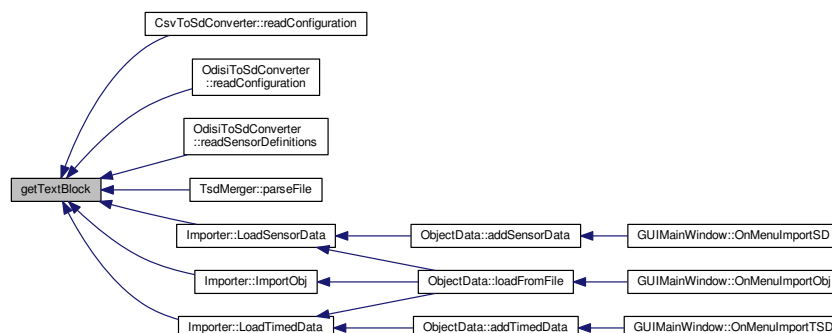
<i>data</i>	Der Ausgangsstring.
<i>n</i>	Index des zu findenden Blocks.

**Rückgabe**

Der n-te durch Leerzeichen getrennte Teilstring. "" Bei ungültigem Index.

Definiert in Zeile 51 der Datei Importer.cpp.

Hier ist ein Graph der zeigt, wo diese Funktion aufgerufen wird:

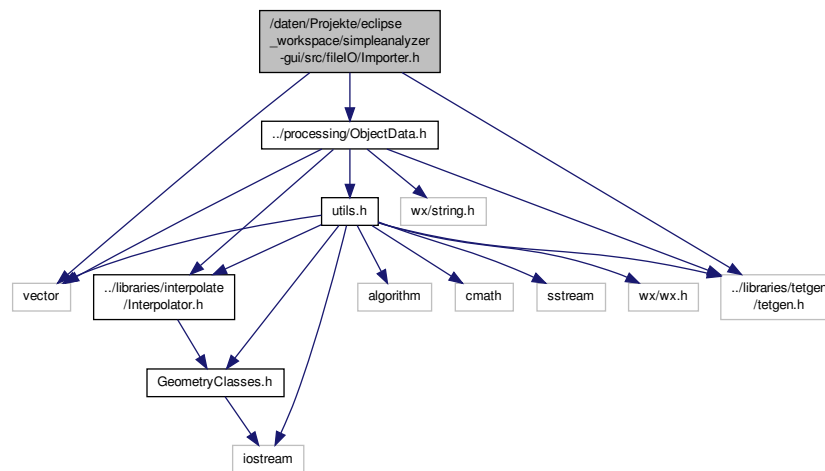
**9.9 /daten/Projekte/eclipse\_workspace/simpleanalyzer-gui/src/fileIO/Importer.h-Dateireferenz**

```

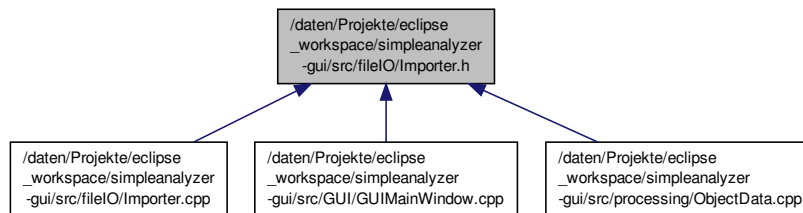
#include "../libraries/tetgen/tetgen.h"
#include "../processing/ObjectData.h"
#include <vector>

```

Include-Abhängigkeitsdiagramm für `Importer.h`:



Dieser Graph zeigt, welche Datei direkt oder indirekt diese Datei enthält:



## Klassen

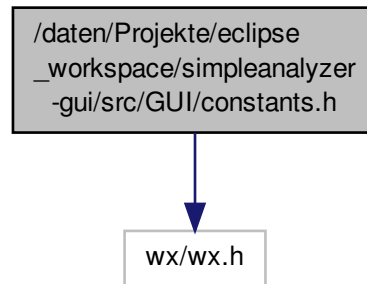
- class `Importer`

*Importieren von 3D-Modell (.obj) und Sensordaten (.tsd oder .sd).*

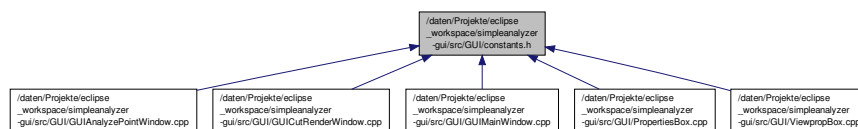
## 9.10 /daten/Projekte/eclipse\_workspace/simpleanalyzer\_gui/src/GUI/constants.h-Dateireferenz

```
#include <wx/wx.h>
```

Include-Abhängigkeitsdiagramm für constants.h:



Dieser Graph zeigt, welche Datei direkt oder indirekt diese Datei enthält:



## Aufzählungen

- enum `EventID` {  
`ID_ABOUT = 1, ID_TEST, ID_IMPORT_OBJ, ID_IMPORT_SD,`  
`ID_RECALCBT, ID_MATERIALBOX, ID_ANALYZE, ID_GENERAL_PROP,`  
`ID_IMMEDIATE_UPDATE_PROP, ID_GENERAL_VIEW_PROP, ID_CHECKLISTBOX_VIEW_PROP, ID_CHANGE_ACTIVE_OBJ,`  
`ID_ANALYZE_POINT, ID_ANALYZE_POINT_BT, ID_CUT_CANVAS, ID_RENDER_CUT,`  
`ID_RENDER_CUT_BT, ID_CUT_TRI_EDIT, ID_DELETE_ACTIVE_OBJ, ID_IMPORT_TSD,`  
`ID_SD_BOX, ID_SD_TIMELINE, ID_ANALYZE_MARKER_CB, ID_CLEAR_MARKER_BT,`  
`ID_MARKER_NEXT_BT, ID_MARKER_PREV_BT, ID_EXPORT_CUT_IMG_BT, ID_EXPORT_VIEWPORT,`  
`ID_FIND_MAX_BT, ID_AUTO_UPDATE_CB, ID_EXPORT_VTK, ID_EXPORT_CUT_CSV_BT,`  
`ID_COLORSCALE_PROP, ID_COLORSCALE_COLORBT, ID_OPEN_MANUAL }`

*IDs für die Events der Programmoberfläche.*

## Variablen

- const int `NUMBER_OF_INTERPOLATION_MODES = 2`  
*Anzahl der verfügbaren Interpolationsmodi.*
- const wxString `INTERPOLATION_MODE_STRINGS [NUMBER_OF_INTERPOLATION_MODES] = {wxT("Linear"), wxT("Logarithmisch")}`

*Bezeichnungen für die von "Interpolator" verwendeten Interpolationsmodi.*

### 9.10.1 Dokumentation der Aufzählungstypen

#### 9.10.1.1 enum EventID

IDs für die Events der Programmoberfläche.

Müssen kleiner als wxID\_LOWEST (wxWidgets 2.8: 4999) sein!

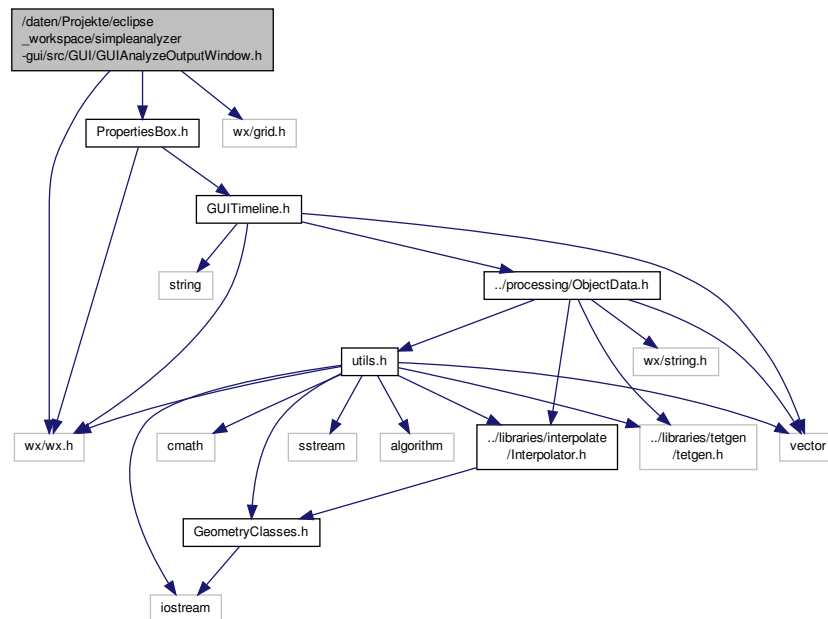
##### Aufzählungswerte

**ID\_ABOUT**  
**ID\_TEST**  
**ID\_IMPORT\_OBJ**  
**ID\_IMPORT\_SD**  
**ID\_RECALCBT**  
**ID\_MATERIALBOX**  
**ID\_ANALYZE**  
**ID\_GENERAL\_PROP**  
**ID\_IMMEDIATE\_UPDATE\_PROP**  
**ID\_GENERAL\_VIEW\_PROP**  
**ID\_CHECKLISTBOX\_VIEW\_PROP**  
**ID\_CHANGE\_ACTIVE\_OBJ**  
**ID\_ANALYZE\_POINT**  
**ID\_ANALYZE\_POINT\_BT**  
**ID\_CUT\_CANVAS**  
**ID\_RENDER\_CUT**  
**ID\_RENDER\_CUT\_BT**  
**ID\_CUT\_TRI\_EDIT**  
**ID\_DELETE\_ACTIVE\_OBJ**  
**ID\_IMPORT\_TSD**  
**ID\_SD\_BOX**  
**ID\_SD\_TIMELINE**  
**ID\_ANALYZE\_MARKER\_CB**  
**ID\_CLEAR\_MARKER\_BT**  
**ID\_MARKER\_NEXT\_BT**  
**ID\_MARKER\_PREV\_BT**  
**ID\_EXPORT\_CUT\_IMG\_BT**  
**ID\_EXPORT\_VIEWPORT**  
**ID\_FIND\_MAX\_BT**  
**ID\_AUTO\_UPDATE\_CB**  
**ID\_EXPORT\_VTK**  
**ID\_EXPORT\_CUT\_CSV\_BT**  
**ID\_COLORSCALE\_PROP**  
**ID\_COLORSCALE\_COLORBT**  
**ID\_OPEN\_MANUAL**

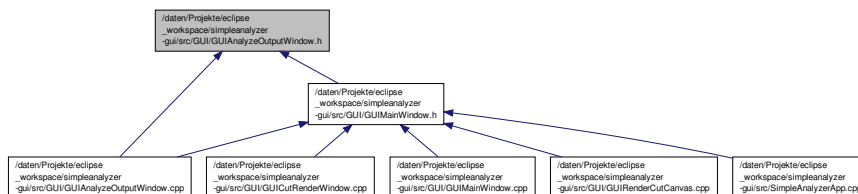
Definiert in Zeile 28 der Datei constants.h.



Include-Abhängigkeitsdiagramm für GUIAnalyzeOutputWindow.h:



Dieser Graph zeigt, welche Datei direkt oder indirekt diese Datei enthält:



## Klassen

- class [GUIAnalyzeOutputWindow](#)  
*Übersichtsfenster über die Analysedaten.*

### 9.13 /daten/Projekte/eclipse\_workspace/simpleanalyzer-gui/src/GUI/GUIAnalyzePointWindow.cpp-Dateireferenz

```

#include "GUIAnalyzePointWindow.h"
#include <iostream>
#include <sstream>
#include "constants.h"
#include "../processing/Analyzer.h"
#include "../SimpleAnalyzerApp.h"

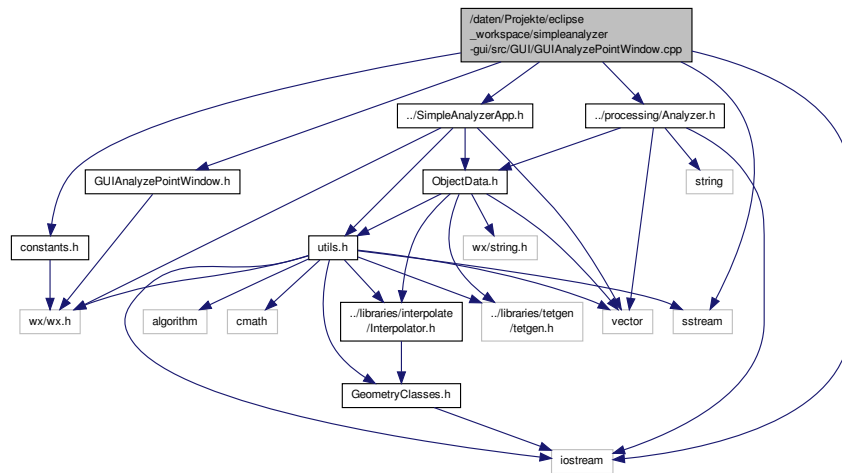
```



## 9.14

/daten/Projekte/eclipse\_workspace/simpleanalyzer-gui/src/GUI/GUIAnalyzePointWindow.h-Dateireferenz231

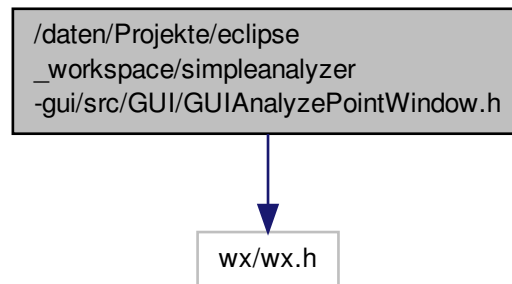
Include-Abhängigkeitsdiagramm für GUIAnalyzePointWindow.cpp:



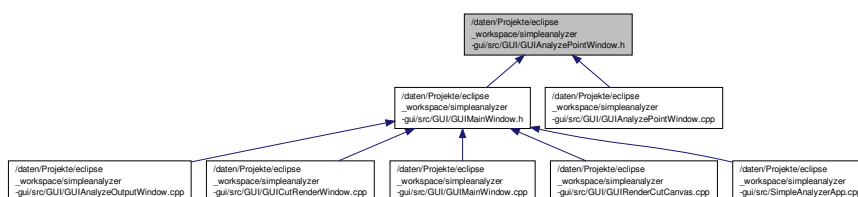
## 9.14 /daten/Projekte/eclipse\_workspace/simpleanalyzer-gui/src/GUI/GUIAnalyzePointWindow.h-Dateireferenz

```
#include <wx/wx.h>
```

Include-Abhängigkeitsdiagramm für GUIAnalyzePointWindow.h:



Dieser Graph zeigt, welche Datei direkt oder indirekt diese Datei enthält:



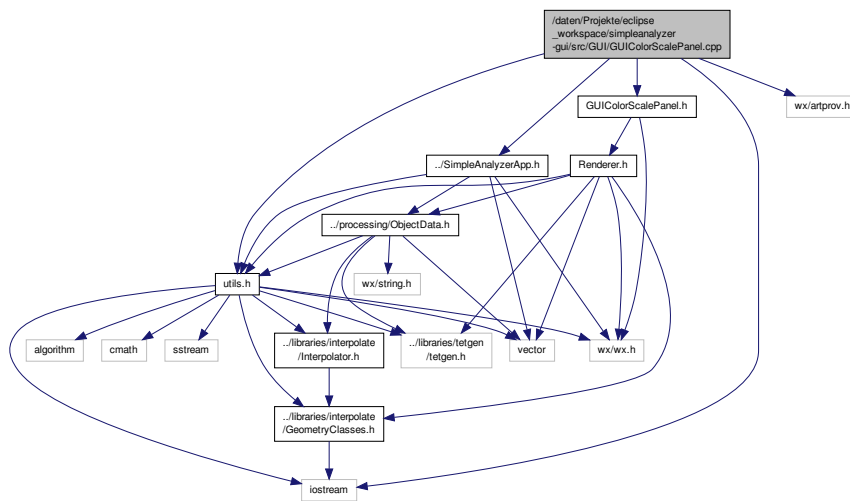
## Klassen

- class [GUIAnalyzePointWindow](#)  
*Analysefenster für einen Punkt.*

### 9.15 /daten/Projekte/eclipse\_workspace/simpleanalyzer-gui/src/GUI/GUIColorScale-Panel.cpp-Dateireferenz

```
#include "GUIColorScalePanel.h"
#include <iostream>
#include "../SimpleAnalyzerApp.h"
#include "../processing/Utils.h"
#include <wx/artprov.h>
```

Include-Abhängigkeitsdiagramm für GUIColorScalePanel.cpp:



## Makrodefinitionen

- `#define MIN_WIDTH 5`
- `#define MIN_HEIGHT 5`

### 9.15.1 Makro-Dokumentation

#### 9.15.1.1 `#define MIN_HEIGHT 5`

Definiert in Zeile 19 der Datei `GUIColorScalePanel.cpp`.

#### 9.15.1.2 `#define MIN_WIDTH 5`

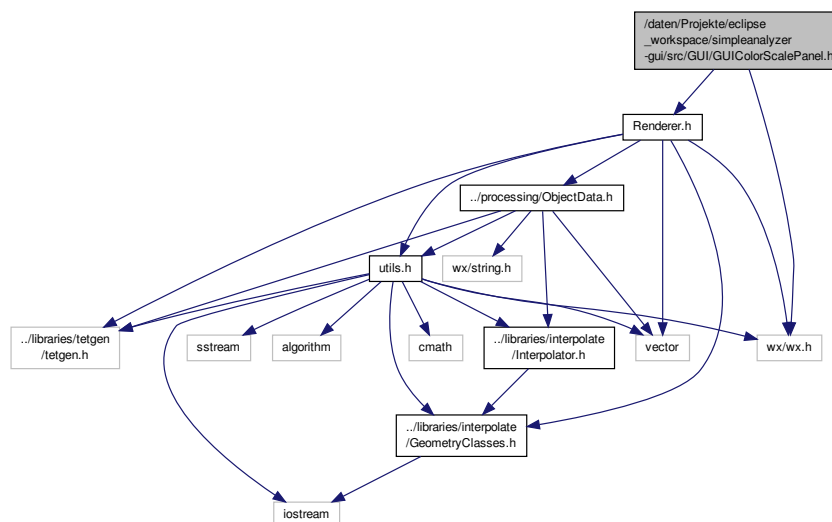
Definiert in Zeile 18 der Datei `GUIColorScalePanel.cpp`.

**9.16** /daten/Projekte/eclipse\_workspace/simpleanalyzer-gui/src/GUI/GUIColorScale-Panel.h-Dateireferenz

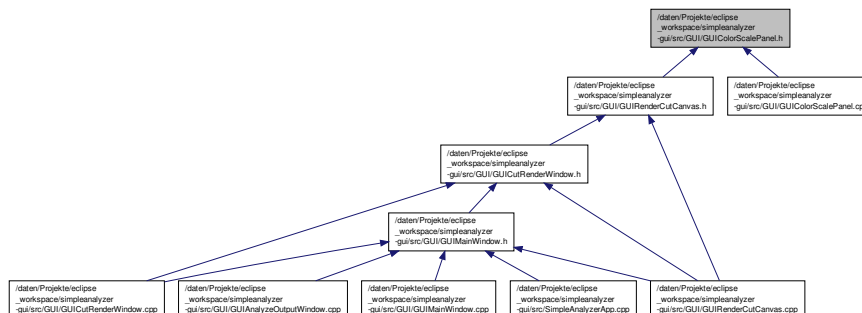
```
#include "Renderer.h"
```

```
#include <wx/wx.h>
```

Include-Abhängigkeitsdiagramm für GUIColorScalePanel.h:



Dieser Graph zeigt, welche Datei direkt oder indirekt diese Datei enthält:



## Klassen

- class `GUIColorScalePanel`

Farbige Temperaturskala für zweidimensionale Temperaturverteilung.

**9.17** /daten/Projekte/eclipse\_workspace/simpleanalyzer-gui/src/GUI/GUICutRender-Window.cpp-Dateireferenz

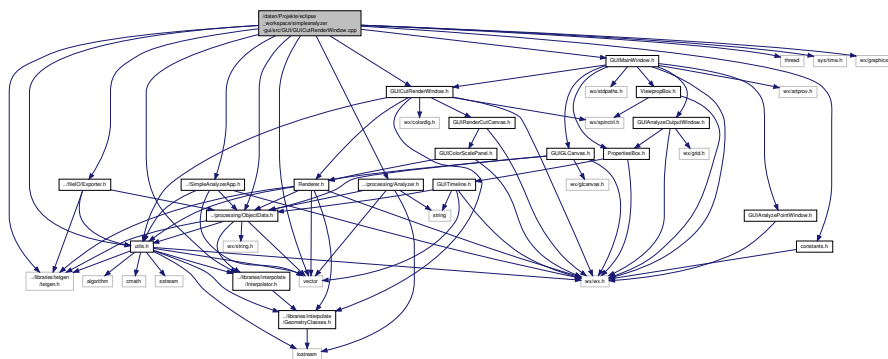
```
#include "GUICutRenderWindow.h"
```

```

#include "constants.h"
#include <vector>
#include "../SimpleAnalyzerApp.h"
#include "../processing/Analyzer.h"
#include "../processing/ObjectData.h"
#include "../processing/Utils.h"
#include "../libraries/interpolate/Interpolator.h"
#include "../libraries/tetgen/tetgen.h"
#include "../fileIO/Exporter.h"
#include "GUIMainWindow.h"
#include <thread>
#include <sys/time.h>
#include <wx/graphics.h>

```

Include-Abhängigkeitsdiagramm für GUICutRenderWindow.cpp:



## Funktionen

- void **render\_thread** (bool \*status\_flag, float \*value\_img, wxImage \*image, int width, int height, int startheight, int delta\_h, CutRender\_info \*info, Vector3D \*xvec, Vector3D \*yvec, Vector3D \*v0, vector< tetgenio \* > \*bases, ObjectData \*obj, vector< SensorPoint > \*sensor\_data, bool use\_last\_tet)

*Funktion zum verteilten berechnen der 2D-Temperaturverteilung.*

### 9.17.1 Dokumentation der Funktionen

- 9.17.1.1 void **render\_thread** ( bool \* *status\_flag*, float \* *value\_img*, wxImage \* *image*, int *width*, int *height*, int *startheight*, int *delta\_h*, CutRender\_info \* *info*, Vector3D \* *xvec*, Vector3D \* *yvec*, Vector3D \* *v0*, vector< tetgenio \* > \* *bases*, ObjectData \* *obj*, vector< SensorPoint > \* *sensor\_data*, bool *use\_last\_tet* )

Funktion zum verteilten berechnen der 2D-Temperaturverteilung.

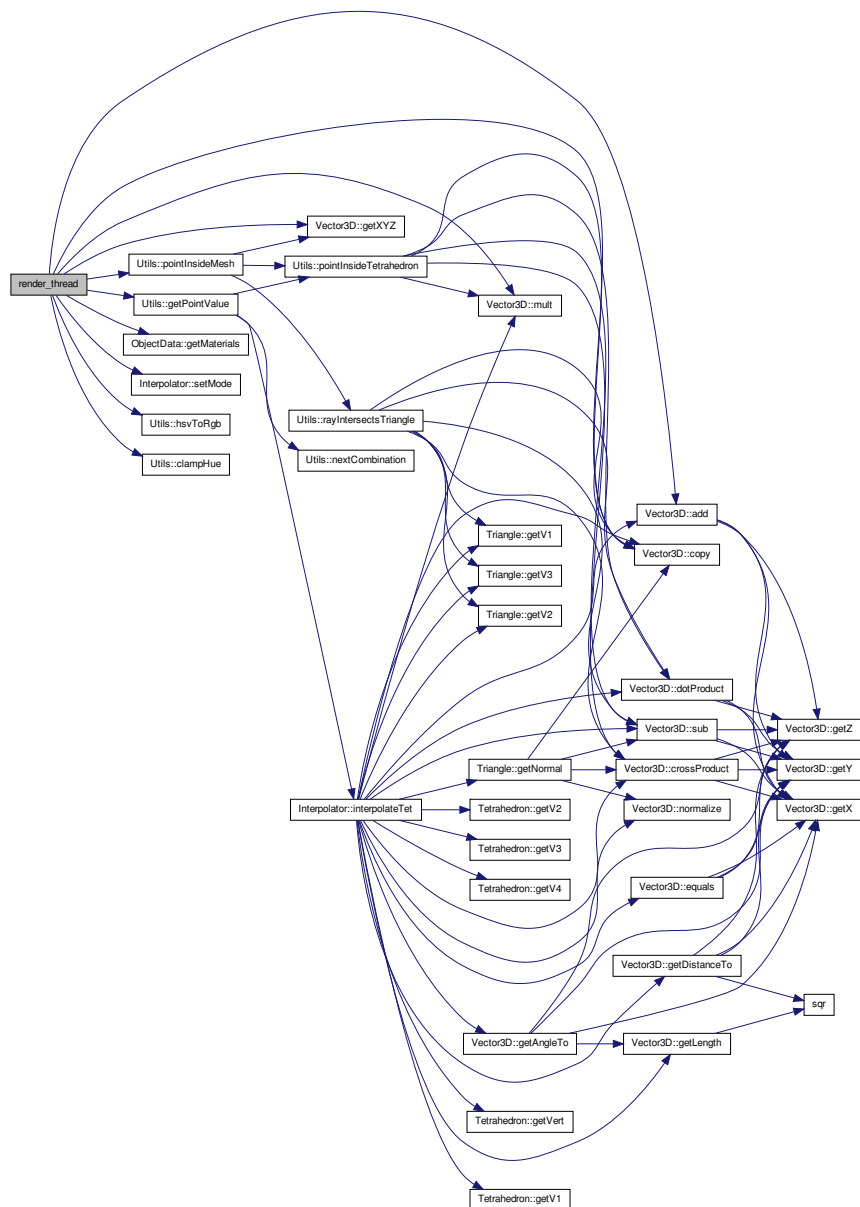
#### Parameter

<i>status_flag</i>	Zeiger auf Variable, die enthält ob der Thread beendet ist. (0 = Beendet)
<i>value_img</i>	Liste für die Daten der Temperaturverteilung.
<i>image</i>	Grafik für die Temperaturverteilung.
<i>width</i>	Breite der Temperaturverteilungsgrafik.
<i>height</i>	Höhe der Temperaturverteilungsgrafik.
<i>startheight</i>	Starthöhe für diesen Thread in der Grafik.

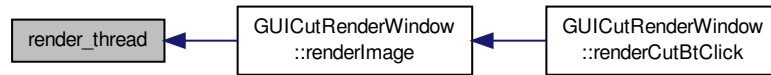
<i>delta_h</i>	Höhe des von diesem Thread zu berechnenden Streifens.
<i>info</i>	Informationen über die Eigenschaften der zu berechnenden Ebene.
<i>xvec</i>	X-Achse der Ebene.
<i>yvec</i>	Y-Achse der Ebene.
<i>v0</i>	Mittelpunkt der Ebene.
<i>bases</i>	Möglichst einfache Geometrien Geometrien der Materialien.
<i>obj</i>	Das aktuelle Objekt.
<i>sensor_data</i>	Die zu verwendenden Sensordaten.
<i>use_last_tet</i>	Versuchen, die Interpolation durch vorgezogenes Testen des zuletzt verwendeten Tetraeders zu beschleunigen. Diese Option ist verursacht Ungenauigkeiten und bietet zumeist wenig Performancegewinn.

Definiert in Zeile 167 der Datei GUICutRenderWindow.cpp.

Hier ist ein Graph, der zeigt, was diese Funktion aufruft:



Hier ist ein Graph der zeigt, wo diese Funktion aufgerufen wird:



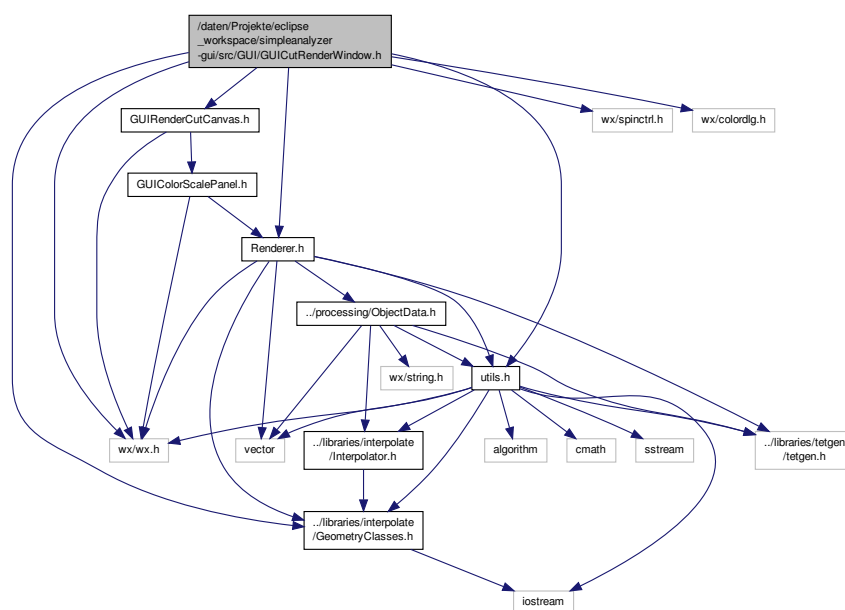
## 9.18 /daten/Projekte/eclipse\_workspace/simpleanalyzer-gui/src/GUI/GUICutRender-Window.h-Dateireferenz

```

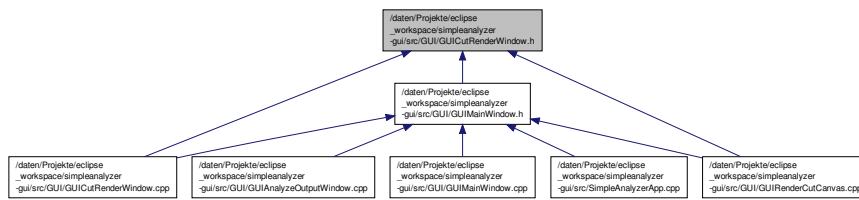
#include <wx/wx.h>
#include "GUIRenderCutCanvas.h"
#include "../libraries/interpolate/GeometryClasses.h"
#include "../processing/Utils.h"
#include <wx/spinctrl.h>
#include "Renderer.h"
#include <wx/colordlg.h>

```

Include-Abhängigkeitsdiagramm für GUICutRenderWindow.h:



Dieser Graph zeigt, welche Datei direkt oder indirekt diese Datei enthält:



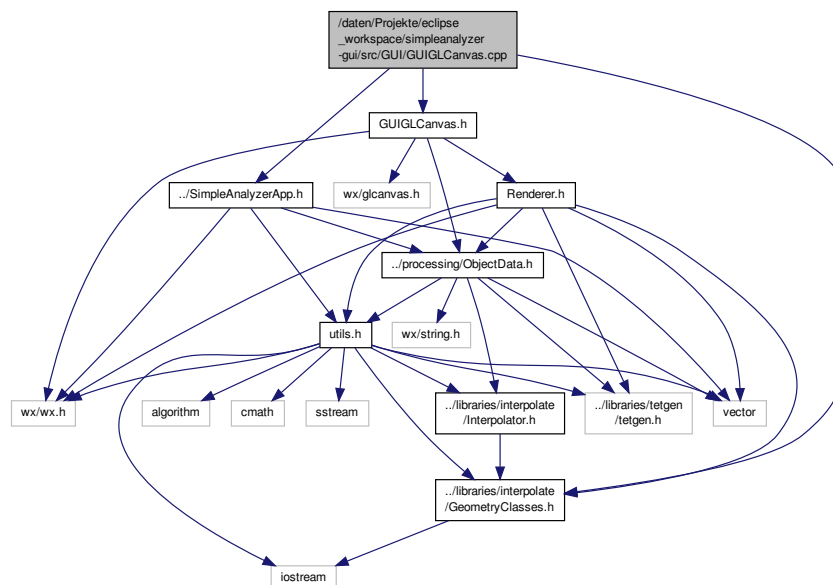
## Klassen

- class [GUICutRenderWindow](#)

*Fenster zum erstellen zweidimensionaler Temperaturverteilungen.*

## 9.19 /daten/Projekte/eclipse\_workspace/simpleanalyzer-gui/src/GUI/GUIGLCanvas.cpp--Dateireferenz

```
#include "GUIGLCanvas.h"
#include "../SimpleAnalyzerApp.h"
#include "../libraries/interpolate/GeometryClasses.h"
Include-Abhängigkeitsdiagramm für GUIGLCanvas.cpp:
```



## Variablen

- int [attrib\\_list](#) [] = { WX\_GL\_RGBA, WX\_GL\_DOUBLEBUFFER, WX\_GL\_DEPTH\_SIZE, 16, 0 }

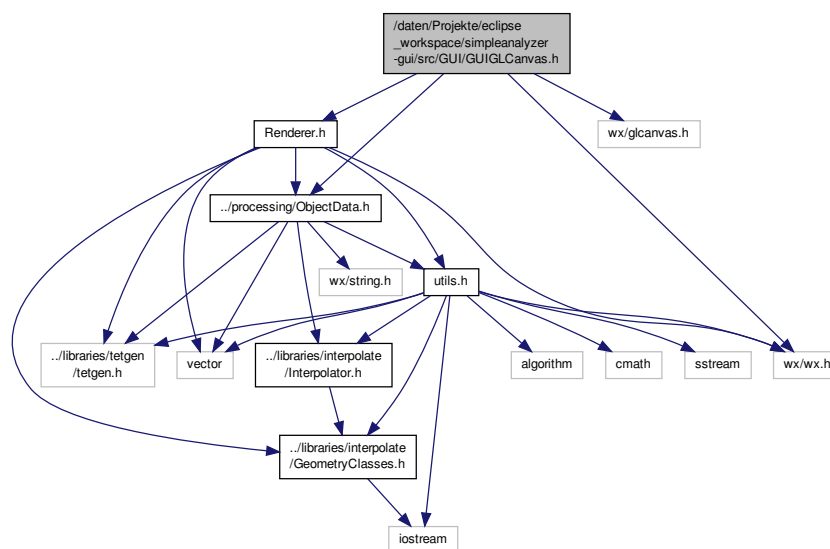
## 9.19.1 Variablen-Dokumentation

9.19.1.1 `int attrib_list[] = { WX_GL_RGBA, WX_GL_DOUBLEBUFFER, WX_GL_DEPTH_SIZE, 16, 0 }`

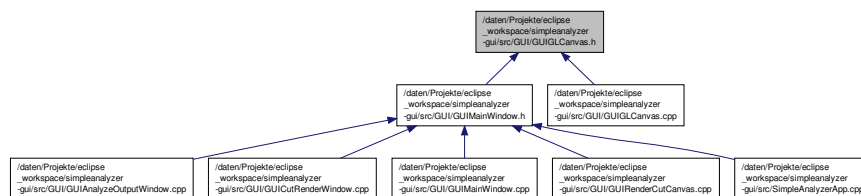
Definiert in Zeile 20 der Datei `GUIGLCanvas.cpp`.

## 9.20 /daten/Projekte/eclipse\_workspace/simpleanalyzer-gui/src/GUI/GLCanvas.h-- Dateireferenz

```
#include "Renderer.h"
#include <wx/wx.h>
#include <wx/glcanvas.h>
#include "../processing/ObjectData.h"
Include-Abhängigkeitsdiagramm für GUIGLCanvas.h:
```



Dieser Graph zeigt, welche Datei direkt oder indirekt diese Datei enthält:



## Klassen

- class `GUIGLCanvas`

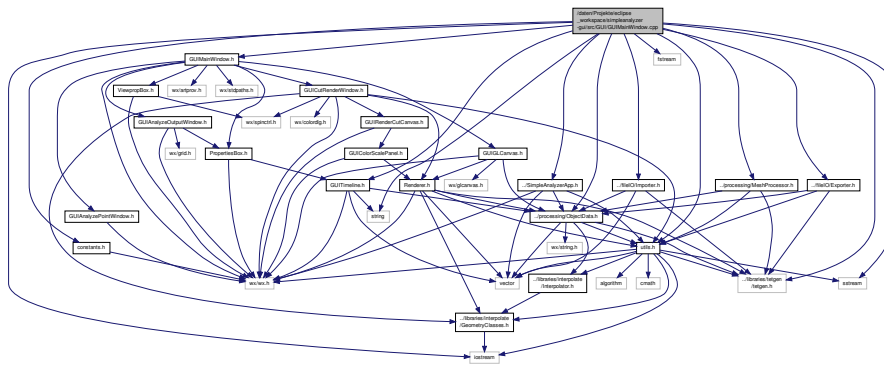
*Zeichenfläche für das 3D-Fenster.*



## 9.21 /daten/Projekte/eclipse\_workspace/simpleanalyzer-gui/src/GUI/GUIMainWindow.cpp-Dateireferenz

```
#include "GUIMainWindow.h"
#include <iostream>
#include <sstream>
#include <fstream>
#include <string>
#include "../SimpleAnalyzerApp.h"
#include "../libraries/tetgen/tetgen.h"
#include "../processing/MeshProcessor.h"
#include "../fileIO/Importer.h"
#include "../processing/ObjectData.h"
#include "constants.h"
#include "../fileIO/Exporter.h"
#include "GUITimeline.h"
#include "../processing/Utils.h"
```

Include-Abhängigkeitsdiagramm für GUIMainWindow.cpp:



### Makrodefinitionen

- #define **PATH\_SEPARATOR** "/"
- #define **PROPBOXWIDTH** 300
- #define **VIEWBOXWIDTH** 300

#### 9.21.1 Makro-Dokumentation

##### 9.21.1.1 #define PATH\_SEPARATOR "/"

Definiert in Zeile 31 der Datei GUIMainWindow.cpp.

##### 9.21.1.2 #define PROPBOXWIDTH 300

Definiert in Zeile 37 der Datei GUIMainWindow.cpp.

##### 9.21.1.3 #define VIEWBOXWIDTH 300

Definiert in Zeile 38 der Datei GUIMainWindow.cpp.







## Variablen

- const wxEventType `wxEVT_TIMELINE_CHANGE` = wxNewEventType()  
*Typ wxEVT\_TIMELINE\_CHANGE zum Auslösen eines Events bei Veränderung der Zeitleiste.*
- const float `refine_factors` [`SCALE_REFINE_STEPS`] = { .5, .2, .1 }

### 9.25.1 Makro-Dokumentation

#### 9.25.1.1 #define SCALE\_REFINE\_STEPS 3

Definiert in Zeile 31 der Datei GUITimeline.cpp.

### 9.25.2 Variablen-Dokumentation

#### 9.25.2.1 const float refine\_factors[SCALE\_REFINE\_STEPS] = { .5, .2, .1 }

Definiert in Zeile 34 der Datei GUITimeline.cpp.

#### 9.25.2.2 const wxEventType wxEVT\_TIMELINE\_CHANGE = wxNewEventType()

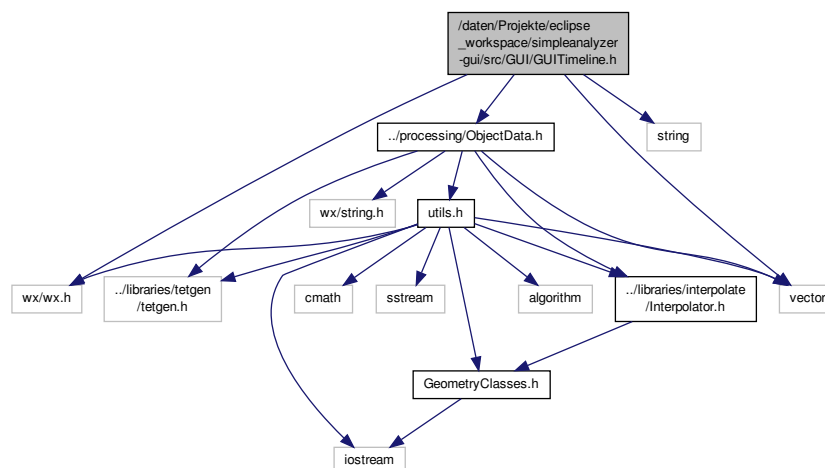
Typ wxEVT\_TIMELINE\_CHANGE zum Auslösen eines Events bei Veränderung der Zeitleiste.

Die Definition als globale Konstante ist durch das GUI-System vorgegeben.

Definiert in Zeile 18 der Datei GUITimeline.cpp.

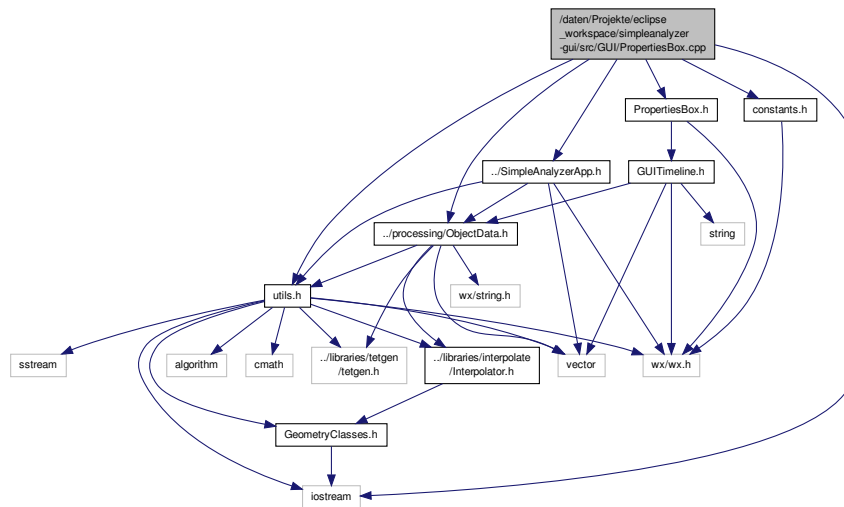
## 9.26 /daten/Projekte/eclipse\_workspace/simpleanalyzer-gui/src/GUI/GUITimeline.h--Dateireferenz

```
#include <wx/wx.h>
#include <string>
#include <vector>
#include "../processing/ObjectData.h"
Include-Abhängigkeitsdiagramm für GUITimeline.h:
```





Include-Abhängigkeitsdiagramm für PropertiesBox.cpp:



## Variablen

- wxString `sdfilestring` [] = { wxT("") }

### 9.27.1 Variablen-Dokumentation

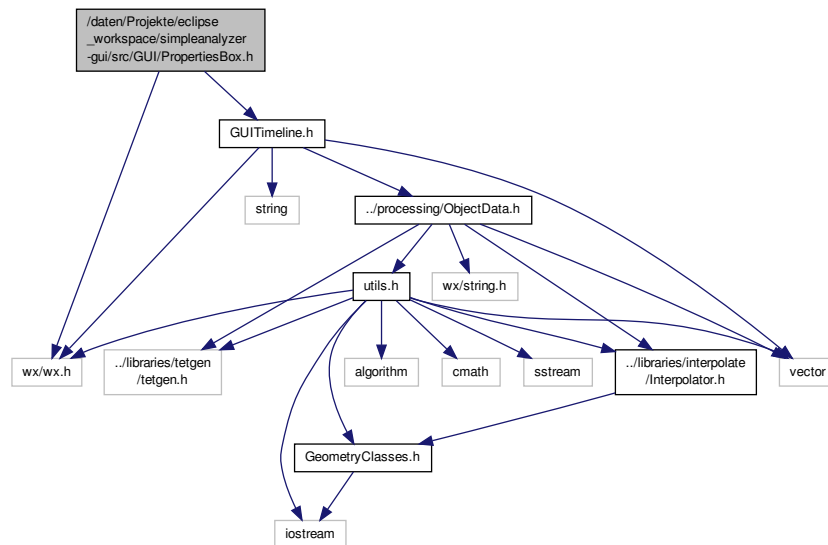
#### 9.27.1.1 wxString sdfilestring[] = { wxT("") }

Definiert in Zeile 17 der Datei PropertiesBox.cpp.

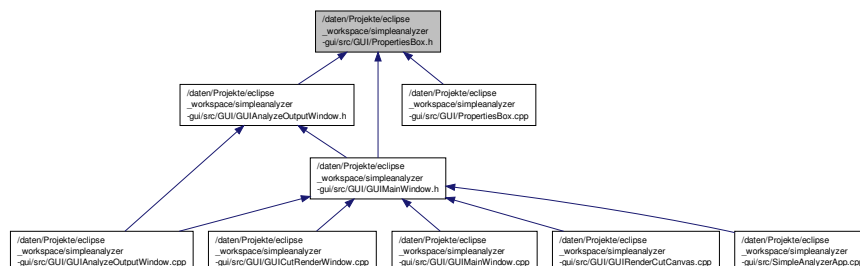
## 9.28 /daten/Projekte/eclipse\_workspace/simpleanalyzer-gui/src/GUI/PropertiesBox.h--Dateireferenz

```
#include <wx/wx.h>
#include "GUITimeline.h"
```

Include-Abhängigkeitsdiagramm für PropertiesBox.h:



Dieser Graph zeigt, welche Datei direkt oder indirekt diese Datei enthält:



## Klassen

- class [PropertiesBox](#)

*Oberfläche zum Verändern/Anzeigen der Eigenschaften eines Objekts.*

## 9.29 /daten/Projekte/eclipse\_workspace/simpleanalyzer-gui/src/GUI/Renderer.cpp-- Dateireferenz

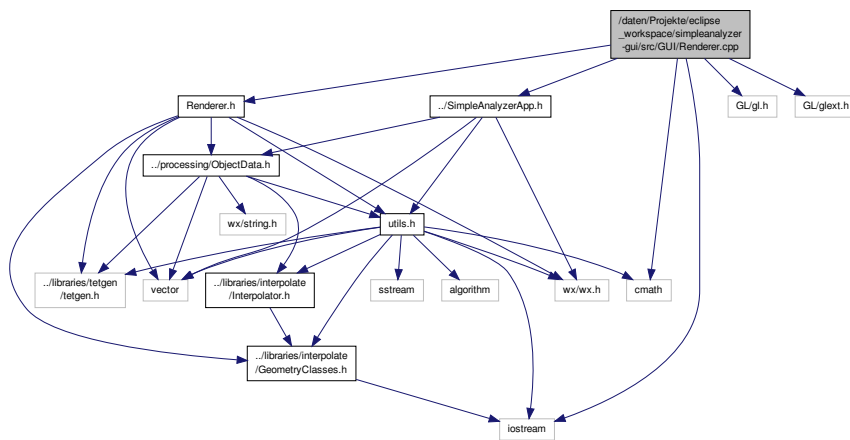
```

#include "Renderer.h"
#include <iostream>
#include <cmath>
#include "../SimpleAnalyzerApp.h"
#include <GL/gl.h>
#include <GL/glext.h>

```



Include-Abhängigkeitsdiagramm für Renderer.cpp:



## Funktionen

- bool `pointBehindCut` (`Vector3D *point`, `Triangle *cut`)
- void `drawVector` (`Vector3D *pos`, `Vector3D *dir`)

*Zeichnet einen Vektor als Pfeil.*

- void `renderGrid` ()

*Zeichnet markante Linien zum leichteren Erfassen des Koordinatensystems.*

- void `drawCutRenderInfo` (`CutRender_info *info`)

*Visualisiert Informationen über eine 2D-Temperaturverteilung.*

### 9.29.1 Dokumentation der Funktionen

#### 9.29.1.1 void `drawCutRenderInfo` ( `CutRender_info * info` )

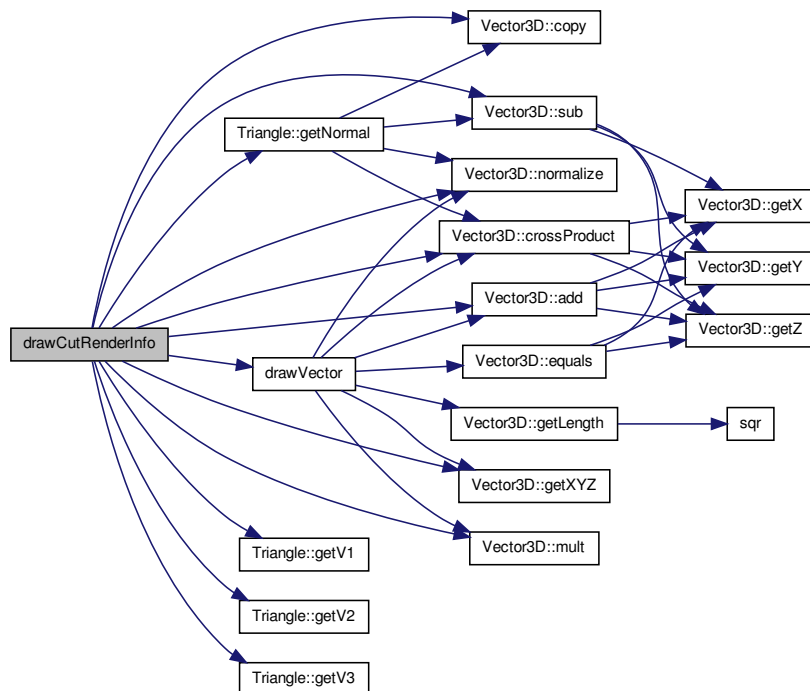
Visualisiert Informationen über eine 2D-Temperaturverteilung.

Parameter

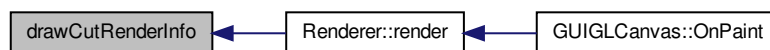
<code>info</code>	Die zu visualisierenden Informationen.
-------------------	--

Definiert in Zeile 576 der Datei `Renderer.cpp`.

Hier ist ein Graph, der zeigt, was diese Funktion aufruft:



Hier ist ein Graph der zeigt, wo diese Funktion aufgerufen wird:



#### 9.29.1.2 void drawVector ( Vector3D \* pos, Vector3D \* dir )

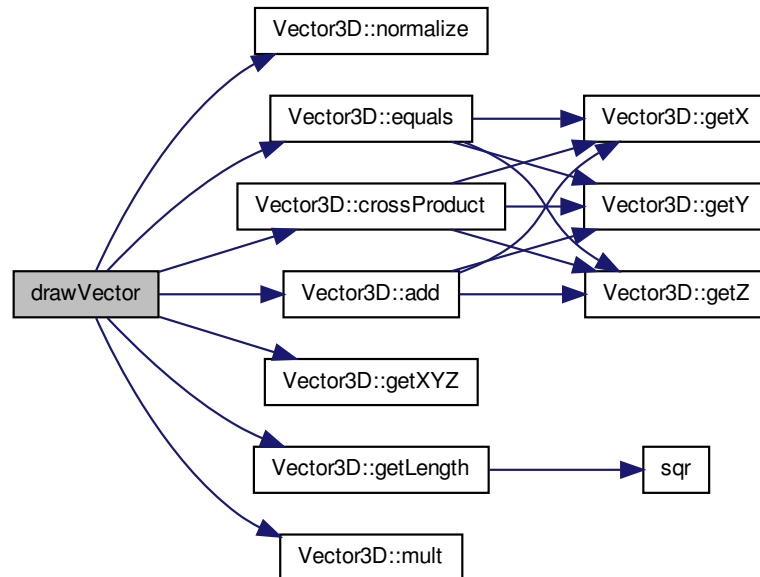
Zeichnet einen Vektor als Pfeil.

Parameter

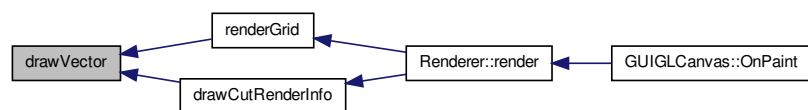
<i>pos</i>	Startpunkt des Pfeils.
<i>dir</i>	Richtung und Länge des Pfeils.

Definiert in Zeile 373 der Datei Renderer.cpp.

Hier ist ein Graph, der zeigt, was diese Funktion aufruft:



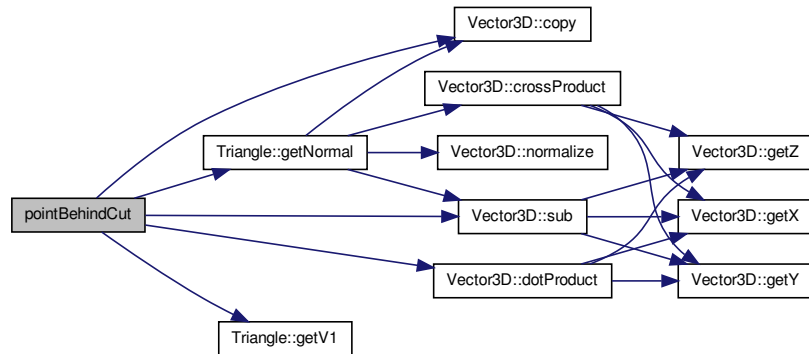
Hier ist ein Graph der zeigt, wo diese Funktion aufgerufen wird:



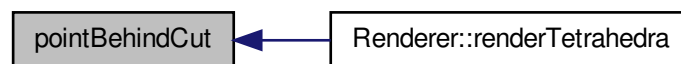
### 9.29.1.3 bool pointBehindCut ( Vector3D \* point, Triangle \* cut )

Definiert in Zeile 46 der Datei `Renderer.cpp`.

Hier ist ein Graph, der zeigt, was diese Funktion aufruft:



Hier ist ein Graph der zeigt, wo diese Funktion aufgerufen wird:

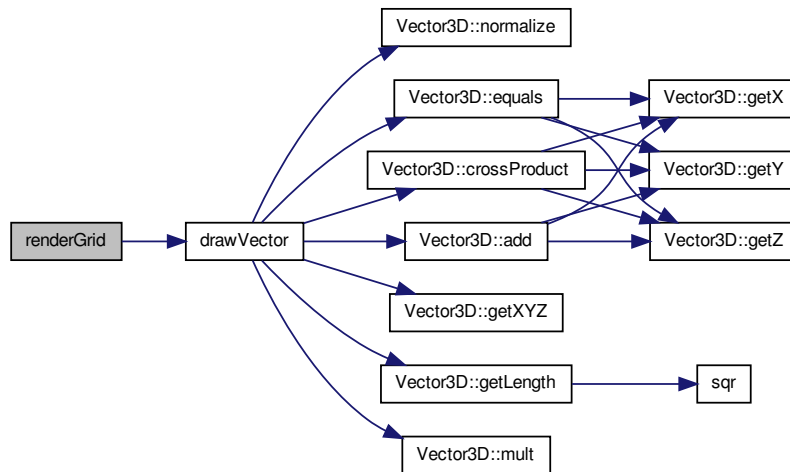


#### 9.29.1.4 void renderGrid ( )

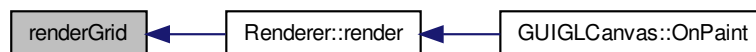
Zeichnet markante Linien zum leichteren Erfassen des Koordinatensystems.

Definiert in Zeile 455 der Datei `Renderer.cpp`.

Hier ist ein Graph, der zeigt, was diese Funktion aufruft:



Hier ist ein Graph der zeigt, wo diese Funktion aufgerufen wird:



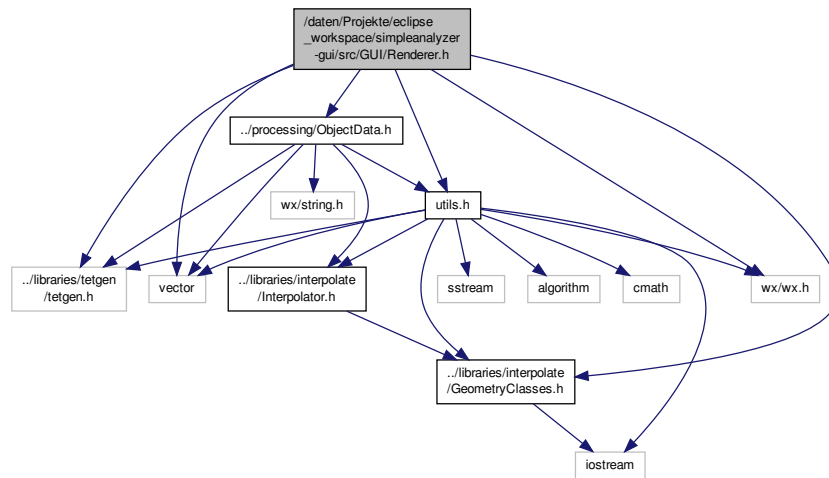
## 9.30 /daten/Projekte/eclipse\_workspace/simpleanalyzer-gui/src/GUI/Renderer.h-Dateireferenz

```

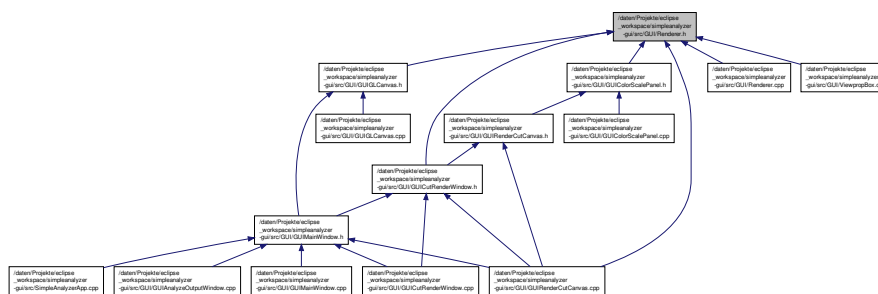
#include "../libraries/tetgen/tetgen.h"
#include <vector>
#include "../libraries/interpolate/GeometryClasses.h"
#include "../processing/ObjectData.h"
#include "../processing/utils.h"
#include <wx/wx.h>

```

Include-Abhängigkeitsdiagramm für `Renderer.h`:



Dieser Graph zeigt, welche Datei direkt oder indirekt diese Datei enthält:



## Klassen

- class [Renderer](#)

*Zeichnet den Inhalt der 3D-Fensters.*

- struct [Renderer::Viewprop\\_info](#)

*Informationen über die Ansicht des Modells (Virtuelle Kamera) und welche Elemente dargestellt werden.*

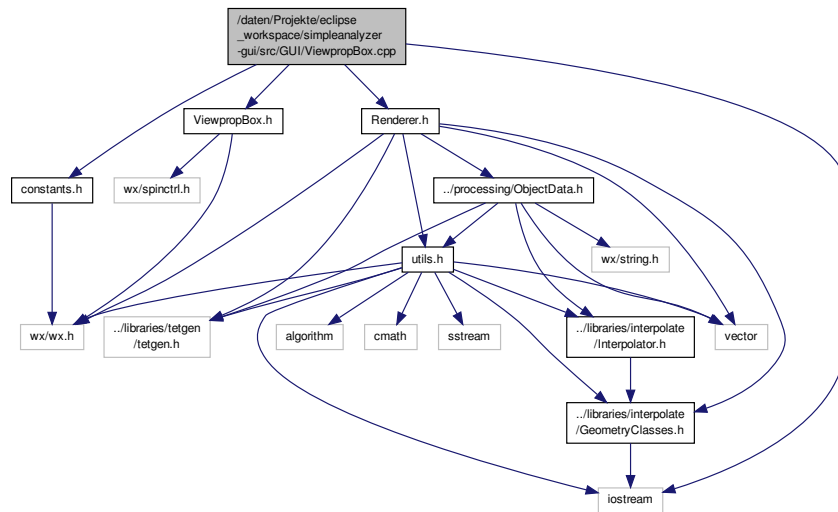
## 9.31 /daten/Projekte/eclipse\_workspace/simpleanalyzer-gui/src/GUI/ViewpropBox.cpp-- Dateireferenz

```

#include "ViewpropBox.h"
#include "constants.h"
#include "Renderer.h"
#include <iostream>

```

Include-Abhängigkeitsdiagramm für ViewpropBox.cpp:



## Variablen

- wxString `renderchoices` [] = { wxT("Kein"), wxT("Material"), wxT("Wert") }

### 9.31.1 Variablen-Dokumentation

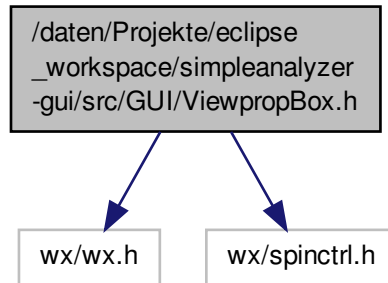
9.31.1.1 wxString `renderchoices`[] = { wxT("Kein"), wxT("Material"), wxT("Wert") }

Definiert in Zeile 15 der Datei ViewpropBox.cpp.

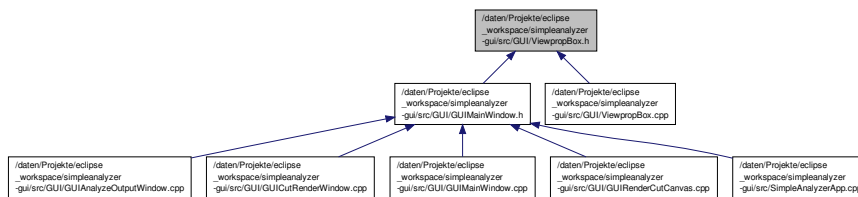
## 9.32 /daten/Projekte/eclipse\_workspace/simpleanalyzer-gui/src/GUI/ViewpropBox.h--Dateireferenz

```
#include <wx/wx.h>
#include <wx/spinctrl.h>
```

Include-Abhängigkeitsdiagramm für ViewpropBox.h:



Dieser Graph zeigt, welche Datei direkt oder indirekt diese Datei enthält:



## Klassen

- class [ViewpropBox](#)

*Oberfläche zum Verändern/Anzeigen der Visualisierungsoptionen.*

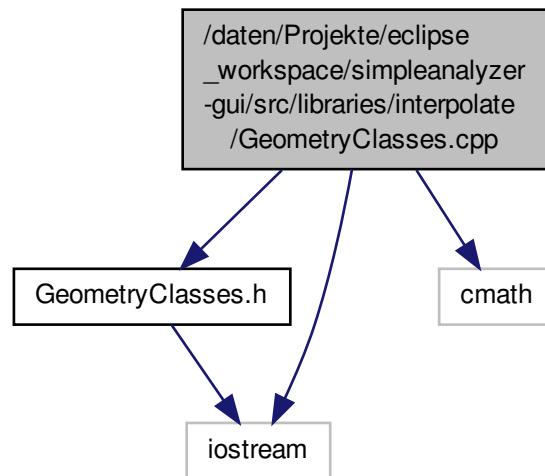
### 9.33 /daten/Projekte/eclipse\_workspace/simpleanalyzer-gui/src/libraries/interpolate/-GeometryClasses.cpp-Dateireferenz

```

#include "GeometryClasses.h"
#include <iostream>
#include <cmath>
  
```



Include-Abhängigkeitsdiagramm für GeometryClasses.cpp:



## Makrodefinitionen

- `#define EPSILON 0.0000001`

## Funktionen

- `double sqr (double v)`
- `std::ostream & operator<< (std::ostream &out, const Vector3D &vec)`

*Definition des <<-Operators für die Ausgabe eines Vektors.*

### 9.33.1 Makro-Dokumentation

#### 9.33.1.1 `#define EPSILON 0.0000001`

Definiert in Zeile 13 der Datei `GeometryClasses.cpp`.

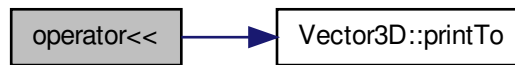
### 9.33.2 Dokumentation der Funktionen

#### 9.33.2.1 `std::ostream& operator<< ( std::ostream & out, const Vector3D & vec )`

Definition des <<-Operators für die Ausgabe eines Vektors.

Definiert in Zeile 132 der Datei `GeometryClasses.cpp`.

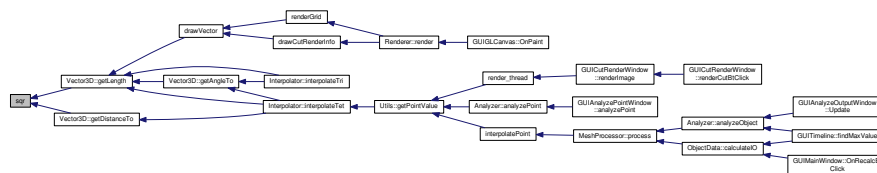
Hier ist ein Graph, der zeigt, was diese Funktion aufruft:



### 9.33.2.2 `double sqr ( double v ) [inline]`

Definiert in Zeile 14 der Datei `GeometryClasses.cpp`.

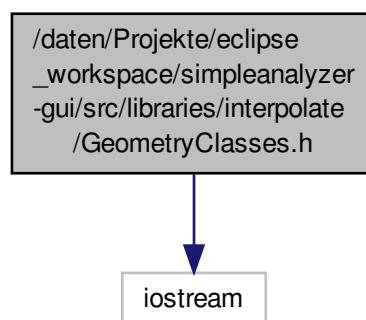
Hier ist ein Graph der zeigt, wo diese Funktion aufgerufen wird:



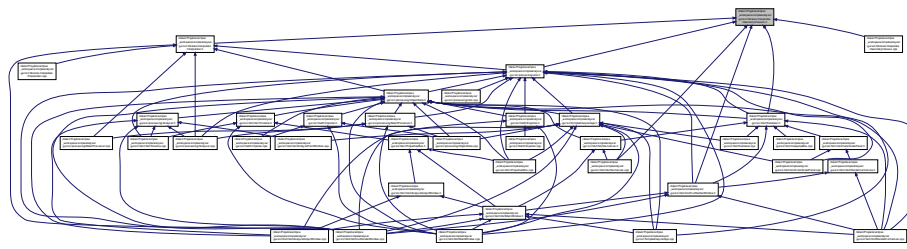
## 9.34 /daten/Projekte/eclipse\_workspace/simpleanalyzer-gui/src/libraries/interpolate/-GeometryClasses.h-Dateireferenz

```
#include <iostream>
```

Include-Abhängigkeitsdiagramm für `GeometryClasses.h`:



Dieser Graph zeigt, welche Datei direkt oder indirekt diese Datei enthält:



## Klassen

- class [Vector3D](#)  
*3D-Vektorklasse mit nützlichen Operationen.*
- class [Matrix3D](#)  
*3x3-Matrixklasse mit Operationen.*
- class [Triangle](#)  
*Ein durch 3 Ortsvektoren beschriebenes Dreieck.*
- class [Tetrahedron](#)  
*Ein durch 4 Ortsvektoren beschriebener Tetraeder.*

## Funktionen

- `std::ostream & operator<< (std::ostream &out, const Vector3D &vec)`  
*Definition des <<-Operators für die Ausgabe eines Vektors.*

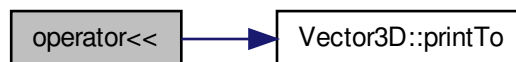
### 9.34.1 Dokumentation der Funktionen

#### 9.34.1.1 `std::ostream& operator<< ( std::ostream & out, const Vector3D & vec )`

Definition des <<-Operators für die Ausgabe eines Vektors.

Definiert in Zeile 132 der Datei GeometryClasses.cpp.

Hier ist ein Graph, der zeigt, was diese Funktion aufruft:

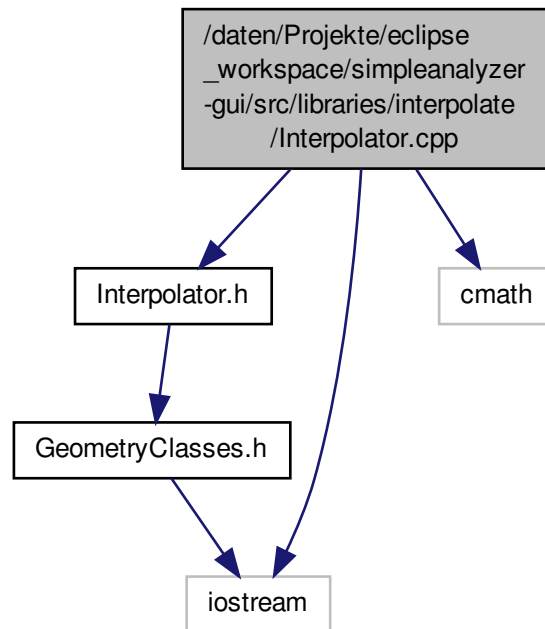


### 9.35 /daten/Projekte/eclipse\_workspace/simpleanalyzer-gui/src/libraries/interpolate/- Interpolator.cpp-Dateireferenz

```
#include "Interpolator.h"
```

```
#include <iostream>
#include <cmath>
```

Include-Abhängigkeitsdiagramm für Interpolator.cpp:



## Makrodefinitionen

- `#define PI 3.14159265358979323846`

## Funktionen

- `double sqr (double v)`
- `double getSign (double x)`

### 9.35.1 Makro-Dokumentation

#### 9.35.1.1 `#define PI 3.14159265358979323846`

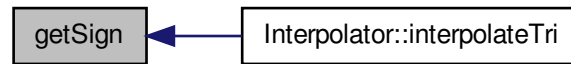
Definiert in Zeile 4 der Datei `Interpolator.cpp`.

### 9.35.2 Dokumentation der Funktionen

#### 9.35.2.1 `double getSign ( double x ) [inline]`

Definiert in Zeile 20 der Datei `Interpolator.cpp`.

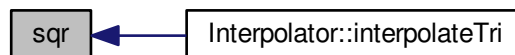
Hier ist ein Graph der zeigt, wo diese Funktion aufgerufen wird:



9.35.2.2 `double sqr ( double v ) [inline]`

Definiert in Zeile 7 der Datei `Interpolator.cpp`.

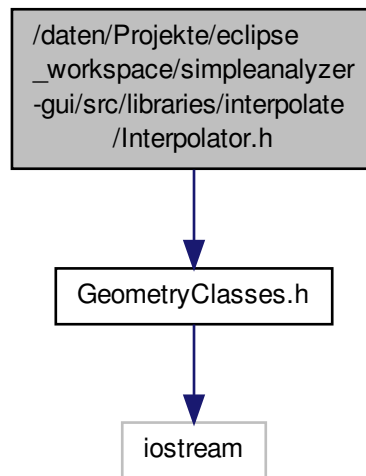
Hier ist ein Graph der zeigt, wo diese Funktion aufgerufen wird:



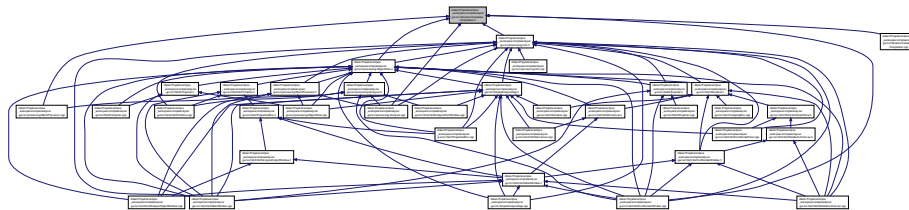
## 9.36 /daten/Projekte/eclipse\_workspace/simpleanalyzer-gui/src/libraries/interpolate/- Interpolator.h-Dateireferenz

```
#include "GeometryClasses.h"
```

Include-Abhängigkeitsdiagramm für Interpolator.h:



Dieser Graph zeigt, welche Datei direkt oder indirekt diese Datei enthält:



## Klassen

- class [Interpolator](#)

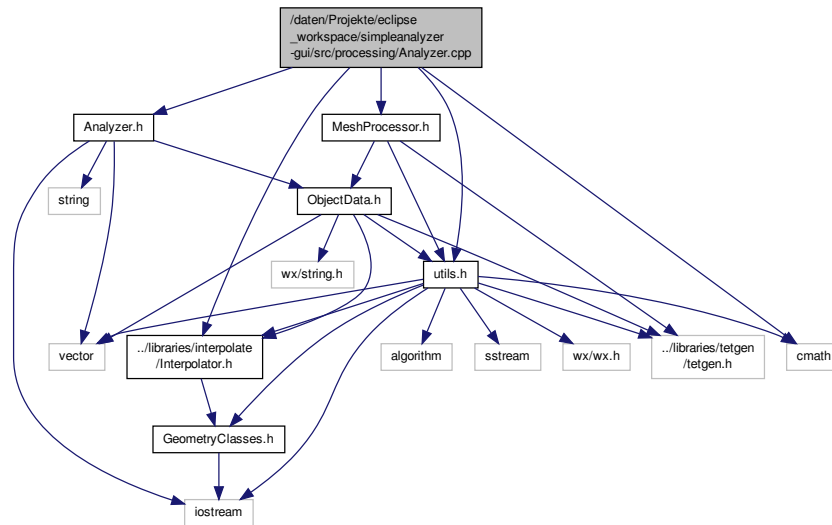
*2- und 3-dimensionale Inter-/Extrapolation*

## 9.37 /daten/Projekte/eclipse\_workspace/simpleanalyzer-gui/src/processing/Analyzer.cpp-Dateireferenz

```

#include "Analyzer.h"
#include <cmath>
#include "MeshProcessor.h"
#include "../libraries/interpolate/Interpolator.h"
#include "utils.h"
  
```

Include-Abhängigkeitsdiagramm für Analyzer.cpp:



## Funktionen

- `std::ostream & operator<< (std::ostream &out, const Analyzer::AnalyzerData_object &data)`

### 9.37.1 Dokumentation der Funktionen

9.37.1.1 `std::ostream& operator<< ( std::ostream & out, const Analyzer::AnalyzerData_object & data )`

Definiert in Zeile 164 der Datei Analyzer.cpp.

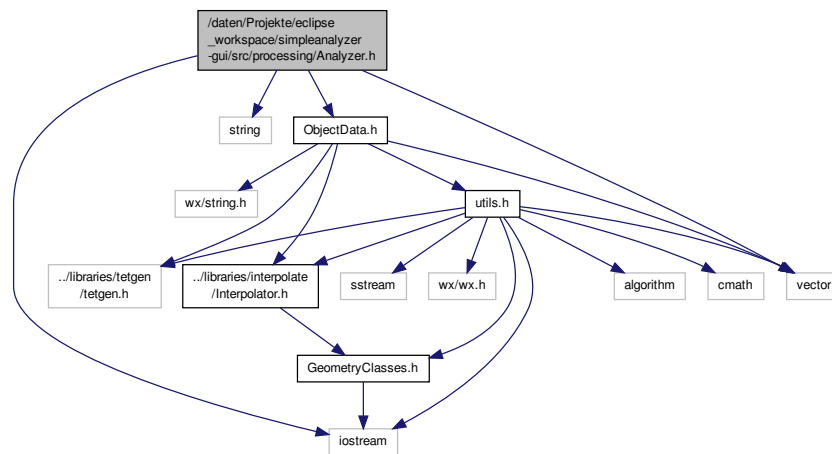
## 9.38 /daten/Projekte/eclipse\_workspace/simpleanalyzer-gui/src/processing/Analyzer.h--Dateireferenz

```

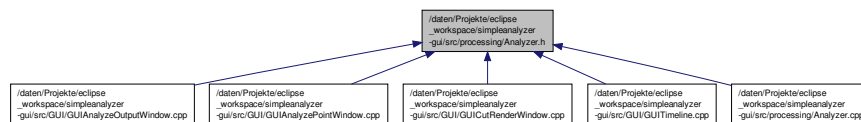
#include <vector>
#include <string>
#include <iostream>
#include "ObjectData.h"

```

Include-Abhängigkeitsdiagramm für Analyzer.h:



Dieser Graph zeigt, welche Datei direkt oder indirekt diese Datei enthält:



## Klassen

- class [Analyzer](#)  
*Ermittelt Daten aus der Temperaturverteilung.*
- struct [Analyzer::AnalyzerData\\_material](#)  
*Analyseergebnisse für ein Material.*
- struct [Analyzer::AnalyzerData\\_dataset](#)  
*Analyseergebnisse für einen Sensordatensatz.*
- struct [Analyzer::AnalyzerData\\_object](#)  
*Analyseergebnisse für ein Objekt.*
- struct [Analyzer::AnalyzerData\\_point](#)  
*Analyseergebnisse für einen Punkt.*

## 9.39 /daten/Projekte/eclipse\_workspace/simpleanalyzer-gui/src/processing/MeshProcessor.cpp-Dateireferenz

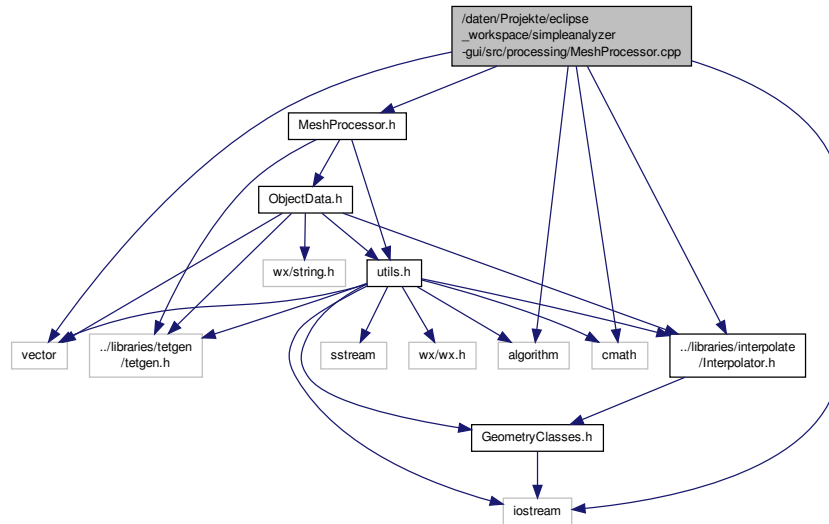
```

#include "MeshProcessor.h"
#include <iostream>
#include <vector>
#include <algorithm>
#include <cmath>
#include "../libraries/interpolate/Interpolator.h"

```



Include-Abhängigkeitsdiagramm für MeshProcessor.cpp:



## Funktionen

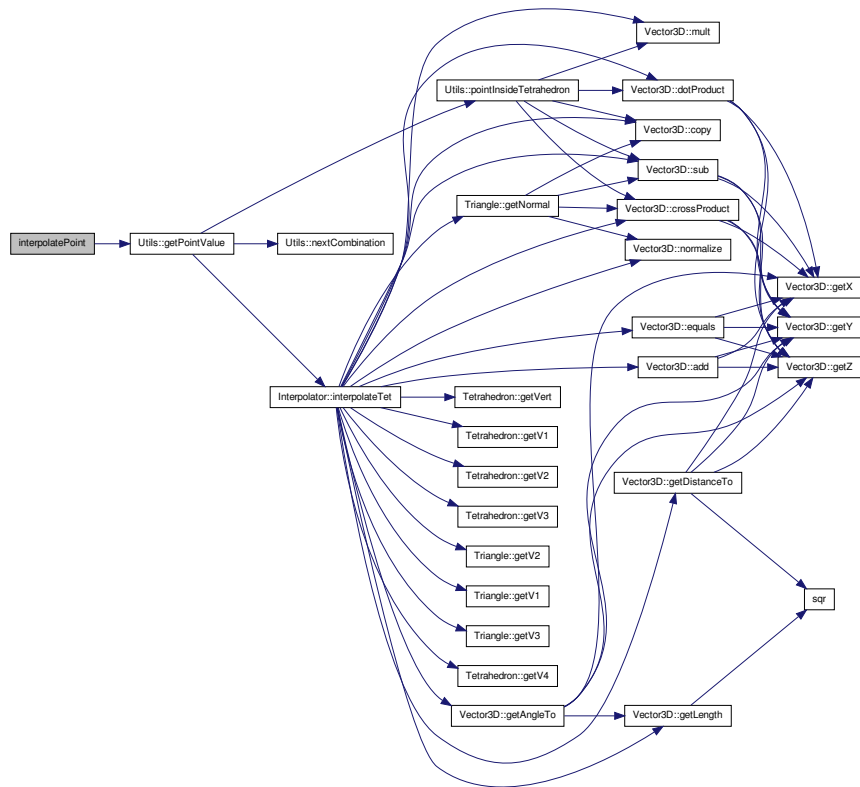
- void `interpolatePoint` (`ObjectData::MaterialData` \*data, `vector`< `SensorPoint` > \*sensorpoints, int pointIndex, `Interpolator` \*interpolator)

### 9.39.1 Dokumentation der Funktionen

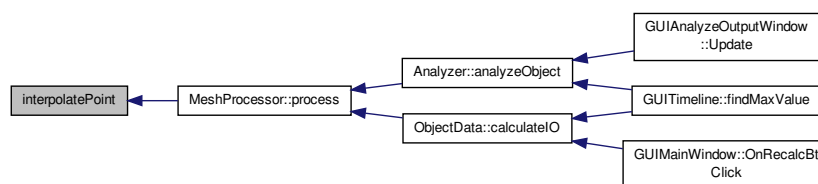
9.39.1.1 void `interpolatePoint` ( `ObjectData::MaterialData` \* data, `vector`< `SensorPoint` > \* sensorpoints, int pointIndex, `Interpolator` \* interpolator )

Definiert in Zeile 20 der Datei MeshProcessor.cpp.

Hier ist ein Graph, der zeigt, was diese Funktion aufruft:



Hier ist ein Graph der zeigt, wo diese Funktion aufgerufen wird:



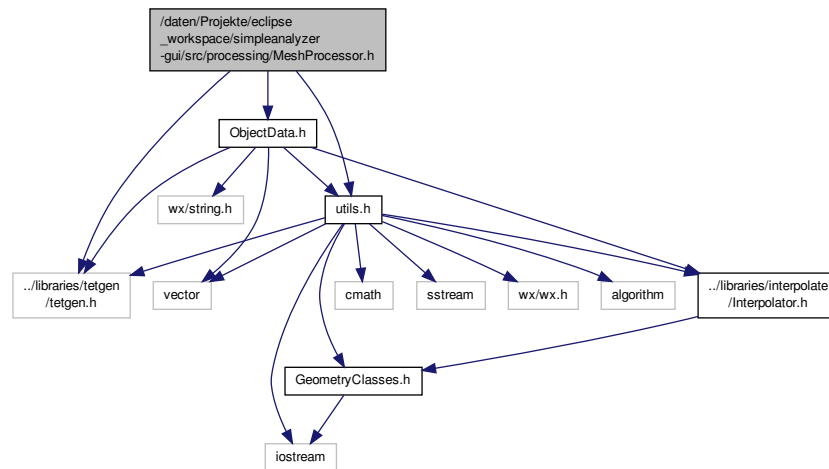
## 9.40 /daten/Projekte/eclipse\_workspace/simpleanalyzer-gui/src/processing/MeshProcessor.h- Dateireferenz

```

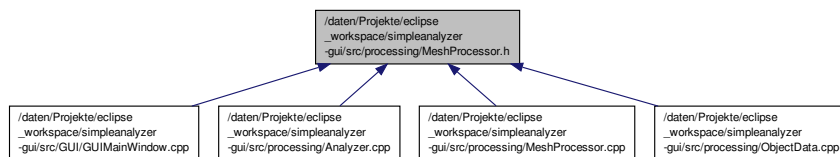
#include "../libraries/tetgen/tetgen.h"
#include "ObjectData.h"
#include "utils.h"

```

Include-Abhängigkeitsdiagramm für MeshProcessor.h:



Dieser Graph zeigt, welche Datei direkt oder indirekt diese Datei enthält:



## Klassen

- class `MeshProcessor`

*Errechnet die Temperaturverteilung für ein Objekt.*

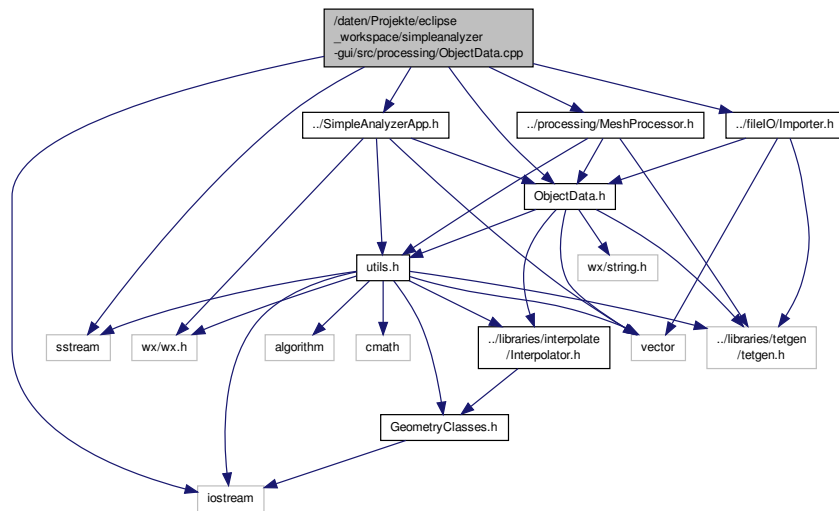
## 9.41 /daten/Projekte/eclipse\_workspace/simpleanalyzer-gui/src/processing/ObjectData.cpp-Dateireferenz

```

#include "ObjectData.h"
#include "../SimpleAnalyzerApp.h"
#include "../fileIO/Importer.h"
#include "../processing/MeshProcessor.h"
#include <iostream>
#include <sstream>

```

Include-Abhängigkeitsdiagramm für ObjectData.cpp:



## Makrodefinitionen

- `#define PATH_SEPARATOR '/'`

### 9.41.1 Makro-Dokumentation

#### 9.41.1.1 `#define PATH_SEPARATOR '/'`

Definiert in Zeile 20 der Datei ObjectData.cpp.

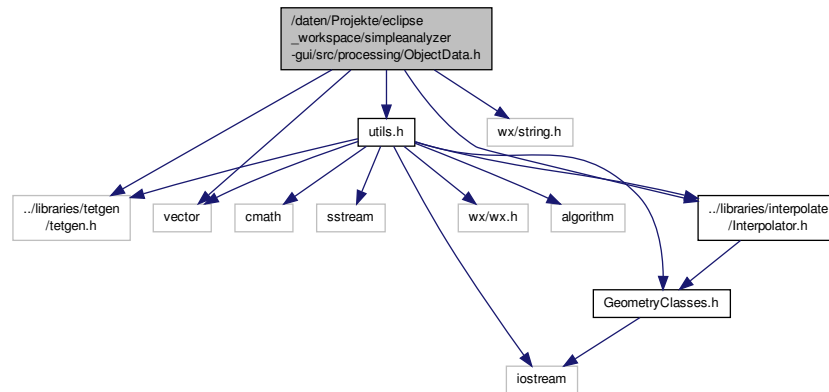
## 9.42 `/daten/Projekte/eclipse_workspace/simpleanalyzer-gui/src/processing/ObjectData.h`-Dateireferenz

```

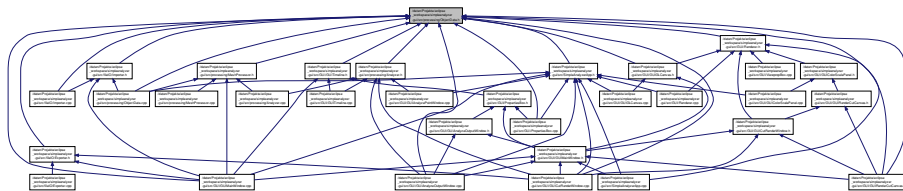
#include "../libraries/tetgen/tetgen.h"
#include <vector>
#include "../libraries/interpolate/Interpolator.h"
#include <wx/string.h>
#include "utils.h"

```

Include-Abhängigkeitsdiagramm für ObjectData.h:



Dieser Graph zeigt, welche Datei direkt oder indirekt diese Datei enthält:



## Klassen

- class [ObjectData](#)  
Die Daten eines Versuchsobjekts.
- struct [ObjectData::MaterialData](#)  
Die Daten eines Materials.

## Makrodefinitionen

- #define [NUMBEROFSENSORATTRIBUTES](#) 1

### 9.42.1 Makro-Dokumentation

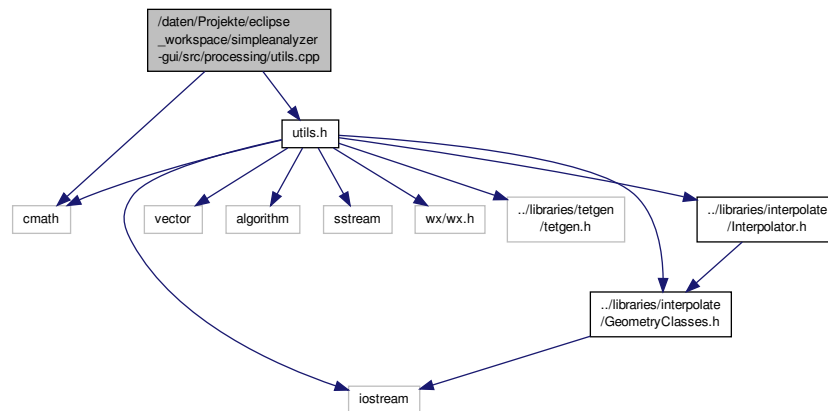
#### 9.42.1.1 #define NUMBEROFSENSORATTRIBUTES 1

Definiert in Zeile 16 der Datei ObjectData.h.

## 9.43 /daten/Projekte/eclipse\_workspace/simpleanalyzer-gui/src/processing/Utils.cpp--Dateireferenz

```
#include "utils.h"
#include <cmath>
```

Include-Abhängigkeitsdiagramm für utils.cpp:



## Makrodefinitionen

- `#define EPSILON 0.000001`

### 9.43.1 Makro-Dokumentation

#### 9.43.1.1 `#define EPSILON 0.000001`

Definiert in Zeile 14 der Datei `utils.cpp`.

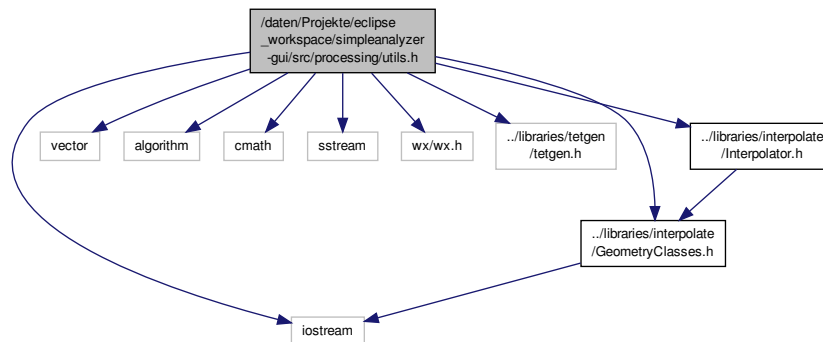
## 9.44 /daten/Projekte/eclipse\_workspace/simpleanalyzer-gui/src/processing/utils.h-- Dateireferenz

```

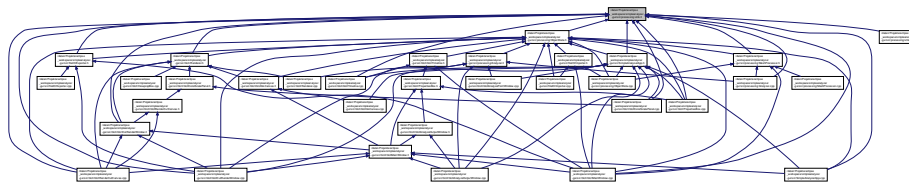
#include <iostream>
#include <vector>
#include <algorithm>
#include <cmath>
#include <sstream>
#include <wx/wx.h>
#include "../libraries/tetgen/tetgen.h"
#include "../libraries/interpolate/GeometryClasses.h"
#include "../libraries/interpolate/Interpolator.h"

```

Include-Abhängigkeitsdiagramm für utils.h:



Dieser Graph zeigt, welche Datei direkt oder indirekt diese Datei enthält:



## Klassen

- struct `Utils::Visualization_info`  
*Informationen über die Farbgebung bei der Visualisierung.*
- struct `Utils::SortStruct`  
*Hilfsstruktur zum Sortieren von Punkten nach dem Abstand zu einem anderen Punkt.*
- struct `Utils::SensorPoint`  
*Daten eines Sensordatenpunktes.*
- struct `Utils::CutRender_info`  
*Daten zur Darstellung einer 2D-Temperaturverteilungs-Ebene.*
- struct `Utils::SensorData`  
*Ein Sensordatensatz.*
- struct `Utils::SensorPointComparator`  
*Hilfsstruktur zum Vergleichen des Abstands von Messpunkten.*

## Namensbereiche

- `Utils`  
*allgemeine Funktionen und Typen.*

## Aufzählungen

- enum `Utils::PIM_algorithm` { `Utils::ALGORITHM_TETRAHEDRONS` = 0, `Utils::ALGORITHM_RAY` }  
*Zum Punkt-in-Volumen Testen verwendeter Algorithmus.*

## Funktionen

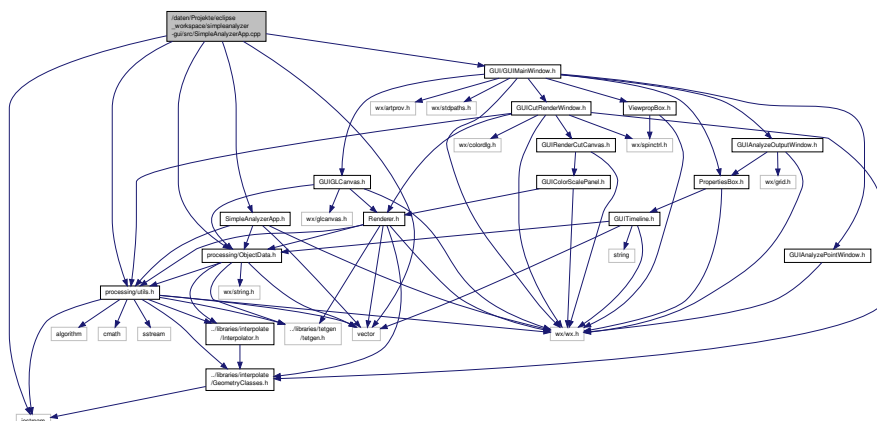
- double `Utils::sqr` (double d)  
*Quadriert eine Zahl.*
- float `Utils::clampHue` (float h)  
*Begrenzt einen Wert auf den Bereich 0..1.*
- string `Utils::floattostr` (double val)  
*Hilfsfunktion zur Umwandlung einer Zahl in einen String.*
- wxString `Utils::floattowxstr` (double val)  
*Wandelt eine Fließkommazahl in einen wxWidgets-String um.*
- wxString `Utils::floattowxstr` (double val, int digits)  
*Wandelt eine Fließkommazahl in einen wxWidgets-String um.*
- int `Utils::rayIntersectsTriangle` (`Vector3D` \*p, `Vector3D` \*direction, `Triangle` \*tri, double \*depth)  
*Testet, ob ein Strahl ein Dreieck schneidet.*
- int `Utils::pointInsideMesh` (`Vector3D` \*p, tetgenio \*io, PIM\_algorithm algorithm)  
*Testet, ob sich ein Punkt innerhalb eines Körpers befindet.*
- int `Utils::pointInsideTetrahedron` (`Vector3D` \*pges, `Vector3D` \*v1, `Vector3D` \*v2, `Vector3D` \*v3, `Vector3D` \*v4)  
*Testet, ob sich ein Punkt innerhalb eines Tetraeders befindet.*
- int `Utils::pointInsideTetrahedron` (double \*pges, double \*v1, double \*v2, double \*v3, double \*v4)  
*Testet, ob sich ein Punkt innerhalb eines Tetraeders befindet.*
- int `Utils::pointInsideTetrahedron` (double \*p, `vector`< `SensorPoint` \* > \*tet)  
*Testet, ob sich ein Punkt innerhalb eines Tetraeders befindet.*
- void `Utils::nextCombination` (`vector`< int > \*indices, int depth, int dataPointCount)  
*Ermöglicht das generieren aller möglichen Verteilungen von 4 Elementen auf dataPointCount Plätze.*
- double `Utils::getPointValue` (int &status, `vector`< `SensorPoint` > \*sensorpoints, double \*p, `Interpolator` \*interpolator, `vector`< `SensorPoint` \* > \*prev\_tet=NULL, `vector`< `SensorPoint` \* > \*current\_tet=NULL)  
*Gibt den inter/extrapolierten Wert eines Punktes zurück.*
- float \* `Utils::hsvToRgb` (float h, float s, float v)  
*Wandelt eine Farbe im HSV-Format ins RGB-Format um.*
- void `Utils::copySensorPoint` (`SensorPoint` \*from, `SensorPoint` \*to)  
*Kopiert die Eigenschaften eines Sensorpunktes in einen Anderen.*

## 9.45 /daten/Projekte/eclipse\_workspace/simpleanalyzer-gui/src/SimpleAnalyzerApp.cpp- Dateireferenz

```
#include <iostream>
#include "SimpleAnalyzerApp.h"
#include "GUI/GUIMainWindow.h"
#include "processing/ObjectData.h"
#include "processing/Utils.h"
#include <vector>
```

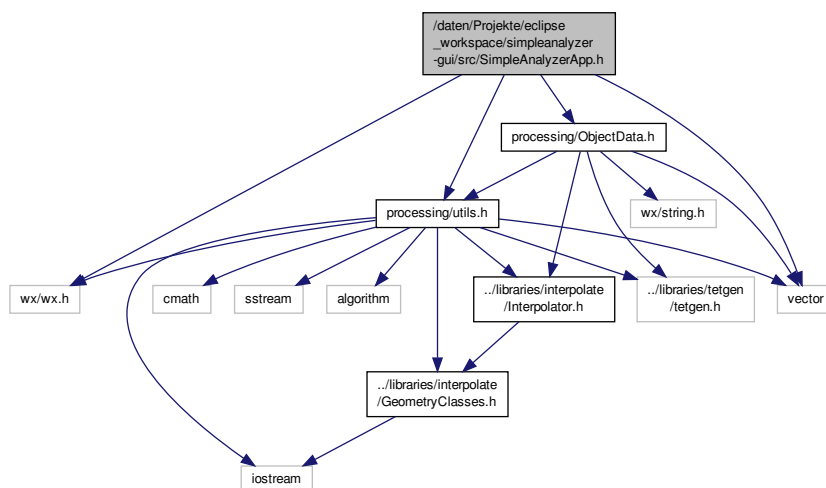


Include-Abhängigkeitsdiagramm für SimpleAnalyzerApp.cpp:

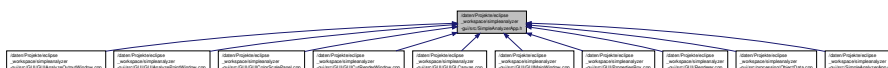


9.46 /daten/Projekte/eclipse\_workspace/simpleanalyzer-gui/src/SimpleAnalyzerApp.h--  
Dateireferenz

```
#include <wx/wx.h>
#include <vector>
#include "processing/Utils.h"
#include "processing/ObjectData.h"
Include-Abhängigkeitsdiagramm für SimpleAnalyzerApp.h:
```



Dieser Graph zeigt, welche Datei direkt oder indirekt diese Datei enthält:



## Klassen

- class [SimpleAnalyzerApp](#)  
*Regelt den allgemeinen Ablauf des Programms.*

# Index

- ~Analyzer
  - Analyzer, [28](#)
- ~Exporter
  - Exporter, [44](#)
- ~GUIAnalyzeOutputWindow
  - GUIAnalyzeOutputWindow, [47](#)
- ~GUIAnalyzePointWindow
  - GUIAnalyzePointWindow, [50](#)
- ~GUIColorScalePanel
  - GUIColorScalePanel, [54](#)
- ~GUICutRenderWindow
  - GUICutRenderWindow, [65](#)
- ~GUIGLCanvas
  - GUIGLCanvas, [80](#)
- ~GUIMainWindow
  - GUIMainWindow, [88](#)
- ~GUIRenderCutCanvas
  - GUIRenderCutCanvas, [101](#)
- ~GUITimeline
  - GUITimeline, [109](#)
- ~Importer
  - Importer, [118](#)
- ~Interpolator
  - Interpolator, [122](#)
- ~MeshProcessor
  - MeshProcessor, [133](#)
- ~ObjectData
  - ObjectData, [137](#)
- ~PropertiesBox
  - PropertiesBox, [160](#)
- ~Renderer
  - Renderer, [169](#)
- ~SimpleAnalyzerApp
  - SimpleAnalyzerApp, [182](#)
- ~Triangle
  - Triangle, [189](#)
- ~Vector3D
  - Vector3D, [198](#)
- ~ViewpropBox
  - ViewpropBox, [215](#)
- /daten/Projekte/eclipse\_workspace/README.md, [222](#)
- /daten/Projekte/eclipse\_workspace/csvtsd/main.cpp, [219](#)
- /daten/Projekte/eclipse\_workspace/mergetsd/src/mergetsd.cpp, [221](#)
- /daten/Projekte/eclipse\_workspace/odisitosd/main.cpp, [220](#)
- /daten/Projekte/eclipse\_workspace/simpleanalyzer-gui/src/GUI/GUIAnalyzeOutputWindow.cpp, [229](#)
- /daten/Projekte/eclipse\_workspace/simpleanalyzer-gui/src/GUI/GUIAnalyzePointWindow.h, [229](#)
- /daten/Projekte/eclipse\_workspace/simpleanalyzer-gui/src/GUI/GUIAnalyzePointWindow.cpp, [230](#)
- /daten/Projekte/eclipse\_workspace/simpleanalyzer-gui/src/GUI/GUIAnalyzePointWindow.h, [231](#)
- /daten/Projekte/eclipse\_workspace/simpleanalyzer-gui/src/GUI/GUIColorScalePanel.cpp, [232](#)
- /daten/Projekte/eclipse\_workspace/simpleanalyzer-gui/src/GUI/GUIColorScalePanel.h, [233](#)
- /daten/Projekte/eclipse\_workspace/simpleanalyzer-gui/src/GUI/GUICutRenderWindow.cpp, [233](#)
- /daten/Projekte/eclipse\_workspace/simpleanalyzer-gui/src/GUI/GUICutRenderWindow.h, [236](#)
- /daten/Projekte/eclipse\_workspace/simpleanalyzer-gui/src/GUI/GUIGLCanvas.cpp, [237](#)
- /daten/Projekte/eclipse\_workspace/simpleanalyzer-gui/src/GUI/GUIGLCanvas.h, [238](#)
- /daten/Projekte/eclipse\_workspace/simpleanalyzer-gui/src/GUI/GUIMainWindow.cpp, [239](#)
- /daten/Projekte/eclipse\_workspace/simpleanalyzer-gui/src/GUI/GUIMainWindow.h, [240](#)
- /daten/Projekte/eclipse\_workspace/simpleanalyzer-gui/src/GUI/GUIRenderCutCanvas.cpp, [240](#)
- /daten/Projekte/eclipse\_workspace/simpleanalyzer-gui/src/GUI/GUIRenderCutCanvas.h, [241](#)
- /daten/Projekte/eclipse\_workspace/simpleanalyzer-gui/src/GUI/GUITimeline.cpp, [242](#)
- /daten/Projekte/eclipse\_workspace/simpleanalyzer-gui/src/GUI/GUITimeline.h, [243](#)
- /daten/Projekte/eclipse\_workspace/simpleanalyzer-gui/src/GUI/PropertiesBox.cpp, [244](#)
- /daten/Projekte/eclipse\_workspace/simpleanalyzer-gui/src/GUI/PropertiesBox.h, [245](#)
- /daten/Projekte/eclipse\_workspace/simpleanalyzer-gui/src/GUI/Renderer.cpp, [246](#)
- /daten/Projekte/eclipse\_workspace/simpleanalyzer-gui/src/GUI/Renderer.h, [251](#)
- /daten/Projekte/eclipse\_workspace/simpleanalyzer-gui/src/GUI/ViewpropBox.cpp, [252](#)
- /daten/Projekte/eclipse\_workspace/simpleanalyzer-gui/src/GUI/ViewpropBox.h, [253](#)
- /daten/Projekte/eclipse\_workspace/simpleanalyzer-gui/src/GUI/constants.h, [226](#)
- /daten/Projekte/eclipse\_workspace/simpleanalyzer-gui/src/SimpleAnalyzerApp.cpp, [270](#)
- /daten/Projekte/eclipse\_workspace/simpleanalyzer-

- gui/src/SimpleAnalyzerApp.h, 271
- /daten/Projekte/eclipse\_workspace/simpleanalyzer-  
gui/src/fileIO/Exporter.cpp, 222
- /daten/Projekte/eclipse\_workspace/simpleanalyzer-  
gui/src/fileIO/Exporter.h, 223
- /daten/Projekte/eclipse\_workspace/simpleanalyzer-  
gui/src/fileIO/Importer.cpp, 224
- /daten/Projekte/eclipse\_workspace/simpleanalyzer-  
gui/src/fileIO/Importer.h, 225
- /daten/Projekte/eclipse\_workspace/simpleanalyzer-  
gui/src/libraries/interpolate/GeometryClasses.-  
cpp, 254
- /daten/Projekte/eclipse\_workspace/simpleanalyzer-  
gui/src/libraries/interpolate/GeometryClasses.-  
h, 256
- /daten/Projekte/eclipse\_workspace/simpleanalyzer-  
gui/src/libraries/interpolate/Interpolator.cpp,  
257
- /daten/Projekte/eclipse\_workspace/simpleanalyzer-  
gui/src/libraries/interpolate/Interpolator.h, 259
- /daten/Projekte/eclipse\_workspace/simpleanalyzer-  
gui/src/processing/Analyzer.cpp, 260
- /daten/Projekte/eclipse\_workspace/simpleanalyzer-  
gui/src/processing/Analyzer.h, 261
- /daten/Projekte/eclipse\_workspace/simpleanalyzer-  
gui/src/processing/MeshProcessor.cpp, 262
- /daten/Projekte/eclipse\_workspace/simpleanalyzer-  
gui/src/processing/MeshProcessor.h, 264
- /daten/Projekte/eclipse\_workspace/simpleanalyzer-  
gui/src/processing/ObjectData.cpp, 265
- /daten/Projekte/eclipse\_workspace/simpleanalyzer-  
gui/src/processing/ObjectData.h, 266
- /daten/Projekte/eclipse\_workspace/simpleanalyzer-  
gui/src/processing/Utils.cpp, 267
- /daten/Projekte/eclipse\_workspace/simpleanalyzer-  
gui/src/processing/Utils.h, 268
- ALGORITHM\_RAY
  - Utils, 14
- ALGORITHM\_TETRAHEDRONS
  - Utils, 14
- add
  - Vector3D, 199
- addObject
  - GUIMainWindow, 88
  - SimpleAnalyzerApp, 182
- addSensorData
  - ObjectData, 137
- addTimedData
  - ObjectData, 138
- analyze\_window\_valid
  - GUIMainWindow, 96
- analyzeMarkerCheckBox
  - PropertiesBox, 163
- analyzeObject
  - Analyzer, 28
- analyzePoint
  - Analyzer, 29
  - GUIAnalyzePointWindow, 50
- Analyzer, 27
  - ~Analyzer, 28
  - analyzeObject, 28
  - analyzePoint, 29
  - Analyzer, 28
  - operator<<, 30
- Analyzer.cpp
  - operator<<, 261
- Analyzer::AnalyzerData\_dataset, 31
  - heat\_energy, 31
  - mat\_data, 31
  - name, 32
- Analyzer::AnalyzerData\_material, 32
  - heat\_energy, 32
  - name, 32
  - volume, 32
- Analyzer::AnalyzerData\_object, 33
  - data\_sets, 34
  - volume, 34
- Analyzer::AnalyzerData\_point, 34
  - extrapolated, 35
  - value, 35
- analyzerframe
  - GUIMainWindow, 96
- assignCurrentObjectProps
  - GUIMainWindow, 88
- assignViewProps
  - GUIMainWindow, 88
- attrib\_list
  - GUIGLCanvas.cpp, 238
- auto\_delta
  - TsdMerger::Options, 154
- autoUpdateCeckBox
  - PropertiesBox, 163
- basetemp
  - OdisiToSdConverter::Options, 155
- CSV\_SEPARATOR
  - Exporter, 45
- calcStepWidth
  - GUITimeline, 109
- calcbt
  - GUIAnalyzePointWindow, 51
  - GUICutRenderWindow, 74
- calculateIO
  - ObjectData, 138
- cameraPosition
  - Renderer::Viewport\_info, 211
- canvas
  - GUICutRenderWindow, 75
- clampHue
  - Utils, 14
- clearAnalyzeMarkerBt
  - PropertiesBox, 163
- clearMarkers
  - GUITimeline, 109
- color
  - ObjectData::MaterialData, 127

- colorRangeLbl
  - ViewpropBox, [216](#)
- colorRangeMaxEdit
  - ViewpropBox, [216](#)
- colorRangeMinEdit
  - ViewpropBox, [216](#)
- configpaths
  - CsvToSdConverter, [41](#)
  - GUIMainWindow, [97](#)
  - OdisiToSdConverter, [151](#)
- constants.h
  - ID\_ABOUT, [228](#)
  - ID\_ANALYZE, [228](#)
  - ID\_ANALYZE\_MARKER\_CB, [228](#)
  - ID\_ANALYZE\_POINT, [228](#)
  - ID\_ANALYZE\_POINT\_BT, [228](#)
  - ID\_AUTO\_UPDATE\_CB, [228](#)
  - ID\_CHANGE\_ACTIVE\_OBJ, [228](#)
  - ID\_CHECKLISTBOX\_VIEW\_PROP, [228](#)
  - ID\_CLEAR\_MARKER\_BT, [228](#)
  - ID\_COLORSCALE\_COLORBT, [228](#)
  - ID\_COLORSCALE\_PROP, [228](#)
  - ID\_CUT\_CANVAS, [228](#)
  - ID\_CUT\_TRI\_EDIT, [228](#)
  - ID\_DELETE\_ACTIVE\_OBJ, [228](#)
  - ID\_EXPORT\_CUT\_CSV\_BT, [228](#)
  - ID\_EXPORT\_CUT\_IMG\_BT, [228](#)
  - ID\_EXPORT\_VIEWPORT, [228](#)
  - ID\_EXPORT\_VTK, [228](#)
  - ID\_FIND\_MAX\_BT, [228](#)
  - ID\_GENERAL\_PROP, [228](#)
  - ID\_GENERAL\_VIEW\_PROP, [228](#)
  - ID\_IMMEDIATE\_UPDATE\_PROP, [228](#)
  - ID\_IMPORT\_OBJ, [228](#)
  - ID\_IMPORT\_SD, [228](#)
  - ID\_IMPORT\_TSD, [228](#)
  - ID\_MARKER\_NEXT\_BT, [228](#)
  - ID\_MARKER\_PREV\_BT, [228](#)
  - ID\_MATERIALBOX, [228](#)
  - ID\_OPEN\_MANUAL, [228](#)
  - ID\_RECALCBT, [228](#)
  - ID\_RENDER\_CUT, [228](#)
  - ID\_RENDER\_CUT\_BT, [228](#)
  - ID\_SD\_BOX, [228](#)
  - ID\_SD\_TIMELINE, [228](#)
  - ID\_TEST, [228](#)
- constants.h
  - EventID, [227](#)
- contains
  - CsvToSdConverter, [36](#), [37](#)
  - OdisiToSdConverter, [147](#)
- convert
  - CsvToSdConverter, [37](#)
  - OdisiToSdConverter, [147](#)
- coords
  - Utils::SensorPoint, [178](#)
  - Vector3D, [209](#)
- copy
  - Vector3D, [199](#)
- copySensorPoint
  - Utils, [15](#)
- core\_count
  - GUICutRenderWindow, [75](#)
- crossProduct
  - Vector3D, [200](#)
- CsvToSdConverter, [35](#)
  - configpaths, [41](#)
  - contains, [36](#), [37](#)
  - convert, [37](#)
  - getTextBlock, [37](#)
  - NUMBEROFPATHS, [41](#)
  - opts, [41](#)
  - parseArguments, [38](#)
  - parseLine, [38](#)
  - readConfiguration, [38](#)
  - readInputFile, [39](#)
  - readSensorDefinitions, [39](#)
  - replaceAll, [39](#)
  - writeOutputFile, [41](#)
- CsvToSdConverter::Options, [152](#)
  - max\_time, [152](#)
  - min\_time, [152](#)
  - namecol, [152](#)
  - replace\_comma\_with\_point, [152](#)
  - separator, [153](#)
  - start\_col, [153](#)
  - time\_step\_delta, [153](#)
  - timecol, [153](#)
- csvtosd/main.cpp
  - main, [219](#)
- current\_data\_object\_index
  - SimpleAnalyzerApp, [184](#)
- current\_material
  - PropertiesBox, [164](#)
- current\_mx
  - GUIColorScalePanel, [60](#)
  - GUICutCanvas, [104](#)
- current\_my
  - GUIColorScalePanel, [60](#)
  - GUICutCanvas, [104](#)
- current\_sensor\_index
  - ObjectData, [144](#)
- current\_time\_index
  - Utils::SensorData, [176](#)
- cut
  - Renderer::Viewport\_info, [211](#)
- cut\_visualisation\_info
  - Renderer, [175](#)
- data
  - Utils::SensorData, [176](#)
- data\_directory
  - GUIMainWindow, [97](#)
- data\_objects
  - SimpleAnalyzerApp, [184](#)
- data\_sets
  - Analyzer::AnalyzerData\_object, [34](#)

- delta
  - TsdMerger::Options, 154
- delta\_v\_view
  - GUITimeline, 117
- deltaX
  - GUIRenderCutCanvas, 104
- deltaY
  - GUIRenderCutCanvas, 104
- density
  - ObjectData::MaterialData, 127
- densityEdit
  - PropertiesBox, 164
- densityLbl
  - PropertiesBox, 164
- displayList
  - Renderer, 175
- distance
  - Utils::SortStruct, 185
- do\_refresh
  - GUIGLCanvas, 83
- dotProduct
  - Vector3D, 201
- doxygen\_dep\_dummy.h, 221
- drawCutRenderInfo
  - Renderer.cpp, 247
- drawVector
  - Renderer.cpp, 248
- EPSILON
  - GeometryClasses.cpp, 255
  - utils.cpp, 268
- edgesCheckBox
  - ViewpropBox, 216
- element
  - std::vector, 196
- elements
  - Matrix3D, 132
- equals
  - Vector3D, 201
- error\_threshold
  - OdisiToSdConverter::Options, 155
- EventID
  - constants.h, 227
- export\_csv\_bt
  - GUICutRenderWindow, 75
- export\_img\_bt
  - GUICutRenderWindow, 75
- ExportCutCSV
  - Exporter, 44
- ExportLegacyVTK
  - Exporter, 45
- Exporter, 43
  - ~Exporter, 44
  - CSV\_SEPARATOR, 45
  - ExportCutCSV, 44
  - ExportLegacyVTK, 45
  - Exporter, 44
- Exporter.cpp
  - tetface\_indices, 223
- extrapolated
  - Analyzer::AnalyzerData\_point, 35
  - ObjectData::MaterialData, 127
- facesCheckBox
  - ViewpropBox, 216
- fiber\_step\_delta
  - OdisiToSdConverter::Options, 155
- findMaxBt
  - PropertiesBox, 164
- findMaxValue
  - GUITimeline, 109
- fitBounds
  - GUIColorScalePanel, 54
- flipobj
  - OdisiToSdConverter::Options, 155
- floattostr
  - OdisiToSdConverter, 148
  - Utils, 15
- floattowxstr
  - Utils, 16
- font\_size
  - GUIColorScalePanel, 60
- GTL\_DEFAULT
  - GUITimeline, 108
- GUIColorScalePanel
  - SCM\_HORIZONTAL, 54
  - SCM\_NONE, 54
  - SCM\_VERTICAL, 54
- GUITimeline
  - GTL\_DEFAULT, 108
- GUI\_TIMELINE\_STYLE
  - GUITimeline, 108
- GUIAnalyzeOutputWindow, 46
  - ~GUIAnalyzeOutputWindow, 47
  - GUIAnalyzeOutputWindow, 47
  - GUIAnalyzeOutputWindow, 47
  - OnKeyPress, 47
  - SelectAll, 47
  - table, 48
  - ToClipboard, 48
  - Update, 48
- GUIAnalyzePointWindow, 49
  - ~GUIAnalyzePointWindow, 50
  - analyzePoint, 50
  - calcbt, 51
  - GUIAnalyzePointWindow, 50
  - GUIAnalyzePointWindow, 50
  - interpolationModeLabel, 51
  - interpolationModeList, 51
  - label, 51
  - xedit, 51
  - yedit, 52
  - zedit, 52
- GUIColorScalePanel, 52
  - ~GUIColorScalePanel, 54
  - current\_mx, 60
  - current\_my, 60

- fitBounds, [54](#)
- font\_size, [60](#)
- GUIColorScalePanel, [54](#)
- getDisplayArea, [55](#)
- getFontSize, [55](#)
- getImage, [55](#)
- getMode, [55](#)
- getStepWidth, [55](#)
- getTextColor, [56](#)
- getX, [56](#)
- getY, [56](#)
- GUIColorScalePanel, [54](#)
- handleMouse, [56](#)
- height, [60](#)
- image, [60](#)
- mode, [61](#)
- mouseOnDisplayArea, [57](#)
- paintTo, [57](#)
- prev\_mouse\_down, [61](#)
- refresh, [58](#)
- ScaleMode, [54](#)
- scaling, [61](#)
- setFontSize, [59](#)
- setMode, [59](#)
- setStepWidth, [59](#)
- setTextColor, [60](#)
- step\_width, [61](#)
- text\_color, [61](#)
- transforming, [61](#)
- width, [61](#)
- x, [61](#)
- y, [61](#)
- GUIColorScalePanel.cpp
  - MIN\_HEIGHT, [232](#)
  - MIN\_WIDTH, [232](#)
- GUICutRenderWindow, [62](#)
  - ~GUICutRenderWindow, [65](#)
  - calcbt, [74](#)
  - canvas, [75](#)
  - core\_count, [75](#)
  - export\_csv\_bt, [75](#)
  - export\_img\_bt, [75](#)
  - GUICutRenderWindow, [65](#)
  - getCutRenderProperties, [66](#)
  - GUICutRenderWindow, [65](#)
  - image, [75](#)
  - imgHeightEdit, [75](#)
  - imgWidthEdit, [75](#)
  - mmpixeledit, [75](#)
  - mmpixellabel, [75](#)
  - OnCSColorBtClick, [68](#)
  - OnColorScaleChanged, [66](#)
  - OnColorScaleChanged\_spin, [67](#)
  - OnCutPropsChanged, [68](#)
  - OnExportCSV, [69](#)
  - OnExportImage, [69](#)
  - OnResize, [70](#)
  - OnSCutPropsChanged\_spin, [70](#)
  - optionslbl, [75](#)
  - p1label, [76](#)
  - p1xedit, [76](#)
  - p1yedit, [76](#)
  - p1zedit, [76](#)
  - p2label, [76](#)
  - p2xedit, [76](#)
  - p2yedit, [76](#)
  - p2zedit, [76](#)
  - p3label, [76](#)
  - p3xedit, [77](#)
  - p3yedit, [77](#)
  - p3zedit, [77](#)
  - refreshVisualisation, [71](#)
  - renderCutBtClick, [72](#)
  - renderImage, [73](#)
  - scalefontcolorbt, [77](#)
  - scalefontpropslbl, [77](#)
  - scalefontsizeedit, [77](#)
  - scalelbl, [77](#)
  - scalemodecb, [77](#)
  - scalemodelbl, [77](#)
  - scalesteppedit, [78](#)
  - scroll\_pane, [78](#)
  - threadcountedit, [78](#)
  - threadcountlbl, [78](#)
  - trilabel, [78](#)
  - value\_img, [78](#)
  - widthHeightlbl, [78](#)
- GUICutRenderWindow.cpp
  - render\_thread, [234](#)
- GUIGLCanvas, [79](#)
  - ~GUIGLCanvas, [80](#)
  - do\_refresh, [83](#)
  - GUIGLCanvas, [80](#)
  - getRenderer, [81](#)
  - GUIGLCanvas, [80](#)
  - is\_initialized, [83](#)
  - OnMouseMove, [81](#)
  - OnMouseWheel, [81](#)
  - OnPaint, [82](#)
  - OnResize, [82](#)
  - prev\_mouse\_x, [83](#)
  - prev\_mouse\_y, [83](#)
  - refresh, [82](#)
  - renderer, [83](#)
  - setRenderObject, [83](#)
- GUIGLCanvas.cpp
  - attrib\_list, [238](#)
- GUIMainWindow, [84](#)
  - ~GUIMainWindow, [88](#)
  - addObject, [88](#)
  - analyze\_window\_valid, [96](#)
  - analyzerframe, [96](#)
  - assignCurrentObjectProps, [88](#)
  - assignViewProps, [88](#)
  - configpaths, [97](#)
  - data\_directory, [97](#)

- GUIMainWindow, 87
- getGLCanvas, 89
- gl\_context, 97
- GUIMainWindow, 87
- mwAnalyzeMenu, 97
- mwEditMenu, 97
- mwExportMenu, 97
- mwFileMenu, 97
- mwHelpMenu, 97
- mwImportMenu, 98
- mwMenuBar, 98
- NUMBEROFPATHS, 98
- OnActiveObjectChange, 89
- OnActiveObjectChangePopup, 89
- OnActiveObjectDelete, 90
- OnAnalyze, 90
- OnAnalyzeMarkerChange, 90
- OnAnalyzePoint, 90
- OnAutoUpdateChange, 90
- OnExportVTK, 90
- OnExportViewportImage, 90
- OnFindMaxTSD, 91
- OnGeneralPropChange, 91
- OnImmediateUpdatePropChange, 91
- OnMaterialSelect, 91
- OnMenuFileQuit, 91
- OnMenuHelpAbout, 91
- OnMenuImportObj, 92
- OnMenuImportSD, 92
- OnMenuImportTSD, 92
- OnMenuOpenManual, 93
- OnRecalcBtClick, 93
- OnRenderCut, 93
- OnResize, 93
- OnSDTLMarkerClear, 94
- OnSDTLNextMarker, 94
- OnSDTLPrevMarker, 94
- OnSDTimelineChange, 93
- OnSensorDataChange, 94
- OnViewPropChange, 94
- OnViewPropSpinChange, 94
- prop\_scroll\_win, 98
- propbox, 98
- render\_cut\_window\_valid, 98
- rendercutwindow, 98
- setActiveObject, 94
- setAnalyzeWindowStatus, 94
- setCutRenderWindowStatus, 95
- toolbar, 98
- updateObjectPropGUI, 95
- updateViewPropGUI, 96
- updating, 98
- view\_scroll\_win, 99
- viewbox, 99
- GUIMainWindow.cpp
- PATH\_SEPARATOR, 239
- PROPBOXWIDTH, 239
- VIEWBOXWIDTH, 239
- GUIRenderCutCanvas, 99
  - ~GUIRenderCutCanvas, 101
  - current\_mx, 104
  - current\_my, 104
  - deltaX, 104
  - deltaY, 104
  - GUIRenderCutCanvas, 101
  - getScalePanel, 101
  - GUIRenderCutCanvas, 101
  - image, 105
  - mouse\_to\_scalepanel, 105
  - onCanvasPaint, 102
  - OnMouseDown, 102
  - OnMouseMove, 103
  - OnMouseWheel, 103
  - OnResize, 103
  - scalepanel, 105
  - setImage, 103
  - setValueImg, 104
  - value\_img, 105
  - zoom, 105
- GUITimeline, 105
  - ~GUITimeline, 109
  - calcStepWidth, 109
  - clearMarkers, 109
  - delta\_v\_view, 117
  - findMaxValue, 109
  - GUITimeline, 108
  - getMarkers, 110
  - getMaxValue, 110
  - getMinValue, 111
  - getValue, 111
  - GUITimeline, 108
  - isMarked, 111
  - markers, 117
  - maxdigits, 117
  - maxvalue, 117
  - minvalue, 117
  - names, 117
  - OnKeyDown, 112
  - OnMouseDown, 112
  - OnMouseMove, 112
  - OnMouseWheel, 112
  - OnPaint, 113
  - OnResize, 113
  - posToVal, 113
  - prev\_mouse\_x, 117
  - sendTimelineEvent, 114
  - setMarked, 114
  - setMarkerList, 114
  - setMarkers, 115
  - setMaxValue, 115
  - setMinValue, 115
  - setNameList, 116
  - setValue, 116
  - value, 118
  - zoom, 118
- GUITimeline.cpp



- refine\_factors, 243
- GeometryClasses.cpp
  - EPSILON, 255
  - operator<<, 255
  - sqr, 256
- GeometryClasses.h
  - operator<<, 257
- getActiveObject
  - SimpleAnalyzerApp, 182
- getAnalyzeMarkerCheckBox
  - PropertiesBox, 160
- getAngleTo
  - Vector3D, 202
- getAutoUpdateCeckBox
  - PropertiesBox, 160
- getClearAnalyzeMarkerBt
  - PropertiesBox, 160
- getColorRangeMaxEdit
  - ViewpropBox, 215
- getColorRangeMinEdit
  - ViewpropBox, 215
- getCurrentDataObjectIndex
  - SimpleAnalyzerApp, 183
- getCurrentMaterial
  - PropertiesBox, 161
- getCurrentSensorIndex
  - ObjectData, 139
- getCutRenderProperties
  - GUICutRenderWindow, 66
- getDataObjects
  - SimpleAnalyzerApp, 183
- getDensityEdit
  - PropertiesBox, 161
- getDisplayArea
  - GUIColorScalePanel, 55
- getDistance\_d
  - Utils::SensorPointComparator, 179
- getDistanceTo
  - Vector3D, 203
- getEdgesCheckBox
  - ViewpropBox, 215
- getFaceIndex
  - Importer.cpp, 224
- getFacesCheckBox
  - ViewpropBox, 215
- getFindMaxBt
  - PropertiesBox, 161
- getFontSize
  - GUIColorScalePanel, 55
- getGLCanvas
  - GUIMainWindow, 89
- getImage
  - GUIColorScalePanel, 55
- getInterpolationModeList
  - PropertiesBox, 161
- getLength
  - Vector3D, 204
- getMarkers
  - GUITimeline, 110
- getMatListBox
  - PropertiesBox, 161
- getMatNameEdit
  - PropertiesBox, 161
- getMatPropBox
  - PropertiesBox, 161
- getMatVisibilityListBox
  - ViewpropBox, 215
- getMaterials
  - ObjectData, 140
- getMaxValue
  - GUITimeline, 110
- getMaxVolumeEdit
  - PropertiesBox, 161
- getMaxvolume
  - ObjectData, 140
- getMinValue
  - GUITimeline, 111
- getMode
  - GUIColorScalePanel, 55
- getName
  - ObjectData, 141
- getNextMarkerBt
  - PropertiesBox, 161
- getNormal
  - Triangle, 189
- getObjNameEdit
  - PropertiesBox, 162
- getPointValue
  - Utils, 18
- getPointsCheckBox
  - ViewpropBox, 215
- getPrevMarkerBt
  - PropertiesBox, 162
- getQuality
  - ObjectData, 141
- getQualityEdit
  - PropertiesBox, 162
- getRecalcButton
  - PropertiesBox, 162
- getRenderer
  - GUIGLCanvas, 81
- getScalePanel
  - GUIRenderCutCanvas, 101
- getSdTimeline
  - PropertiesBox, 162
- getSensorDataList
  - ObjectData, 141
  - PropertiesBox, 162
- getShowExtrapolatedCheckBox
  - ViewpropBox, 215
- getShowShowSensorData
  - ViewpropBox, 215
- getSign
  - Interpolator.cpp, 258
- getSpecificHeatCapEdit
  - PropertiesBox, 162

- getStepWidth
  - GUIColorScalePanel, 55
- getTextBlock
  - CsvToSdConverter, 37
  - Importer.cpp, 225
  - OdisiToSdConverter, 148
  - TsdMerger, 193
- getTextColor
  - GUIColorScalePanel, 56
- getUpToDateLbl
  - PropertiesBox, 162
- getV1
  - Tetrahedron, 186
  - Triangle, 190
- getV2
  - Tetrahedron, 186
  - Triangle, 190
- getV3
  - Tetrahedron, 187
  - Triangle, 191
- getV4
  - Tetrahedron, 187
- getValue
  - GUITimeline, 111
- getVert
  - Tetrahedron, 187
  - Triangle, 191
- getViewScaleEdit
  - ViewpropBox, 216
- getViewport
  - Renderer, 169
- getViewportImage
  - Renderer, 169
- getVisualizationInfo
  - SimpleAnalyzerApp, 183
- getX
  - GUIColorScalePanel, 56
  - Vector3D, 205
- getXYZ
  - Vector3D, 205
- getY
  - GUIColorScalePanel, 56
  - Vector3D, 206
- getZ
  - Vector3D, 206
- gl\_context
  - GUIMainWindow, 97
- handleMouse
  - GUIColorScalePanel, 56
- heat\_energy
  - Analyzer::AnalyzerData\_dataset, 31
  - Analyzer::AnalyzerData\_material, 32
- height
  - GUIColorScalePanel, 60
  - OdisiToSdConverter::Options, 155
  - Renderer::Viewport\_info, 211
- hsvToRgb
  - Utils, 19
- ID\_ABOUT
  - constants.h, 228
- ID\_ANALYZE
  - constants.h, 228
- ID\_ANALYZE\_MARKER\_CB
  - constants.h, 228
- ID\_ANALYZE\_POINT
  - constants.h, 228
- ID\_ANALYZE\_POINT\_BT
  - constants.h, 228
- ID\_AUTO\_UPDATE\_CB
  - constants.h, 228
- ID\_CHANGE\_ACTIVE\_OBJ
  - constants.h, 228
- ID\_CHECKLISTBOX\_VIEW\_PROP
  - constants.h, 228
- ID\_CLEAR\_MARKER\_BT
  - constants.h, 228
- ID\_COLORSCALE\_COLORBT
  - constants.h, 228
- ID\_COLORSCALE\_PROP
  - constants.h, 228
- ID\_CUT\_CANVAS
  - constants.h, 228
- ID\_CUT\_TRI\_EDIT
  - constants.h, 228
- ID\_DELETE\_ACTIVE\_OBJ
  - constants.h, 228
- ID\_EXPORT\_CUT\_CSV\_BT
  - constants.h, 228
- ID\_EXPORT\_CUT\_IMG\_BT
  - constants.h, 228
- ID\_EXPORT\_VIEWPORT
  - constants.h, 228
- ID\_EXPORT\_VTK
  - constants.h, 228
- ID\_FIND\_MAX\_BT
  - constants.h, 228
- ID\_GENERAL\_PROP
  - constants.h, 228
- ID\_GENERAL\_VIEW\_PROP
  - constants.h, 228
- ID\_IMMEDIATE\_UPDATE\_PROP
  - constants.h, 228
- ID\_IMPORT\_OBJ
  - constants.h, 228
- ID\_IMPORT\_SD
  - constants.h, 228
- ID\_IMPORT\_TSD
  - constants.h, 228
- ID\_MARKER\_NEXT\_BT
  - constants.h, 228
- ID\_MARKER\_PREV\_BT
  - constants.h, 228
- ID\_MATERIALBOX
  - constants.h, 228
- ID\_OPEN\_MANUAL
  - constants.h, 228

- ID\_RECALCBT
  - constants.h, [228](#)
- ID\_RENDER\_CUT
  - constants.h, [228](#)
- ID\_RENDER\_CUT\_BT
  - constants.h, [228](#)
- ID\_SD\_BOX
  - constants.h, [228](#)
- ID\_SD\_TIMELINE
  - constants.h, [228](#)
- ID\_TEST
  - constants.h, [228](#)
- image
  - GUIColorScalePanel, [60](#)
  - GUICutRenderWindow, [75](#)
  - GUIRenderCutCanvas, [105](#)
- img\_height
  - Utils::CutRender\_info, [42](#)
- img\_width
  - Utils::CutRender\_info, [42](#)
- imgHeightEdit
  - GUICutRenderWindow, [75](#)
- imgWidthEdit
  - GUICutRenderWindow, [75](#)
- ImportObj
  - Importer, [119](#)
- Importer, [118](#)
  - ~Importer, [118](#)
  - ImportObj, [119](#)
  - Importer, [118](#)
  - LoadSensorData, [119](#)
  - LoadTimedData, [120](#)
- Importer.cpp
  - getFacelIndex, [224](#)
  - getTextBlock, [225](#)
  - PATH\_SEPARATOR, [224](#)
- in\_volume\_algorithm
  - Utils::CutRender\_info, [43](#)
- initGL
  - Renderer, [169](#)
- interpolatePoint
  - MeshProcessor.cpp, [263](#)
- interpolateTet
  - Interpolator, [123](#)
- interpolateTri
  - Interpolator, [124](#)
- interpolation\_mode
  - ObjectData::MaterialData, [127](#)
- InterpolationMode
  - Interpolator, [122](#)
- interpolationModeLabel
  - GUIAnalyzePointWindow, [51](#)
- interpolationModeLbl
  - PropertiesBox, [164](#)
- interpolationModeList
  - GUIAnalyzePointWindow, [51](#)
  - PropertiesBox, [164](#)
- Interpolator, [121](#)
  - ~Interpolator, [122](#)
  - interpolateTet, [123](#)
  - interpolateTri, [124](#)
  - InterpolationMode, [122](#)
  - Interpolator, [122](#)
  - LINEAR, [122](#)
  - LOGARITHMIC, [122](#)
  - mode, [126](#)
  - setMode, [125](#)
- Interpolator.cpp
  - getSign, [258](#)
  - PI, [258](#)
  - sqr, [259](#)
- invertcut
  - Renderer::Viewport\_info, [211](#)
- is\_initialized
  - GUIGLCanvas, [83](#)
- isMarked
  - GUITimeline, [111](#)
- LINEAR
  - Interpolator, [122](#)
- LOGARITHMIC
  - Interpolator, [122](#)
- label
  - GUIAnalyzePointWindow, [51](#)
- loadFromFile
  - ObjectData, [142](#)
- LoadSensorData
  - Importer, [119](#)
- LoadTimedData
  - Importer, [120](#)
- MIN\_HEIGHT
  - GUIColorScalePanel.cpp, [232](#)
- MIN\_WIDTH
  - GUIColorScalePanel.cpp, [232](#)
- main
  - csvtosd/main.cpp, [219](#)
  - mergetsd.cpp, [222](#)
  - odisitosd/main.cpp, [220](#)
- markers
  - GUITimeline, [117](#)
  - Utils::SensorData, [177](#)
- mat\_data
  - Analyzer::AnalyzerData\_dataset, [31](#)
- matListBox
  - PropertiesBox, [164](#)
- matListBoxLbl
  - PropertiesBox, [164](#)
- matNameEdit
  - PropertiesBox, [164](#)
- matNameLbl
  - PropertiesBox, [165](#)
- matPropBox
  - PropertiesBox, [165](#)
- matVisibilityListBox
  - ViewpropBox, [216](#)
- matVisualizationLbl

- ViewpropBox, 216
- materials
  - ObjectData, 144
- Matrix3D, 128
  - elements, 132
  - Matrix3D, 129
  - Matrix3D, 129
  - mult, 129
  - print, 131
  - rotateX, 131
  - rotateY, 131
  - rotateZ, 132
  - transpose, 132
- max\_dt
  - TsdMerger::Options, 154
- max\_time
  - CsvToSdConverter::Options, 152
  - OdisiToSdConverter::Options, 155
- max\_visualisation\_temp
  - Utils::Visualization\_info, 218
- maxVolumeEdit
  - PropertiesBox, 165
- maxVolumeLbl
  - PropertiesBox, 165
- maxdigits
  - GUITimeline, 117
- maxfwcount
  - OdisiToSdConverter::Options, 156
- maxvalue
  - GUITimeline, 117
- maxvolume
  - ObjectData, 144
- merge
  - TsdMerger, 193
- mergetsd.cpp
  - main, 222
- MeshProcessor, 133
  - ~MeshProcessor, 133
  - MeshProcessor, 133
  - MeshProcessor, 133
  - process, 133
- MeshProcessor.cpp
  - interpolatePoint, 263
- meshpoint
  - Utils::SensorPointComparator, 180
- min\_time
  - CsvToSdConverter::Options, 152
  - OdisiToSdConverter::Options, 156
- min\_visualisation\_temp
  - Utils::Visualization\_info, 218
- minvalue
  - GUITimeline, 117
- mmperpixel
  - Utils::CutRender\_info, 43
- mmperpixeledit
  - GUICutRenderWindow, 75
- mmperpixellabel
  - GUICutRenderWindow, 75
- mode
  - GUIColorScalePanel, 61
  - Interpolator, 126
- mouse\_to\_scalepanel
  - GUICutRenderCanvas, 105
- mouseOnDisplayArea
  - GUIColorScalePanel, 57
- mult
  - Matrix3D, 129
  - Vector3D, 207
- mwAnalyzeMenu
  - GUIMainWindow, 97
- mwEditMenu
  - GUIMainWindow, 97
- mwExportMenu
  - GUIMainWindow, 97
- mwFileMenu
  - GUIMainWindow, 97
- mwHelpMenu
  - GUIMainWindow, 97
- mwImportMenu
  - GUIMainWindow, 98
- mwMenuBar
  - GUIMainWindow, 98
- NUMBEROFPATHS
  - CsvToSdConverter, 41
  - GUIMainWindow, 98
  - OdisiToSdConverter, 151
- name
  - Analyzer::AnalyzerData\_dataset, 32
  - Analyzer::AnalyzerData\_material, 32
  - ObjectData, 144
  - ObjectData::MaterialData, 127
  - Utils::SensorData, 177
- namecol
  - CsvToSdConverter::Options, 152
- names
  - GUITimeline, 117
- nextCombination
  - Utils, 20
- nextMarkerBt
  - PropertiesBox, 165
- normalize
  - Vector3D, 207
- OD\_FAILURE
  - ObjectData, 137
- OD\_LOAD\_ALREADY\_LOADED
  - ObjectData, 137
- OD\_LOAD\_INVALID\_FILE
  - ObjectData, 137
- OD\_LOAD\_INVALID\_SENSOR\_FILE
  - ObjectData, 137
- OD\_SUCCESS
  - ObjectData, 137
- objNameEdit
  - PropertiesBox, 165
- objNameLbl

- PropertiesBox, 165
- object
  - Renderer, 175
- ObjectData
  - OD\_FAILURE, 137
  - OD\_LOAD\_ALREADY\_LOADED, 137
  - OD\_LOAD\_INVALID\_FILE, 137
  - OD\_LOAD\_INVALID\_SENSOR\_FILE, 137
  - OD\_SUCCESS, 137
- ObjectData, 134
  - ~ObjectData, 137
  - addSensorData, 137
  - addTimedData, 138
  - calculateIO, 138
  - current\_sensor\_index, 144
  - getCurrentSensorIndex, 139
  - getMaterials, 140
  - getMaxvolume, 140
  - getName, 141
  - getQuality, 141
  - getSensorDataList, 141
  - loadFromFile, 142
  - materials, 144
  - maxvolume, 144
  - name, 144
  - ObjectData, 137
  - ObjectDataStatus, 136
  - ObjectData, 137
  - quality, 145
  - sensorDataList, 145
  - setCurrentSensorIndex, 143
  - setMaxvolume, 143
  - setName, 143
  - setQuality, 144
- ObjectData.cpp
  - PATH\_SEPARATOR, 266
- ObjectData::MaterialData, 126
  - color, 127
  - density, 127
  - extrapolated, 127
  - interpolation\_mode, 127
  - name, 127
  - specificheatcapacity, 127
  - tetgeninput, 127
  - tetgenoutput, 127
  - visible, 128
- ObjectDataStatus
  - ObjectData, 136
- objwidth
  - OdisiToSdConverter::Options, 156
- OdisiToSdConverter, 145
  - configpaths, 151
  - contains, 147
  - convert, 147
  - floattostr, 148
  - getTextBlock, 148
  - NUMBEROFPATHS, 151
  - opts, 151
  - parseArguments, 148
  - parseLine, 148
  - readConfiguration, 149
  - readInputFile, 149
  - readSensorDefinitions, 150
  - replaceAll, 150
  - writeOutputFile, 151
- OdisiToSdConverter::Options, 154
  - basetemp, 155
  - error\_threshold, 155
  - fiber\_step\_delta, 155
  - flipobj, 155
  - height, 155
  - max\_time, 155
  - maxfwcount, 156
  - min\_time, 156
  - objwidth, 156
  - replace\_comma\_with\_point, 156
  - separator, 156
  - startrow, 156
  - tab\_space\_count, 156
  - time\_step\_delta, 156
  - timecol, 156
- odisitosd/main.cpp
  - main, 220
- offset
  - TsdMerger::Options, 154
- OnActiveObjectChange
  - GUIMainWindow, 89
- OnActiveObjectChangePopup
  - GUIMainWindow, 89
- OnActiveObjectDelete
  - GUIMainWindow, 90
- OnAnalyze
  - GUIMainWindow, 90
- OnAnalyzeMarkerChange
  - GUIMainWindow, 90
- OnAnalyzePoint
  - GUIMainWindow, 90
- OnAutoUpdateChange
  - GUIMainWindow, 90
- OnCSColorBtClick
  - GUICutRenderWindow, 68
- onCanvasPaint
  - GUICutRenderWindow, 102
- OnColorScaleChanged
  - GUICutRenderWindow, 66
- OnColorScaleChanged\_spin
  - GUICutRenderWindow, 67
- OnCutPropsChanged
  - GUICutRenderWindow, 68
- OnExportCSV
  - GUICutRenderWindow, 69
- OnExportImage
  - GUICutRenderWindow, 69
- OnExportVTK
  - GUIMainWindow, 90
- OnExportViewportImage

- GUIMainWindow, 90
- OnFindMaxTSD
  - GUIMainWindow, 91
- OnGeneralPropChange
  - GUIMainWindow, 91
- OnImmediateUpdatePropChange
  - GUIMainWindow, 91
- OnInit
  - SimpleAnalyzerApp, 183
- OnKeyDown
  - GUITimeline, 112
- OnKeyPress
  - GUIAnalyzeOutputWindow, 47
- OnMaterialSelect
  - GUIMainWindow, 91
- OnMenuFileQuit
  - GUIMainWindow, 91
- OnMenuHelpAbout
  - GUIMainWindow, 91
- OnMenuImportObj
  - GUIMainWindow, 92
- OnMenuImportSD
  - GUIMainWindow, 92
- OnMenuImportTSD
  - GUIMainWindow, 92
- OnMenuOpenManual
  - GUIMainWindow, 93
- OnMouseDown
  - GUIRenderCutCanvas, 102
  - GUITimeline, 112
- OnMouseMove
  - GUIGLCanvas, 81
  - GUIRenderCutCanvas, 103
  - GUITimeline, 112
- OnMouseWheel
  - GUIGLCanvas, 81
  - GUIRenderCutCanvas, 103
  - GUITimeline, 112
- OnPaint
  - GUIGLCanvas, 82
  - GUITimeline, 113
- OnRecalcBtClick
  - GUIMainWindow, 93
- OnRenderCut
  - GUIMainWindow, 93
- OnResize
  - GUICutRenderWindow, 70
  - GUIGLCanvas, 82
  - GUIMainWindow, 93
  - GUIRenderCutCanvas, 103
  - GUITimeline, 113
- OnSCutPropsChanged\_spin
  - GUICutRenderWindow, 70
- OnSDTLMarkerClear
  - GUIMainWindow, 94
- OnSDTLNextMarker
  - GUIMainWindow, 94
- OnSDTLPrevMarker
  - GUIMainWindow, 94
- OnSDTimelineChange
  - GUIMainWindow, 93
- OnSensorDataChange
  - GUIMainWindow, 94
- OnViewPropChange
  - GUIMainWindow, 94
- OnViewPropSpinChange
  - GUIMainWindow, 94
- operator<<
  - Analyzer, 30
  - Analyzer.cpp, 261
  - GeometryClasses.cpp, 255
  - GeometryClasses.h, 257
  - Vector3D, 209
- operator()
  - Utils::SensorPointComparator, 180
- optionslbl
  - GUICutRenderWindow, 75
- opts
  - CsvToSdConverter, 41
  - OdisiToSdConverter, 151
  - TsdMerger, 195
- p1label
  - GUICutRenderWindow, 76
- p1xedit
  - GUICutRenderWindow, 76
- p1yedit
  - GUICutRenderWindow, 76
- p1zedit
  - GUICutRenderWindow, 76
- p2label
  - GUICutRenderWindow, 76
- p2xedit
  - GUICutRenderWindow, 76
- p2yedit
  - GUICutRenderWindow, 76
- p2zedit
  - GUICutRenderWindow, 76
- p3label
  - GUICutRenderWindow, 76
- p3xedit
  - GUICutRenderWindow, 77
- p3yedit
  - GUICutRenderWindow, 77
- p3zedit
  - GUICutRenderWindow, 77
- PATH\_SEPARATOR
  - GUIMainWindow.cpp, 239
  - Importer.cpp, 224
  - ObjectData.cpp, 266
- PI
  - Interpolator.cpp, 258
- PIM\_algorithm
  - Utils, 14
- PROPBOXWIDTH
  - GUIMainWindow.cpp, 239
- paintTo

- GUIColorScalePanel, 57
- parseArguments
  - CsvToSdConverter, 38
  - OdisiToSdConverter, 148
  - TsdMerger, 194
- parseFile
  - TsdMerger, 194
- parseLine
  - CsvToSdConverter, 38
  - OdisiToSdConverter, 148
- pointBehindCut
  - Renderer.cpp, 249
- pointIndex
  - Utils::SortStruct, 185
- pointInsideMesh
  - Utils, 21
- pointInsideTetrahedron
  - Utils, 22, 23
- pointsCheckBox
  - ViewpropBox, 217
- posToVal
  - GUITimeline, 113
- prev\_mouse\_down
  - GUIColorScalePanel, 61
- prev\_mouse\_x
  - GUIGLCanvas, 83
  - GUITimeline, 117
- prev\_mouse\_y
  - GUIGLCanvas, 83
- prevMarkerBt
  - PropertiesBox, 165
- print
  - Matrix3D, 131
  - Triangle, 191
  - Vector3D, 208
- printTo
  - Vector3D, 208
- process
  - MeshProcessor, 133
- prop\_scroll\_win
  - GUIMainWindow, 98
- propbox
  - GUIMainWindow, 98
- PropertiesBox, 157
  - ~PropertiesBox, 160
  - analyzeMarkerCheckBox, 163
  - autoUpdateCeckBox, 163
  - clearAnalyzeMarkerBt, 163
  - current\_material, 164
  - densityEdit, 164
  - densityLbl, 164
  - findMaxBt, 164
  - getAnalyzeMarkerCheckBox, 160
  - getAutoUpdateCeckBox, 160
  - getClearAnalyzeMarkerBt, 160
  - getCurrentMaterial, 161
  - getDensityEdit, 161
  - getFindMaxBt, 161
  - getInterpolationModeList, 161
  - getMatListBox, 161
  - getMatNameEdit, 161
  - getMatPropBox, 161
  - getMaxVolumeEdit, 161
  - getNextMarkerBt, 161
  - getObjNameEdit, 162
  - getPrevMarkerBt, 162
  - getQualityEdit, 162
  - getRecalcButton, 162
  - getSdTimeline, 162
  - getSensorDataList, 162
  - getSpecificHeatCapEdit, 162
  - getUpToDateLbl, 162
  - interpolationModeLbl, 164
  - interpolationModeList, 164
  - matListBox, 164
  - matListBoxLbl, 164
  - matNameEdit, 164
  - matNameLbl, 165
  - matPropBox, 165
  - maxVolumeEdit, 165
  - maxVolumeLbl, 165
  - nextMarkerBt, 165
  - objNameEdit, 165
  - objNameLbl, 165
  - prevMarkerBt, 165
  - PropertiesBox, 160
  - PropertiesBox, 160
  - qualityEdit, 165
  - qualityLbl, 166
  - recalcButton, 166
  - resize, 163
  - sdTimeline, 166
  - sensorDataLbl, 166
  - sensorDataList, 166
  - setCurrentMaterial, 163
  - specificHeatCapEdit, 166
  - specificHeatCapLbl, 166
  - upToDateLbl, 166
- PropertiesBox.cpp
  - sdfilestring, 245
- quality
  - ObjectData, 145
- qualityEdit
  - PropertiesBox, 165
- qualityLbl
  - PropertiesBox, 166
- RM\_MATERIALCOLOR
  - Renderer, 168
- RM\_NONE
  - Renderer, 168
- RM\_VALUECOLOR
  - Renderer, 168
- rayIntersectsTriangle
  - Utils, 24
- readConfiguration



- CsvToSdConverter, 38
  - OdisiToSdConverter, 149
- readInputFile
  - CsvToSdConverter, 39
  - OdisiToSdConverter, 149
- readSensorDefinitions
  - CsvToSdConverter, 39
  - OdisiToSdConverter, 150
- recalcButton
  - PropertiesBox, 166
- refine\_factors
  - GUITimeline.cpp, 243
- refresh
  - GUIColorScalePanel, 58
  - GUIGLCanvas, 82
- refreshVisualisation
  - GUICutRenderWindow, 71
- removeCurrentObject
  - SimpleAnalyzerApp, 183
- render
  - Renderer, 170
- render\_cut\_window\_valid
  - GUIMainWindow, 98
- render\_thread
  - GUICutRenderWindow.cpp, 234
- renderCutBtClick
  - GUICutRenderWindow, 72
- renderGrid
  - Renderer.cpp, 250
- renderImage
  - GUICutRenderWindow, 73
- renderMaterial
  - Renderer, 170
- RenderMode
  - Renderer, 168
- renderSensorData
  - Renderer, 172
- renderTetrahedra
  - Renderer, 172
- renderchoices
  - ViewpropBox.cpp, 253
- rendercutwindow
  - GUIMainWindow, 98
- Renderer, 167
  - ~Renderer, 169
  - cut\_visualisation\_info, 175
  - displayList, 175
  - getViewport, 169
  - getViewportImage, 169
  - initGL, 169
  - object, 175
  - RM\_MATERIALCOLOR, 168
  - RM\_NONE, 168
  - RM\_VALUECOLOR, 168
  - render, 170
  - renderMaterial, 170
  - RenderMode, 168
  - renderSensorData, 172
  - renderTetrahedra, 172
  - Renderer, 169
  - resize, 173
  - setCutRenderInfo, 173
  - setObject, 174
  - viewport, 175
- renderer
  - GUIGLCanvas, 83
- Renderer.cpp
  - drawCutRenderInfo, 247
  - drawVector, 248
  - pointBehindCut, 249
  - renderGrid, 250
- Renderer::Viewport\_info, 210
  - cameraPosition, 211
  - cut, 211
  - height, 211
  - invertcut, 211
  - rotationX, 211
  - rotationY, 211
  - scale, 211
  - show\_extrapolated, 212
  - show\_sensordata, 212
  - showEdges, 212
  - showFaces, 212
  - showPoints, 212
  - width, 212
  - zoom, 212
- replace\_comma\_with\_point
  - CsvToSdConverter::Options, 152
  - OdisiToSdConverter::Options, 156
- replaceAll
  - CsvToSdConverter, 39
  - OdisiToSdConverter, 150
- resize
  - PropertiesBox, 163
  - Renderer, 173
  - ViewpropBox, 216
- rotateX
  - Matrix3D, 131
- rotateY
  - Matrix3D, 131
- rotateZ
  - Matrix3D, 132
- rotationX
  - Renderer::Viewport\_info, 211
- rotationY
  - Renderer::Viewport\_info, 211
- SCM\_HORIZONTAL
  - GUIColorScalePanel, 54
- SCM\_NONE
  - GUIColorScalePanel, 54
- SCM\_VERTICAL
  - GUIColorScalePanel, 54
- SCALE\_REFINE\_STEPS
  - GUITimeline.cpp, 243
- scale
  - Renderer::Viewport\_info, 211



- ScaleMode
  - GUIColorScalePanel, 54
- scalefontcolorbt
  - GUICutRenderWindow, 77
- scalefontproplsbl
  - GUICutRenderWindow, 77
- scalefontsizeedit
  - GUICutRenderWindow, 77
- scaletbl
  - GUICutRenderWindow, 77
- scalemodecb
  - GUICutRenderWindow, 77
- scalemodelbl
  - GUICutRenderWindow, 77
- scalepanel
  - GUIRenderCutCanvas, 105
- scalestepedit
  - GUICutRenderWindow, 78
- scaling
  - GUIColorScalePanel, 61
- scroll\_pane
  - GUICutRenderWindow, 78
- sdTimeline
  - PropertiesBox, 166
- sdfilestring
  - PropertiesBox.cpp, 245
- SelectAll
  - GUIAnalyzeOutputWindow, 47
- sendTimelineEvent
  - GUITimeline, 114
- sensorDataLbl
  - PropertiesBox, 166
- sensorDataList
  - ObjectData, 145
  - PropertiesBox, 166
- separator
  - CsvToSdConverter::Options, 153
  - OdisiToSdConverter::Options, 156
- setActiveObject
  - GUIMainWindow, 94
- setAnalyzeWindowStatus
  - GUIMainWindow, 94
- setCurrentDataObjectIndex
  - SimpleAnalyzerApp, 183
- setCurrentMaterial
  - PropertiesBox, 163
- setCurrentSensorIndex
  - ObjectData, 143
- setCutRenderInfo
  - Renderer, 173
- setCutRenderWindowStatus
  - GUIMainWindow, 95
- setFontSize
  - GUIColorScalePanel, 59
- setImage
  - GUIRenderCutCanvas, 103
- setMarked
  - GUITimeline, 114
- setMarkerList
  - GUITimeline, 114
- setMarkers
  - GUITimeline, 115
- setMaxValue
  - GUITimeline, 115
- setMaxvolume
  - ObjectData, 143
- setMinValue
  - GUITimeline, 115
- setMode
  - GUIColorScalePanel, 59
  - Interpolator, 125
- setName
  - ObjectData, 143
- setNameList
  - GUITimeline, 116
- setObject
  - Renderer, 174
- setQuality
  - ObjectData, 144
- setRenderObject
  - GUIGLCanvas, 83
- setStepWidth
  - GUIColorScalePanel, 59
- setTextColor
  - GUIColorScalePanel, 60
- setValue
  - GUITimeline, 116
- setValueImg
  - GUIRenderCutCanvas, 104
- show\_extrapolated
  - Renderer::Viewport\_info, 212
- show\_sensordata
  - Renderer::Viewport\_info, 212
- showEdges
  - Renderer::Viewport\_info, 212
- showExtrapolatedCheckBox
  - ViewpropBox, 217
- showFaces
  - Renderer::Viewport\_info, 212
- showPoints
  - Renderer::Viewport\_info, 212
- showShowSensorData
  - ViewpropBox, 217
- SimpleAnalyzerApp, 180
  - ~SimpleAnalyzerApp, 182
  - addObject, 182
  - current\_data\_object\_index, 184
  - data\_objects, 184
  - getActiveObject, 182
  - getCurrentDataObjectIndex, 183
  - getDataObjects, 183
  - getVisualizationInfo, 183
  - OnInit, 183
  - removeCurrentObject, 183
  - setCurrentDataObjectIndex, 183
  - visualization\_info, 184

- specificHeatCapEdit
  - PropertiesBox, 166
- specificHeatCapLbl
  - PropertiesBox, 166
- specificheatcapacity
  - ObjectData::MaterialData, 127
- sqr
  - GeometryClasses.cpp, 256
  - Interpolator.cpp, 259
  - Utils, 25
- start\_col
  - CsvToSdConverter::Options, 153
- startrow
  - OdisiToSdConverter::Options, 156
- std, 13
- std::vector
  - element, 196
- std::vector< T >, 195
- step\_width
  - GUIColorScalePanel, 61
- sub
  - Vector3D, 208
- subnames
  - Utils::SensorData, 177
- tab\_space\_count
  - OdisiToSdConverter::Options, 156
- table
  - GUIAnalyzeOutputWindow, 48
- temperature
  - Utils::SensorPoint, 178
- tetface\_indices
  - Exporter.cpp, 223
- tetgeninput
  - ObjectData::MaterialData, 127
- tetgenoutput
  - ObjectData::MaterialData, 127
- Tetrahedron, 185
  - getV1, 186
  - getV2, 186
  - getV3, 187
  - getV4, 187
  - getVert, 187
  - Tetrahedron, 186
  - verts, 188
- text\_color
  - GUIColorScalePanel, 61
- threadcountedit
  - GUICutRenderWindow, 78
- threadcountlbl
  - GUICutRenderWindow, 78
- time\_step\_delta
  - CsvToSdConverter::Options, 153
  - OdisiToSdConverter::Options, 156
- timecol
  - CsvToSdConverter::Options, 153
  - OdisiToSdConverter::Options, 156
- timed
  - Utils::SensorData, 177
- timestamps
  - Utils::SensorData, 177
- ToClipboard
  - GUIAnalyzeOutputWindow, 48
- toolbar
  - GUIMainWindow, 98
- transforming
  - GUIColorScalePanel, 61
- transpose
  - Matrix3D, 132
- tri
  - Utils::CutRender\_info, 43
- Triangle, 188
  - ~Triangle, 189
  - getNormal, 189
  - getV1, 190
  - getV2, 190
  - getV3, 191
  - getVert, 191
  - print, 191
  - Triangle, 189
  - verts, 192
- trilabel
  - GUICutRenderWindow, 78
- TsdMerger, 192
  - getTextBlock, 193
  - merge, 193
  - opts, 195
  - parseArguments, 194
  - parseFile, 194
  - writeOutputFile, 195
- TsdMerger::Options, 153
  - auto\_delta, 154
  - delta, 154
  - max\_dt, 154
  - offset, 154
- upToDateLbl
  - PropertiesBox, 166
- Update
  - GUIAnalyzeOutputWindow, 48
- updateObjectPropGUI
  - GUIMainWindow, 95
- updateViewPropGUI
  - GUIMainWindow, 96
- updating
  - GUIMainWindow, 98
- Utils, 13
  - ALGORITHM\_RAY, 14
  - ALGORITHM\_TETRAHEDRONS, 14
  - clampHue, 14
  - copySensorPoint, 15
  - floattostr, 15
  - floattowxstr, 16
  - getPointValue, 18
  - hsvToRgb, 19
  - nextCombination, 20
  - PIM\_algorithm, 14
  - pointInsideMesh, 21

- pointInsideTetrahedron, 22, 23
- rayIntersectsTriangle, 24
- sqr, 25
- utils.cpp
  - EPSILON, 268
- Utils::CutRender\_info, 42
  - img\_height, 42
  - img\_width, 42
  - in\_volume\_algorithm, 43
  - mmperpixel, 43
  - tri, 43
- Utils::SensorData, 175
  - current\_time\_index, 176
  - data, 176
  - markers, 177
  - name, 177
  - subnames, 177
  - timed, 177
  - timestamps, 177
- Utils::SensorPoint, 177
  - coords, 178
  - temperature, 178
- Utils::SensorPointComparator, 178
  - getDistance\_d, 179
  - meshpoint, 180
  - operator(), 180
- Utils::SortStruct, 184
  - distance, 185
  - pointIndex, 185
- Utils::Visualization\_info, 217
  - max\_visualisation\_temp, 218
  - min\_visualisation\_temp, 218
- VIEWBOXWIDTH
  - GUIMainWindow.cpp, 239
- value
  - Analyzer::AnalyzerData\_point, 35
  - GUITimeline, 118
- value\_img
  - GUICutRenderWindow, 78
  - GUIRenderCutCanvas, 105
- Vector3D, 196
  - ~Vector3D, 198
  - add, 199
  - coords, 209
  - copy, 199
  - crossProduct, 200
  - dotProduct, 201
  - equals, 201
  - getAngleTo, 202
  - getDistanceTo, 203
  - getLength, 204
  - getX, 205
  - getXYZ, 205
  - getY, 206
  - getZ, 206
  - mult, 207
  - normalize, 207
  - operator<, 209
  - print, 208
  - printTo, 208
  - sub, 208
  - Vector3D, 198
  - Vector3D, 198
- verts
  - Tetrahedron, 188
  - Triangle, 192
- view\_scroll\_win
  - GUIMainWindow, 99
- viewScaleEdit
  - ViewpropBox, 217
- viewScaleLbl
  - ViewpropBox, 217
- viewbox
  - GUIMainWindow, 99
- viewport
  - Renderer, 175
- ViewpropBox, 212
  - ~ViewpropBox, 215
  - colorRangeLbl, 216
  - colorRangeMaxEdit, 216
  - colorRangeMinEdit, 216
  - edgesCheckBox, 216
  - facesCheckBox, 216
  - getColorRangeMaxEdit, 215
  - getColorRangeMinEdit, 215
  - getEdgesCheckBox, 215
  - getFacesCheckBox, 215
  - getMatVisibilityListBox, 215
  - getPointsCheckBox, 215
  - getShowExtrapolatedCheckBox, 215
  - getShowShowSensorData, 215
  - getViewScaleEdit, 216
  - matVisibilityListBox, 216
  - matVisualizationLbl, 216
  - pointsCheckBox, 217
  - resize, 216
  - showExtrapolatedCheckBox, 217
  - showShowSensorData, 217
  - viewScaleEdit, 217
  - viewScaleLbl, 217
  - ViewpropBox, 214
  - ViewpropBox, 214
- ViewpropBox.cpp
  - renderchoices, 253
- visible
  - ObjectData::MaterialData, 128
- visualization\_info
  - SimpleAnalyzerApp, 184
- volume
  - Analyzer::AnalyzerData\_material, 32
  - Analyzer::AnalyzerData\_object, 34
- width
  - GUIColorScalePanel, 61
  - Renderer::Viewport\_info, 212
- widthHeightLbl
  - GUICutRenderWindow, 78

writeOutputFile

    CsvToSdConverter, [41](#)

    OdisiToSdConverter, [151](#)

    TsdMerger, [195](#)

x

    GUIColorScalePanel, [61](#)

xedit

    GUIAnalyzePointWindow, [51](#)

y

    GUIColorScalePanel, [61](#)

yedit

    GUIAnalyzePointWindow, [52](#)

zedit

    GUIAnalyzePointWindow, [52](#)

zoom

    GUIRenderCutCanvas, [105](#)

    GUITimeline, [118](#)

    Renderer::Viewport\_info, [212](#)